BAOU Education for All

DR.BABASAHEB AMBEDKAR OPEN UNIVERSITY

BCA

BCAR-504

Internet Programming (ASP.NET using C#)

INTERNET PROGRAMMING (ASP.NET USING C#)

DR. BABASAHEB AMBEDKAR OPEN UNIVERSITY
AHMEDABAD

Editorial Panel

Author : Dr. Neelam Mehta
Assistant Professor
Khyati School of Computer Application, BCA
Affiliated to Gujarat University

Editor : Dr. Keyur B. Shah
Ph.D. (Computer Science)
Network Administrator
Gujarat Technological University

Language Editor : Dr. Bhupesh Gupta
M.A., M.Phil., Ph. D.
Assistant Professor
Lokmanya College of Commerce
Ahmedabad

Edition : 2022

Acknowledgment

Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.

ROLE OF SELF–INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self–instructional material is completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self–instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual–skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behaviour should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminate interactive communication cues, such as pauses, intonation and gestures, associated with the face–to–face method of teaching. This is

particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self–Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self–instructional materials, be they print or otherwise. These materials are designed to achieve certain pre–determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user–friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included many tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

INTERNET PROGRAMMING
(ASP.NET USING C#)

Contents

ExpressionValidator Control, CompareValidator Control, CustomValidator Control, ValidationSummary Control

Unit 6 **WORKING WITH DATABASE, ACCESSING AND DISPLAYING DATA USING DATA SOURCE WEB CONTROLS**

Introduction, Various Database Systems, Key Constraints, Creation of Database, Tables, Records Insertion in Data Tables, Data Source Controls, SqlDataSource Control, Data Connection, Configure Select Statement, Test The Query, Code Behind the Query, Working with Sql Queries, Where Clause, Order by, Sorting in Ascending and Descending Order, Filtering SqlDataSource Controls, Sorting SqlDataSource Controls, Studying The Sql Code

Unit 7 **WORKING WITH DATABASE**

Introduction, CRUD (Create, Retrieve, Update and Delete) Operation on Data, GridView Control, DataList Control, DetailsView Control, ListView Control, FormView Control, SqlDataSource Control, AccessDataSource Control, Repeater Control, XmlDataSource Control, EntityDataSource Control

---

BLOCK 3 : WORKING WITH DATABASE

Unit 8 **WORKING WITH DROPDOWN LISTS, RADIO BUTTONS, CHECKBOXES, REVISITING DATA BINDING**

Introduction, Data Binding to A List Web Control, Benefits of Populating the List Dynamically Using SqlDataSource Selectedindexchanged, Filtering Using Dropdownlist and Checkboxlists, Selecteditem, Selectedvalue, Radiobuttonlist, Repeatdirection, Checkboxfield, Hyperlinkfield, Imagefield, Sql Wild Cards, Two-Way Data Binding

Unit 9 **LINQ QUERIES**

Introduction, Introduction of LINQ Queries, Query Operators, LINQ to Objects, LINQ to ADO.NET, LINQ to XML, LinqDataSourceControl

Unit 10 ADO.NET FRAMEWORK

Introduction, Concept of Data Modeling, Object Services, Entity SQL Language, Entity Client Provider, Entity Framework Controls

---

BLOCK 4 : WORKING WITH USER

Unit 11 WORKING WITH USER LOGINS

Introduction, User Accounts, Membership, Putting Users into Various Roles, Access Rules, Creating New User Accounts Via Websites, Autogeneratepassword & Required email Properties, Logging to The Website Login Control, Logging Out, Logout action Property, Working with Login View and Login Name Web Controls, Forgotten Password, Change Password Control

Unit 12 INTORDUCTION OF MASTER PAGES

Introduction, Importance of Master Pages and Themes, Master Pages and Themes Creation, Configuration of Master Pages and Themes

Unit 13 FILES AND STREAMS

Introduction, Introducing System.IO Namespace, Working with Drives, Directories, Files, Properties and ACL, Directory Class, DriveInfo Class, DirectoryInfo Class, Creating, Copying Directory and Subdirectories, Retrieving Files from a Directory, Files : Creating, Copying, Reading, Appending, Renaming, Compressing

Unit 14 APPLICATIONS

Introduction, Employee Application, CRUD Operation Application

Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-504

INTERNET PROGRAMMING (ASP.NET USING C#)

BLOCK 1 : BUILDING BLOCKS AND WEB CONTROLS

BUILDING BLOCKS AND WEB CONTROLS

Block Introduction :

Basically we use ASP.NET for web development. It is a platform which consists of programming model and several services for developing attractive and useful web applications by a software infrastructure. Web controls are mainly elements of HTML covered simple and easy scripting tags and make available good functionality in our web pages.

In this block, we will discuss about the benefits of .NET Framework, it includes major languages and framework developments. We will discuss about an integrated development environment (IDE) i.e. software for developing applications that syndicates common developer tools into a specific graphical user interface.

In this block, we will learn and understand about various types of files, directives, location of web application where the sites host and how can we compile the code of web applications dynamically.

Block Objectives :

After learning this block, you will be able to understand :

- About Benefits of .NET Framework
- Overview of ASP.NET
- Overview of Technologies related to .NET
- Web Application location
- Types of files
- ASP.NET Coding Models
- Directives
- Code Sharing Implementation
- Dynamic Compilation

Block Structure :

Unit 1 : Introduction Asp.Net Programming Model

Unit 2 : Studying Building Blocks

Unit 3 : Studying Web Controls of Text

Unit 4 : Studying Navigation Controls

Unit 01

INTRODUCTION ASP.NET PROGRAMMING MODEL

UNIT STRUCTURE

1.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About the .NET framework on which we can do work
- How do we write code and ways of implementation
- Process of compilation

1.1 Introduction :

ASP.NET is a platform for the web development, provides a model for programming, a complete software structure and several other services essential to accumulate solid web applications, and mobile devices. To set a communication between browser–server it uses HTTP protocol, HTTP commands and policies. These commands and policies are written using the components present in the .Net framework. It works on top of the HTTP protocol, hierarchy in .NET framework used in the codes. These web applications are developed using HTML, CSS, and JavaScript.

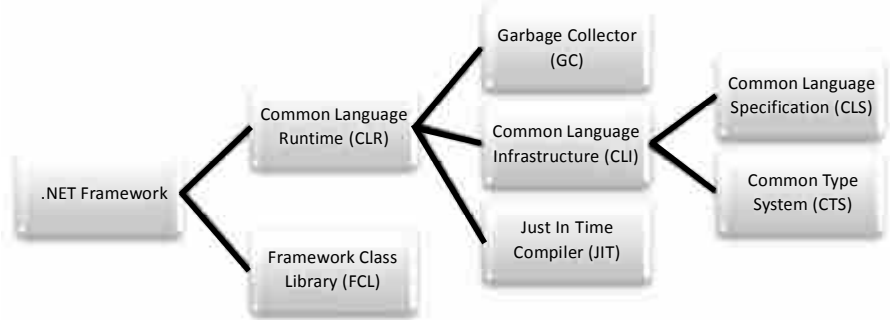
1.2 Benefits of .NET Framework :

.NET Framework is an accomplished implementation platform that delivers a variety of services to its running applications. It has two main components: Common Language Runtime (CLR) handles execution of web applications and the .NET Framework Class Library (FCL) has the several reusable source codes which are already tested and stored in the library. FCL provides this library to the developers to use for developing their web applications. The benefits of .NET Framework are as follows:

- **Cross–platform Support :** It states that any windows platform that supports the Common Language Runtime (CLR) can execute .NET applications. These applications facilitate internal operation ability between several Windows Operating Systems.
- **Language Interoperability :** .NET Framework enables codes written in other different languages to interact with each other. This increases the ability of reusing codes and increases the effectiveness of the development process.
- **Automatic Management of Resource :** .NET Framework provides a facility called CLR that automatically trails the resource (files, memory, database connection and network) procedure and supports us in execution the task of manual resource management.
- **Ease of Deployment :** .NET Framework installs applications that do not disturb the existing applications. Several times we want to copy the applications with its components on the objective computer. But in .NET, applications are organized as assemblies. Assemblies store information about the different versions of the component used by any application.
- **Consistent Programming Model :** In .NET Framework, we can use consistent object–oriented Programming model to create programs for performing different jobs, like saving data and connecting to databases, writing and reading in files.

1.3 Overview of .NET Framework :

.NET Framework is a common stage to run an application that was constructed using the different languages such as C#, VB.NET, Visual Basic, FORTRAN, COBOL etc. It is also used to make different types like form based, console–based, mobile and web–based of applications which are based on form, console, web and mobile or services that are presented in Microsoft environment. .NET framework is an object oriented language that can run on the windows platform.

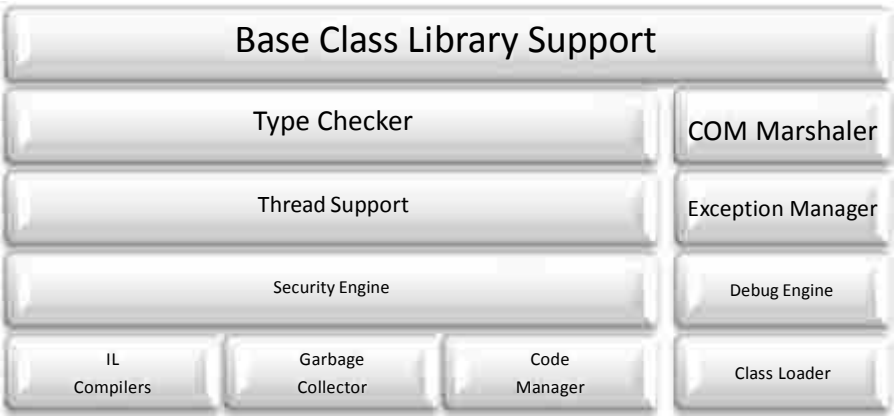


.NET Framework

- **Common Language Runtime (CLR) :**

The Common Language Runtime is the implementation device for .NET applications. It offers several services that includes loading and execution of code, Application memory segregation, Authentication of type protection, Transformation of Intermediate Language to native code, Access to metadata, memory management for managed objects, Prosecution of code access safety, Exception handling, Operations in between managed codes, COM objects, and pre–existing DLLs, Computerisation of object layout, Care for developer services.

CLR is a main part of a .NET framework that works like an essential element of the .NET Framework to perform the different languages program like C#, Visual Basic, etc. A CLR also supports to change a source code into the byte code, and this byte code is called as CIL (Common Intermediate Language) or MSIL (Microsoft Intermediate Language). After changing into a byte code, a CLR uses a Just–In Time compiler at run time that supports to change a CIL or MSIL code into the device or native code.



CLR (Common Language Runtime)

Class loader loads classes into CLR. *Intermediate Language (IL) Compiler* changes Microsoft Intermediate Language (MSIL) code into native code. *Code manager* accomplishes the code throughout implementation. *Garbage Collector* achieves automatic memory management. *Security engine* implements security restrictions as code level security folder level and device level security. *Type checker* implements strict type checking. *Thread support* offers multithreading support to applications. *Exception manager* offers a device to handle the run–time exceptions handling. *Debug engine* permits developer to debug different types of applications. *COM Marshaler* permits .NET applications to interchange data with COM applications. *Base class library* offers the classes or types that the applications want at run time.

❑ **Check Your Progress – 1 :**

1. .NET Framework has _____ main components.

   a. 4 b. 5 c. 2 d. 1

2. _____ permits .NET applications to interchange data with COM applications.

   a. Type Checker b. COM Marshaler

   c. Thread support d. All of these

Process of CLR execution :

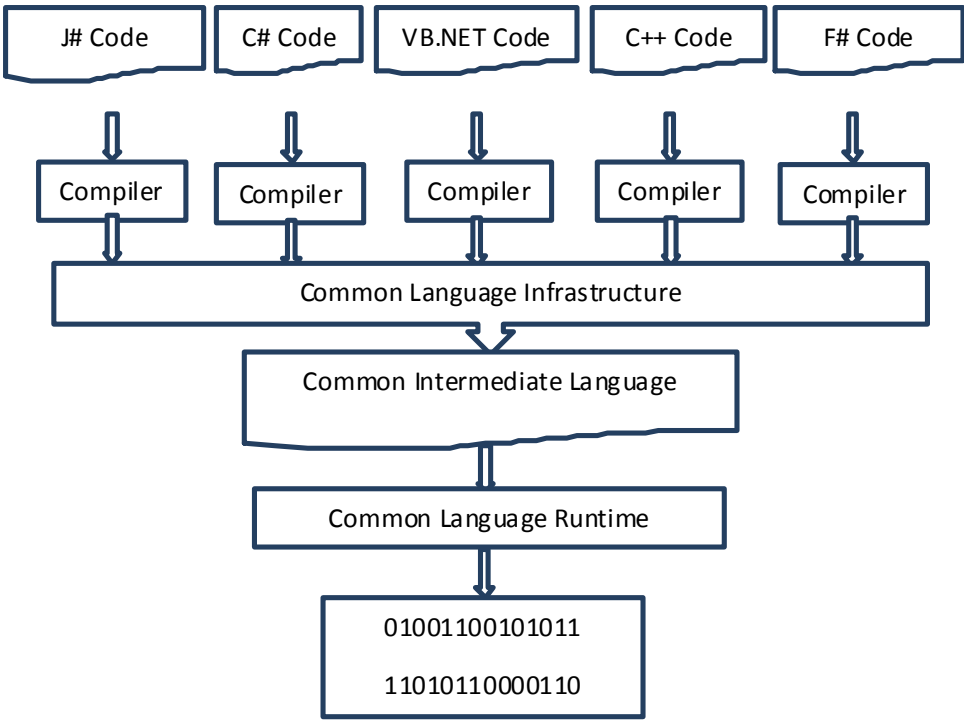
Away from only code execution, parts of the execution process right affect the design of our application and how a program acts at runtime. When we or process picks a .NET application for execution, the CLR executes a unusual process to run our application as shown in the figure.

- At the beginning point Windows i.e. Operating System will start. After that CLR starts execution.
- During the execution of an application, Operating System checks the file that it has a special header which indicates is it a .NET application or not. If it is not then Windows remains run the application.
- If it indicates the application is of .NET then Windows begins the CLR and application goes through the CLR for the execution process.
- For the execution process the CLR firstly loads the executable assembly to find the staring entry point and starts process.
- The executable assembly could refer further assemblies like Dynamic Link Libraries (DLLs) to load using CLR according to its need. Only when the CLR needs access of assembly's code then it's loaded otherwise not.
- C# compiler creates Intermediate Language (IL) to code. This IL converts into binary code through CLR which the operating system knows. It is the concern of the Just–In Time (JIT) compiler.
- The JIT compiler compiles code at the first time of execution. After the code conversion by JIT compiler the CLR grasps the compiled code in a functioning set.
- Whenever next time the code executes the CLR verifies its functioning set and runs the code openly if it is previously compiled.
- The Just–In Time compiler works at the procedure level. When the CLR starts execution this compiler compiles the Main () function or procedure in C#. All methods compiled just before the execution starts.
- This whole procedure of verifying functioning set, compilation of JIT, assembly loading and the execution remains till the program stops. After code has been JIT compiled, it executes as fast as any other binary code in memory.
- **Garbage Collector (GC) :**

The Garbage Collector (GC) is a memory manager who manages the memory allocation and release of memory. We need not to worry about how it manages the memory of the objects. "new" keyword is used to declare an allocation for an object. If there is not sufficient memory space to allocate an object, the GC collects and arranges the garbage memory so that memory can be available for new allocations. This all process is known as Garbage Collection.

- **Common Language Infrastructure (CLI) :**

Common Language Infrastructure (CLI) is used for executing High Level Language (HLL) applications on several types of machines without modifying anything in the source code because CLI compiles applications in the Intermediate Language that directly compiled in the form of built–in system code.

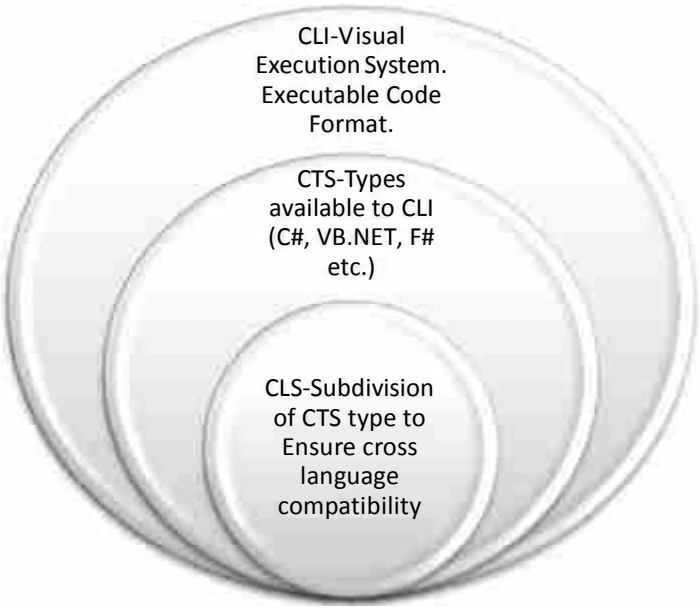


Common Language Infrastructure

Current CLI languages are Ada for .Net, C#, C++, ClojureCLR, Component Pascal, Eiffel, F#, F*, Fantom, IronPython, IronScheme, Limnor Studio, Oxygene, PascalABC.NET, PeachPie, PowerBuilder, RemObjects Mercury, Small Basic, SilverFrost FTN95, Team Developer, Visual Basic (VB.NET), Visual COBOL, PowerShell, XSharp etc.

- **Common Language Specification (CLS) :**

Common Language Specification (CLS) explains the simple and straightforward guidelines essential for any language aiming common language infrastructure (CLI) to control internally with CLS languages. CLS agrees elasticity in spending non–compliant types in the internal execution of modules with CLS–compliant requests. CLS acts as a device for incorporating several languages into one authority in a smooth method. Common Language Specification explains a division of Common Type System (CTS).



Common Language Specification

- **Common Type System (CTS) :**

The Common Type System (CTS) normalizes the types of data of all languages using .NET below the authority of .NET to a shared data type for easy and smooth communication between the .NET languages. Let us take an example how data type converts into common data type using CTS. When we declare data type "int" in VB.Net and C#, it converts into int32 or int16. That means after conversion both languages have same data type which delivers stretchy communication among the C# and VB.Net. The common type system performs the following functions :

- Starts a framework that supports cross–language amalgamation, type security, and high–performance code implementation.
- Delivers an object–oriented model which helps the whole execution.
- Describes guidelines that languages essentially track that supports confirm that objects printed in several programming languages can relate together.
- Offers a library that comprises the simple data types like Int16, Int32, UInt64, and Char, and Byte, Boolean etc. used in developing any application.

- **Just In Time Compiler (JIT) :**

Just–In–Time compiler (JIT) is a portion of Common Language Runtime (CLR) in .NET which is accountable for dealing with the implementation of .NET programs irrespective of any .NET language. A definite language compiler translates the source code into the intermediate language (IL). It is then altered into the machine code by the Just–In–Time (JIT) compiler. This machine code is particular to the situation which the JIT compiler goes on.

❑ **Check Your Progress – 2 :**

1. _____ is a main part of a .NET framework that works like an essential element of the .NET Framework to perform the different languages program. It also supports in changing source code into byte code.

   a. CTS b. FCL c. JIT d. CLR

2. _____ is a definite language compiler translates the source code into the intermediate language (IL).

   a. Type Checker b. JIT

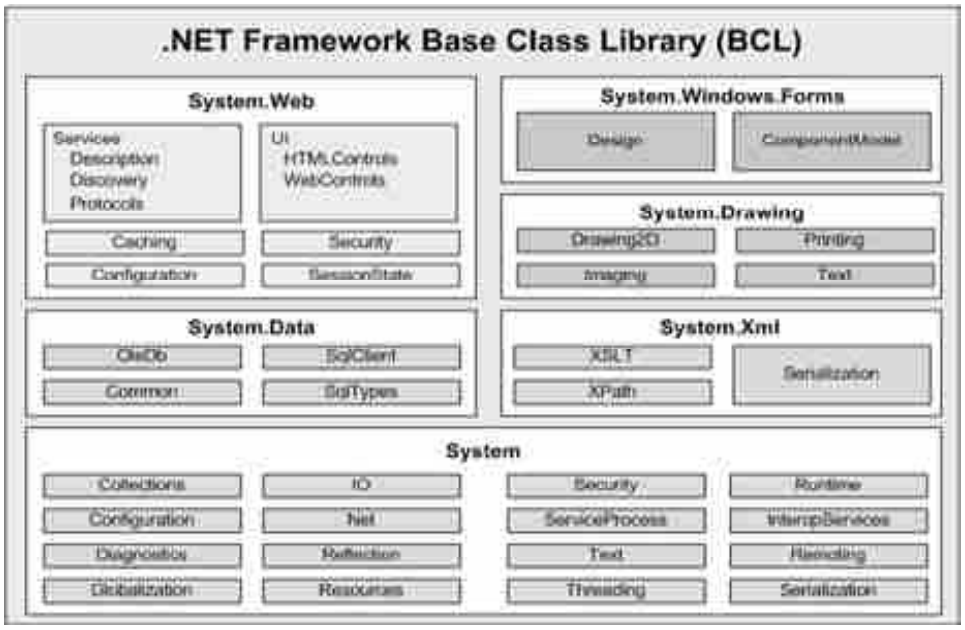
   c. Common Type System d. All of these

- **Framework Class Library (FCL) :**

The Framework Class Library (FCL) makes available the roles of system in the .NET framework since it consists of various data types, classes, etc. to execute several methods and form several kinds of applications like desktop applications, website applications, and mobile applications and so on. The FCL is incorporated with the CLR and is practiced by all the languages of .NET like C#, VB .NET, F#, etc.

Before knowing more about FCL we need to understand the concept of Base Class Library (BCL) as it is the primary set of classes which help as the elementary interface for application programming of the CLR. BCL includes the classes in namespaces like System.Diagnostics, System.Globalization, System.Resources, System.Text, System.Runtime.Serialization and System.Data

etc. Several classes in mscorlib.dll and in System.dll and System.core.dll are included as a part of BCL. The BCL is simply runtime library for recent programming languages. It assists as the Standard for the runtime library for the language C# in addition to one of the CLI Standard Libraries.



Base Class Library

According to the figure you can understand that the BCL makes available classes which summarize common methods, like rendering of graphics, interaction with databases, XML manipulation in documents, reading of files and writing for the same.

The .NET Framework class library is nothing but a simple library of several classes which are written in C#. These classes can be used from any type of programming language which is based on CLR. BCL is the subset of the FCL.

According to all these we understood that any .NET application compiles using CLR and designed using several utility features and the wrappers around Operating System functionality running on different .NET frameworks using FCL.

1.4 Brief of Visual Studio IDE :

An Integrated Development Environment (IDE) is a tool or you can say software which enables development of any application. According to our point of view or .NET applications, Visual Studio (VS) is the extremely used IDE.

The key ability the IDE offers is creation of forms during the time of designing. Simply we place the controls in the layout and application can be rendered or extracted during runtime. IDE offers easier technique of constructing applications in minimum time.

Features of IDE :

There are few main features of Integrated Development Environment (IDE) which are as follows :

- Single development environment from where we can develop an application of .NET in any language. Thus we need not require any type of conversion in .NET applications development.

- Single .NET platform provided for any application which has been made on the source code which is written in several languages.
- Compilation starting within the situation on the basis of definite formation options
- Incorporated debugger which efforts at source level and machine level
- Plug–in tools which supports to improve tools for area specific languages
- Flexible environment to support the user to organize the IDE on the basis of the necessary settings
- To see or search the contents from internet like source–code, help etc. online there is a browser inbuilt within the Integrated Development Environment (IDE).
- Few other features are high performance, cross–platform and container support, asynchronous via async/await rich development environment, language independent, support for web sockets, action filters, globalization and localization, security, supports HTML5 form types, NET Web API.

1.5 Overview of Asp.Net :

In this developing era everybody wants to build several types of applications and for this they think about any platform from where they can start development. For this .NET is the best developer platform made up of several libraries, tools, technologies, and programming languages.

ASP.NET spread out this development platform for developing web applications with the help of tools and libraries. ASP.NET is freeware framework for website for creating up–to–date website applications and facilities with .NET. ASP.NET is cross stage and executes on Windows, Linux, and macOS. It uses HTTP commands and workings with HTTP protocol to ensure exact communication between server and browser. It simplifies the working of websites and applications as it builds a library of many codes and tools. We can use ASP.Net for front–end and back–end development.

The developer sends the request along with the ASP.NET server–side controls for various browsers, based on the header values of User–Agent. We can also use the project named Wireless Universal Resource File Locator (WURFL). This project includes a current set of browser definitions which allow users to determine the features of several devices on the server, like input device method, supportive format of audio and several other server features. This determination technique is known as browser detection technique. It is quite difficult to use this technique.

To resolve the problem with the above browser detection technique, there is one another useful feature detection option, that is, HTML5 (Hyper Text Markup Language). The support of various features of browser got tested. If it supports the specific feature then the browser uses it otherwise not. Feature detection is good in the creation of maintainable code but browser detection is common enough from several years.

1.6 Overview of Technologies Related to .NET :

.NET framework is an improvement bionetwork planned to upkeep the production of web applications and software. Many developers support .NET above related platforms due to its easy features and advanced functionality.

To increase the processes and support the security operations, the .NET framework introduced. The first .NET framework was started in 2002 and used C# language for coding.

Firstly, the .NET was only for those developers who were developing Windows–based applications and servers. Now a day, it's different. The .NET Framework or .NET Core permits users to develop applications on Windows, Linux and MacOS.

.NET framework handles various types of programming languages like C++, VB.NET, F# etc. Several developers choose C# while coding with .NET, as it is an object–oriented language. That means it claims superb functionality and can increase developer efficiency. C# includes type declaration, type safety, scalability support and the quick development results.

- ***.NET CORE :*** .NET CORE is a cross–platform which allows developers to develop applications, facilities, and websites with .NET.
- ***.NET 5 :*** It is nothing but an extension of .NET Core with additional features and functionalities in .NET for combined cross–platform plans and tools.
- ***ASP.NET :*** ASP.NET is an extension to the .NET platform which is committed to constructing web applications with several tools and libraries. It has web–page templating language rules, altered libraries for public web forms for example Model View Controller (MVC), Authentication System.
- ***ASP.NET CORE :*** ASP.NET Core is the newer version of ASP.NET. ASP.NET Core 2.0 is specified to be finished double quicker than the last ASP.NET versions. It is free and can be implemented in all Operating Systems from Windows to Mac and Linux.
- ***.NET Standard :*** .NET Standard describes the base class libraries which must be executed which is really an API requirement. It is used to execute base class libraries (BCL) that comprise classes such as collections, networking, I/O, exception handling, and more.
- ***Visual Studio :*** Microsoft Visual Studio is an entirely–featured IDE for developers through domains and knowledge points from learners to innovativeness level engineers that perform as an editor of code for web applications, mobile applications, and development of web services.
- ***Visual Studio Code :*** VS Code is developed by Microsoft as a source code editor for all type of Operating System platforms from Windows, macOS, and Linux and includes numerous features supporting syntax highlighting, debugging, refactoring, git operation, and many more.
- ***Microsoft Silverlight :*** Microsoft Silverlight permitted cooperative media knowledge as a free browser plugin widely to develop applications and make available mobile application involvement.
- ***Blazor :*** One another feature of ASP.NET is Blazor which permits user to make user interfaces for web applications using C#. These types of applications have some type of components of Web User Interfaces which we can reuse whenever we want that are developed in CSS, C#, HTML etc.

- ***Blazor Mobile Binding :*** Mobile Blazor Binding allows developers to code for the Native User Interface of the applications. For rapid use of rich native user interface the contents of the code, layout and styles are developed.
- ***Hybrid Apps :*** Hybrid applications are the web applications which can be wrapped in a native wrapper not like Native applications that are particularly designed for mobile phones.
- ***Xamarin :*** Xamarin is a raised area, freeware software which provides permission to developers to make cross–platform applications for superior level applications or softwares.

❑ **Check Your Progress – 3 :**

1. _____ an extension to the .NET platform which is committed to constructing web applications with several tools and libraries.

   a. ASP.NET b. Blazor Mobile Binding

   c. Xamarin d. Hybrid Apps

2. _____ project includes a current set of browser definitions which allow users to determine the features of several devices on the server.

   a. XML

   b. ASP.NET

   c. Hyper Text

   d. Wireless Universal Resource File Locator

1.7 Web Application Locations :

For creating any web application we need Visual Studio on our computer device. For a free version of it download from this link Download Visual Studio from here. After downloading Visual Studio, we will start creating new ASP.NET Core project.

For developing any project type, we need to follow the following steps:

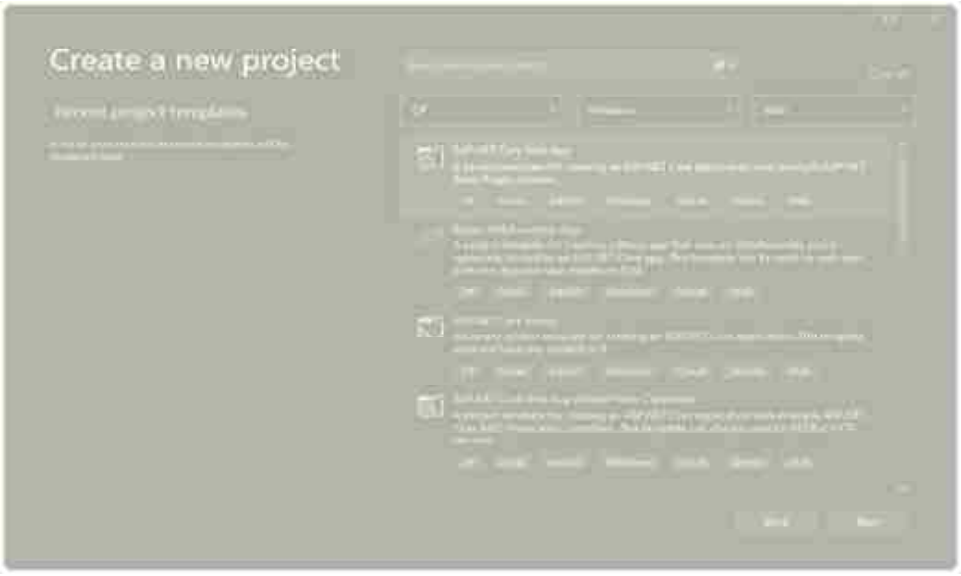
I Step : On the start window, select create a new project.

II Step : In the Create a new project window, from all language list select C#.

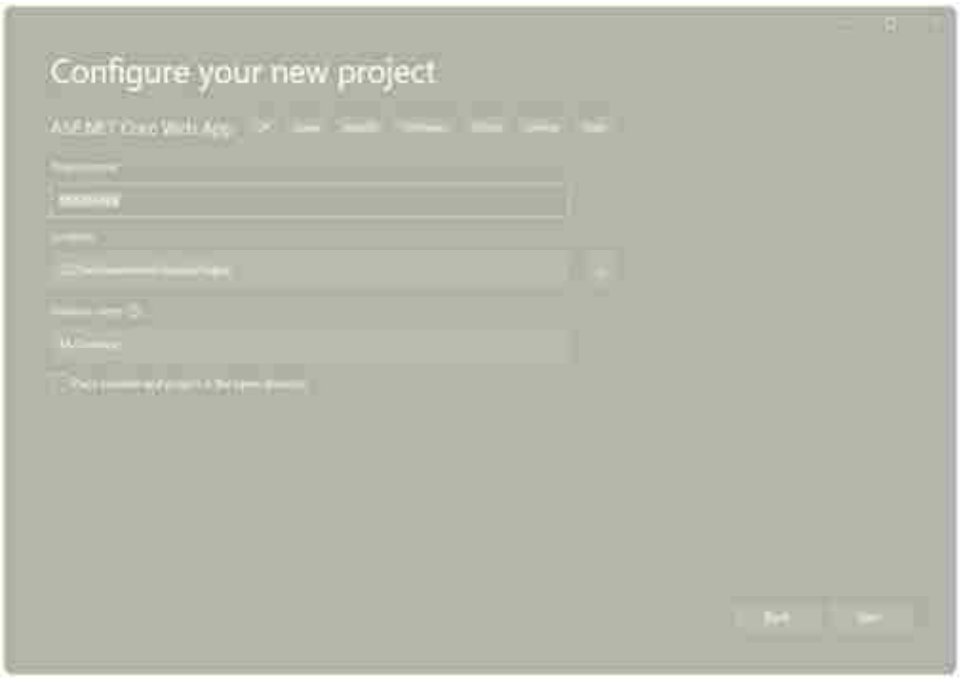
III Step : Select Windows from the platform list.

IV Step : Web from the list of project types.

V Step : Select the ASP.NET Core Web App Template and select Next



VI Step : In the Configure your new project window, enter MyCoreApp in the Project name field. Then, select Next.



VII Step : In the Additional information window, confirm that .NET 6.0 looks in the Target Framework field. From this window, we can allow Docker support and add authentication support. For selecting the Authentication Type, there will be 4 options in drop–down menu:

- **None :** No Authentication
- **Individual Accounts :** This type of authentication is stored in a local database or Azure–based database.
- **Microsoft Identity Platform :** This option uses Active Directory or Microsoft 365 for authentication.
- **Windows :** It is convenient for intranet applications.

Leave the Enable Docker box unchecked, and select None for Authentication Type. Next, select Create.

VIII Step : Visual Studio will open up your new project

1.8 Types of Files :

As you are undoubtedly aware, web applications can use a variety of file kinds. The IIS Server or ASP.NET Applications support and manage this. Generally speaking, the Add New Dialog box in Visual Studio 2010 may be used to create the majority of file types. You value certain file kinds higher since they aid in your understanding of the file's behaviour and coding. We will see later that ASP.NET 4.0 supported a wide variety of file types. Using application mappings, all file formats are mapped to web apps. For instance, when we open a.txt file, Notepad is automatically launched because the window.txt file is immediately linked to the Notepad.exe file. All.NET developers need to be aware of these extension files' uses.

Some file types are handled by ASP.NET, which are defined below :

- ***.asax :*** This abbreviation refers to the Global.asax file, which contains code from the HttpApplication class. It is located in the Application Root Directory.
- ***.ascx :*** This is the abbreviation for a web user control file. It could be in a subdirectory or the application root directory.
- The extension.***aspx*** stands for an ASP.NET Web form. It could be in a subdirectory or the application root directory.
- The term ***".asmx"*** designates an xml web services file that contains classes and methods. It could be in a subdirectory or the application root directory.
- ***.compile :*** This is a precompiled stub file that points to an assembly that represents a website file that has been compiled. First, there are precompiled.aspx,.ascx, and .master files. It is found in the Bin subfolder.
- ***.browser :*** This is the name of a file that defines a browser and lists its features. The App Browsers Subdirectory is updated.
- The term ***".dll"*** designates compiled class library files (assembly file). It is found in the Bin subfolder.
- The extensions ***.cs,.vb, and.jsl*** designate source–code files that include application logic. The App Code Subdirectory or the same directory as the web page contains it.

- ***.csproj, vbproj, and vjsproj :*** It alludes to a project file for a client application project created in Visual Studio. It is located in the Visual Studio Project Directory.
- ***.master :*** This term refers to a master page that specifies a web application's page layout.
- It is in a subfolder or the application root.
- ***.mdf, .sdf :*** The SQL database file is referred to. It is located in the App Data directory.
- ***.mdb, ldb :*** This is a file extension for an Access database. It is located in the App Data directory.

- ***.skin :*** This is the name of a skin file. It is applied to web controls to provide uniform formatting. It is found in the subdirectory App Themes.
- ***Sitemap :*** It alludes to a site–map file that contains the website's organisational structure. It is located in the Application Root Directory.
- The terms ***".resx"*** and ***".resources"*** refer to resource files that contain resource strings. The App Global Resources or App Local Resources subdirectory is where it is located.

1.9 ASP.NET Coding Models :

Two components make up an ASP.NET Web page :

- Markup, server controls, and static text are visual components.
- Programming code, such as event handlers and other pieces of code, for the page.

The single–file page model and the code–behind page model are two models offered by ASP.NET for managing the visual components and the code. The controls and code for the two versions are interchangeable, and both models perform the identical functions.

Modeling a Single–File Page

The Markup and programming code for a page are both contained in the same actual.aspx file under the single–file page paradigm. The script block containing the programming code has the property runat="server" to designate that ASP.NET should run it.

The single–file page with a Button control and a Label control is displayed in the following code example. The Click event handler for the Button control is displayed in the highlighted area of the script block.

The script block may have as much code as is necessary for the page. The code can include methods, properties, event handlers for page controls

(like in the example), and any other kind of code that would typically be found in a class file. Single–file pages are processed at runtime as classes that inherit from the Page class. There is no explicit class declaration on the page. Instead, a new class that has the controls as members is created by the compiler. (Some controls are children of other controls; not all controls are available as page members.) The page's code is incorporated into the class; for instance, any event handlers you design are added as members of the derived Page class.

You cannot have a using statement in your code because every piece of code on the page is made a member of the Page class. Place a @ Import directive that details the namespace to import in its place. It's possible that you'll additionally need to include a reference to the DLL containing the namespace.

The Model for Code–Behind Pages

The code–behind page architecture enables you to retain the programming code in a separate file from the HTML in one file (the.aspx file). Depending on the programming language, you're using, the code file's name changes.

Note : You cannot generate code–behind files for ASP.NET Web sites with all.NET programming languages. Partial classes must be supported by languages. For instance, because J# does not support partial classes, code–behind files for ASP.NET pages cannot be created.

When dealing with a page called Default, for instance, the Markup is in the file SamplePage.aspx and the code is in Default.aspx.vb (for Visual Basic), Default.aspx.cs (for C#), and so on.

Note : Version 2.0 of the .NET Framework has a different code–behind paradigm than earlier iterations.

The Default page example from the earlier section would have two components in the code–behind approach. The Markup would be comparable to the Default page and would be contained in a single file (in this case, Default.aspx), as demonstrated in the following code example.

The single–file paradigm and the code–behind model differ in two ways on the.aspx page. There isn't a script block with the runat="server" attribute in the code–behind model. If you want to include client–side script on the page, script blocks may not have the runat="server" attribute. The @ Page directive in the code–behind model has properties that refer to a class and an external file (Default.aspx.vb or Default.aspx.cs), which is the second difference. The.aspx page and its code are connected by these properties.

A separate file contains the code. The code snippet that follows demonstrates a code–behind file with the identical Click event handler as the Default page example.

The whole class declarations in the default namespace are contained in the code–behind file. However, the class is marked with the partial keyword in its declaration, indicating that it is not wholly included in a single file. Instead, the.aspx page and the file it references in the @ Page directive are read by the compiler when the page is running, assembled into a single class, and then compiled into a single class.

The page class is an ancestor of the partial class file.

Selection of a Page Model

The single–file and code–behind page models are equivalent in terms of functionality. There is no performance difference between the models because they both operate in the same way during run time. So picking a page model is dependent on other things like how you want to structure the code in your application, whether it's crucial to keep page design and coding distinct, etc.

The benefits of single–file pages

The single–file architecture is typically appropriate for pages where the majority of the code consists of event handlers for the controls on the page. The following are some advantages of the single–file page model:

- The convenience of retaining the code and mark–up in the same file can outweigh other benefits of the code–behind architecture on pages with little to no code. For instance, because the code and the mark–up are all visible on one page, studying a single–file page may be simpler.
- Because there is only one file, pages created using the single–file format are a little bit simpler to deploy or transfer to another programmer.
- A single–file page is simpler to rename because the files are not dependent on one another.
- Because the page is self–contained in a single file, managing files in a source code control system is a little bit simpler.

❑ **Check Your Progress – 4 :**

1. _____ term refers to a master page that specifies a web application's page layout.

   a. .dll b. .mdf c. .master d. .asax

2. _____ stands for an ASP.NET Web form. It could be in a subdirectory or the application root directory.

   a. .cs b. .aspx c. .mdb d. .dll

3. _____ refers to the Global file, which contains code from the HttpApplication class. It is located in the Application Root Directory.

   a. .asx b. .aspx c. .asd d. .asax

The benefits of code–behind pages

Code–behind pages provide benefits that make them appropriate for Web applications with a lot of code or situations where a Web site is being developed by a team of developers. The following benefits of the code–behind model:

- The HTML (user interface) and code are neatly separated by code–behind pages. It makes sense to have a designer work on the mark–up as a programmer creates code.
- Page designers and those who simply work with the page mark–up are not exposed to the code.
- Code can be applied to numerous pages.

1.10 Directives :

A custom control can be registered and a page language can be specified using ASP.NET directives. These parameters specify how the.Net framework processes web forms (.aspx) or user controls (.ascx) pages.

The syntax for a directive declaration is:

<%@ directive_name attribute=value [attribute=value] %>

We will only introduce the ASP.NET directives in this part. These directives are as follows :

- ***Application Directive :*** Application–specific properties are defined using the Application directive. It is accessible through the global.aspx file's header. The fundamental Application directive syntax is :

  <%@ Application Language="C#" %>

  The characteristics of Application Directives are as follows :

  - **Inherits :** To inherit we need to define the name of the class.
  - **Description :** The application description in the text format.
  - **Language :** Block codes use the language.

- ***Assembly Directive :*** At parsing time, the Assembly directive connects an assembly to the page or application. The global.asax file for application–wide linking, the page file, or a user control file for linking to a page or user control are all possible places for this to exist. The fundamental Assembly directive syntax is as follows :

  <%@ Assembly Name ="myassembly" %>

The characteristics of Assembly Directives are as follows :

- **Name :** The name of the assembly to link.
- **Src :** The location of the source file that will be dynamically linked and compiled.

- ***Control Directive :*** The control directive can be found in the user control (.ascx) files and is used with user controls. The fundamental Control directive syntax is as follows :

  <%@ Control Language="C#" EnableViewState="false" %>

  The characteristics of Control Directives are as follows :

  - **AutoEventWireup :** The Boolean value that controls whether events are automatically associated with handlers.
  - **ClassName :** The name of the file for the control.
  - **Debug :** The Boolean indicator of whether or not debug symbols should be compiled using.
  - **Description :** The compiler ignores the control page's text description.
  - **EnableViewState :** Whether view state is preserved between control requests. If view state is preserved, true; otherwise, false. default is true
  - **Explicit :** Checks whether the control was built using Visual Basic Option Explicit. true means the Visual Basic explicit compilation option is enabled and all variables must be defined using a Dim, Private, Public, or ReDim declaration. Default value is false.
  - **Inherits :** Specifies a code–behind class for the control. It's any UserControl–derived class. Used with CodeFile, this contains the class's source file location.
  - **Language :** Sets the language used to compile inline rendering (% %> and %= %>) and code declaration blocks in the control. Visual Basic, C#, and JScript are all.NET Framework–supported languages. Each control only supports one language.
  - **Src :** Path to a source file containing linked control code. In the connected source file, you may add control logic in a class or code declaration blocks. Src links build providers to a control.
  - **Strict :** Indicates that the control should be built using Visual Basic Strict. If Option Strict is on, true; otherwise, false. Assume default value as false.

- ***Implements Directive :*** If you see the Implement directive on a web page, it means that either the master page or the user control page needs to implement the specified. Interface for the Net framework.The following is the fundamental syntax for the implements directive:

  <%@ Implements Interface="interface_name" %>

- ***Import Directive :*** An application's user control panel or web page may be made to include an imported namespace by using the Import directive. When the Import directive is defined in the global.asax file, it causes that directive to be used across the whole application. If it is found in a user control page or a page, then it will be applied to the control or

page it is found in. The following is the fundamental syntax for the import directive :

<%@ namespace="System.Drawing" %>

- ***Master Directive :*** The Master directive allows a page file to be designated as the master page for the database. The following is the fundamental syntax of an example MasterPage directive :

  <%@ MasterPage Language="C#" AutoEventWireup="true"

  CodeFile="SiteMater.master.cs" Inherits="SiteMaster" %>

- ***MasterType Directive :*** In order to ensure that a page is strongly typed, the Master attribute is given a class name via the use of the MasterType directive. The following is the fundamental syntax of the MasterType directive :

  <%@ MasterType attribute="value"[attribute="value" ...] %>

- ***OutputCache Directive :*** A web page or a user control's output caching policies may be managed with the help of the OutputCache directive. The following is the fundamental syntax of the OutputCache directive:

  <%@ OutputCache Duration="15" VaryByParam="None"%>

- ***Page Directive :*** The Page directive is responsible for defining the properties that are unique to the page file, which are then used by the page parser and the compiler. The following is the fundamental syntax of the Page directive :

  <%@ Page Language="C#" AutoEventWireup="true"

  CodeFile="Default.aspx.cs" Inherits="_Default" Trace="true" %>

  The characteristics of Page Directives are as follows :

  - **AutoEventWireup :** The value of the Boolean that determines whether or not page events that are automatically being tied to methods, such as Page Load, are enabled or disabled.
  - **Buffer :** The value of the Boolean that determines whether HTTP response buffering is enabled or disabled.
  - **ClassName :** The name of the class that represents this page.
  - **CodeFile :** The name of the file that contains the code underneath.
  - **Debug :** The value of the Boolean that determines whether or not compilation occurs using debug symbols.
  - **Description :** The written description of the page, which the parser does not take into account.
  - **EnableSessionState :** It either activates or deactivates the session state, or it makes it read–only.
  - **EnableViewState :** The value of the Boolean flag that determines whether or not view state is preserved between page requests.
  - **ErrorPage :** URL to use for page redirection in the event that an unhandled page error is encountered.
  - **Inherits :** The name of the class that is used behind the scenes or another class.
  - **Language :** The programming language that the code is written in.

- o **Src :** The name of the file that contains the class's code.
- o **Trace :** It determines whether or not tracing is enabled.
- o **TraceMode :** It provides information about how trace messages are presented and arranged according to time or category.
- o **Transaction :** It shows whether or not transaction support is available.
- o **ValidateRequest :** The value of the Boolean flag that indicates whether or not a predetermined set of values is used to verify all of the data that is being entered.

- ***PreviousPageType Directive :*** A class is given to a page with the PreviousPageType directive in order to ensure that the page is firmly typed. The following is an example of the fundamental syntax for a PreviousPageType directive :

  <%@ PreviousPageType attribute="value"[attribute="value" ...] %>

- ***Reference Directive :*** A web page or a user control's output caching policies may be managed with the help of the OutputCache directive. The following is the fundamental syntax of the OutputCache directive :

  <%@ Reference Page ="somepage.aspx" %>

- ***Register Directive :*** It is necessary to utilise the Register derivative in order to register both the user controls and the custom server controls. The following is the fundamental syntax of the Register directive :

  <%@ Register Src="~/footer.ascx" TagName="footer"

  TagPrefix="Tfooter" %>

1.11 Code Sharing Implementation :

We are presently capable, with.NET Core, of developing apps for three distinct.NET frameworks that are compatible with diverse platforms. Windows uses the classic or standard version of the.NET Framework, whereas iOS, OS X, and Android use the Mono framework..NET Core was developed for Windows, Mac OS X, and Linux.

.NET Frameworks

Different framework class libraries are used by the .Net frameworks. It indicates that code created in one framework is incompatible with code produced in other frameworks. For instance, a console application that was created with the.NET Framework cannot be launched on the.NET Core platform, and vice versa. Sharing of source codes is thus not permitted. It would be convenient to just have to develop code once and then be able to reuse it across several projects using variety of.NET frameworks.

Code Sharing

We have three options at our disposal for addressing the issue of code sharing, and they are as follows :

- Make a Moveable Copy of the Class Library
- Target Multiple Frameworks app written on ASP.NET Core
- Target .NET Standard

When we make a modification to the targets, the application programming interfaces (APIs) that are accessible to us for the development of our project will also be modified to correspond with our choice. Visual Studio notifies the user of any issues or warnings that may develop as a consequence of the targets being modified.

1.12 Dynamic Compilation :

In order for our Web application to process requests, ASP.NET must first parse and compile the code of our Web application into one or more assemblies. Only then will our Web application be able to process requests. When the programme is compiled, the source code is converted into a representation known as Microsoft Intermediate Language, which is independent of both the programming language and the CPU (MSIL). At run time, MSIL is executed inside the context of the.NET Framework. The.NET Framework then translates MSIL into instructions that are particular to the processor of the machine on which the programme is being run.

With the help of ASP.NET dynamic compilation, we are able to make changes to our source code without being required to compile it in a traditional sense before deploying our Web application. If we make changes to a source file, ASP.NET will immediately recompile the file and update any resources that are connected to it. The IIS server does not need to be restarted in order for the modifications to take effect, provided that the 'process Model' section has not been modified in any way.

1.13 Let Us Sum Up :

ASP.NET is a web development platform that provides the software architecture and services needed to construct sophisticated online applications for computers and mobile devices. ASP.NET uses Request.Browser ASP.NET controls support several browsers based on the user–agent information.

ASP.NET web pages are pure text, like HTML files, and serve as application building blocks. An event is a system–generated notice that an application can respond to and control.

In this course, we learned about ASP.NET Framework, ASP.NET features, Class Libraries, CLR functionality, and other components and assemblies required for developing ASP.NET applications.

1.14 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : c **2 :** b

- **Check Your Progress 2 :**

  1 : d **2 :** b

- **Check Your Progress 3 :**

  1 : a **2 :** d

- **Check Your Progress 4 :**

  1 : c **2 :** b **3 :** d

1.15 Glossary :

1. **Framework :** It is the the basic structure of something that gives it shape and strength.
2. **Browser :** A browser is an application program that provides a way to look at and interact with all the information on the World Wide Web. This includes Web pages, videos and images.
3. **CLR :** Common Language Runtime. It manages the execution of programs written in any of several supported languages, allowing them to share common object–oriented classes written in any of the languages.
4. **Compiler :** It is a program that converts instructions into a machine–code or lower–level form so that they can be read and executed by a computer.
5. **HTML :** HyperText MarkUp Languageis a system used to mark text for World Wide Web pages in order to obtain colours, style, pictures, etc.)
6. **HTTP :** The Hypertext Transfer Protocol is an application protocol for distributed, collaborative, hypermedia information systems that allows users to communicate data on the World Wide Web.

1.16 Assignment :

1. Explain the execution process of CLR.

1.17 Activities :

1. Explain various directives in detail.

1.18 Case Study :

Study about the various features of ASP.NET in detail.

1.19 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

Unit
02

STUDYING BUILDING BLOCKS

UNIT STRUCTURE

2.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About the structure of the application
- Various States and variables
- Types of Operators
- Data Types
- Conditions and Loops
- Subroutines and Functions
- The creation of web pages and it's testing
- Objects
- Creation of Objects using parameters
- Properties of Objects
- Methods
- Event Handlers

2.1 Introduction :

An ASP.NET Web Application may contain.asmx files that implement web services, and a web service application may contain.aspx files that implement user interfaces for web services or functionality contained in.NET assemblies. Although it is convenient to divide ASP.NET applications into web applications and web services for the purpose of discussing application types, the reality is that from a practical standpoint, an ASP.NET application may be composed of both types. Therefore, ASP.NET Web Applications and ASP.NET Web Services are relatively comparable to one another in terms of the structure of the programme.

2.2 Application Structure :

An ASP.NET application may be broken down into its component parts, which include an IIS site or virtual directory, as well as at least one ASP.NET page or web service. Each ASP.NET application may, at its discretion, have the following :

- A single instance of the global.asax file, found in the application's root directory.
- One or more files with the extension web.config. Only one web.config file is permitted to exist inside of an application's directory or subfolder at any one time.
- There may be more than one User Control file, denoted by the extension.ascx.
- One or more class files, either for the code–behinds of your ASP.NET application or for the assemblies that are utilised in your programme.
- A directory in the /bin folder that contains the.NET assemblies that you want to make use of in your application. Your programme will have access to the assembled files that are located in the /bin directory automatically.
- The Solution and Project–related files (such as.sln,.suo,.vbproj, and.csproj, for example) are included in ASP.NET Web Applications that are developed using Visual Studio.NET. ASP.NET Web Applications may also include an optional default cascading style sheets file (.css). These programmes could furthermore, but not always, include resource files (.resx), datasets, and/or XML schema definitions (.xsd), along with other sorts of file formats.
- Every other kind of file (.htm,.asp, pictures, and so on) that you would normally anticipate to see in a traditional online application. It is important to keep in mind, however, that ASP pages included inside an ASP.NET application will not share an Application or Session state with the ASP.NET application itself.

2.3 States and Variables :

- ASP.NET Managing States
- Variables

ASP.NET Managing States : The Hyper Text Transfer Protocol, often known as HTTP, is what's known as a stateless protocol. The page objects are deleted from memory by the ASP.NET engine once the client is disconnected from the server. In this manner, any web application may scale up to handle

a greater number of requests concurrently without risking exhausting the available server RAM.

However, there should be a method that can be used to store the information in between requests and then get it when it is necessary to do so. The term "State" refers to the information that describes the current values of all of the controls and variables for the currently logged in user during the currently active session. ASP.NET is capable of managing the following sorts of states:

- View State
- Control State
- Session State
- Application State

View State : The state of the page and all of its controls is referred to as the view state. The ASP.NET framework will ensure that it is kept up to date across all postings automatically.

When a page is returned back to a client, the changes in the properties of the page and the controls on the page are determined, and the value of a hidden input field called _VIEWSTATE is where these changes are saved. The _VIEWSTATE parameter is included in the HTTP request that is sent to the server every time the page is posted back again. For the following, the view state might be enabled or disabled:

- By changing the value of the EnableViewState property in the pages section of the web.config file, we may affect the whole programme.
- We may disable view state on a page by using the Page directive's EnableViewState property and setting it to "false." For example, %@ Page Language="C#" EnableViewState="false"%>
- A control that is activated by configuring the Control. Property known as EnableViewState.

The storing view state notion will be shown with the help of the following example. Let us maintain a counter that, when the page is posted back by clicking a button on the page, will cause an increase in the counter's value. The current value of the counter is shown on a label control.

Control State : It is not possible to change the control status, get direct access to it, or disabled.

Session State : Each time a user establishes a connection to an ASP.NET–powered website, a brand–new session object is generated. When the session state feature is on, a new object representing the session state is generated for each new request. This item of the session state is included into the context, and access to it may be gained via the page.

In most cases, the data of an application, such as its inventory, supplier list, customer record, or shopping cart, is saved in what is known as the session state. In addition to this, it is able to save information on the user and his preferences, as well as maintain a record of pending operations.

The HttpSessionState class, which describes a collection of session state elements, is what's used to generate the object that represents the session state. The following attributes/properties are available for use with the HttpSessionState class :

Properties of HttpSessionState Class

Attributes/Properties	Description
SessionID	The unique session identification.
Item(name)	The value of the session state item that has the given name as its identifier. This is the property that the HttpSessionState class uses as its default setting.
Count	The number of things that are part of the collection held by the session state.
TimeOut	This method both retrieves and sets the maximum amount of time, in minutes, that may pass between requests before the session–state provider shuts off the connection.

These are the methods that are available for the HttpSessionState class :

Methods of HttpSessionState Class

Methods	Description
Add(name, value)	This function adds an item to the collection that stores the session state.
Clear	Eliminates each and every item from the session state collection.
Remove(name)	Deletes the item from the session state collection that has been given as an argument.
RemoveAll	Deletes each and every key and value from the collection containing the session state.
RemoveAt	Deletes an item from the session–state collection that is located at the provided index.

Application State : An application written in ASP.NET is a compilation of all of the web pages, code, and other files that are stored in a single virtual directory on a web server. When information is saved in the application's state, which information is made accessible to all of the users.

ASP.NET generates and saves in the server's memory an instance of the HTTPApplicationState class for each application. This instance of the class is used to produce the application state object that allows for the usage of application state. The class file known as global.asax is the one that represents this object. The following properties are available for use with the HttpApplicationState class :

Properties of HttpApplicationState Class

Properties	Description
Item(name)	The value of the application state item that corresponds to the name that was supplied. This is the property that the HttpApplicationState class uses as its default setting.
Count	The number of different elements that are included in the application state collection.

These are the methods that are available for the HttpApplicationState class :

Methods of HttpApplicationState Class

Methods	Description
Add(name, value)	Inserts an item into the collection that stores the application's state.
Clear	Performs a complete purge of the application state collection.
Remove(name)	Deletes the item from the application state collection that has been given as an argument.
RemoveAll	Deletes each and every item included inside a HttpApplicationState collection.
RemoveAt	Using the specified index, removes a HttpApplicationState object from the collection.
Lock()	Locks the application state collection, limiting access to it to just the currently logged in user.
Unlock()	This function unlocks the application state collection and grants access to it to all users.

Variables : The concept of a variable in C# is equivalent to that of a variable in mathematics. First, let's have a grasp on what an expression is before we try to comprehend what a variable is. The phrases that are listed below are some instances. Example: Expressions 20 + 20, 5 * 3 and 20/2. The answers to the formulas given above are always the same : 20 + 20 = 40, 5 * 3 = 15 and 20/2 = 10. Take into consideration the following idioms and phrases.

Syntax: <data type> <variable name> = <value>;

The following both declares and initializes an integer–type variable that will be used later.

Example : C# Variable int num = 50;

In the previous example, int is a data type, and num is the name of a variable (identifier). When assigning a value to a variable, the = operator is the symbol that is utilised. A value that will be allocated to the variable on the left side of the = operator is included on the right side of the operator. In the previous example, the value 50 is stored in the variable num. Variables of various data types are declared and given their initial values in the following:

```
public static void Main(string[] args)
{
int num = 100;
float rate = 10.2f;
decimal amount = 100.50M;
char code = 'C';
bool isValid = true;
string name = "Steve";
Console.WriteLine(num);
```

```
Console.WriteLine(rate);
Console.WriteLine(amount);
Console.WriteLine(code);
Console.WriteLine(isValid);
Console.WriteLine(name);
}
```

The following is a list of naming conventions for variables that may be used when declaring them in C# :

- Variable names must be unique.
- Only letters, numbers, and the underscore character (_) are permitted in variable names.
- The first letter of a variable's name must always be capitalised.
- Because to the case sensitivity of variable names, num and Num are treated as two separate names.
- It is not possible for variable names to include any reserved keywords. If you wish to reserve keywords as identifiers, you have to prefix them with the @ symbol beforehand.

C# is known for being a tightly typed programming language. It indicates that you are able to assign a value of the data type that was given. You cannot assign a value of type string to a variable of type integer, and vice versa.

Example : Cannot assign string to int type variable i.e. int num = "Steve";

Declaring variables may come first, with initialization coming afterwards.

Example : Late Initialization i.e. int num; num = 100;

If a value is not given to a variable before it is used in C# code, the programming language will generate a compile–time error. Error: Invalid Assignment

Example : int i; int j = i; //compile–time error: Use of unassigned local variable 'i'

After being initialized, the value of a variable may have its value altered at any moment.

Example : C# Variable int num = 100; num = 200; Console.WriteLine (num); //output: 200

Declaring and initializing several variables that have the same data type may take place on a single line if the variables are separated by commas.

Example : Multiple Variables in a Single Line

```
public static void Main()
{
int i = 0, j = 10, k = 100;
Console.WriteLine (k);
} }
```

Expressions may make use of variables, and the results of those expressions can be assigned to the same variable or to a different one. Variables can also be given the result of the expression itself.

Example: Variable & Expression

```
{
int i = 100;
int j = i + 20;
Console.WriteLine("j = {0}", j);
i = 200;
j = i + 20;
Console.WriteLine("j = {0}", j);
i = 300;
Console.WriteLine("j = {0}", j);
}
```

In the previous example, the value of j is determined by the value of i. You will need to re–execute the expression each time you make a change to the value of I because, if you don't, the value of j will not change dependent on the value of i.

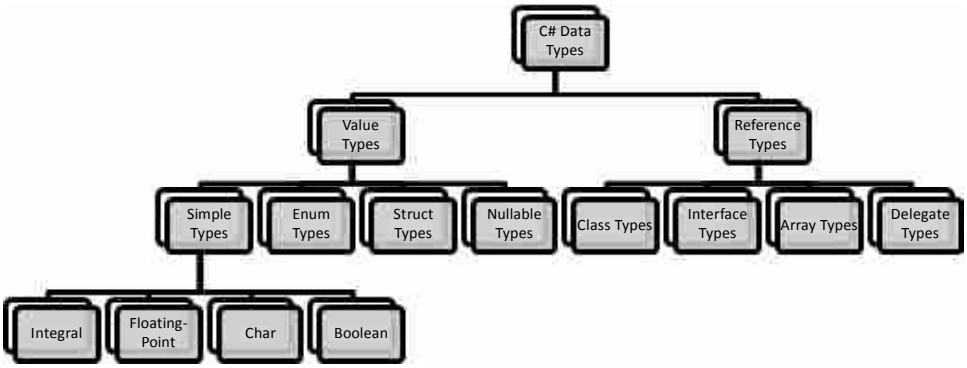
Variables in C# are separated into several categories according to the method that they use to store their values in memory. Variables may be divided into three different types: value type, reference type, and pointer type variables.

2.4 Data Types and Operators :

- Data Types
- Operators

Data Types : The programming language C# has a high level of typing. It implies that we have to specify the type of a variable, which determines the sort of data that it will contain, such as an integer, a float, a decimal, text, and so on.

In C#, data types are primarily separated into two categories: value types and reference types. Simple data types (such as int, float, bool, and char), enum data types, struct data types, and Nullable value types are examples of value types. Reference types include of things like array types, delegate types, class types, and interface types.



Data Types in C#

- ***Predefined Data Types in C# :***

C# comes with a number of value types and reference types that are predefined.

❑ **Check Your Progress – 1 :**

1. Which of the following is a valid variable name ?

   a. 1rohan b. struct c. Book:number d. None of these

2. Which of the following is not the valid data type ?

   a. int b. char c. floot d. string

- ***Alias vs .NET Type :*** The names of the preset data types are aliases for the names of their respective.NET types (CLR classes). The table that follows provides a list of aliases for predefined data types together with the.NET class name that corresponds to them.

Table : Alias and .NET Type

Alias	.NET Type	Type
Byte	System.Byte	struct
Sbyte	System.SByte	struct
Int	System.Int32	struct
Uint	System.UInt32	struct
Short	System.Int16	struct
Ushort	System.UInt16	struct
Long	System.Int64	struct
Ulong	System.UInt64	struct
Float	System.Single	struct
Double	System.Double	struct
Char	System.Char	struct
Bool	System.Boolean	struct
Object	System.Object	Class
String	System.String	Class
Decimal	System.Decimal	struct
DateTime	System.DateTime	struct

It indicates that it does not matter whether a variable is defined as an int or an Int32; the results are the same either way.

```
int i = 345;
Int32 i = 345;// same as above
```

- ***Default Values :***

Every data type has a default value. The default value for the Boolean type is false, while the default value for the character type is '0'. To set a default value to a data type, use the default (typename) function.

```
int i = default; // 0
float f = default;// 0
decimal d = default;// 0
bool b = default;// false
char c = default;// '\0'
```

- ***Conversions :***

In C#, the values of some data types are transformed to the values of other data types automatically. This kind of conversion is known as an implicit conversion.

Example : Implicit Conversion

```
int i = 345;
float f = i;
Console.WriteLine(f); //output: 345
```

Due to the fact that the C# programming language already has this conversion procedure preset, the value of the integer variable i has been allocated to the float variable f in the preceding example. The table that follows is a conversion from one implicit data type to another.

Table : Implicit Data Type Conversion

Implicit Conversion From	To
sbyte	short, int, long, float, double, decimal
Byte	short, ushort, int, uint, long, ulong, float, double, decimal
Short	int, long, float, double, or decimal
Ushort	int, uint, long, ulong, float, double, or decimal
Int	long, float, double, or decimal.
Uint	long, ulong, float, double, or decimal
Long	float, double, or decimal
Ulong	float, double, or decimal
Char	ushort, int, uint, long, ulong, float, double, or decimal
Float	double

There is a possibility that accuracy will be lost when moving from int, uint, long, or ulong to float, as well as when moving from long or ulong to double. There was no implicit conversion from any data type to the char type.

However, not all data types are automatically changed to other data types when they are sent via an implicit conversion. For instance, the int data type cannot be automatically transformed to the uint data type. As will be seen in the next example, it has to be mentioned explicitly.

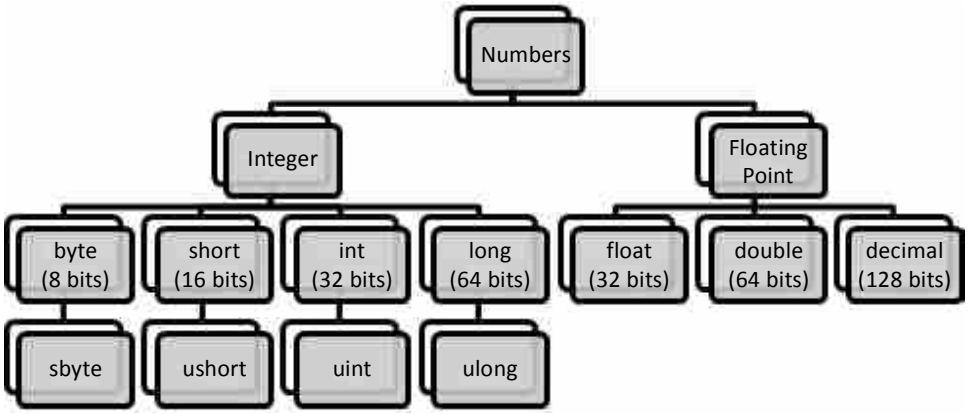
Example : Explicit Conversion

```
public static void Main()
{
int i = 100;
uint u = (uint) i;
Console.Write(i);
}
```

The integer i is transformed to an unsigned integral explicitly in the preceding example by providing uint inside the brackets (uint). An integer will be converted into a uint by doing this.

- ***Numbers in C# :*** In general, numbers may be broken down into two distinct categories: the integer type, and the floating–point type.
  - o Integers of the integer type are entire numbers that do not include any decimal points. It's possible for them to be positive or negative numbers.
  - o Numbers of the floating–point type have one or more decimal places after their value. It's possible for them to be positive or negative numbers.

    C# has a variety of data types for integer types and floating–point types, each of which may hold a varied range of values depending on how much memory space they take up. The following illustration provides an example of a numeric type in C#.



Numeric Data Types

- ***Interger Types :***

Positive or negative whole numbers that do not include decimal points are examples of integer types of numbers. Integer numbers may be represented by one of four different data types in C#: byte, short, int, and long.

- ***Floating Point Types :***

Numbers having one or more decimal places might be positive or negative. These are known as floating–point numbers. float, double, and decimal are the three different data types that may be used for floating–point values in C#. Float numbers may be represented by one of these data types in C#: float, double, decimal, and scientific notation.

❑ **Check Your Progress – 2 :**

1. A _____ data type is a signed integer that can hold integers in the range of –32768 to 32,767 and has a maximum capacity of 32,767.

   a. long b. short c. char d. None of these

2. Which should be used to represent the power of 10 ?

   a. A b. C c. D d. E

- ***Strings in C# :***

A string is a sequence of letters that is used to represent text in the programming language C#. It might be a single letter, a single phrase, or even a lengthy piece that is encircled by the double quotations". The items that follow are known as string literals.

Examples : String Literals

"S"

"String"

"This is a string."

String is a built–in data type in C# that may be used to hold string literals. As will be seen in the following example, a string variable may be defined, and then a string literal can be assigned to it.

Example : String Type Variables

```
{
string ch = "S";
string word = "String";
string text = "This is a string.";
Console.WriteLine(ch);
Console.WriteLine(word);
Console.WriteLine(text);
}
```

In C#, declaring a string variable may be done in one of two methods. Using System. The String class as well as the use of the string keyword both are equivalent; therefore there is no discernible difference between them.

It is a difficult process to prefix "" in order to include all of the special characters. A special character and line breaks may be included in a verbatim string when using C#. The @ symbol may be prefixed in front of double quotation marks to produce a verbatim string.

In C#, a String cannot be changed in any way. It indicates that it can only be read and cannot be altered after it has been generated in the memory. When you concatenate strings,.NET Common Language Runtime will generate a new memory address to store the concatenated text after each operation. If you want to concatenate more than five strings, it is strongly suggested that you make use of the StringBuilder rather than string.

- ***Date and Time in C# :***

The DateTime type is available to users of C# who want to deal with dates and times. Create an object of the DateTime type in C# by using the new keyword. This will allow you to interact with dates and times. The following code will generate a DateTime object with the value that is set as the default.

Example : Create DateTime Object

```
DateTime dt = new DateTime(); // assigns default value 01/01/0001 00:00:00
```

January 1, 0001 00:00:00 is a DateTime object's default value, and it is also its minimum possible value (midnight). The maximum value that may be entered is 11:59:59 p.m. on December 31, 9999.

Example : Set Date & Time

```
{
DateTime dt1 = new DateTime();
Console.WriteLine(dt1);
//assign year, month, day
DateTime dt2 = new DateTime(2015, 12, 31);
Console.WriteLine(dt2);
//assign year, month, day, hour, min, seconds
DateTime dt3 = new DateTime(2015, 12, 31, 5, 10, 20);
Console.WriteLine(dt3);
//assign year, month, day, hour, min, seconds, UTC timezone
DateTime dt4 = new DateTime(2015, 12, 31, 5, 10, 20, DateTimeKind.Utc);
Console.WriteLine(dt4);
}
```

- ***Structure in C# :***

The value type data type that represents data structures in C# is called struct. It is possible for it to have nested types, parameterized constructors, static constructors, constants, fields, methods, properties, indexers, operators, and events.

Declaring a structure requires the use of the struct keyword. The default modifier is considered an internal part of the struct and the components that make up the struct.

Example :

```
struct Coordinate
{
public int x;
public int y;
}
```

In the same way that basic type variables may be generated with or without the new operator, struct objects can also be formed with or without it.

Example :

```
public static void Main()
{
Coordinate point = new Coordinate();
Console.WriteLine(point.x);
Console.WriteLine(point.y);
}
struct Coordinate
{
public int x;
```

public int y;

}

- ***C# Enumerations Type – Enum :***

In C#, an enum (or enumeration type) assigns constant names to integer values. It improves readability of constants like WeekDays. Monday is more readable than 0 as a weekday.

A namespace, class, or structure defines an enum using the enum keyword. All constant names may be expressed within curly brackets and commas. This is called enum for the Weekdays.

For examples : enum defination

enum WeekDays

{Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}

In the previous section, each line contains a declaration for a member of the WeekDays enum. These declarations are separated by commas.

Enumeration members may have a variety of values assigned to them. If you edit the value of the default member of an enum, all of the other members will have their values automatically adjusted to reflect the change in order.

You are even able to set varying values to each individual component. It is possible for the enum to have any of the following numeric data types: byte, sbyte, short, ushort, int, uint, long, or ulong. On the other hand, a string type can never be an enum. After the enum name, you must provide the type using the notation type.

- ***C# – Anonymous Type :***

An anonymous type is a type (class) in C# that does not have a name and may only include attributes that are accessible to the public in read–only mode. Other members, such as fields, methods, events, and so on, cannot be included inside it.

Using the new operator in conjunction with object initializer syntax will result in the creation of an anonymous type. The reference to anonymous types is stored in the variable var, which has an implicit type and is referred to as var. The following illustration explains how to create an anonymous type variable named student, which has three values called Id, FirstName, and LastName.

For example :

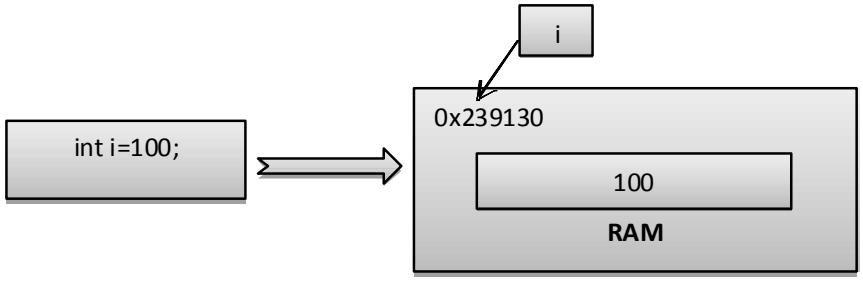
var student = new { Id = 1, FirstName = "James", LastName = "Bond" };

Properties of anonymous types may only be read, and they cannot be initialised using null, anonymous functions, or pointer types. Read–only access is required for anonymous type properties. The dot notation (.) is used in the same way to access the properties as is done with object properties.

- ***C# – Value Types :***

When a data type stores a data value inside its own allocated memory space, we refer to that data type as a value type. This indicates that the variables of these data types contain values in their native form. Every single value type is a descendant of the System type. ValueType is a subclass of System.Object, the parent class of which ValueType is. For example, assume integer variable int i= 100; The value 100 is saved in the portion of memory that has been

designated for the variable i by the system. The figure that follows provides an illustration of how the value 100 is stored at a fictitious position in the memory (0x239130) for the letter i.



Memory Allocation of Value Type Variable

When you transfer a variable of type value from one method to another, the system makes a distinct duplicate of the variable in the second procedure. If the value was modified in one method, it would have no impact on the variable that was used in another method.

For example :

```
static void ChangeValue(int x)
{x = 400;
Console.WriteLine(x);}
public static void Main()
        {int i = 500;
Console.WriteLine(i);
ChangeValue(i);
Console.WriteLine(i);
} }
```

Even when we provide the variable i to the ChangeValue() function and modify its value there, the value of the variable i in the Main() method does not update. This is shown by the above example.

- ***C# – Reference Type :***

A reference type, as contrast to a value type, does not store its value in an unmediated manner. Instead of storing the item itself, it saves the address of the location where the data is being saved. In other words, a reference type stores a pointer to another memory address somewhere on the computer system that is responsible for holding the data.

For example : string s = "Hello Students!!";

The variable s is stored in memory at a place that is chosen at random by the system (0x192334. The memory address of the real data value is 0x300000, which is the value of the variable s, which is denoted by the letters. Therefore, rather of storing the item itself, a reference type will instead save the address of the place where the data is really stored.

When you send a variable of the reference type from one method to another, it does not make a new copy of the variable but rather provides the address of the variable. Consequently, if we alter the value of a variable inside a method, the effect will also be reflected within the method that the variable was called from.

The String reference type cannot be changed since it is immutable. It implies that once a value has been allocated, we cannot alter it in any way. When we modify the value of a string, the compiler produces a new string object in memory and points a variable to the new memory address. This happens whenever we update the value of a string. Therefore, when you send a string value to a function, a new variable will be created in the memory.

Operators : In C#, an operator is a particular symbol that, when applied to operands, produces a certain result. The following types of operators are included in the C# programming language:

- ***Arithmetic Operators :***

The arithmetic operators are responsible for carrying out arithmetic operations on all of the different types of numeric operands, including sbyte, byte, short, ushort, int, uint, long, ulong, float, double, and decimal.

Arithmetic Operators

Operator	Name	Description	Example
+	Addition	Adds left and right operands.	int x = 10+2;
–	Subtraction	Right operand minus left operand.	int x = 15–11;
*	Multiplication	Left and right operand multiplication	int x = 5 * 4;
/	Division	Divides left by right	int x = 20 / 2;
%	Remainder	Divides left operand by right operand and returns residual.	int x = 7 % 2;
++	Unary Increment	++ operator unary increments operand by 1	x++
--	Unary Decrement	Decrement operator -- reduces operand by 1	x--
+	Unary Plus	Returns operand's value	+8
–	Unary Minus	Negates its input.	-8

Let us see few examples of each arithmetic operator.

Addition (+) Operator :

```
{
int x = 5 + 5;
string greet = "Hello" + " Student";
Console.WriteLine("x = 5 + 5 = {0}", x);
Console.WriteLine("greet = {0}", greet);
}
```

Subtraction (–) Operator :

```
{
int x = 5 – 1;
int y = x – 2;
int z = x – y;
```

```
float f1 = 6.5f – 5.25f;
Console.WriteLine("x = 5 – 1 = {0}", x);
Console.WriteLine("y = x – 2 = {0}", y);
Console.WriteLine("z = x – y = {0}", z);
Console.WriteLine("f = 6.5 – 5.25 = {0}", f1);
}
```

Multiplication (*) Operator :

```
{
int x = 5 * 1;
int y = x * 2;
int z = x * y;
float f = 6.5f * 5.25f;
Console.WriteLine("x = 5 * 1 = {0}", x);
Console.WriteLine("y = x * 2 = {0}", y);
Console.WriteLine("z = x * y = {0}", z);
Console.WriteLine("f = 6.5 * 5.25 = {0}", f);
}
```

Division (/) Operator :

```
{
int x = 10 / 2;
int y = x / 5;
int z = x * y;
float f = 6.5f / 5.25f;
Console.WriteLine("x = 10/2 = {0}", x);
Console.WriteLine("y = x/5 = {0}", y);
Console.WriteLine("z = x/y = {0}", z);
Console.WriteLine("f = 6.5 / 5.25 = {0}", f);
}
```

Remainder (%) Operator :

```
{
int x = 5 % 2;
int y = x % 2;
int z = x % y;
float f = 6.5f % 5.25f;
Console.WriteLine("x = 5 % 2 = {0}", x);
Console.WriteLine("y = x/% 25 = {0}", y);
Console.WriteLine("z = x % y = {0}", z);
Console.WriteLine("f = 6.5 % 5.25 = {0}", f);
}
```

Increment (++) Operator :

```
{
int x = 5;
int y = 6;
float f = 6.5f;
Console.WriteLine("x++ = {0}", x++);
Console.WriteLine("x = {0} after x++", x);
Console.WriteLine("++y = {0}", ++y);
Console.WriteLine("f++ = {0}", f++);
Console.WriteLine("f = {0} after f++", f);
}
}
```

Decrement (--) Operator :

```
{
int x = 5;
float f = 6.5f;
Console.WriteLine("x-- = {0}", x--);
Console.WriteLine("x = {0} after x--", x);
Console.WriteLine("f-- = {0}", f--);
Console.WriteLine("f = {0} after f--", f);
}
```

Unary Plus (+) and Unary Minus (–) Operator :

```
{
int x = 5;
float f = 6.5f;
Console.WriteLine("+x = {0}", +x);
Console.WriteLine("+f = {0}", +f);
Console.WriteLine("–x = {0}", –x);
Console.WriteLine("–x + 3 = {0}", –x + 3);
Console.WriteLine("–f = {0}", –f);
}
```

- ***Assignment Operators :***

When the assignment operator = is used, the value in the right–hand variable is transferred to the variable, property, or indexer in the left–hand position. In addition to that, you may combine it with other arithmetic operators, Boolean operators, and bitwise operators.

Assignment Operators

Operator	Name	Description	Example
=	Assignment	Assigns the value in its right-hand variable, property, or indexer to the value in its left-hand variable, property, or indexer.	x = 10;
x op= y	Compound Assignment	A short form of the expression x = x op y, where op may stand for any arithmetic, Boolean, or bitwise operator.	x += 5;
??=	Null–coalescing assignment	Starting with version C# 8, the??= operator will only assign the value of the right operand if the left operand is null.	x ??= 5;

Let us see few examples of each assignment operator.

Assignment (=) Operator :

```
{
int x,y;
Console.WriteLine("x = 5 = {0}", x = 5);
Console.WriteLine("y = (x = 5) = {0}", y = (x = 5));
}
```

- ***Comparison Operators :***

Comparison operators take in two numeric operands and return either true or false after performing a comparison between them.

Comparison Operators

Operator	Name	Description	Example
<	Less Than	If the right operand is smaller than the left operand, then this function will return true.	x<y;
>	Greater Than	If the right operand is bigger than the left operand, then this function will return true.	x>y;
<=	Less Than Equal To	If the right operand is smaller than or equal to the left operand, then this function will return true.	x<=y;
>=	Greater Than Equal To	If the right operand is more than or equal to the left operand, then this function will return true.	x>=y;

Let us see few examples of each comparison operator.

Less Than (<) Operator :

```
{
int x = 5, y = 10;
float f = 5.3f;
Console.WriteLine(3 < 2);
```

```
Console.WriteLine(x < y);
Console.WriteLine(x < f);
}
```

Greater Than (>) Operator :

```
{
int x = 5, y = 10;
float f = 5.3f;
Console.WriteLine(3 > 2);
Console.WriteLine(x > y);
Console.WriteLine(x > f);
}
```

Less Than Equal To (<=) Operator :

```
{
int x = 5, y = 10;
float f = 5.3f;
Console.WriteLine(3 <= 2);
Console.WriteLine(x <= y);
Console.WriteLine(x <= f);
}
```

Greater Than Equal To (>=) Operator :

```
{
int x = 5, y = 10;
float f = 5.3f;
Console.WriteLine(3 >= 2);
Console.WriteLine(x >= 5);
Console.WriteLine(x >= y);
Console.WriteLine(x >= f);
}
```

- ***Ternary Operators :***

The operator that makes decisions in C# is denoted by the question mark (?), also known as the conditional operator or ternary operator. It is a condensed version of if and else conditions.

Syntax : condition ? statement 1 : statement 2

A boolean condition serves as the foundation for the ternary operator. If the condition is evaluated as true, then the first statement that follows the question mark (?) will be carried out; otherwise, the second statement that follows the colon (:) will be carried out. The ternary operator will be shown using the following **example**.

```
{
int x = 20, y = 10;
var result = x > y ? "x is greater than y" : "x is less than or equal to y";
```

```
Console.WriteLine(result);
}
```

In the previous example, the result of the conditional expression x > y was true, which means that the first line after the question mark (?) will be executed.

As a result, a ternary operator is a condensed version of the sentence "if else." The above example might be rewritten using if else condition, as demonstrated in the following example:

```
{
int x = 20, y = 10;
if (x > y)
Console.WriteLine("x is greater than y");
else
Console.WriteLine("x is less than y");
}
```

Right associative behaviour may be expected from the ternary operator. It is important to note that the phrase a? b: c? d: e is interpreted and evaluated as a? b: (c? d: e), and not as (a? b: c)? d: e.

```
{
int x = 5, y = 10, z = 15;
int result = x * 3 > y ? x : y > z? y : z;
Console.WriteLine(result);
}
```

- ***Equality Operators :***

The equality operator determines whether or not the two operands are the same by comparing them to each other.

Equality Operators

Operator	Name	Description	Example
==	Equal To	If the operands are the same, this function returns true; otherwise, it returns false.	x==y;
!=	Not Equal To	If the operands are not equal, this function returns true; otherwise, it returns false.	x!=y;

Let us see few examples of each equality operator.

Equal To (==) Operator :

```
{
int x = 5, y = 5;
float f = 5.3f;
Console.WriteLine(3 == 2);
Console.WriteLine(x == y);
Console.WriteLine(x == f);
}
```

Not Equal To (!=) Operator :

```
{
int x = 5, y = 5;
float f = 5.3f;
Console.WriteLine(3 != 2);
Console.WriteLine(x != y);
Console.WriteLine(x != f);
}
```

- ***Boolean Logical Operators :***

On bool operands, the Boolean logical operators will execute the appropriate logical operation.

Boolean Logical Operators

Operator	Name	Description	Example
!	NOT	Turns the outcome of a boolean expression into a false value. If result is true, this function will return false, and if result is false, it will return true.	!false
&&	AND	Performs a computation that is equivalent to the logical AND of its bool operands. When both operands are true, this function returns true; otherwise, it returns false.	x && y;
||	OR	Performs a computation that is the logical OR of the bool operands it has. This function will return true if any of the operands are true.	x || y;

Let us see few examples of each Boolean logical operator.

NOT (!) Operator :

```
{
bool bvar = false;
Console.WriteLine(!false);
Console.WriteLine(!true);
Console.WriteLine(!bvar);
Console.WriteLine(!(5 > 3));
}
```

AND (&&) Operator :

```
{
bool bvar1 = false, bvar2 = false;
Console.WriteLine(true && true);
Console.WriteLine(true && false);
Console.WriteLine(bvar1 && bvar2);
Console.WriteLine(!bvar1 && !bvar2);
Console.WriteLine((5 > 2) && (10 < 5));
}
```

OR (||) Operator :

```
{
bool bvar1 = false, bvar2 = false;
Console.WriteLine(true || true);
Console.WriteLine(true || false);
Console.WriteLine(bvar1 || bvar2);
Console.WriteLine(!bvar1 || !bvar2);
Console.WriteLine((5 > 2) || (10 < 5));
}
```

- ***Bitwise and Shift Operators :***

The following operators perform bitwise or shift operations with operands that are integral numeric types or are of the character type :

- Unary ~ (bitwise complement) operator
- Binary << (left shift), >> (right shift), and >>> (unsigned right shift) operators
- Binary & (logical AND), | (logical OR), and ^ (logical exclusive OR) operators

- ***Type–Cast Operators :***

In order to perform type checking or type conversion, you can make use of the following operators and expressions :

- is Operator
- as Operator
- cast Expression
- typeof Operator

is Operator : The is operator determines whether or not the type returned by an expression at runtime is compatible with the type that is specified. The form of the expression that includes the is operator for type testing looks like this: E is T, where E is a value–returning expression and T is either the name of a type or a type parameter.

as Operator : The outcome of an expression may be explicitly converted by using the as operator to a specific reference or nullable value type. The as operator will return null if the conversion cannot be completed successfully. For example: E as T. This yields the same result as the following expressions, where E is an expression that returns a value and T is the name of a type or a type parameter exception made for E, which is only ever assessed once: E is T ? (T)(E) : (T)null

Cast Expression : When you use a cast expression of the form (T)E, you are telling the computer to make an explicit conversion of the value returned by the expression E to the type T. An error will occur during the compilation process if there is no explicit conversion that may take place from the type of E to the type of T. At run time, an explicit conversion may fail, and a cast expression may produce an exception. Both of these scenarios are possible.

typeof Operators : The System.Type object associated with a type may be retrieved using the typeof operator. The typeof operator may be used with

unbound generic types. Commas must be included in the name of an unbound generic type in the proper quantity, which is one less than the total number of type parameters. The following illustrative example demonstrates the use of the typeof operator in conjunction with an unbound generic type:

```
Console.WriteLine(typeof(Dictionary<,>));
```

It is not possible to use an expression as a parameter in the typeof operator in order to get the System.Type instance to determine the type of a runtime result that an expression produces method known as Object.GetType.

- ***Pointer Related Operators :***

When dealing with pointers, you can work with them using the following operators:

- Unary & (address–of) operator
- Unary * (pointer indirection) operator
- The –> (member access) and [] (element access) operators
- Arithmetic operators +, –, ++, and –
- Comparison operators ==, !=, <, >, <=, and >=

Unary & (address–of) operator : The address of the operand may be obtained by using the unary & operator:

```
Unsafe {
int number = 27;
int* pointerToNumber = &number;
Console.WriteLine($"Value of the variable: {number}");
Console.WriteLine($"Address of the variable: {(long)pointerToNumber:X}");
}
```

The "and" operator requires a fixed variable to be used as its operand. Fixed variables are those that are stored in places that are unaffected by the action of the garbage collector. They are also known as persistent variables. Because it is stored on the stack, the local variable number in the example that came before it is what's known as a fixed variable.

Pointer indirection operator * : The unary pointer indirection operator * finds the variable to which its operand points by obtaining the variable's address from the operand. In certain circles, it is also referred to as the dereference operator. The * operator requires that the operand it is working with be of type pointer.

Pointer member access operator –> : Indirection of pointers and access to members are both handled by the –> operator. To put it another way, if x is a pointer of type T* and y is a member of type T who may be accessed, then an expression of the form would be valid. For example: x–>y is comparable to (*x).y.

Pointer element access operator [] : A pointer element access of the form p[n] is evaluated as *(p + n), where n must be of a type that is implicitly convertible to int, uint, long, or ulong. This evaluation occurs for an expression p that is of a type that is a pointer. A stackalloc expression is responsible for allocating a portion of memory to be used on the stack. When working with an expression of type void*, you are unable to utilise [] for pointer element

access. Access to array elements or indexers may also be accomplished via the use of the [] operator.

Pointer arithmetic operators : The following arithmetic operations are all ones that may be performed with pointers:

- To or from a pointer, you may either add or remove an integral value.
- Reduce the score by two points.
- Increment or decrement a pointer

Pointer subtraction : The formula p1 – p2 provides the difference between the addresses supplied by p1 and p2 divided by sizeof when applied to two pointers of type T* named p1 and p2, respectively (T). The format of the result is of the long variety. In other words, the difference between p1 and p2 may be calculated as ((long)(p1) – (long)(p2)) / sizeof (T).

Pointer increment and decrement : The ++ increment operator will add one to the value of the pointer that it is operating on. The value 1 is subtracted from the pointer operand when the – decrement operator is used.

Both of these operators may be used in their postfix and prefix forms, which are denoted by the notation p++ and p– respectively. The value of p before the operation is determined by the combination of p++ and p–. The value of p after the operation is what is referred to as the outcome of the ++p and –p procedure.

Pointer comparison operators : Comparing operands of different pointer types, including void*, may be accomplished with the help of the ==, !=, >, =, and >= operators. These operators treat the addresses that are provided by the two operands as if they were unsigned integers and compare the results.

❑ **Check Your Progress – 3 :**

1. What is the other way to write a=a–b ?

   a. a=–b b. a–=b c. a–b d. None of these

2.5 Conditions and Loops :

Conditions : There are a lot of decision–making statements available to use in C#, and they improve the flow of the C# programme by evaluating specific logical criteria and acting accordingly. In this section, you will learn how to manage the flow of the programme depending on the circumstances using if statements, else if statements, and nested if else statements. The following variations of if statements are available in C# :

- if Statement
- if–else Statement
- else Statement
- nested if Statement
- switch Statement

- ***C# if Statement :***

The if statement includes a boolean condition, which is followed by either a single line of code or a block of code consisting of many lines. If a boolean condition evaluates to true at runtime, then the code block will be executed; if not, the code block will not be executed.

Syntax :

```
if(condition)
{ // code block to be executed when if condition evaluates to true}
```

For **example :**

```
int i = 10, j = 20;
if (i > j)
{ Console.WriteLine("i is greater than j"); }
if (i < j)
{ Console.WriteLine("i is less than j"); }
if (i == j)
{ Console.WriteLine("i is equal to j"); }
```

As a result of the Boolean condition in the first if statement I j) evaluating to true in the example that was just shown, the C# compiler will execute the following code block. Because the condition I > j of the second if statement evaluates to false, the compiler will not execute the code block that is associated with it.

If the conditional expression does not produce a Boolean value, the C# compiler will provide an error message throughout the compilation process.

- ***C# if–else Statement :***

After if statement, you may use one or more else if statements if necessary. Only in the event that if condition is found to be untrue will it be carried out. Therefore, either the if statement or one of the otherwise if statements may be carried out, but not both at the same time.

Syntax :

```
if(condition1)
{ // code block to be executed when if condition1 evaluates to true}
else if(condition2)
{ // code block to be executed when condition1 evaluates to flase
condition2 evaluates to true}
else if(condition3)
{ // code block to be executed when condition1 evaluates to flase
condition2 evaluates to false
// condition3 evaluates to true }
```

For **example :**

```
int i = 10, j = 20;
if (i == j)
{ Console.WriteLine("i is equal to j"); }
else if (i > j)
{ Console.WriteLine("i is greater than j"); }
else if (i < j)
{ Console.WriteLine("i is less than j"); }
```

- ***C# else Statement :***

The else statement may appear in an if–otherwise statement only after if or else if statement, and it can only be used once inside an if–else statement. The otherwise statement cannot have any conditions attached to it and will only be carried out if all of the criteria attached to the preceding if and else if statements are found to be false.

Example :

```
int i = 20, j = 20;
if (i > j)
{ Console.WriteLine("i is greater than j"); }
else if (i < j)
{ Console.WriteLine("i is less than j"); }
else
{ Console.WriteLine("i is equal to j"); }
```

- ***C# nested if Statement :***

C# permits the use of if else statements inside the context of other if else statements. These kind of sentences are referred to as nested if else statements. The code is easier to follow as a result of the nested if statements.

Syntax :

```
if(condition1)
{
   // code block to be executed when if condition1 evaluates to true
}
else if(condition2)
{
   // code block to be executed when
   //    condition1 evaluates to flase
   //    condition2 evaluates to true
}
else if(condition3)
{
   // code block to be executed when
   //    condition1 evaluates to false
   //    condition2 evaluates to false
   //    condition3 evaluates to true
}
```

The following illustration provides a demonstration of the nested if then statements.

```
{ int i = 10, j = 20;
if (i != j)
{ if (i < j)
```

```
{ Console.WriteLine("i is less than j"); }
else if (i > j)
{ Console.WriteLine("i is greater than j"); }
}else Console.WriteLine("i is equal to j");}}
```

- ***C# switch Statement :***

When you wish to test a variable against three or more criteria, you may use the switch statement rather than if else statement. This is because the switch statement is more flexible.

Syntax :

```
switch(match expression/variable)
{
case constant–value:
statement(s) to be executed;
break;
default:
statement(s) to be executed;
break;
}
```

The switch statement begins with the switch keyword, which may be followed by a match expression or a variable enclosed in brackets after the switch keyword (match expression). Within the brackets with the curly braces (), the outcome of this match expression or a variable will be compared to the conditions that are stated as cases. A case must be stated with a value that is constant and unique, and the specification must finish with the colon:. Every single instance has one or more assertions that need to be carried out. If the value of a constant and the value of an expression or variable that matches it are the same, then the case statement will be performed. A default label is optional and may be included in the switch statement if desired. If none of the other labels are performed, the default label will be used. To remove control from a switch case after it has been evaluated, you may use the keywords break, return, or goto. An example of simple switch statement is :

```
{ int x = 10;
switch (x)
{
case 5: Console.WriteLine("Value of x is 5");
break;
case 10: Console.WriteLine("Value of x is 10");
break;
case 15: Console.WriteLine("Value of x is 15");
break;
default: Console.WriteLine("Unknown value");
break;
}}
```

In the previous example, there is a statement called switch(x) that contains a variable called x, the value of which will be compared to the value of each case value. The switch statement shown earlier has three scenarios, each of which has a constant value of 5, 10, and 15. Additionally, it includes the default label, which is the one that will be carried out even if none of the case values match with the switch variable or expression. Every instance begins with: and consists of one assertion that has to be carried out. The value of x coincides with the second case scenario, case 10:, which means that the outcome would be 10 is the value of x. The switch statement may also include an expression, the result of which will be compared to each instance when the programme is being executed.

The switch cases must have unique constant values. It may be of type bool, char, string, integer, enum, or the nullable type that corresponds to it.

For example :

```
{
string statementType = "switch";
switch (statementType)
{
case "if.else":
Console.WriteLine("if...else statement");
break;
case "ternary":
Console.WriteLine("Ternary operator");
break;
case "switch":
Console.WriteLine("switch statement");
break;
} }
```

It is necessary for each case to have an explicit exit from the case, which can be accomplished through the use of a break, return, goto statement, or some other method. This will ensure that the programme control leaves the case and does not pass through to the default case. It is possible to nest one switch statement inside of another switch statement.

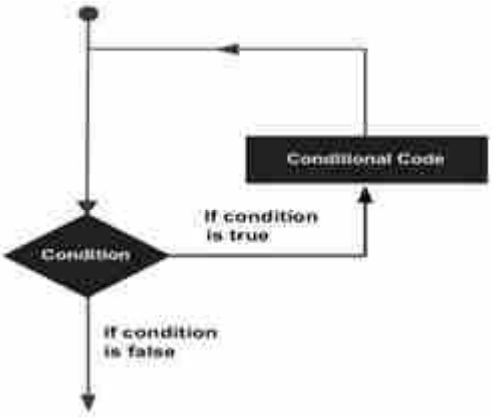
❑ **Check Your Progress – 4 :**

1. When you wish to test a variable against three or more criteria, you may use _____ statement:

   a. If b. Else–if c. Switch d. None of these

Loops : In most cases, the statements are carried out in the order that they are written. For example, the first statement in a function is carried out before moving on to the second, and so on. Programming languages are equipped with a variety of control structures, which in turn make it possible to create more intricate execution routes.

We are able to repeatedly execute a statement or a series of statements if we use something called a loop statement, and the following is the typical form of a loop statement in the majority of programming languages :



Loops

The following forms of loop are available in C# to address various looping needs.

- ***C# for Loop :***

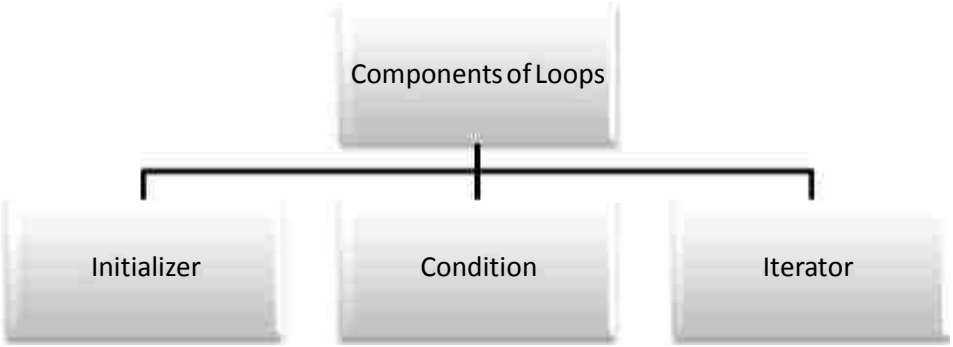
In this section, you will learn how to use for loop to repeat the execution of a statement or code block numerous times, as well as the structure of the for loop, how to create nested for loops, and how to leave the for loop.

In C#, a loop is indicated by the use of for keyword. A group of statements is carried out iteratively by for loop up to the point at which the stated condition yields false.

Syntax :

```
for (initializer; condition; iterator)
{
    //code block
}
```

The following three optional components are included inside for loop and are each separated by a semicolon :

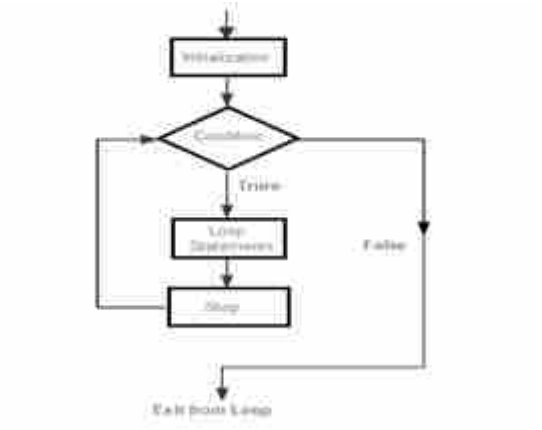


Components of Loops

Initializer : A variable that is going to be local to for loop and that cannot be accessible outside of the loop is initialised using the initializer part of for loop. In addition, it might be zero or more assignment statements, a call to a function, an increment or decrement expression such as i or i++, or an await expression.

Condition : This is a Boolean expression, thus it will either return true or false depending on the state of the condition. If an expression is evaluated as true, then the loop will be executed again; if the expression evaluates as false, then the loop will be terminated.

Iterator : The iterator determines whether the loop variable will increase or decrease at each iteration.

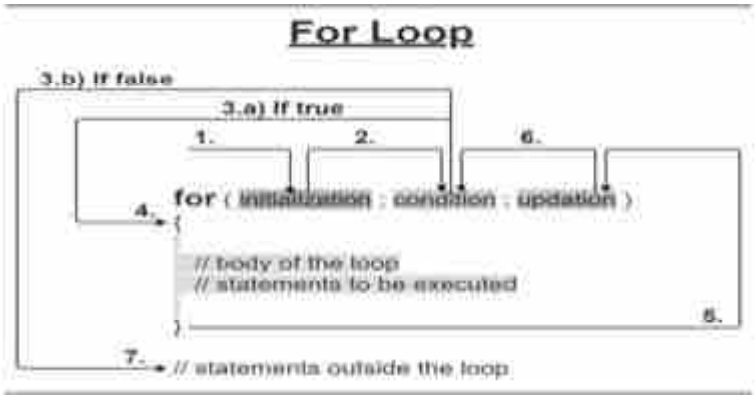


For loop execution

The following example for loop executes a code block 5 times:

```
{
for (int i = 0; i < 5; i++)
{ Console.WriteLine("Value of i: {0}", i);}
}
```

An initializer might be something like int i = 0 in the above example. This is where we declare an int variable i and then initialise it with 0. The second portion is a condition expression that looks like this: i<5. If the result of this condition is true, then a code block will be carried out. Following the execution of the code block, it will go to the iterator section, which is the third part. The incrementing statement denoted by i++ raises the current value of the iteration loop variable I by one. Now, it will carry out the previous step of checking the conditional phrase again and will continue doing so until the conditional expression yields a negative result. The processes involved in putting for loop into action are shown in the picture below.



For Loop Execution

- ***C# while Loop :***
  - C# do while Loop
  - C# nested Loop

The value of a variable may go up or down depending on whether or not the steps in for loop are executed. Let us see the reverse for loop with an **example** :

```
{
for(int i = 10; i> 0;i--)
{
Console.WriteLine("Value of i: {0}", i);
}
}
```

The break keyword may also be used as an alternative means of escaping a for loop. **For example** :

```
{ for (int i = 0; i < 10; i++)
{ if( i == 5 )
break;
Console.WriteLine("Value of i: {0}", i); }
```

❑ **Check Your Progress – 5 :**

1. There are 3 components included in any loop: Initialization, _____ and Iterator :

   a. Initializer b. Condition c. Switch d. None of these

- ***C# while Loop :***

The while loop is available to programmers working in C#, and it allows them to continually run a section of code for as long as a certain condition remains true.

Syntax :

```
while(condition)
{
   //code block
}
```

The while loop is initiated by using the keyword while, and it is required to include a Boolean conditional expression that is enclosed in brackets and returns either true or false. It continues to run the code block until the conditional expression that was supplied yields a false result.

The initialization as well as the increment and decrement sections are included inside the for loop. When working with a while loop, initialization should be performed before the loop begins, and any actions to increase or decrease values should be included inside the loop itself. **For example** :

```
{
int i = 0; // initialization
while (i < 10) // condition
{
   Console.WriteLine("i = {0}", i);
   i++; // increment
}
}
```

In the previous example, there is a while loop that contains the phrase i<10. The value of i was raised to 1 inside of a while loop by using the i++ operator. The while loop that was just described will be carried out once the value of i is equal to 10 and the test that i<10 produces a false result.

In order to prevent creating an endless loop, you should either check to see if the expression being evaluated as a condition evaluates to false or leave the while loop on some condition. Because the following loop does not include an acceptable condition or a break the loop statement, it is a while loop that continues forever. Let us see the example of infinite while loop :

```
int i = 1;
while (i > 0)
{ Console.WriteLine("i = {0}", i);
   i++; }
```

- ***C# do while Loop :***

The difference between the do while loop and the while loop is that the do while loop guarantees that the code block will be run at least once.

Syntax :

```
do
{
//code block
} while(condition);
```

The do–while loop is initiated with the term do, which is then followed by a code block and a Boolean expression that begins with the keyword while. The do while loop stops execution exits when a Boolean condition evaluates to false. Due to the fact that the while (condition) is provided at the very end of the block, it unquestionably runs through the code block at least once. **For example** :

```
{
int i = 0;
do
{
Console.WriteLine("i = {0}", i);
i++;
} while (i < 5);
}
```

The initialization should be specified outside of the loop, and the counter should be incremented and decremented within the do while loop. Use the break or the return keyword to quit from the do while loop.

2.6 Subroutines and Functions :

When using functions or subroutines, the programmer will first create the function so that it may take inputs (parameters). The function will be called to be performed at some point in the programme, maybe in the last section, and the output will then be sent back to the browser once it has been processed.

Subroutines : One kind of code construct is called a subroutine. A pattern of coding is referred to as a code construct. Instead of computing and returning data to the user, subroutines are used to modularize scripts in order to improve performance, cut down on redundant code, and enhance the readability of the scripts. In ASP.NET, the syntax for a subroutine looks like this :

```
Sub Subroutinename(ByVal variable as variable type)
Subroutine_instructions_here
End Sub
```

A subroutine is a self–contained block of code that may or may not include inputs or outputs depending on the circumstances.

```
Private Sub PrintHelloWorld()
Console.WriteLine("Hello World")
End Sub
```

A certain kind of subroutine known as an Event Handler is where you should place your code in order to handle a button click.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
End Sub
```

Functions : A function allows you to encapsulate a piece of code and call it from other parts of your code. You may very soon run into a situation where you need to repeat a piece of code, from multiple places, and this is where functions come in. In C#, they are basically declared like this :

```
<visibility> <return type> <name>(<parameters>)
{
<function code>
}
```

Simply writing the name of the function, followed by an open parenthesis, any arguments (if there are any), and then a closed parenthesis is all that is required to call a function. The first part, public, is the visibility, and is optional. If you don't define any, then the function will be private. Next is the type to return. It could be any valid type in C#, or as we have done it here, void. A void means that this function returns absolutely nothing. Also, this function takes no parameters, as you can see from the empty set of parentheses, so it's actually just a tad bit boring. **For example :**

```
public int AddNumbers(int number1, int number2)
{
int result = number1 + number2;
return result;
}
```

The function no longer outputs anything; instead, it does some computation and then gives the result to the caller. It requires two arguments, and both of those parameters must also be integers. Because of this, we do not need to manually compose the calculation code each time; rather, we can just use this method whenever we need to add two integers somewhere else in our code instead. This function is called like this :

```
int result = AddNumbers(10, 5);
Console.WriteLine(result);
```

As was said, this function does in fact return anything; indeed, it is required to do so since we instructed C# that it is expected to do so. When we declare a type of return that is anything other than void, we are compelled to return something to the calling function. You can see the compiler give you a warning if you attempt to remove the return line from the example that was just given :

'AddNumbers(int, int)': not all code paths return a value

The compiler is bringing to our attention the fact that we have a function that does not return anything, despite the fact that we promised it would. And the compiler is a really astute individual! Try something more interesting like this instead of eliminating the line :

```
public int AddNumbers(int number1, int number2)
{
int result = number1 + number2;
if(result > 10)
{
return result;
}
}
```

You are going to come across the identical mistake; however, why ? Mostly due to the fact that there is no assurance that our if statement will evaluate to true, and hence there is no assurance that the return line will be performed. Having a second return statement that functions similarly to the default one at the very end might help you overcome this problem :

```
public int AddNumbers(int number1, int number2)
{
int result = number1 + number2;
if(result > 10)
{ return result; }
return 0;
}
```

This will not only address the issue that we caused for ourselves, but it will also demonstrate to you that we may have more than one return statement inside our function. The function is terminated as soon as the return statement is reached, and none of the remaining code in it is carried out. In this particular scenario, it indicates that the "return 0" condition is never satisfied so long as the result is more than 10.

There are several types of functions available in c# programming which are as follows :

Parameter less Function : A method is said to be parameter less if it doesn't contain any parameters at all. This is a section of code that contains the statement :

```
namespace MethodExamples
{ class ParameterLessMethod
{ protected void Message()
{ Console.WriteLine("Hello World!");}
static void Main()
{ var p = new ParameterLessMethod();
p.Message();
Console.ReadLine(); }}}
```

Parameterized Function : A method is referred to as being parameterized when it has at least one and perhaps more than one parameter. **For example :**

```
namespace MethodExamples
{ class ParameterMethod
{ public void PrintMessage(string message)
{ Console.WriteLine("Hello " + message); }
static void Main()
{ var p = new ParameterMethod();
p.PrintMessage("Students!!");
Console.ReadLine();
}}}
```

Parameter and Return Function : The void type is the return type that should be specified for a method that does not return any value. **For example :**

```
namespace MethodExamples
{ class ParameterAndReturnMethod
{ protected int MethodValue(int value)
{ Console.WriteLine("Entered value is:"+value);
return value; }
static void Main()
{ var p = new ParameterAndReturnMethod();
p.MethodValue(87);
Console.ReadLine();
} } }
```

Call by Value Function : Instead of passing a reference to the original value, value type arguments instead send a copy of it to the method being called. It does not alter or affect the value that was initially present. Changing the value that is supplied does not affect the value that is being used. The following is an example of the call by value technique :

```
namespace MethodExamples
{ class CallByValueMethod
{ public void ShowValue(int value)
{ value *= value;
```

```
Console.WriteLine("Value calling method:" + value);
}
static void Main(string[] args)
{
int value = 10;
var p = new CallByValueMethod();
Console.WriteLine("Value before calling method:"+value);
p.ShowValue(value);
Console.WriteLine("Value after calling method:" + value);
Console.ReadLine(); } } }
```

Reference Function : The reference value may be passed along and returned with the help of the ref keyword. Any change in value that was supplied as a reference would likewise reflect that change and change. **For example :**

```
namespace MethodExamples
{
   class CallByReferenceMethod
   { protected void PrintNumber(ref int number)
   { number *= number;
      Console.WriteLine("Value of reference method: " + number);
   }
   static void Main()
   {
   int number = 100;
   var p = new CallByReferenceMethod();
   Console.WriteLine("Value before reference method called: " + number);
   p.PrintNumber(ref number);
   Console.WriteLine("Reflected value afrter method called: " + number);
   Console.ReadLine();
   } } }
```

Out Parameter Function : When passing the argument as an out parameter, the out keyword is what's utilised. It is exactly the same as a reference type, with the exception that it does not demand to be initialised before passing it on to the function. When we need to return numerous values from a method, it comes in handy to have this capability. **For example :**

```
class OutParameterMethod
{ protected void PrintNumber(out int number)
{ int squre = 10;
number = squre;
number *= number;
Console.WriteLine("Value when calling out parameter: " + number);
}
```

```
static void Main()
{ int number = 20;
   var p = new OutParameterMethod();
   Console.WriteLine("Value befor calling out parameter method: "
   + number);
   p.PrintNumber(out number);
   Console.WriteLine("Value after calling out parameter method: "
   + number);
   Console.ReadLine();
} } }
```

Function Overloading : Method overloading refers to the situation in which two or more methods have the same name but vary in their signatures. **For example :**

```
{ var demo = new Demo();
int firstTotal= demo.Add(10, 20);
int secondTotal = demo.Add(10, 20, 30);
double thirdTotal= demo.Add(10.5, 20.5);
Console.WriteLine("Total:{0}",firstTotal);
Console.WriteLine("Second Total:{0}",secondTotal);
Console.WriteLine("Third Total:{0}", thirdTotal);
Console.ReadLine();
}}
public class Demo
{ public int Add(int valueOne, int valueTwo)
{ return valueOne + valueTwo; }
public int Add(int valueOne, int valueTwo, int valueThree)
{ return valueOne + valueTwo + valueThree; }
public double Add(double valueOne, double valueTwo)
{ return valueOne + valueTwo; } } }
```

Methods are able to be overloaded when different data types are used for the parameter, when the order of the parameters is changed, and when there are more parameters than usual.

Abstract Method : In programming, the presence of the abstract modifier in a method indicates that the method is an abstract method. Because abstract methods do not have an associated implementation, the method definition is only followed by a semicolon rather than the typical method block. All abstract methods have to be implemented by any classes that derive from the abstract class. **For example :**

```
{ public abstract class AbstractMethod
{ public abstract void MyAbstractMethod(); }}
```

There are several characteristics of the abstract method which are as follows :

- A method that is abstract is also a method that is virtual by definition.
- Abstract classes are the only ones that may have method declarations that are abstract.
- The method declaration for an abstract method merely comes to a conclusion with a semicolon, and there are no curly brackets ( ) placed after the signature of the method.
- An abstract method declaration does not include any real implementation, and there is no method body.
- A method override, which is a component of a class that is not abstract, is responsible for providing the implementation.
- When declaring an abstract method, you must not utilise the static or virtual modifiers. This constitutes a mistake.

Virtual Method : It is possible to change a method, property, indexer, or event declaration such that it may be overridden in a class that is derived from the base class by using the virtual keyword. A Virtual method is required to have a default implementation at all times. However, it is possible for it to be overridden in the derived class, albeit doing so is not required. It is possible to bypass this restriction by using the override keyword. **For example :**

```
{ public class VirtualMethod
{ public virtual string Message(string msg)
{ return "Hello" + msg; }}
public class myClass : VirtualMethod
{ public override string Message(string msg)
{ return base.Message(msg);}}
public class program
{ static void Main()
{ var p = new VirtualMethod();
string name= p.Message("Farhan Ahmed");
var p1 = new myClass();
string name1= p1.Message("Rahul Sharma");
Console.WriteLine(name);
Console.WriteLine(name1);
Console.ReadLine();} } }
```

The virtual keyword is used to change the declaration of a method, property, indexer, or event so that it may be overridden in a class that is derived from the base class. It is required that a Virtual method always have a default implementation. Nevertheless, it is permissible for the derived class to override it even if doing so is not required. The override keyword may be used to modify its behaviour.

❑ **Check Your Progress – 6 :**

1. Function _____ refers to the practice of creating many methods with the same name that share the different arguments :

   a. Overriding b. Overloading c. Abstract d. Virtual

Static Method : The creation of a static method requires the inclusion of the keyword static. A method that may be invoked without first constructing an instance of the class's object is referred to as a static method. Either the name of the class itself or a reference to an Object of that class may be used to refer to it. For example:

```
static class StaticMethod
{ public static void PrintValue(int value)
{ Console.WriteLine("The Value is: " + value);}}
public class StaticExample
{ static void Main(){
StaticMethod.PrintValue(100);
Console.ReadLine(); }}
```

2.7 Creation and Testing Asp.Net Pages :

Creation of ASP.NET Pages : Online Forms, ASP.NET Model–View–Controller (MVC), and ASP.NET Web Pages are the three frameworks that are offered by ASP.NET for the creation of web applications. Each of the three frameworks is reliable and well–developed, and any one of them may be used to develop excellent online applications. Online Forms, ASP.NET Model–View–Controller (MVC), and ASP.NET Web Pages are the three frameworks that are offered by ASP.NET for the creation of web applications. Each of the three frameworks is reliable and well–developed, and any one of them may be used to develop excellent online applications.

Each framework caters to a certain method of software development. Your programming assets (knowledge, abilities, and development experience), the kind of application you're producing, and the method of application creation with which you are most comfortable all play a role in determining which one is the best option for you. In future versions of ASP.NET, each of these three frameworks will get support, along with updates and improvements.

The first thing that we are going to do is create a MasterPage, which will give all of the pages a consistent look and feel. I wanted to make things as easy as possible, so I just went ahead and made a basic master page with the company logo on the left–hand side and a single ContentPlaceHolder for the page's content. For example:

```
<%@MasterLanguage="C#" AutoEventWireup="true" CodeFile="Main Master.master.cs" Inherits ="MainMaster" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/ TR/xhtml1/DTD/xhtml1–transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>Untitled Page</title>
</head>
<body>
<form id="form1" runat="server">
<div>
```

```
<table width="100%">
<tr><td width="5%"> <h2>Your Site Logo</h2></td><td></td></tr>
<tr><td colspan="2"><asp:Label ID="lblPageHead" runat="server"></
asp:Label> </td></tr>
<tr><td colspan="2">
<asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
</asp:contentplaceholder>
</td></tr>
</table>
</div>
</form>
</body></html>
```

The creation of a template ASPX page that contains the fundamental framework of the dynamic aspx pages will be the next stage in our process. **For example :** Template ASPX Page

```
<%@ Page Language="C#" MasterPageFile="MainMaster.master"
CodeFileBaseClass=" BasePage" AutoEventWireup="true" CodeFile=
"CommonCodeBehind.cs" Inherits=" CommonCodeBehind" PageID="[ID]"
Title="[Title]" MetaKeywords="[key]" MetaDescription= "[MetaDes]" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlace
Holder1" Runat="Server">
</asp:Content>
```

During the process of constructing the page, you will need to change the placeholders that are described in the template with the real values that should be used. Additionally, we have created the PageID, MetaKeywords, and MetaDescription custom attributes that are linked through the CodeFileBaseClass property. These attributes are all specified in the BasePage class. Based on the value that is specified on the @Page directive, the Meta keyword and description tags will be dynamically generated inside the BasePage class.

We are going to develop a page that replaces the user–provided Title, Meta Keyword, Meta Description, and Content with the placeholders that are in the Template.aspx page, and then we are going to save that page as a distinct ASP.NET page. Please refer to the diagram that has been provided below.

Creation of Web Page using ASP.NET

Testing : Testing of ASP.NET pages involves the testing the objects that are included inside it. The criteria stipulate that the page in question must load properly and accurately portray the data retrieved from the database. Every control that is shown on the website must also act in the manner that the developer had in mind.

For the purpose of demonstrating testing of the functionality of an ASP.NET page, we will test a page that has a grid control. In order to do this, we will first establish a TestComplete project, then write all of the required tests, and then use it to automate all of the operations.

When analysing the functionality of websites, it is necessary to examine each individual page of the website. In this article, you will learn how to develop tests that examine the operation of an ASP.NET website, as well as how to automate the quality assurance testing process using TestComplete. In addition to that, we will provide some examples of methods that execute typical activities during the testing of grid controls. Following the inspection of your custom grid, you will have the ability to develop your own methods that carry out duties that are analogous.

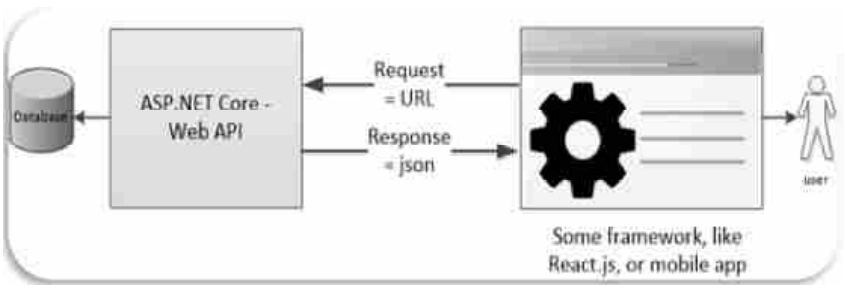
For the purpose of testing, we need to compose a number of processes that carry out typical duties seen on testing pages. These procedures include the following :

- the process of locating the required cell
- obtaining a value from a cell
- looking for a value in a table using a key
- getting values from the columns that have been supplied

Because Test Complete exposes certain methods and attributes that provide access to controls and other aspects of your ASP.NET page, this is something that is doable. The process of developing tests is substantially streamlined because to these approaches and attributes. Helper procedures are going to be implemented as scripting routines by us. You are able to tweak the code so that it corresponds with the specifics of your application, and then you can use these processes to do quality assurance testing on your specific grid. Consequently, while we are doing tests, we will develop the following steps :

- Establishing the nature of the test item
- Developing a project with TestComplete.com
- Developing Operational Assessments
- Performing Functional Tests Through Automation
- Carrying Out Functional Tests and Examining the Results

When testing web pages, the typical architecture consists of the following components :



Architecture of the ASP.NET

In this section, we will implement a test method inside an aspx page and enable the running of a hidden handshake or backdoor within an aspx page. It is seen that many Unit Tests for asp.net page will take: Create an aspx page, Do some scenario and verify the output of the page.

Testing your application is necessary if you want to ensure that it performs in the manner that you anticipate. There is no escaping the reality of the situation. On the other hand, it is much simpler to say than to really accomplish. There is a wide variety of approaches to testing, each of which has certain advantages over the others. Each technique places an emphasis on evaluating a certain facet of your application.

When you execute this as part of your testing, those three stages are going to be dispersed throughout your code into three distinct locations:

- The building of the page and the redirect to the scenario of your choosing will be handled by the PageLoad event of your aspx page (the TestMethod).
- Within your ASPX code behind page, there will be a TestMethod that is responsible for carrying out the scenario component.
- The HTML output from the aspx page will be checked by the Unit Test, which will make a Callout to that page. This is exactly when everything begins to unfold.

Unit Testing : When doing unit testing, the primary focus is on exercising a single component in isolation from the rest of the system. If it is given a certain set of inputs or a known state of the system, it will simulate or stub out its dependencies to ensure that it generates the correct results.

In general, we think of our classes as units and test the public methods of those classes while mocking the relationships between those classes. If you don't do this, you won't be able to separate your classes from their dependencies, which is a major flaw in the design of your code that also has the added bonus of moving your code towards a loosely coupled design.

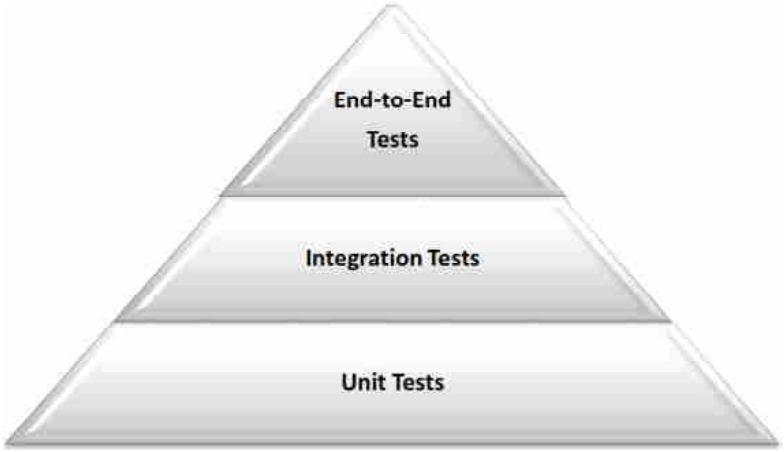
Integration Testing : During these tests, many units will be worked together, and each will be treated as if they were a black box. They will be given certain inputs, and they will be expected to generate the desired outputs. When discussing online applications, it is standard practise to think of your web server as a black box and to write integration tests that exercise the server's public endpoints without making use of a browser or client–side code.

This indicates that your integration tests will put your actual web server code through its paces by submitting real HTTP requests; nevertheless, it will be up to you to choose whether or not to include any external dependencies, such as databases, inside your test. These tests are of tremendous use when it comes to developing REST APIs.

End–to–End Testing : End–to–end testing will put your system through its paces by exercising it via the same user interface (UI) that is used by ordinary users. Due to the fact that these tests are strongly tied to your user interface and hence often become invalid as a consequence of changes made to the latter, they are the most costly tests to create and maintain.

The Hierarchy of the Tests : One of the most important things to take away from this summary is the fact that various kinds of tests serve a variety of goals and have varying costs associated with their installation and upkeep.

Because of this, a number of individuals, like Martin Fowler, have been discussing the idea of the testing pyramid for a considerable amount of time.



Testing Pyramid

As can be seen, it is a graphic representation of a testing strategy that takes into account the benefits and drawbacks associated with the various types of testing.

At the bottom you have a massive suite of unit tests that thoroughly tests every piece of code with important business logic. These are regularly executed by CI servers, which in turn offer developers with constant feedback about the changes they make to their code.

When you go on to the next level, there will be fewer integration tests, and you will need to ensure that a few of your components, when combined, will still achieve the desired outcomes (In web applications, this would be your web server without client–side code). These are often executed by developers at certain points in time, such as before making a change or after adding a new feature to the system. Depending on the specific nature and implementation of the tests, as well as how external dependencies like databases were separated, CI servers may either execute them after deploying a staging or integration environment, or they will run them after every change, depending on the situation.

Last but not least, you should have a select few End–to–End tests at the very top of your list. These tests ensure that your application is not fundamentally flawed when seen from the viewpoint of a real user. These will be executed by CI servers in the majority of cases once environments have been deployed, although developers will run them even less often than integration tests.

2.8 Objects in Asp.Net :

You are able to get access to information about the Web server, the client who is visiting a Web page, the Web application that includes the Web page and the fields in the HTTP request and response streams by using the built–in objects that come standard with ASP.NET.

ASP.NET includes the Request, Response, Server, Application, and Session objects, all of which are used in a manner that is quite similar to how they were utilised in ASP. On the other hand, ASP.NET defines these objects as belonging to brand new classes in the System.Web namespace.

ASP.NET Objects

Objects	Description
Application	Describes the methods, attributes, and collections of the object that keeps information relevant to the whole Web application. This information may include variables and objects that continue to exist for the duration of the application's lifespan.
Request	This document provides a description of the object's methods, attributes, and collections that are responsible for storing information relevant to HTTP requests. This includes things like cookies and form submissions.
Response	This document provides a description of the object's methods, attributes, and collections that hold information connected to the server's response. This involves displaying material, modifying headers, and other similar activities.
Server	Contains a description of the object's methods and attributes, which together provide methods for a variety of server operations. You will be able to run code, get error conditions, encode text strings, build objects that may be used by the web page, and map physical roads by using these techniques.
Session	Describes the methods, attributes, and collections of the object that is responsible for storing information relevant to the user's session. This information may include variables and objects that are active for the duration of the session.

- ***Application Object :***

An application is a collection of ASP files that are designed to operate together in order to accomplish a certain goal. The Application object is what binds all of these files together into a single unit. The following is a description of the collections, methods, and events available on the Application object:

Collections :

- **Contents Collection :** All of the things that were added to the application or session by means of a script command are located inside the Contents collection. Use the Remove and RemoveAll methods in order to remove individual items or an entire item from the Contents collection.

  Syntax : Application.Contents(Key)

Key parameter is must to retrieve as it is the name of the item. There are few examples of it.

Example : Please take note that the name and objtest entries will both be added to the Contents collection :

<% Application("name")="BAOUASPNET" Set Application("objtest") =Server.CreateObject ("ADODB.Connection")%>

- **StaticObjects Collection :** Contains all the objects appended to the application with the HTML <object> tag.

  Syntax : Application.StaticObjects (Key)

Key parameter is must to retrieve as it is the name of the item. There are few examples of it.

Example : To iterate over the collection of StaticObjects as follows :

```
<%
for each x in Application.StaticObjects
Response.Write(x & "<br>")
next
%>
```

Methods :

- **Contents.Remove :** The Contents.Remove method deletes an item from the Contents collection.

Syntax : Application.Contents.Remove(name|index)

Parameter name is used for the name of the item to remove and index is the item's index of the item to remove. There are few examples of it :

Example :

```
<%
Application("test1")=("First test")
Application("test2")=("Second test")
Application("test3")=("Third test")
Application.Contents.Remove("test2")
for each x in Application.Contents
   Response.Write(x & "=" & Application.Contents(x) & "<br>")
next
%>
```

Output:

```
test1=First test
test3=Third test
```

- **Contents.RemoveAll :** The Contents.RemoveAll method deletes each and every item in the collection called Contents.

Syntax : Application.Contents.RemoveAll()

Example :

```
<%
Application.Contents.RemoveAll()
%>
```

- **Lock :** Through the usage of the Lock method, other users are prevented from making changes to the variables contained inside the Application object (used to ensure that only one client at a time can modify the Application variables).
- **Unlock :** Other users are allowed to make changes to the variables that are contained in the Application object after using the Unlock function (after it has been locked using the Lock method).

Syntax : Application.Lock and Application.Unlock

Example : The following example demonstrates how to use the Lock method to stop several users from accessing a variable at the same time by locking the variable visits and then using the Unlock method to unlock the locked object so that the subsequent client may increment the variable visits :

```
<%
Application.Lock
Application("visits")=Application("visits")+1
Application.Unlock
%>
This page has been visited
<%=Application("visits")%> times!
```

Events :

- **Application_OnStart Event :** Before the very first new session is ever initiated, the Application OnStart event will take place (when the Application object is first referenced). The Global.asa file will be updated to reflect this occurrence. Note that making a reference in the Application_OnStart event script to the objects Session, Request, or Response will result in an error being generated.
- **Application_OnEnd Event :** When an application is terminated, the Application OnEnd event is triggered (when the web server stops). The Global.asa file will be updated to reflect this occurrence. Note that you cannot use the MapPath method in the code for the Application_OnEnd event.

Syntax :

```
<script language="c#" runat="server">
protected void Application_Start(object sender, EventArgs e)
{ //this event is execute only once when application start and it stores the server memory until the worker process is restart
    Application["startMessage"] = "The application has started.";
}
void Application_End(object sender, EventArgs e)
{
    // this event is execute Code that runs on application ends
    Application["endtMessage"] = "The application has ended.";
}
</script>
```

- **ServerVariables :** It is possible to access the values of the server variables by using the ServerVariables collection.

Syntax : Request.ServerVariables (server_variable)

The server_variable parameter is necessary and is the name of the variable on the server that should be retrieved.

Example : You are possible to iterate over each of the server variables in the following manner :

```
<%
for each x in Request.ServerVariables
    response.write(x & "<br>")
next
%>
```

The following example explains how to discover information about a visitor, including the kind of browser they are using, their IP address, and more :

```
<html>
<body>
<p>
<b>You are browsing this site with:</b>
<%Response.Write(Request.ServerVariables("http_user_agent"))%>
</p>
<p>
<b>Your IP address is:</b>
<%Response.Write(Request.ServerVariables("remote_addr"))%>
</p>
<p>
<b>The DNS lookup of the IP address is:</b>
<%Response.Write(Request.ServerVariables("remote_host"))%>
</p>
<p>
<b>The method used to call the page:</b>
<%Response.Write(Request.ServerVariables("request_method"))%>
</p>
<p>
<b>The server's domain name:</b>
<%Response.Write(Request.ServerVariables("server_name"))%>
</p>
<p>
<b>The server's port:</b>
<%Response.Write(Request.ServerVariables("server_port"))%>
</p>
<p>
<b>The server's software:</b>
<%Response.Write(Request.ServerVariables("server_software"))%>
</p>
</body>
</html>
```

Property :

- **TotalBytes :** The total amount of bytes that the client submitted in the request's body may be found in the TotalBytes field, which is a read–only property that returns the value.

Syntax : varbytes=Request.Totalbytes

Example : The line of code that follows changes the value of the variable a to be equal to the total amount of bytes that were sent in the request's body :

```
<%
dim a
a=Request.TotalBytes
%>
```

Method :

- **BinaryRead :** It is possible to get the data that was supplied to the server from the client as part of a POST request by using the BinaryRead method. The information will be saved in a secure array (an array that stores information about the number of dimensions and the bounds of its dimensions).

Syntax : Request.BinaryRead(count)

The count parameter is necessary and it indicates the number of bytes that should be read from the client.

Example : The following example demonstrates how to make use of the BinaryRead method in order to save the contents of a request inside a secure array:

```
<%
dim a,b
a=Request.TotalBytes
b=Request.BinaryRead(a)
%>
```

- ***Response Object :***

It is possible to convey output from the server to the user by using the ASP Response object. The following list provides descriptions of its collections, properties, and methods :

Collections :

- **Cookies :** You may either set or receive cookie values by using the Cookies collection. If the cookie does not already exist, it will be generated and given the value that was supplied in the case that it is created. Note that the Response.Cookies command has to come before the html element in order for it to work properly.

Syntax :

```
Response.Cookies(name)[(key)|.attribute]=value
variablename=Request.Cookies(name)[(key)|.attribute]
```

Key parameter is optional. It identifies the key to which the value will be associated. **Attribute** parameter is also optional. It details the information that pertains to the cookie. **Value** parameter is the value of the cookie. It is

essential for the Response.Cookies command. The **name** parameter is the name of the cookie and it is essential. It is possible that this parameter is one of the following:

- Write–only access granted to domain. Only requests made to this site will result in the cookie being delivered.
- Expires and can only be written with. The date after which the cookie will no longer function. In the event that no date is supplied, the cookie will become invalid at the conclusion of the session.
- HasKeys is read–only at this time. Indicates whether or not the cookie contains keys. (Since this is the only property that is compatible with the Request.Cookies command, you will need to use this one.)
- Write–only access to the Path. In the event that it is set, the cookie is only given to requests that are directed to this route. If not specified, the path to the programme will be utilised.
- Safe and limited to writing only. Indicates whether the cookie has been encrypted.

Example : To generate a cookie or to change the value of an existing cookie, you may use the "Response.Cookies" command :

```
<% Response.Cookies("firstname")="Mahesh" %>
```

In the code that you just looked at, we have already generated a cookie with the name "firstname" and given it the value "Mahesh"

A collection of various values may also be stored in a cookie if the user allows it. We make the assertion that the cookie has Keys. In the following illustration, we will establish a cookie collection that we will call "user." The "user" cookie includes Keys that each carry information specific to a user, including the following :

```
<%
Response.Cookies("user")("firstname")="Mahesh"
Response.Cookies("user")("lastname")="Patel"
Response.Cookies("user")("country")="India"
Response.Cookies("user")("age")="48"
%>
```

The following line of code will read all of the cookies that a user's browser has received from your server. Take note that the HasKeys attribute is used in the code to determine whether or not a cookie has Keys :

```
<html>
<body>
<% dim x,y
for each x in Request.Cookies
response.write("<p>")
if Request.Cookies(x).HasKeys then
for each y in Request.Cookies(x)
response.write(x & ":" & y & "=" & Request.Cookies(x)(y))
```

```
response.write("<br>")
next
else
Response.Write(x & "=" & Request.Cookies(x) & "<br>")
end if
response.write "</p>"
next%>
</body>
</html> %>
```

Output:

```
firstname=Mahesh
user:firstname=Mahesh
user:lastname=Patel
user:country=India
user:age=48
```

Property :

- **Buffer :** The Buffer attribute allows one to specify whether or not the output should be buffered. When the output is buffered, the server will delay sending a response to the browser until either all of the server scripts have been processed or the script uses the Flush or End method. This will continue until the output buffer is flushed or the script is terminated.

Syntax : response.Buffer[=flag]

The flag parameter is a value of type Boolean that indicates whether or not the output of the page should be buffered. False indicates no buffering. The server will send the output as it is processed. True indicates buffering. The server will not send output until all of the scripts on the page have been processed, or until the Flush or End method has been called.

Example : Within the context of this demonstration, the browser will not get any output until the iteration has completed. It would post a line to the browser each time it went through the loop if buffer was set to False, but if it was set to true, it would not.

```
<%response.Buffer=true%>
<html>
<body>
<%
for i=1 to 100
   response.write(i & "<br>")
next
%>
</body>
</html>
```

- **CacheControl :** The CacheControl attribute determines whether or not an ASP proxy server is allowed to cache the output that is created by ASP. A proxy server will not save a cached copy of the requested resource by default.

  Syntax : response.CacheControl[=control_header]

The control_header parameter is a cache control header that may either be set to "Public" or "Private," depending on the user's preference. The default setting is private, which implies that only private caches are permitted to store a copy of this page. When this option is in place, proxy servers will not cache any pages.

The term "public" refers to caches that are accessible to the public. When this option is applied, pages will be cached by proxy servers.

Example :

<%response.CacheControl="Public"%> or

<%response.CacheControl="Private"%>

- **Charset :** The name of a character set is added to the Response object's content–type header when the Charset attribute is used. ISO–LATIN–1 is the character set that is used by default. Note that this property will take any string as the name, regardless of whether or not the characters in the string belong to a recognised character set.

  Syntax : response.Charset(charsetname)

The charsetname is the parameter used as a string that indicates the character set that should be used for the page.

Example : In the event that the Charset attribute of an ASP page is not set, the content–type header will read as follows :

content–type:text/html

- **ContentType :** The HTTP content type for the response object may be altered via the use of the ContentType attribute.

  Syntax : response.ContentType[=contenttype]

The ContentType is the parameter used as a string that describes the format of the material. Consult the documentation that comes along with your browser or the HTTP protocol for a complete list of the content types.

Example : In the event that the ContentType property of an ASP page is left unset, the following values will be used for the content–type header :

content–type:text/html

Additional typical examples of ContentType values :

<%response.ContentType="text/HTML"%>

<%response.ContentType="image/GIF"%>

<%response.ContentType="image/JPEG"%>

<%response.ContentType="text/plain"%>

- **Expires :** The Expires attribute allows you to specify, in minutes, how long a page will remain in a browser's cache before it becomes invalid. The cached version of the page will be presented to users who revisit that page before the time it is set to expire.

Syntax : response.Expires[=number]

The number is the parameter used as the number of minutes left until the page is permanently deleted.

Example : The code that follows indicates that the page will never be stored in a cache :

```
<%response.Expires=–1%>
```

- **ExpiresAbsolute :** The ExpiresAbsolute feature allows a date and time to be specified for when a page that has been cached on a browser will become invalid. The cached version of the page will be shown to users who revisit the same page before the specified date and time.

  Syntax : response.ExpiresAbsolute[=[date][time]]

  The date is the parameter used as it indicates the date after which the page will no longer be accessible. In the event that this option is left unspecified, the page will be rendered inaccessible at the hour that is indicated on the day that the script is executed.

  The time is the parameter used as it indicates the period after which the page will no longer be accessible. In the event that this option is not provided, the page will become inaccessible at the stated time the next day (midnight).

  Example : It can be seen from the following piece of code that the page will be taken down at 4:00 PM on September 11, 2022 :

  ```
  <%response.ExpiresAbsolute=#September 11, 2022 16:00:00#%>
  ```

- **IsClientConnected :** If the client and the server are still connected to one another, the IsClientConnected property will return true.

  Syntax : response.IsClientConnected

  Example :

  ```
  <%
  If response.IsClientConnected=true then
     response.write("The user is still connected!")
  else
     response.write("The user is not connected!")
  end if
  %>
  ```

- **Status :** The value of the status line that is delivered by the server is the information that is specified by the Status property.

  Syntax : response.Status=statusdescription

  The statusdescription parameter used as a three–digit number plus an explanation of what that number means, such as "404 Not Found." Note: The HTTP standard includes a section that defines status codes.

  Example :

  ```
  <%
  ip=request.ServerVariables("REMOTE_ADDR")
  if ip<>"194.248.333.500" then
  ```

```
response.Status="401 Unauthorized"
response.Write(response.Status)
response.End
end if
%>
```

Method :

- **AddHeader :** Adding a new HTTP header and a value to the HTTP response is the responsibility of the AddHeader method. It is not possible to delete a header once it has been added to the document. When using IIS 4.0, you are required to call this method in advance of sending any output to the browser. If you are using IIS 5.0, the AddHeader method can be called at any point in the script.

  Syntax : response.AddHeader name,value

The name parameter is necessary and it is the name of the new variable to be included in the header (cannot contain underscores). The value parameter is necessary and it is the value that will serve as the starting point for the new header variable.

Example : <%Response.AddHeader "WARNING","Error message text"%>

- **AppendToLog :** Adding a text at the very end of the item in the server log for this request is what the AppendToLog function does. Within a single script, you are free to invoke this method as many times as you want. The supplied string will be added to the log entry each time it is called, as it will append the string.

  Syntax : response.AppendToLog string

The string parameter is necessary and the content that should be added to the log file (cannot contain any comma characters)

Example : the content that should be added to the log file (cannot contain any comma characters)

- **BinaryWrite :** The BinaryWrite method skips the step of converting the data into character form before writing it straight to the output.

  Syntax : response.BinaryWrite data

The data parameter is necessary and the information that will be delivered in binary form.

Example : If you have an object that produces an array of bytes, you can use BinaryWrite to transmit the bytes to an application. Here are some examples of how to use this function:

```
<% Set objBinaryGen=Server.CreateObject("MyComponents.Binary
Generator") pic=objBinaryGen.MakePicture response.BinaryWrite pic %>
```

- **Clear :** The Clear method purges any HTML output that has been buffered. Only the response body is cleared when using this approach; the response headers remain intact. This method will result in a run–time error if the response.Buffer variable is set to false.

  Syntax : response.Clear

Example :

```
<%response.Buffer=true%>
<html>
<body>
<p>This is an example I want to send to the BAOU student.</p>
<p>No, I changed my mind. I want to clear the text.</p>
<%response.Clear%>
</body>
</html>
```

- **End :** The End method brings back the result that was obtained before it terminated processing of the script. If Response is received, this procedure will flush the buffer. The value of the buffer has been changed to true. Calling Response is the appropriate action to do if you do not intend to provide any output to the user.

  Syntax : Response.End

  Example :

```
<html>
<body>
<p>I am writing some text. This text will never be
<% Response.End %>
finished! It is very late to write more!</p>
</body>
</html>
```

- **Flush :** Immediately sending buffered HTML output is what the Flush technique does. This method will result in a run–time error if the response.Buffer variable is set to false.

  Syntax : Response.Flush

  Example :

```
<% Response.Buffer=true %>
<html>
<body>
<p>I write some text, but I will control when the text will be sent to the browser.</p>
<p>The text is not sent yet. I hold it back!</p>
<p>OK, let it go!</p>
<% Response.Flush%>
</body>
</html>
```

- **Redirect :** The Redirect method sends the user to a new URL after it has been called.

  Syntax : Response.Redirect URL

The URL parameter is necessary and it is the Uniform Resource Locator (URL), which the user's browser will be routed to.

Example : <% Response.Redirect "https://www.BAOU.com"%>

- **Write :** The Write method will write a string that has been given to it to the output.

Syntax : Response.Write variant

The variant parameter is necessary and it is the information that has to be written.

Example :

<%Response.Write "Hello Students"%>

- ***Server Object :***

Accessing the attributes and methods of the server may be done via the usage of the ASP Server object. The following is a description of its characteristics and operational procedures :

Properties :

- **ScriptTimeout :** The ScriptTimeout attribute allows the user to either specify or get the maximum number of seconds that a script may execute before it is stopped.

Syntax : Server.ScriptTimeout[=NumSeconds]

The numseconds parameter is the longest amount of time a script will be allowed to execute before being stopped by the server. Default is 90 seconds.

Example : Adjust the timeout for the script:

<% Server.ScriptTimeout=200 %>

Properties :

- **CodePage :** When displaying dynamic content, the character set that is going to be utilised may be specified using the CodePage attribute.

Syntax : Session.CodePage(=Codepage)

The codepage parameter is specifies a code page, also known as a character set, for use by the computer that is executing the script engine.

Example :

<% Response.Write(Session.CodePage) %>

- **LCID :** A location or area may either be set or retrieved using the LCID property, which uses an integer to do so. The contents, such as date, time, and currency, will be shown in accordance with the area or region being viewed.

Syntax : Session.LCID(=LCID)

The LCID parameter is a locale identifier.

- **SessionID :** The SessionID property gives each user their own unique id and returns it. The server is responsible for generating the one–of–a–kind id.

Syntax : Session.SessionID

Example : <% Response.Write(Session.SessionID) %>

Methods :

- **Abandon :** A user session is terminated when the Abandon method is called. Note that the current Session object will not be removed when this method is invoked because it waits until all of the script on the current page has been parsed before doing so. This indicates that it is possible to access session variables on the same page as the call to abandon, but not on a different Web page. However, this does not mean that it is possible to access session variables in any other way.

  Syntax : Session.Abandon

  Example : File1.asp:

  ```
  <%
  Session("name")="Neelam"
  Session.Abandon
  Response.Write(Session("name"))
  %>
  ```

  Output: Neelam

  File2.asp:

  ```
  <% Response.Write(Session("name")) %>
  ```

  Output: (none)

- **Contents.Remove :** The Contents.Remove method deletes an item from the Contents collection.

  Syntax : Session.Contents.Remove(name|index)

  Parameter name is used for the name of the item to remove and index is the item's index of the item to remove.

  Example :

  ```
  <%Session("test1")=("First test")
  Session("test2")=("Second test")
  Session("test3")=("Third test")
  Session.Contents.Remove("test2")
  for each x in Session.Contents
     Response.Write(x & "=" & Session.Contents(x) & "<br>")
  next%>
  ```

  Output: test1=First test

  test3=Third test

- **Contents.RemoveAll :** The Contents.RemoveAll method removes every item in the Contents collection from the collection.

  Syntax : Session.Contents.RemoveAll()

  Example : <% Session.Contents.RemoveAll() %>

  Events :

- **Session_OnStart Event :** When a session is first created by the server, the Session OnStart event is triggered. The Global.asa file will be updated to reflect this occurrence.

- **Session_OnEnd Event :** When a session comes to a conclusion, the Session_OnEnd event takes place (abandoned or times out). The Global.asa file will be updated to reflect this occurrence. Note that you cannot use the MapPath method in the code for the Session_OnEnd event.

Syntax :

```
<script language="c#" runat="server">
void Session_Start(object sender, EventArgs e)
{ … }
protected void Session_End(object sender, EventArgs e)
{ … }
```

Examples :

Global.asa:

```
<script language="c#" runat="server">
void Session_Start(object sender, EventArgs e)
{ Session["Test"] = 1; }
protected void Session_End(object sender, EventArgs e)
{ Response.Redirect("LoginPage.aspx"); }
</script>
```

To display the number of current visitors in an ASP file:

```
<html>
<head>
</head>
<body>
<p> There are <%response.write(Application("visitors"))%>online now!</p>
</body>
</html>
```

2.9 Object Creation with Parameters :

The term "fully object–oriented language" refers to a programming language in which everything is represented as an object, but the language is unable to distinguish between objects of primitive types and objects of classes. C#, on the other hand, is not an entirely object–oriented language because it supports many concepts from procedural programming, such as pointers, to some extent. In object–oriented programming (OOP), an object is the fundamental building block that stands in for a real–world entity. A typical C# application will generate a large number of objects, all of which will communicate with one another by executing various methods. The following are some of the methods that we may create objects using C# :

- ***Using the 'new' operator :***

A class is a reference type, and at run time, any object of a reference type is given a value of null unless the object was defined using the new operator. A class is a reference type. Only during the execution of the programme does the new operator allocate memory space to the object, making this kind of allocation a dynamic process.

Syntax :

```
// The className() is a call
// to the constructor
className ObjectName = new className();
```

The constructor can be a default constructor or a user–defined one.

Example :

```
using System;
namespace NewOperator {
class Rectangle {
    public int length, breadth;
    public Rectangle(int l, int b)
    {
        length = l;
        breadth = b;
    }
    public int Area()
    {
        return length * breadth;
    }
}
class Program {
static void Main(string[] args)
{
    Rectangle rect1 = new Rectangle(10, 12);
    int area = rect1.Area();
    Console.WriteLine("The area of the"+" Rectangle is " + area);
}
} }
```

- ***Creating Reference to Existing Object :***

Only the class name and the reference name are required to define a reference to a class. The reference does not allow for this to be possible. It has to be associated with an object of the same class that is already in existence. Any changes that are made to the reference will be stored to the item that it refers to. It is similar to using an alias.

Syntax :

```
className RefName;
RefName = objectName;
```

Example :

```
using System;
namespace Reference {
```

```
class Triangle {
 public int side, altitude;
public double Area()
 {
return (double)0.5 * side * altitude;
}}
class Program {
static void Main(string[] args)
{
Triangle tri1 = new Triangle();
Triangle tri2;
 Console.WriteLine("Area of tri1 is " + tri1.Area());
 tri2 = tri1;
 tri2.side = 5;
 tri2.altitude = 7;
Console.WriteLine("Area of tri1 is " + tri1.Area());
}}}
```

- ***Creating an Array of objects :***

You may generate an array of objects if you require several instances of the same kind of object in a single go. In order to complete this task, you will first need to define the array, and then you will need to initialise each entry, which will be an object. Initialization may be accomplished by the usage of the for loop.

Syntax : className[] arrayName = new className[size];

Example :

```
using System;
namespace ArrayofObjects {
class Circle {
public int radius;
// Defining Constructor
public Circle()
{ radius = 0; }
// Method to set value of radius
public void setValue(int r)
{
radius = r;
}
// Method to calculate the area of the Circle
public double Area()
{
```

```
return (double)3.14 * radius * radius;
}
}
// Driver Class
class Program {
// Main Method
static void Main(string[] args)
{
// To declare an array of two objects
Circle[] circleArray = new Circle[2];
// Initialize the objects
circleArray[0] = new Circle();
circleArray[1] = new Circle();
// to set values for the radius
circleArray[0].setValue(1);
circleArray[1].setValue(2);
// for loop to display areas
for (int i = 0; i < circleArray.Length; i++)
{
Console.Write("Area of circle with radius " + (i + 1));
Console.Write(" is " + circleArray[i].Area() + "\n");
}
}}}
```

2.10 Setting Objects Properties :

In C#, a property is a member of a class that offers a flexible method for classes to expose private fields. Properties may be inherited. On the inside, C# properties are implemented as specialised methods known as accessors. Get property accessor and set property accessor are the two accessors that are associated with a C# property. The value of a property is returned by a get accessor, and a new value is assigned by a set accessor. The value of a property is denoted by the use of the value keyword.

An access modifier is what determines the different access levels that are available for properties in C# and.NET. Read–only, read–write, or write–only access may be granted to properties. Both a get and a set accessor are included into the read–write property's implementation. The only access method that is implemented for a write–only property is the set accessor. The get accessor is all that is implemented for a read–only property; there is no set accessor.

Example :

```
public int Age
{  get { return this.age; }
set {
```

```
If (this.age>65)
this.retired=true;
this.age=value; }}
```

In C#, a property is a member of a class that offers a flexible method for classes to expose private fields. Properties may be inherited. On the inside, C# properties are implemented as specialised methods known as accessors. Get property accessor and set property accessor are the two accessors that are associated with a C# property. The value of a property is returned by a get accessor, and a new value is assigned by a set accessor. The value of a property is denoted by the use of the value keyword.

However, C# has a built–in mechanism known as properties that may accomplish the aforementioned goals. The property declaration syntax is used in C# to declare the various properties of an object. The following is an example of the typical format for defining a property.

```
<acces_modifier> <return_type> <property_name>
{ get { }
   set { } }
```

Whereby <access_modifier> may be either private, public, protected, or internal. The <return_type> variable may be set to any type that is legal in C#. It is important to take note that the first part of the syntax seems to be relatively comparable to a field declaration, and that the second portion consists of a get accessor, a set accessor, and nothing else.

For instance, the programme described above may be altered by adding a property X in the manner shown below.

```
class MyClass
{ private int x;
public int X
{ get { return x; }
set { x = value; }}}
```

The thing that the class is all about the following explains how MyClass can access the X property.

```
MyClass mc = new MyClass();
```

mc.X = 10; / invokes the set accessor of the property X, and passes 10 as the value of the standard field referred to as 'value'. This is how the value of the data member x is set up to be utilized. Console.WriteLine(mc.X);/ / shows the number 10. This method invokes the get accessor of the X property.

- ***Static Properties :***

Static properties, which are properties that belong to the class rather than the objects of the class, are supported as well in C#. Every rule that can be applied to a static member may likewise be applied to static properties. The following piece of code illustrates a class that has a static attribute.

```
class MyClass
{  private static int x;
   public static int X
   {
      get { return x; }
      set { x = value; }
   } }
class MyClient
{  public static void Main()
   {
      MyClass.X = 10;
      int xVal = MyClass.X;
      Console.WriteLine(xVal);//Displays 10
   }}
```

It is important to keep in mind that the set and get accessors of static properties may only interact with other static members of the class. Additionally, static properties are activated by specifying the class name in the invocation.

- ***Properties & Inheritance :***

A Derived class is able to inherit the attributes of a Base class that it derives from.

```
using System;
class Base
{
public int X
{ get {
Console.Write("Base GET");
return 10;
}
set{ Console.Write("Base SET"); } }
}
class Derived : Base
{ }
class MyClient
{
   public static void Main()
   {
      Derived d1 = new Derived();
      d1.X = 10;
      Console.WriteLine(d1.X);
   }
}
```

The software that was just described is fairly easy to understand. The inheritance of properties works in exactly the same way as the inheritance of any other family member.

- ***Properties & Polymorphism :***

A property of a Base class may have its value overwritten by a Derived class using the polymorphic inheritance technique. It is important to keep in mind, however, that modifiers such as virtual and override are used at the property level, not the accessor level.

- ***Abstract Properties :***

By utilising the term "abstract," a property that is included inside a class may be declared to be abstract. Keep in mind that an abstract property in a class has absolutely no associated code with it. A simple semicolon is used to indicate the get/set accessors in the code. It is necessary for us to implement both set and get assessors in the derived class.

If the derived class simply has to implement set, then the set accessor is all that needs to be included in the abstract class.

The properties are fundamental additions to the C# programming language that were made at the level of the language. They are of great assistance in graphical user interface development. It is important to keep in mind that when the C# property syntax is parsed, the compiler actually constructs the relevant getter and setter methods for each property.

- ***Properties Access Modifiers :***

The access level of a property may be defined by access modifiers, which determine whether a property can be accessed by any caller programme, just inside an assembly, or only within a class. The access level modifications are summarised in the table that follows.

Access Modifiers

Access Modifier	Description
Public	Any other piece of code included inside the same assembly or within another assembly that references it is able to access the type or member.
Private	Only other pieces of code inside the same class or struct will be able to access the type or member.
Protected	The code from inside the same class, or from a class that is derived from that class, is the only way that the type or member may be accessed.
Internal	Any piece of code included within the same assembly has access to the type or member, but code contained inside another assembly does not.
Protected Internal	Any piece of code included inside the assembly in which the type or member was defined, or from within a derived class contained within an entirely different assembly, is able to access it.
Private Protected	Access to the type or member is only possible from inside the assembly that declared it, either via the code of an instance of the same class or of a type that is derived from that class.

- ***Automatically Implemented Properties :***

Listing is an example of how a public property would typically be implemented in code. The getter and setter for a property are required in the default implementation of the property.

```
private string name;
public string Name
{
   get { return this.name; }
   set { this.name = value; }
}
```

If there is no need for further computation, having auto–implemented properties in C# makes the code more understandable and keeps it cleaner. The code that is shown above in Listing may be replaced by the one line of code that is shown below in Listing.

```
public string Name { get; set; }
```

When auto–implemented properties are used, the compiler constructs a private field variable that can only be accessed using the getter and setter methods associated with the property.

A new instance of the class is created by the code in Listing, as well as calls being made to its methods and attributes.

```
class Program
{ static void Main()
{ Author author = new Author("Mahesh Chand", "Apress", "Programming
C#", 2003, 49.95);
   Console.WriteLine(author.AuthorDetails());
   Console.WriteLine("Author published his book in {0}", author.Year);
   author.Price = 50;
   Console.WriteLine(author.CostOfThousandBooks().ToString());
   Console.ReadKey();}}
```

2.11 Object Methods :

When we wrote anything in Visual Studio, we observed that everything appeared to act as if it were an object. This was something that we noticed immediately. To clarify further, consider the following scenario: we have just written a literal number, and after we have typed a dot, the ToString() function miraculously becomes available to us (.).

The System.Object class serves as the root from which all other.NET classes are eventually derived. In point of fact, if you don't tell the compiler which base class a class derives from while you're creating it, the compiler will presume that it comes from System.Object by default. Last but not least, as a result of this behaviour, you are granted access to a large number of public and protected member methods that have been specified for the Object class.

In order to demonstrate some of the claims made in the previous paragraphs, I have developed a unit test that demonstrates the following :

- System.Object class serves as the root of all other classes in the.NET Framework.
- Every built–in class and every specified class is a subclass of the System.Object class.

The conclusion that we have just reached is that everything originates from the System.Object class. First things first, before we go into the many attributes and methods that we may directly inherit from the Object class, let's see if we can find a solution to this question: Why is it necessary for all.NET types to originate from the object/System.Object class?

- It provides the developer with a behaviour that is applicable to all of the object class's methods and attributes. Because of this, it is compatible with all versions of the .NET core and foundation.
- Those who work in software development have the capacity to "pass anything" as an object.
- It is possible for developers to make references to objects of an unknown type, particularly when reflection is used.

The above–mentioned list was compiled based on our expertise and observations with the .NET core and framework from the very beginning.

Object Methods in C#

Methods	Description
Equals(Object)	Checks to see whether the provided object and the current object have the same properties. **Example :** public virtual bool Equals (object? obj);
Equals (Object, Object)	Determines if the object instances that have been supplied are deemed to be the same. **Example :** public static bool Equals (object? objA, object? objB);
Finalize()	Gives an item the opportunity, before being recovered by trash collection, to attempt to liberate resources and carry out additional cleaning actions. **Example :** ~Object ();
GetHashCode()	Performs the function of the hash function by default. **Example :** public virtual int GetHashCode ();
GetType()	Retrieves the instance type currently being used. **Example :** public Type GetType ();
MemberwiseClone()	Produces a simplified replica of the object that is now active. **Example :** protected object MemberwiseClone ();
ReferenceEquals (Object, Object)	Determines whether or not the Object instances that have been supplied belong to the same instance. **Example :** public static bool ReferenceEquals (object? objA, object? objB);
ToString()	This function will return a string that contains a representation of the current object. **Example :** public virtual string? ToString ();

- **Object.Equals Method :** Checks to see whether the provided object and the current object have the same properties.

  Syntax : public virtual bool Equals (object? obj);

The obj parameter is the object that will be compared with the one that is currently being used. It returns the boolean type result. If the provided object and the current object are the same, this value will be true; otherwise, it will return false.

There are a few programming languages that enable operator overloading, including C# and Visual Basic. In order to offer the same functionality as the original method, a type that overloads the equality operator is required to also override the Equals(Object) method.

It is not possible to derive Complex from any other type since it is a value type. Therefore, the override to the Equals(Object) method does not have to call GetType to determine the precise run–time type of each object; rather, it can use the is operator in C# or the TypeOf operator in Visual Basic to check the type of the obj parameter. GetType is a method that determines the precise run–time type of each object.

- **Equals(Object, Object) Method :** Determines if the object instances that have been supplied are deemed to be the same.

  Syntax : public static bool Equals (object? objA, object? objB);

The objA parameter is the first object to compare and the objB parameter is the second object to compare. It returns the boolean type result. If the items are regarded equal, then the value will be true; otherwise, it will be false. If objA and objB are both empty, then the procedure will return the value true.

- **Object.Finalize Method :** Gives an item the opportunity, before being recovered by trash collection, to attempt to liberate resources and carry out additional cleaning actions.

  Syntax : ~Object ();

Example : When an object that overrides the Finalize method is destroyed, the following example checks to see whether the Finalize method is invoked as expected. It is important to keep in mind that the Finalize function would need to be changed in a production application in order to free up any unmanaged resources that the object was holding.

- **Object.GetHashCode Method :** Performs the function of the hash function by default.

  Syntax : public virtual int GetHashCode ();

It returns the int32 type result. A hash code for the item that is currently being examined.

It is common for a type to have many data fields that are eligible to take part in the generation of the hash code. Combining these fields via the use of an XOR (exclusive OR) operation is one method for producing a hash code.

- **Object.GetType Method :** Retrieves the instance type currently being used.

  Syntax : public Type GetType ();

It returns the precise runtime type of the instance that is currently being used.

Example : The following snippet of code illustrates how to use the GetType method, which returns the runtime type of the currently active instance.

```
public static void Main()
{
    MyBaseClass myBase = new MyBaseClass();
    MyDerivedClass myDerived = new MyDerivedClass();
    object o = myDerived;
    MyBaseClass b = myDerived;
    Console.WriteLine("mybase: Type is {0}", myBase.GetType());
    Console.WriteLine("myDerived: Type is {0}", myDerived.GetType());
    Console.WriteLine("object o = myDerived: Type is {0}", o.GetType());
    Console.WriteLine("MyBaseClass b = myDerived: Type is {0}",
    b.GetType());
}
```

When Object.ReferenceEquals(x.GetType(),y.GetType()) is called on two objects, x and y, and they both have the same runtime type, it returns the value true.

- **Object.MemberwiseClone Method :** Produces a simplified replica of the object that is now active.

  Syntax : protected object MemberwiseClone ();

  It returns the object. A clone that is just slightly deeper than the existing Object.

- **Object.ReferenceEquals(Object, Object) Method :** Determines whether or not the Object instances that have been supplied belong to the same instance.

  Syntax : public static bool ReferenceEquals (object? objA, object? objB);

  The objA parameter is the first object to compare and the objB parameter is the second object to compare. It returns the boolean type result. If the items are regarded equal, then the value will be true; otherwise, it will be false. If objA and objB are both empty, then the procedure will return the value true.

  It is not possible to modify the ReferenceEquals method in any way, in contrast to the Equals method and the equality operator. Because of this, you may use the ReferenceEquals method whenever you want to test two object references for equality and you are unclear about the implementation of the Equals method. In other words, you can call ReferenceEquals whenever you wish to compare two object references.

  On the other hand, the return result of the ReferenceEquals function could give the impression of being strange in the following two scenarios:

  o When comparing different sorts of values. Before being sent to the ReferenceEquals function, value types objA and objB are checked to see whether or not they need to be boxed. This implies that even if objA and objB both represent the same instance of a value type, the ReferenceEquals function will still return a false result.

- While comparing different string formats. When both objA and objB are strings, the ReferenceEquals function checks to see whether the string is interned and returns true if it is. It does not conduct a test to determine whether or not the values are equal.

- **Object.ToString Method :** This function will return a string that contains a representation of the current object.

  Syntax : public virtual string? ToString ();

It returns the string type result. A string that represents the item that is currently being worked with. The most important function for formatting in the.NET Framework is called Object.ToString. It will transform an object into the string representation of the object so that it may be shown properly.

The Object.ToString function is commonly overridden by types in order for those types to give a string representation that is better appropriate for that type. Additionally, the Object is typically overloaded by types. The ToString function is used to offer support for culture–sensitive formatting as well as format strings.

- **The default Object.ToString() method :** The default implementation of the ToString function delivers the name of the type of the Object together with all of its qualification information.

  Example :

  ```
  Object obj = new Object();
  Console.WriteLine(obj.ToString());
  ```

  This behaviour is inherited by reference types within the.NET Framework that do not override the ToString function. This is due to the fact that Object is the base class of all reference types within the.NET Framework.

- **Overriding the Object.ToString() method :** The Object.ToString function is often overridden by types in order to produce a string that accurately describes the object instance. For instance, the ToString implementations that are provided by the basic types such as Char, Int32, and String return the string representation of the value that the object represents. These base types include :

- **Overloading the ToString method :** Many types override the ToString function, providing variants of the method that accept arguments in addition to overriding the Object.ToString() method, which does not take any parameters and returns the string. This is done for a variety of reasons, but the most prevalent one is to give support for culture–sensitive formatting and variable formatting.

- **Extending the Object.ToString method :** Because a type inherits the Object.ToString function as its default, you can find the behaviour of that type to be unsatisfactory and wish to modify it. This is especially the case with arrays and other types of collection classes. The ToString function of an array or collection class does not, contrary to what you would anticipate, display the values of the array's or collection's members.

You may construct the output string that you want using one of many different approaches. Firstly, You may enumerate the elements of a type using the foreach statement in C# or the For Each Next statement if the type is an array, a collection object, or an object that implements the IEnumerable or IEnumerable<T> interfaces. If this is the case, you can enumerate the type's

elements. Secondly, You may construct a wrapper class that inherits from the base class whose Object.ToString method you wish to change if the class is not sealed in C# or NotInheritable in Visual Basic. This will allow you to edit the Object.ToString method of the base class. At a bare minimum, this calls on you to carry out the following tasks :

- Constructors, if any are required, should be implemented. Constructors that are specific to the base class are not passed down to derived classes.
- You may return the desired result string by overriding the Object.ToString function and passing it your own string.

The following example constructs a wrapper class for the List<T> class. It does this by overriding the Object.ToString function such that, rather than displaying the fully qualified type name, it displays the value of each method in the collection.

Lastly, build an extension method that will give you the desired result string and return it to you. Take note that you are unable to change the behaviour of the default Object. ToString method in this fashion (that is, your extension class (in C#) or module (in Visual Basic) cannot contain a parameterless method named ToString that is called in favor of the ToString method that was included with the original type. You will be required to provide a different name for the parameterless ToString replacement you want to use.

2.12 Event Handlers :

A mouse click, a key press, mouse motions, or any system–generated notification may all be considered events since they are actions taken or occurrences that take place. Events are the means through which a process conveys information. For instance, interruptions are events that are created by the system. The programme need to be able to react to events as they occur and take appropriate action in response.

The client computer is responsible for generating ASP.NET events, while the server machine is the one that processes them. For instance, the user may interact with the browser by clicking a button that is presented there. There is a raise of a Click event. This client–side event is managed by the browser, which handles it by publishing it to the server.

The event–handler is a subroutine that is stored on the server and is responsible for specifying what should be done when an event is triggered. As a result, whenever the event message is sent to the server, it examines the state of the Click event to determine whether or not it has an associated event handler. In the case that it has, the event handler will be carried out.

- **Event Arguments :** Event handlers written in ASP.NET typically accept two arguments and do not return anything. The first parameter identifies the object that caused the event to occur, while the second parameter specifies the argument for the event.

  Syntax : private void EventName (object sender, EventArgs e);

- **Application and Session Events :** The following are the most significant application events:
  - **Application_Start :** It is triggered whenever the programme or webpage is opened for the first time.
  - **Application_End :** It becomes active once the application or website has been closed.

In a similar manner, the most common Session events are as follows :

- **Session_Start :** It is triggered whenever a user accesses a page from the application for the very first time.
- **Session_End :** It is inactive until the application or website is closed, at which point it becomes active.

- **Page and Control Events :** The following are examples of common page and control events:
  - **DataBinding :** It is triggered whenever a control establishes a connection to a data source.
  - **Disposed :** It is triggered once either the page or the control is made available for use.
  - **Error :** When an unhandled exception is thrown, a page event will take place at that location.
  - **Init :** When either the page or the control is first loaded, it will trigger.
  - **Load :** It is triggered whenever either the page itself or a control on the page is loaded.
  - **PreRender :** The signal is sent out once the page or control is about to be rendered.
  - **Unload :** When the page or control is unloaded from memory, it triggers the raising of this event.
- **Event Handling Using Controls :** Every ASP.NET control is built as a class, and each of those classes has events that are triggered whenever a user carries out a certain action on the control. For instance, the 'Click' event is triggered whenever a user hits a button on the page. There are in–built characteristics and event handlers available for managing event processing. The purpose of the code that makes up an event handler is to react to events and execute the required actions in response to them.

Visual Studio will automatically generate an event handler for you by putting a Handles clause on the Sub procedure you build. Within this clause, the control and event that the procedure is responsible for handling are named.

It is also possible to write an event without the Handles clause. The following are the typical control events:

Control Events with Attributes

Event	Attribute	Controls
Click	OnClick	Button, image button, link button, image map
Command	OnCommand	Button, image button, link button
TextChanged	OnTextChanged	Text box
SelectedIndexChanged	OnSelectedIndexChanged	Drop-down list, list box, radio button list, check box list.
CheckedChanged	OnCheckedChanged	Check box, radio button

These kinds of events are referred to as "postback events," and they are the ones that cause the form to instantly be sent back to the server. Take, for instance, the click event that occurs when a button.Click is clicked.

There are a few events that do not get reported back to the server straight away; we refer to them as non–postback events. Take, for instance, the change events or the selection events, like the TextBox.TextChanged or the checkbox.CheckedChanged may be changed. By setting the true value of the AutoPostBack attribute on the nonpostback events, it was possible to force them to post back instantly.

- **Default Events :** The default event for the Page object is Load event. In a similar vein, each control possesses its own default event. For instance, the Click event is the one that is set as the default for the button control.

Simply double clicking the control while in design view of Visual Studio would result in the creation of the default event handler for the application. The table that follows provides an overview of some of the events that are often associated with popular controls :

Default Events for Common Controls

Control	Default Events
AdRotator	AdCreated
BulletedList	Click
Button	Click
Calender	SelectionChanged
CheckBox	CheckedChanged
CheckBoxList	SelectedIndexChanged
DataGrid	SelectedIndexChanged
DataList	SelectedIndexChanged
DropDownList	SelectedIndexChanged
HyperLink	Click
ImageButton	Click
ImageMap	Click
LinkButton	Click
ListBox	SelectedIndexChanged
Menu	MenuItemClick
RadioButton	CheckedChanged
RadioButtonList	SelectedIndexChanged

Example : This example has a simple page that contains a label control as well as a button control on its surface. It transmits a message that is shown by the label control at the same time as page events such as Page Load, Page Init, and Page PreRender, among others, take place on the page. The Button Click event is triggered whenever the button is pressed, and it also causes a message to be sent to the label so that it may be shown.

After you have created a new website, go to the control tool box and drag a label control and a button control onto it, respectively. Make sure that the IDs of the controls are set to .lblmessage. and .btnclick. correspondingly by using the properties window. "Click" should be entered into the Text attribute of the Button control.

The markup.aspx file :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind=
"Default.aspx.cs"
    Inherits="eventdemo._Default" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
    <head runat="server">
        <title>Untitled Page</title>
    </head>
    <body>
        <form id="form1" runat="server">
            <div>
                <asp:Label ID="lblmessage" runat="server" >
                </asp:Label>
                <br />
                <br />
                <br />
                <asp:Button ID="btnclick" runat="server" Text="Click"
                onclick="btnclick_Click" />
                </div>
            </form>
        </body>
    </html>
```

To navigate to the file containing the code underlying the design, just double click on the design view. There is no need to include any code in the Page Load event since it is produced automatically. Please jot down the lines of code that explain themselves below :

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
```

```
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

namespace eventdemo {
    public partial class _Default : System.Web.UI.Page {
        protected void Page_Load(object sender, EventArgs e) {
            lblmessage.Text += "Page load event handled. <br />";
            if (Page.IsPostBack) {
                lblmessage.Text += "Page post back event handled.<br/>";
            }
        }
        protected void Page_Init(object sender, EventArgs e) {
            lblmessage.Text += "Page initialization event handled.<br/>";
        }
        protected void Page_PreRender(object sender, EventArgs e) {
            lblmessage.Text += "Page prerender event handled. <br/>";
        }
        protected void btnclick_Click(object sender, EventArgs e) {
            lblmessage.Text += "Button click event handled. <br/>";
        }
    }
}
```

Proceed with the page. The label provides information on the events of page load, page startup, and page pre–rendering. To see the effect, please click the button :

2.13 Let Us Sum Up :

This unit taught us that creating.net applications is easy with a coding technique. Variables are application data that can vary. Declare a variable first.

C# lets you run code based on circumstances that may be verified using an if statement that returns true or false. Starting with the while loop is easiest. The while loop runs a block of code if the condition is true.

Developers write input–accepting functions/subroutines initially. The application will execute the function and return the output to the browser. ASP.NET has three web application frameworks: Web Forms, MVC, and Web Pages.

All three frameworks can produce fantastic web apps. ASP.NET has three web application frameworks: Web Forms, MVC, and Web Pages.

Objects simplify repetitions. Objects are functional and informative. ASP has a few Intrinsic Objects. The Application property or HttpApplicationState object contains application–wide data.

It is a shared variable among the users and is an easy–to–use area to store global information that will be needed across your application. All classes stem from Object. It has some functions. Object base types reference all derived types. Class events notify clients of events. Delegates announce events. Event handler prints "event raised" and event number.

2.14 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : a **2 :** c

- **Check Your Progress 2 :**

  1 : b **2 :** d

- **Check Your Progress 3 :**

  1 : b

- **Check Your Progress 4 :**

  1 : c

- **Check Your Progress 5 :**

  1 : b

- **Check Your Progress 6 :**

  1 : a

2.15 Glossary :

1. **Abstract :** It exists only as an idea, not as a physical thing.
2. **Virtual :** It made to appear to exist by computer.
3. **Operator :** An operator is a character that represents a specific mathematical or logical action or process.
4. **Loop :** a loop is a sequence of instruction s that is continually repeated until a certain condition is reached.
5. **Condition :** Conditions are statements that are created by the programmer which evaluates actions in the program and evaluates if it's true or false.
6. **Testing :** Testing is the process of executing a program to find errors.

2.16 Assignment :

1. Explain different types of Loops in detail.

2.17 Activities :

1. What are the types of operators in ASP.NET ?

2.18 Case Study :

Study about the various testing techniques.

2.19 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

Unit
03

STUDYING WEB CONTROLS

UNIT STRUCTURE

3.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About the controls which are the basic elements that make up the graphical user interface (GUI).
- How users are able to input data, make decisions, and express their preferences while using these controls

- About the Control class that implements very basic functionality required by classes that display information to the user.

3.1 Introduction :

Everything that you can see in a graphical user interface (GUI) programme that runs on Windows.NET is a Control, and it derives from the Control class. A class that is used to represent control behaviour that is particular to one or more use cases is referred to as a Control Class. Use–case–specific behaviours may be encapsulated inside control classes. The execution of a particular use case is intimately connected to the manner in which a control object behaves.

A component that was created with the help of the.NET Framework and runs in the managed environment of the .NET Common Language Runtime is known as a.NET control (CLR). The user interface functionality of a client–side Windows–based programme is often encapsulated inside a.NET control in its most common form.

Rich functionality may be added to your FORMs or pages by using web controls. Web controls are essentially HTML components that are encased in the simple scripting tags of ASP+. Controls on the web may be as basic as text boxes or as complex as grids and lists.

3.2 Button Control :

A user may interact with an item known as a button to accomplish a certain action, such as clicking the OK button or the Cancel button on a dialogue box. Because it corresponds to a single instruction that the user desires to carry out, the button control is an easy control to disclose. ASP.NET provides three types of button control:

Types of Button Control

Types	Description	Syntax	Example
Button	Displaying a push button requires the usage of the control known as the Button. It's possible that the push button is a command button or a submit button.	<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Click" / >	<asp:Button ID="btnSubmit" runat="server" onclick="btnSubmit_Click" Text="Submit" / >
Link Button	It is a control for the server's web page that functions as a hyperlink. On the web page, it is shown as a button control that has the appearance of a hyperlink. There is a tag available in ASP.NET that may be used to construct LinkButtons.	<asp:LinkButton ID="LinkButton1" runat="server" OnClick="LinkButton 1_Click"> </asp:LinkButton>	<asp:LinkButton ID="lbtnASPNET" runat="server" OnClick="lbtnASPNET_ Click">Web Controls</asp:LinkButton>
Image Button	An ImageButton is an AbsoluteLayout which enables you to specify the exact location of its children. This shows a button with an image (instead of text) that can be pressed or clicked by the user.	<asp:ImageButton ID="ImageButton1" runat="server" ImageUrl=" " Height="64px" Width="158px" Imagealign="Left" PostBackUrl="" />	<asp:ImageButton ID="imgbtnBAOU" runat="server" ImageUrl="~\BAOU.png" Height="64px" Width="158px" Imagealign="Left" PostBackUrl="https:/ www.baou.edu.in/" />

Displaying a push button requires the usage of the control known as the Button. It's possible that the push button is a command button or a submit button. This control has been set up as a submit button by default.

When a user clicks on a submit button, the Web page is sent back to the server. This button does not have a command name associated with it. When the submit button is pressed, certain activities will be carried out, and you may control those actions by writing an event handler.

You are able to establish several button controls on a page using a command button, which also comes with its own command name. When the command button is pressed, certain activities will be carried out, and you may control those actions by writing an event handler.

Properties of Button Control

Property	Description
CausesValidation	When a Button control is clicked, specifies whether or not a page should be verified.
CommandArgument	When a user clicks the button, a string value that is associated with that click is sent to the command event.
CommandName	When a user clicks the button, a string value that is associated with that click is sent to the command event.
OnClientClick	When a button is clicked, this is the name of the function that will be carried out.
PostBackUrl	When the Button control is clicked, the URL of the page to which the user will be posted from the current page.
runat	Identifies the control as one that is hosted on the server. Must be set to "server"
Text	The text that is shown when the button is selected. Only controls for buttons and link buttons will be shown here.
UseSubmitBehavior	A value that indicates whether the Button control makes use of the submit mechanism of the client browser or the postback technique made available by ASP.NET.
ValidationGroup	The collection of controls for which the validation is performed when it is sent back to the server by the Button control.
AccessKey	It is used to configure the shortcut key for the control on the keyboard.
TabIndex	The order of the tabs in the control.
BackColor	It may be used to change the colour of the control's background.
BorderColor	It may be used to change the colour of the control's border.
BorderWidth	It is used to specify the width of the control's border, therefore it is important.
Font	It is used for the purpose of setting the font for the control text.

ForeColor	The colour of the control text may be changed using this option.
ToolTip	When the mouse is hovered over the control, the text is shown.
Visible	To determine whether or not a control will be shown on the form.
Height	The height of the control may be adjusted using this control.
Width	The width of the control may be adjusted using this.

Few other control standard properties for the same are : AppRelativeTemplateSourceDirectory, BindingContainer, ClientID, Controls, EnableTheming, EnableViewState, ID, NamingContainer, Page, Parent, Site, TemplateControl, TemplateSourceDirectory, UniqueID and Visible.

Example : WebControls.aspx page

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Web Controls.aspx.cs" Inherits="WebFormsControlls.WebControls" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="Button1" runat="server" Text="Click here" OnClick="Button1_Click" />
        </div>
    </form>
    <br />
    <asp:Label ID="Label1" runat="server"></asp:Label>
</body>
</html>
```

CodeBehind Code : WebControl.aspx.cs page

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebFormsControlls
{
```

```
public partial class WebControls : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = "You Clicked the Button.";
    }
}
}
```

3.3 Literal Control :

Displaying text on a page requires the usage of the Literal control. The text may be used in programming. Please take a note that you cannot add styles to the content of this control.

A web page may use either the Label Control or the Literal Control to show static text. The Literal Control is quite similar to the Label Control. There is no inheritance of properties from the WebControl namespace into the Literal Control. Literal text may be programmed, despite the fact that the Literal Control does not give any major capability. The web page will not get any new HTML components as a result of this action. Because of this feature, it is now able to write HTML code right in the code designer window, rather than first having to go to design view and then click the HTML button in order to change the HTML. A literal control cannot have a style applied to it in any way. In contrast to the Label control, the Literal control does not have any properties such as BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, or Height, amongst others. Because of this, its strength is increased, and you may even insert HTML code directly into its body.

Control	Description	Syntax	Example
Literal	The Literal control is used to display text on a page. The text is programmable. This control does not let you apply styles to its content.	<asp:Literal ID="Literal1" runat="server" Text=" "> </ asp:Literal>	<asp:Literal ID="LiteralText" runat="server" Text="This is example of Literal"></ asp:Literal>

Properties of Literal Control

Property		Description
Mode	PassThrough	There is no change made to the information included in the control.
	Encode	The contents of the control are encoded using HTML, and this string is then output.
	Transform	The contents of the control have had any markup-language components that are not supported removed. The contents of the Literal control are not changed when it is displayed on a browser that supports HTML or XHTML. This is the case for all other browsers.
runat		Identifies the control as one that is hosted on the server. It is required to be set to "server."
Text		Identifies the text that will be shown.

Example : ShowLiteral.aspx page

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
   <title><asp:Literal id="ltlTitle" Runat="Server" /></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <h1>Look in the title bar</h1>
      </div>
   </form>
</body>
</html>
```

CodeBehind Code : WebControl.aspx.cs page

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebFormsControlls
{
   void Page_Load()
   {
      ltlTitle.Text = DateTime.Now.ToString("D");
   }
}
```

Example : ShowLiteralMode.aspx page

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
   <title>Show Literal Mode</title>
</head>
<body>
```

```
<form id="form1" runat="server">
<div>
<asp:Literal
    id="ltlFirst"
    Mode="PassThrough"
    Text="<hr />"
    Runat="server" />
<br /><br />
<asp:Literal
    id="ltlSecond"
    Mode="Encode"
    Text="<hr />"
    Runat="server" />
    <br /><br />
    <asp:Literal
    id="ltlThird"
    Mode="Transform"
    Text="<hr />"
    Runat="server" />
    </div>
</form>
</body>
</html>
```

3.4 Label Control :

Label controls make it simple to show text that can be altered from one iteration of a page to the next, and they provide this benefit to users. You should utilise the literal text in situations when you wish to show text that does not alter.

Control	Description	Syntax	Example
Label	Text may be displayed on a page with the usage of the Label control. The text is possible to be programmed.	<asp:LabelID ="Label1" runat="server" Text="Label"> </asp:Label>	<asp:Label ID="lblUserName" runat="server" Text="User Name"></asp:Label>

Properties of Label Control

Property	Description
runat	Identifies the control as one that is hosted on the server. It is required to be set to "server."
Text	The content that will be shown in the label.
AccessKey	It is used to set keyboard shortcut for the label.

TabIndex	The order of the tabs in the control.
BackColor	It may be used to change the colour of the control's background.
BorderColor	It may be used to change the colour of the control's border.
BorderWidth	It is used to specify the width of the control's border, therefore it is important.
Font	It is used for the purpose of setting the font for the control text.
ForeColor	The colour of the control text may be changed using this option.
ToolTip	When the mouse is hovered over the control, the text is shown.
Visible	To determine whether or not a control will be shown on the form.
Height	The height of the control may be adjusted using this control.
Width	The width of the control may be adjusted using this.

Example : ShowLabel.aspx page

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">
<script runat="server">
    void Page_Load()
    {
        lblTime.Text = DateTime.Now.ToString("T");
    }
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Show Label</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <asp:Label id="lblTime" Runat="server" />
    </div>
    </form>
</body>
</html>
```

3.5 TextBox Control :

This is a control known as a TextBox control, and it is used to receive input from users. We can either write code or use the drag and drop feature of the visual studio IDE in order to construct a TextBox. Both options are available to us. This is a server–side control, and ASP supplies its own tag so that it can be created. The example is provided further down.

Control	Description	Syntax	Example
TextBox	TextBox controls are utilized so that the user can provide their input. Depending on how the TextMode attribute is configured, a text box control may be made to accept either one line of text or multiple lines of text.	<asp:TextBox ID="" runat="server"> </asp:TextBox>	<asp:TextBox ID="txtstate" runat="server" Text="State"> </asp:TextBox>

Properties of TextBox Control

Property	Description
runat	Identifies the control as one that is hosted on the server. It is required to be set to "server."
Text	The content that will be shown in the label.
AccessKey	It is used to set keyboard shortcut for the label.
TabIndex	The order of the tabs in the control.
BackColor	It may be used to change the colour of the control's background.
BorderColor	It may be used to change the colour of the control's border.
BorderWidth	It is used to specify the width of the control's border, therefore it is important.
Font	It is used for the purpose of setting the font for the control text.
ForeColor	The colour of the control text may be changed using this option.
ToolTip	When the mouse is hovered over the control, the text is shown.
Visible	To determine whether or not a control will be shown on the form.
Height	The height of the control may be adjusted using this control.
Width	The width of the control may be adjusted using this.

Example : ShowTextBox.aspx page

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">
<script runat="server">
    void Page_Load()
    { txtState.Text = "Rajasthan"; }
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Show TextBox</title>
</head>
<body>
```

```
<form id="form1" runat="server">
<div> <asp:TextBox ID="txtstate" runat="server"> </asp:TextBox> </
div>
</form>
</body>
</html>
```

3.6 PlaceHolder Control :

The ASP.NET PlaceHolder Control is not standard. It has no user interface and client–side HTML. PlaceHolder is hidden from website visitors. It's one of the most important Visual Studio controls for creating dynamic user interfaces. PlaceHolder holds other controls or HTML, JavaScript, plain text, etc. It's possible to add controls directly to Page objects, but this is rarely done because it's difficult to manage when and how they appear at runtime.

Control	Description	Syntax	Example
PlaceHolder	PlaceHolder holds other controls or HTML, JavaScript, plain text, etc	<asp:PlaceHolder ID="PlaceHolder1" runat="server"> </asp:PlaceHolder>	<asp:PlaceHolder ID = "PlaceHolder1" runat="server"> Your name: <asp:TextBox runat="server" ID="txtName" /> E-mail: <asp:TextBox runat="server" ID="txtEmail" /> <asp:Button runat="server" ID="btnSubmit" Text="Submit" OnClick="btnSubmit_Click" /> </asp:PlaceHolder>

Properties of PlaceHolder Control

Property	Description
Adapter	Gets the browser–specific adapter for the control.
AppRelativeTemplate SourceDirectory	Gets or sets the application–relative virtual directory of the Page or UserControl object that contains this control.
BindingContainer	Gets the control that contains this control's data binding.
ChildControlsCreated	Gets a value that indicates whether the server control's child controls have been created.
ClientID	Gets the control ID for HTML markup that is generated by ASP.NET.
ClientIDMode	Gets or sets the algorithm that is used to generate the value of the ClientID property.
ClientIDSeparator	Gets a character value representing the separator character used in the ClientID property.
Context	Gets the HttpContext object associated with the server control for the current Web request.
Controls	Gets a ControlCollection object that represents the child controls for a specified server control in the UI hierarchy.

DataItemContainer	Gets a reference to the naming container if the naming container implements IDataItemContainer.
DataKeysContainer	Gets a reference to the naming container if the naming container implements IDataKeysControl.
DesignMode	Gets a value indicating whether a control is being used on a design surface.
EnableTheming	Gets or sets a value indicating whether themes apply to this control.
EnableViewState	Gets or sets a value indicating whether the server control persists its view state, and the view state of any child controls it contains, to the requesting client.
Events	Gets a list of event handler delegates for the control. This property is read–only.
HasChildViewState	Gets a value indicating whether the current server control's child controls have any saved view–state settings.
ID	Gets or sets the programmatic identifier assigned to the server control.
IdSeparator	Gets the character used to separate control identifiers.
IsChildControlState Cleared	Gets a value indicating whether controls contained within this control have control state.
IsTrackingViewState	Gets a value that indicates whether the server control is saving changes to its view state.
IsViewStateEnabled	Gets a value indicating whether view state is enabled for this control.
LoadViewStateByID	Gets a value indicating whether the control participates in loading its view state by ID instead of index.
NamingContainer	Gets a reference to the server control's naming container, which creates a unique namespace for differentiating between server controls with the same ID property value.
Page	Gets a reference to the Page instance that contains the server control.
Parent	Gets a reference to the server control's parent control in the page control hierarchy.
Rendering Compatibility	Gets a value that specifies the ASP.NET version that rendered HTML will be compatible with.
Site	Gets information about the container that hosts the current control when rendered on a design surface.
SkinID	Gets or sets the skin to apply to the control.
TemplateControl	Gets or sets a reference to the template that contains this control.

TemplateSource Directory	Gets the virtual directory of the Page or UserControl that contains the current server control.
UniqueID	Gets the unique, hierarchically qualified identifier for the server control.
ValidateRequestMode	Gets or sets a value that indicates whether the control checks client input from the browser for potentially dangerous values.
ViewState	Gets a dictionary of state information that allows you to save and restore the view state of a server control across multiple requests for the same page.
ViewStateIgnoresCase	Gets a value that indicates whether the StateBag object is case–insensitive.
ViewStateMode	Gets or sets the view–state mode of this control
Visible	Gets or sets a value that indicates whether a server control is rendered as UI on the page.

Example : ShowPlaceHolder.aspx page

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class PlaceHolderCaspx : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
PlaceHolder1.Controls.Add(new LiteralControl(@"<b>Sign up for our newsletter</b><br />
Your name:<br />"));
TextBox txtName = new TextBox();
txtName.ID = "txtName";
PlaceHolder1.Controls.Add(txtName);
PlaceHolder1.Controls.Add(new LiteralControl("<br />Email:<br />"));
TextBox txtEmail = new TextBox();
txtEmail.ID = "txtEmail";
PlaceHolder1.Controls.Add(txtEmail);
PlaceHolder1.Controls.Add(new LiteralControl("<br />"));
Button btnSubmit = new Button();
btnSubmit.ID = "btnSubmit";
btnSubmit.Text = "Submit";
btnSubmit.Click += new EventHandler(btnSubmit_Click);
PlaceHolder1.Controls.Add(btnSubmit);
}
void btnSubmit_Click(object sender, EventArgs e)
```

```
{
// do something...
}
}
```

❑ **Check Your Progress – 1 :**

1. _____ property Identifies the control as one that is hosted on the server. It is required to be set to "server."

   a. Viewstate b. Visible c. Runat d. None of these

2. _____ holds other controls or HTML, JavaScript, plain text, etc.

   a. Literal b. Label c. Textbox d. PlaceHolder

3.7 HiddenField Control :

HiddenField, as name says, is hidden. This is an ASP.NET control that does not display on the screen and allows you to save the value. This is an example of a client–side state management tool of a particular sort. It remembers the value between the two iterations of the roundtrip. Simply reading the document's source code will allow anyone to see the specifics of a HiddenField.

HiddenFields are not encrypted or protected in any other way, therefore anyone has the ability to modify them. On the other hand, this is not something that is recommended from a safety standpoint. The HiddenField control is utilised by ASP.NET for the purpose of controlling the ViewState. Therefore, you shouldn't save any sensitive or significant data with this option, such as your password or the details of your credit card.

Control	Description	Syntax	Example
HiddenField	It is an ASP.NET control that does not display on the screen and allows you to save the value.	<asp:HiddenField ID="HiddenField1" runat="server" />	<asp:HiddenField ID="hfCurrentTime" runat="server" />

Properties of HiddenField Control

Property	Description
EnableTheming	Gets or sets a value indicating whether themes apply to this control.
SkinID	Gets or sets the skin to apply to the control.
Value	Gets or sets the value of the hidden field.

Example : ShowHiddenField.aspx page

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">
<script runat="server">
    void Page_Load()
    {
```

```
        hfCurrentTime.Value = DateTime.Now.ToString();
        lblCurrentTime.Text = Convert.ToString(hfCurrentTime.Value);
    }
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Show HiddenField</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:HiddenField ID="hfCurrentTime" runat="server" />
        <asp:Label ID="lblCurrentTime" runat="server" Text=""/>
    </div>
    </form>
</body>
</html>
```

3.8 FileUpload Control :

The FileUpload control provides the user with a browse button and a text box in which they can enter the filename while also enabling the user to search for and pick the file that is going to be uploaded.

After the user has browsed for the filename or typed it into the text box, the SaveAs method of the FileUpload control can be used to save the file to the disc. This can be done after the user has entered the filename in the text box.

Control	Description	Syntax	Example
FileUpload	It provides the user with a browse button and a text box in which they can enter the filename while also enabling the user to search for and pick the file that is going to be uploaded.	<asp:FileUpload ID= "FileUpload1" runat = "server" />	<asp:FileUpload ID = "fuUploader" runat = "server" />

Read–Only Properties of FileUpload Control

Property	Description
FileBytes	Returns an array of the bytes in a file to be uploaded.
FileContent	Returns the stream object pointing to the file to be uploaded.
FileName	Returns the name of the file to be uploaded.
HasFile	Specifies whether the control has a file to upload.
PostedFile	Returns a reference to the uploaded file.

HTTPPosted Properties of FileUpload Control

Property	Description
ContentLength	Returns the size of the uploaded file in bytes.
ContentType	Returns the MIME type of the uploaded file.
FileName	Returns the full filename.
InputStream	Returns a stream object pointing to the uploaded file.

Example : ShowFileUpload.aspx page

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">
<script runat="server">
protected void btnsave_Click(object sender, EventArgs e)
{
   StringBuilder sb = new StringBuilder();

   if (fuUploader1.HasFile)
   {
      try
      {
         sb.AppendFormat(" Uploading file: {0}", fuUploader.FileName);
         //saving the file
         fuUploader.SaveAs("<c:\\SaveDirectory>" +
         fuUploader.FileName);
         //Showing the file information
         sb.AppendFormat("<br/> Save As: {0}",
         fuUploader.PostedFile.FileName);
         sb.AppendFormat("<br/> File type: {0}",
         fuUploader.PostedFile.ContentType);
         sb.AppendFormat("<br/> File length: {0}",
         fuUploader.PostedFile.ContentLength);
         sb.AppendFormat("<br/> File name: {0}",
         fuUploader.PostedFile.FileName);
         }catch (Exception ex)
      {
         sb.Append("<br/> Error <br/>");
         sb.AppendFormat("Unable to save file <br/> {0}", ex.Message);
      }
   }
   else
   {
```

```
        lblmessage.Text = sb.ToString();
    }
}</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Show FileUpload</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <h3> File Upload:</h3>
        <br />
        <asp:FileUpload ID="fuUploader" runat="server" />
        <br /><br />
        <asp:Button ID="btnsave" runat="server" onclick="btnsave_Click"
        Text="Save" style = "width:85px" />
        <br /><br />
        <asp:Label ID="lblmessage" runat="server" />
    </div>
    </form>
</body>
</html>
```

3.9 Check Box Control :

A check box presents the user with a single option that can be selected to be checked or left unchecked. Set the checked property of the check box to true if you want it to be chosen automatically when the form is shown for the first time.

Control	Description	Syntax	Example
CheckBox	A check box presents the user with a single option that can be selected to be checked or left unchecked.	<asp:CheckBox ID= "CheckBox1" runat= "Server"> </asp:CheckBox>	<asp:CheckBox ID= "chkoption1" runat= "Server"> </asp:CheckBox>

Properties of CheckBox Control

Property	Description
AccessKey	It is used to set keyboard shortcut for the control.
TabIndex	The tab order of the control.
BackColor	It is used to set background color of the control.
BorderColor	It is used to set border color of the control.
BorderWidth	It is used to set width of border of the control.
Font	It is used to set font for the control text.

ForeColor	It is used to set color of the control text.
Text	It is used to set text to be shown for the control.
ToolTip	It displays the text when mouse is over the control.
Visible	To set visibility of control on the form.
Height	It is used to set height of the control.
Width	It is used to set width of the control.
Checked	It is used to set check state of the control either true or false.

Example : ShowCheckBox.aspx page

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">
<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
        {
            ShowCourses.Text = "None";
        }
        protected void btnSelect_Click(object sender, EventArgs e)
        {
            var message = "" ;
            if (chkOption1.Checked)
            {
                message = chkOption1.Text+" ";
            }
            if (chkOption2.Checked)
            {
                message += chkOption2.Text + " ";
            }
            if (chkOption3.Checked)
            {
                message += chkOption3.Text;
            }
            ShowCourses.Text = message;
        } </script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Show CheckBox</title>
</head>
<body>
```

```
<form id="form1" runat="server">
<div>
    <h2>Select Courses</h2>
    <asp:CheckBox ID="chkOption1" runat="server" Text="BCA" />
    <asp:CheckBox ID="chkOption2" runat="server" Text="BBA" />
    <asp:CheckBox ID="chkOption3" runat="server" Text="PGDCA" />
</div>
<p>
    <asp:Button ID="btnSelect" runat="server" Text="Button"
    OnClick="btnSelect_Click" />
</p>
</form>
<p>
    Courses Selected: <asp:Label runat="server" ID="ShowCourses"></
    asp:Label>
</p>
</form>
</body>
</html>
```

❑ **Check Your Progress – 2 :**

1. _____ is an ASP.NET control that does not display on the screen and allows you to save the value.

   a. CheckBox b. HiddenField c. TextBox d. FileUpload

3.10 Radio Buttons Control :

It is a control that is used to take input from the user and is known as an input control. It gives the user the ability to choose one option out of several available options. We may make a RadioButton by dragging the appropriate icon from the Visual Studio toolbox into the editor. This is a control that runs on the server, and ASP.NET offers its own tag so that you can create it.

Control	Description	Syntax	Example
RadioButton	It gives the user the ability to choose one option out of several available options.	< asp:RadioButtonID ="RadioButton1" runat ="server" Text=" " GroupName=" "/>	< asp:RadioButtonID ="RadioButton1" runat="server" Text="Male" GroupName = "gender"/>

Properties of RadioButton Control

Property	Description
AccessKey	It is used to set keyboard shortcut for the control.
TabIndex	The tab order of the control.
BackColor	It is used to set background color of the control.
BorderColor	It is used to set border color of the control.

BorderWidth	It is used to set width of border of the control.
Font	It is used to set font for the control text.
ForeColor	It is used to set color of the control text.
Text	It is used to set text to be shown for the control.
ToolTip	It displays the text when mouse is over the control.
Visible	To set visibility of control on the form.
Height	It is used to set height of the control.
Width	It is used to set width of the control.
Checked	It is used to set check state of the control either true or false.
GroupName	It is used to set name of the radio button group.

Example : ShowRadioButton.aspx page

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">
<script runat="server">
    protected void btnSubmit_Click(object sender, EventArgs e)
        {
            genderId.Text = "";
            if (rbtn1.Checked)
            {
                lblGender.Text = "Your gender is "+rbtn1.Text;
            }
            else lblGender.Text = "Your gender is "+rbtn2.Text;
        } </script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Show RadioButton</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:RadioButton ID="rbtn1" runat="server" Text="Male"
GroupName="gender" />
<asp:RadioButton ID="rbtn2" runat="server" Text="Female"
GroupName="gender" />
</div>
<p>
<asp:Button ID="btnSubmit" runat="server" Text="Submit"
OnClick="btnSubmit_Click" style="width: 61px" />
```

```
</p> </form>
<asp:Label runat="server" ID="lblGender"></asp:Label>
</body>
</html>
```

3.11 List Controls :

ASP.NET has several list controls which are as follows: CheckBoxList Control, RadioButtonList Control, BulletedList Control, NumberedList Control. These all controls gets from the same base class, known as ListControl.

A user is given the option to select one or more items from the list while using these controls. Both list boxes and drop–down lists have one or more items on their respective lists. These lists may be loaded via code or with the ListItemCollection editor. Both methods are available.

Control	Description	Syntax	Example
ListBox	List boxes contain one or more list items. A user can select one or more items from the list using these.	<asp:ListBox ID="ListBox1" runat="server" AutoPostBack="True" OnSelectedIndexChanged="ListBox1_SelectedIndexChanged"> </asp:ListBox>	<asp:ListBox ID="lstBox1" runat="server" AutoPostBack="True" OnSelectedIndexChanged="lstBox1_SelectedIndexChanged"> </asp:DropDownList>

Properties of List Controls

Property	Description
Items	The collection of ListItem objects that stands in for the various things that are contained within the control. This property is responsible for returning an object of type ListItemCollection when called.
Rows	Indicates the total number of items that will be shown in the box. If the actual list has more rows than are currently being displayed, a scroll bar will be added.
SelectedIndex	The index of the item that is now being picked. When there is more than one item selected, the index will reflect the item that was chosen first. The value of this attribute is set to –1 whenever there is no item selected.
SelectedValue	The value of the item that is currently being picked. In the event that more than one item is picked, the value will reflect the initial item chosen. If there is no selection made for this item, the value of this attribute is an empty string ("").
SelectionMode	The value of this property determines whether a list box supports single selections or multiple selections.
Items	The collection of ListItem objects that stands in for the many items that can be selected using the control. This property is responsible for returning an object of type ListItemCollection when called.

Rows	Indicates the total number of items that will be shown in the box. If the actual list has more rows than are currently being displayed, a scroll bar will be added.
SelectedIndex	The index of the item that is now being picked. When there is more than one item selected, the index will reflect the item that was chosen first. The value of this attribute is set to –1 whenever there is no item selected.
SelectedValue	The value of the item that is currently being picked. In the event that more than one item is picked, the value will reflect the initial item chosen. If there is no selection made for this item, the value of this attribute is an empty string ("").
SelectionMode	The value of this property determines whether a list box supports single selections or multiple selections.
Items	The collection of ListItem objects that stands in for the many items that can be selected using the control. This property is responsible for returning an object of type ListItemCollection when called.
Rows	Indicates the total number of items that will be shown in the box. If the actual list has more rows than are currently being displayed, a scroll bar will be added.
SelectedIndex	The index of the item that is now being picked. When there is more than one item selected, the index will reflect the item that was chosen first. The value of this attribute is set to –1 whenever there is no item selected.
SelectedValue	The value of the item that is currently being picked. In the event that more than one item is picked, the value will reflect the initial item chosen. If there is no selection made for this item, the value of this attribute is an empty string ("").
Text	The text displayed for the item.
Selected	Indicates whether the item is selected.
Value	A string value associated with the item.

3.12 Check Box List Control :

The user has the option to pick many items at the same time by making use of the ASP.NET CheckBoxList web control, which is a web control that can be used to collate the items that can be checked. Using the functions for data binding, you are able to build this list of things that are contained in the CheckBoxList in a dynamic manner. The CheckBoxList control class is responsible for implementing a variety of interfaces, including INamingContainer, IPostBackDataHandler, and IRepeatInfoUser, among others.

Control	Description	Syntax	Example
CheckBoxList	CheckBoxList control is a web control that can be used to collate the items that can be checked.	<asp:CheckBoxList id= "checkboxlist1" AutoPostBack = "True" TextAlign = "Right" OnSelectedIndexChanged = "CheckList_Clicked" runat= "server"> <asp:ListItem> Item 1 </asp:ListItem> <asp:ListItem> Item 2 </asp:ListItem> <asp:ListItem> Item 3 </asp:ListItem>	<asp:CheckBoxList id= "chkblGames" AutoPostBack = "True" TextAlign = "Right" OnSelectedIndexChanged = "CheckList_Clicked" runat= "server"> <asp:ListItem>Cricket </asp:ListItem> <asp:ListItem>Basketball </asp:ListItem> <asp:ListItem>Tennis </asp:ListItem>

Properties of CheckBoxList Controls

Property	Description
CellPadding	Sets cell spacing.
RepeatedItemCount	This displays the list's length.
RepeatColumns	Gets the CheckboxList's column count.
RepeatDirection	This attribute sets vertical or horizontal presentation of checkboxlist controls.
CellSpacing	It sets the checkboxlist cell spacing.
HasFooter	Checks for a checkboxlist footer. False indicates no footer.
HasHeader	Checks for a checkboxlist heading. False means no header section.
HasSeparators	Checks if checkboxlist has a separator. False indicates no item separators.
RenderWhenDataEmpty	This property obtains or sets the condition value when the data source is empty. True indicates the control is rendered even without data.
RepeatLayout	Specifies how the list will be rendered: table, ul, or span.
TextAlign	Sets checkbox text alignment.
Items	Gets list items.
AutoPostBack	If true, a postback occurs whenever a checkbox is selected.

To bind a CheckBoxList control to a data source, construct a DataControlObject with the display items. Using DataBind, attach this data source to the CheckBoxList control. "DataTextField" and "DataValueField" describe which data source fields to connect to Text and Value, respectively.

Properties of CheckBoxList Control to a data source

Property	Description
DataTextField	Specifies the name of the checkboxlist data source field.
DataValueField	Return the checkboxlist data source field.
DataSource	Specifies the checkboxlist data source.

Example : ShowCheckBoxList.aspx page

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">
<script runat="server">
   void Check_Clicked (Object sender, EventArgs e)
{
Message.Text = "Selected Item(s):<br /><br />";
// Iterate through the Items collection of the CheckBoxList
// control and display the selected items.
for (int i=0; i<checkboxlist1.Items.Count; i++)
{
if (checkboxlist1.Items[i].Selected)
{Message.Text += checkboxlist1.Items[i].Text + "<br />"; }
}
}</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
   <title>Show CheckBoxList</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:CheckBoxList id="checkboxlist1" AutoPostBack="True"
CellPadding="5" CellSpacing="5" RepeatColumns="2" RepeatDirection=
"Vertical" RepeatLayout="Flow" TextAlign="Right" OnSelectedIndex
Changed="Check_Clicked" runat="server">
<asp:ListItem>Item 1</asp:ListItem>
<asp:ListItem>Item 2</asp:ListItem>
<asp:ListItem>Item 3</asp:ListItem>
<asp:ListItem>Item 4</asp:ListItem>
<asp:ListItem>Item 5</asp:ListItem>
<asp:ListItem>Item 6</asp:ListItem></asp:CheckBoxList>
</div>
</body>
</html>
```

3.14 Radio Button List Control :

RadioButtonList is like DropDownList but shows radio buttons horizontally or vertically. RadioButtonList choices are single–select. They're mutually exclusive.

Control	Description	Syntax	Example
RadioButtonList	A radio button list presents a list of mutually exclusive options.	<asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoPostBack="True" OnSelectedIndexChanged="RadioButtonList1_SelectedIndexChanged" ></asp:RadioButtonList>	<asp: RadioButtonList id= "radblGames"AutoPostBack = "True"TextAlign = "Right" OnSelectedIndexChanged = "CheckList_Clicked" runat= "server"> <asp:ListItem>Cricket</asp:ListItem> <asp:ListItem>Basketball</asp:ListItem> <asp:ListItem>Tennis</asp:ListItem>

Properties of RadioButtonList Controls

Property	Description
RepeatLayout	This attribute specifies whether the table tags or the normal html flow to use while formatting the list when it is rendered. The default is Table. Default is Vertical.
RepeatDirection	It specifies the direction in which the controls to be repeated. The values available are Horizontal and Vertical.
RepeatColumns	It specifies the number of columns to use when repeating the controls; default is 0.

To bind a RadioButtonList control to a data source, construct a DataControlObject with the display items. Using DataBind, attach this data source to the RadioButtonList control. "DataTextField" and "DataValueField" describe which data source fields to connect to Text and Value, respectively.

Properties of RadioButtonList Control to a data source

Property	Description
DataTextField	Specifies the name of the checkboxlist data source field.
DataValueField	Return the checkboxlist data source field.
DataSource	Specifies the checkboxlist data source.

Example : ShowRadioButtonList.aspx page

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">
<script runat="server">
    void Check_Clicked (Object sender, EventArgs e)
{
Message.Text = "Selected Item(s):<br /><br />";
```

```
// Iterate through the Items collection of the CheckBoxList
// control and display the selected items.
for (int i=0; i<radiobuttonlist1.Items.Count; i++)
{
if (radiobuttonlist1.Items[i].Selected)
{Message.Text += radiobuttonlist1.Items[i].Text + "<br />";      }
}
}</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
   <title>Show RadioButtonList1</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:RadioButtonList id="radiobuttonlist1" AutoPostBack="True"
CellPadding="5" CellSpacing="5" RepeatColumns="2" Repeat
Direction="Vertical" RepeatLayout="Flow" TextAlign="Right"
OnSelectedIndexChanged="Check_Clicked" runat="server">
<asp:ListItem>Item 1</asp:ListItem>
<asp:ListItem>Item 2</asp:ListItem>
<asp:ListItem>Item 3</asp:ListItem>
<asp:ListItem>Item 4</asp:ListItem>
<asp:ListItem>Item 5</asp:ListItem>
<asp:ListItem>Item 6</asp:ListItem></asp:RadioButtonList>
</div>
</body>
</html>
```

3.15 Bulleted List Control :

BulletedList is rich in item styles. It displays unsorted or ordered lists. Each list item can be text, a LinkButton control, or a web link. BulletStyle property defaults to NotSet, rendering a bulleted list. BulletedList's DisplayMode attribute modifies list item appearance. Values include: Circle, CustomImage, Disc, LowerAlpha, LowerRoman, NotSet, Numbered, Square, UpperAlpha, UpperRoman. BulletedList has BulletStyle attribute. Values include: HyperLink, LinkButton and Text.

Control	Description	Syntax	Example
BulletedList	Create bulleted or numbered lists with the help of the control for creating bulleted lists. These controls hold a collection of ListItem objects, which are accessible via the Items property of the control. You can use this property to make references to the collection.	<asp:BulletedList ID="BulletedList1" runat="server"> </asp:BulletedList>	<asp:BulletedList id="blProducts" DataSourceID="SqlDataSource1" DataTextField="Title" Runat="server" />

Properties of BulletedList Controls

Property	Description
BulletImageUrl	Specifies a bullet list image URL. "CustomImage" if "BulletStyle"
BulletStyle	Styles bullet lists
DisplayMode	List type
FirstBulletNumber	Start number of ordered list items
runat	Required. Identifies a server control. To "server"
Target	Where to open target URL
ImageUrl	The location of the image file that will be shown by the control.

To bind a BulletedList control to a data source, construct a DataControlObject with the display items. Using DataBind, attach this data source to the BulletedList control. "DataTextField" and "DataValueField" describe which data source fields to connect to Text and Value, respectively.

Properties of BulletedList Control to a data source

Property	Description
DataTextField	Specifies the name of the checkboxlist data source field.
DataValueField	Return the checkboxlist data source field.
DataSource	Specifies the checkboxlist data source.

Example : ShowBulletedList.aspx page

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            List<string> courseList = new List<string>();
            courseList.Add("PGDBA");
            courseList.Add("PGDCA");
            courseList.Add("BCA");
            courseList.Add("BBA");
            courseList.Add("BA");
            courseList.Add("BEd");
```

```
            BulletedList1.DataSource = courseList;
            BulletedList1.DataBind();
        }
    }
    protected void Style_Command(object sender, CommandEventArgs e)
    {
        switch (e.CommandName)
        {
            case "Circle":
                BulletedList1.BulletStyle = BulletStyle.Circle;
                break;
            case "Disc":
                BulletedList1.BulletStyle = BulletStyle.Disc;
                break;
            case "Numbered":
                BulletedList1.BulletStyle = BulletStyle.Numbered;
                break;
            case "Square":
                BulletedList1.BulletStyle = BulletStyle.Square;
                break;
            case "LowerRoman":
                BulletedList1.BulletStyle = BulletStyle.LowerRoman;
                break;
            case "UpperAlpha":
                BulletedList1.BulletStyle = BulletStyle.UpperAlpha;
                break;
        }
    }
```

3.16 Numbered List Control :

NumberedList is rich in item styles. It displays ordered lists. Each list item can be text, a LinkButton control, or a web link. BulletStyle property is numbered, rendering a numbered list. NumberedList's DisplayMode attribute modifies list item appearance. Values include: LowerAlpha, LowerRoman, NotSet, Numbered, UpperAlpha, UpperRoman. NumberedList has BulletStyle attribute.

Control	Description	Syntax	Example
BulletedList	Create bulleted or numbered lists with the help of the control for creating bulleted lists. These controls hold a collection of ListItem objects, which are accessible via the Items property of the control. You can use this property to make references to the collection.	<asp:BulletedList ID="BulletedList1" runat="server"> </asp:BulletedList>	<asp:BulletedList id="blProducts" BulletStyle= "Numbered" Runat="server" />

Properties of NumberedList Controls

Property	Description
BulletImageUrl	Specifies a bullet list image URL. "CustomImage" if "BulletStyle"
BulletStyle	Styles bullet lists
DisplayMode	List type
FirstBulletNumber	Start number of ordered list items
runat	Required. Identifies a server control. To "server"
Target	Where to open target URL

To bind a NumberedList control to a data source, construct a DataControlObject with the display items. Using DataBind, attach this data source to the NumberedList control. "DataTextField" and "DataValueField" describe which data source fields to connect to Text and Value, respectively.

Properties of NumberedList Control to a data source

Property	Description
DataTextField	Specifies the name of the checkboxlist data source field.
DataValueField	Return the checkboxlist data source field.
DataSource	Specifies the checkboxlist data source.

Example : ShowNumberedList.aspx page

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            List<string> courseList = new List<string>();
            courseList.Add("PGDBA");
            courseList.Add("PGDCA");
            courseList.Add("BCA");
            courseList.Add("BBA");
            courseList.Add("BA");
            courseList.Add("BEd");
            BulletedList1.DataSource = courseList;
            BulletedList1.DataBind();
```

```
    }
}
protected void Style_Command(object sender, CommandEventArgs e)
{
    switch (e.CommandName)
    {
        case "Numbered":
            BulletedList1.BulletStyle = BulletStyle.Numbered;
            break;
        case "LowerRoman":
            BulletedList1.BulletStyle = BulletStyle.LowerRoman;
            break;
        case "UpperAlpha":
            BulletedList1.BulletStyle = BulletStyle.UpperAlpha;
            break;
    }
}
```

❑ **Check Your Progress – 3 :**

1. _____ presents a list of mutually exclusive options.:

   a. ListBox b. Numbered List

   c. RadioButton List d. None of these

3.17 HyperLink Control :

Either by writing code or by making use of the drag–and–drop functionality of the visual studio IDE, we can establish HyperLinks. The toolbox contains a listing for this control. It is a control that can be utilized in the process of hyperlink creation. It reacts whenever a click event occurs. It allows us to refer to any web page that is stored on the server.

Control	Description	Syntax	Example
HyperLink	The HyperLink control is like the HTML <a> element. This causes the server to render it as an HTML control, which then causes the browser to receive the following code.	<asp:HyperLink ID="HyperLink1" runat="server"> HyperLink </asp:HyperLink>	< asp:HyperLink ID="HyperLink1" runat="server" Text="Contact Us" NavigateUrl=" https://baou.edu.in " > </asp:HyperLink> Contact Us

Properties of Hyperlink Control

Property	Description
ImageUrl	Path of the image to be displayed by the control.
NavigateUrl	Target link URL.
Target	The window or frame which loads the linked page.
Text	The text to be displayed as the link.

Example : ShowHyperlink.aspx page

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
<asp:HyperLink ID="HyperLink1" runat="server" Text="Contact Us " NavigateUrl = "https://baou.edu.in"> </asp:HyperLink>
        </div>
    </form>
</body>
</html>
```

3.18 Image Control :

Recently, there has been a significant increase in the number of people seeking how to preview an image before it is saved to the database on various online forums. Before we save the uploaded image to the database, let's have a look at it using the preview function in ASP.NET, which we will cover in this post. This also explains how to pass a byte array into an image control, so keep reading. We are going to accomplish it with the help of the IhttpHandler.

Control	Description	Syntax	Example
Image	The image control can be used to display images on a web page, or it can be used to display some alternative text in the event that the image cannot be displayed.	<asp:Image ID="Image1" runat="server">	<asp:Image runat="server" ID="ProfileImage" Height="122px" Width="110px" />

Properties of Image Control

Property	Description
AlternateText	In the event that the image cannot be displayed, other text will be shown instead.
ImageAlign	Options for aligning the control are available.
ImageUrl	The location of the image file that will be shown by the control.

Example : ShowImage.aspx page

```
<table>
   <tr>
      <td class="style3">
         <asp:Label ID="Label1" runat="server" Text="Photo upload" />
      </td>
      <td class="style4">
         <asp:FileUpload runat="server" ID="PhotoUpload" />
      </td>
      <td class="style4">
<asp:Button runat="server" OnClick="btnPreview_Click"
ID="btnPhotoPreview" Text="Preview" />
      </td>
      <td class="style1">
         <asp:Image runat="server" ID="ImagePreview" Height="164px"
         Width="125px" />
      </td>
   </tr>
</table>
```

In order to display the image, we need to build a handler class. The upload control will make use of a session variable for the value of FileBytes. Retrieve the session variable inside the new handler class, then proceed to build the image. When we add the .ashx file, it will have some code inside it:

```
<%@ WebHandler Language="C#" Class="ImageHandler" %>
using System;
using System.Web;
 public class ImageHandler : IHttpHandler {
   public void ProcessRequest (HttpContext context) {
      context.Response.ContentType = "text/plain";
      context.Response.Write("Hello Students");
   } public bool IsReusable {
      get {
         return false;
      } } }
<%@ WebHandler Language="C#" Class="ImageHandler" %>
using System;
using System.Web;
public class ImageHandler : IHttpHandler, System.Web.SessionState.
IRequiresSessionState
{
```

```
public void ProcessRequest (HttpContext context) {
    //Checking whether the imagebytes session variable have anything
    else not doing anything
    if ((context.Session["ImageBytes"]) != null)
    {
        byte[] image = (byte[])(context.Session["ImageBytes"]);
        context.Response.ContentType = "image/JPEG";
        context.Response.BinaryWrite(image);
    }
}
public bool IsReusable {
    get {
        return false;
    } } }
```

Now inside the button click in our page, get the filebytes from the FileUpload control and save it in a session variable :

```
protected void btnPreview_Click(object sender, EventArgs e)
{
    Session["ImageBytes"] = PhotoUpload.FileBytes;
    ImagePreview.ImageUrl = "~/ImageHandler.ashx";
}
```

3.19 AdRotator Control :

A random banner graphic is chosen by the AdRotator control from a list that is supplied in an external XML schedule file. This file, which is an external XML schedule, is referred to as the advertisement file. You are able to specify both the advertisement file and the target window type that the link should follow by utilising the AdvertisementFile property and the Target property of the AdRotator control, respectively.

Control	Description	Syntax	Example
AdRotator	The Adrotator control shows the sequence of photos that is specified in the file that is stored externally in XML format. A number of different properties, including as image impressions, NavigateUrl, ImageUrl, and AlternateText, are used within an xml file to specify which images are to be displayed.	<asp:AdRotator ID="AdRotator1" runat="server" />	<asp:AdRotator runat = "server" AdvertisementFile = "adfile.xml" Target = "_blank" />

The XML advertisement file contains information about shown ads. XML is a W3C document markup standard. It's a text–based markup language that stores data via tags. Extensible means you can describe a document by adding tags to the application. XML is a set of principles for building new markup languages, like HTML. It's meta–markup. Developers can make custom tag sets. It stores and transports data.

Properties of XML

Property	Description
Advertisements	Encloses the advertisement file.
Ad	Delineates separate ad.
ImageUrl	The path of image that will be displayed.
NavigateUrl	The link that will be followed when the user clicks the ad.
AlternateText	The text that will be displayed instead of the picture if it cannot be displayed.
Keyword	Keyword identifying a group of advertisements. This is used for filtering.
Impressions	The number indicating how often an advertisement will appear.
Height	Height of the image to be displayed.
Width	Width of the image to be displayed.

Properties of AdRotator Control

Property	Description
AdvertisementFile	The path to the advertisement file.
AlternateTextFeild	The element name of the field where alternate text is provided. The default value is AlternateText.
DataMember	The name of the specific list of data to be bound when advertisement file is not used.
DataSource	Control from where it would retrieve data.
DataSourceID	Id of the control from where it would retrieve data.
Font	Specifies the font properties associated with the advertisement banner control.
ImageUrlField	The element name of the field where the URL for the image is provided. The default value is ImageUrl.
KeywordFilter	For displaying the keyword based ads only.
NavigateUrlField	The element name of the field where the URL to navigate to is provided. The default value is NavigateUrl.
Target	The browser window or frame that displays the content of the page linked.
UniqueID	Obtains the unique, hierarchically qualified identifier for the AdRotator control.

Following are the important events of the AdRotator class :

Events of AdRotator Control

Events	Description
AdCreated	It's raised after control creation but before page rendering. Binding to a data source in a server control.
DataBinding	After server control data binding.
DataBound	This is the final stage of a server control's lifecycle when an ASP.NET page is requested.

Disposed	First step in the server control's lifetime.
Init	Loads the server control into Page.
Load	After loading but before rendering the Control object.
PreRender	For displaying the keyword based ads only.
Unload	Unloads server control from memory.

Example : Advertise.XML page

```
<?xml version="1.0" encoding="utf–8" ?>
<Advertisements>
  <Ad>
    <ImageUrl>img1.jpg</ImageUrl>
    <NavigateUrl>https://baou.edu.in</NavigateUrl>
    <AlternateText>BabaSaheb Ambedkar Open University</
    AlternateText>
    <Impressions>20</Impressions>
    <Keyword>BAOU</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>img2.jpg</ImageUrl>
    <NavigateUrl>http://baou.edu</NavigateUrl>
    <AlternateText>Ambedkar University</AlternateText>
    <Impressions>50</Impressions>
    <Keyword>Distance Learning</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>img3.jpg</ImageUrl>
    <NavigateUrl> https://baou.edu.in/about</NavigateUrl>
    <AlternateText>BabaSaheb Ambedkar University</AlternateText>
    <Impressions>30</Impressions>
    <Keyword>Distance Learning</Keyword>
  </Ad>
</Advertisements>
```

Allocate the XML file to AdRotator control at AdvertisementFile property of AdRotator control.

❑ **Check Your Progress – 4 :**

1. _____ control can be used to display images on a web page, or it can be used to display some alternative text in the event that the image cannot be displayed:

   a. AdRotator b. HyperLink c. Image d. None of these

3.20 Calendar Control :

A web page can incorporate a calendar into its design by utilising the Calendar control that is included with ASP.NET. The ASP.NET Calendar control presents the user with a month calendar that enables the user to pick dates and navigate to the following month as well as the month before it.

This control will, by default, show the name of the current month, day headings for the weeks and days of the month, as well as arrow characters that allow navigating to the previous or subsequent month. The following is a list of the classes that make up this control's class hierarchy: the order of operations is Object > Control > WebControl > Calendar.

You can incorporate a calendar function into your web page by using the Calendar control, which is a sophisticated and powerful Web server control. By utilising the Calendar control, we are able to display any date that falls between the years 0 and 9999 A.D.

Control	Description	Syntax	Example
Calendar	The calendar web control includes the following features: Month-by-month, Choose a day, week, or month. Choosing days, Monthly, Programmatically displaying days.	<asp:Calender ID = "Calendar1" runat = "server"> </asp:Calender> <asp:Calender ID = "Calendar1" runat = "server" SelectionMode= "DayWeekMonth"> </asp:Calender>	<asp:Calendar ID="Calendar1" runat="server" DayNameFormat="Shortest" OnDayRender = "Calendar1_DayRender" OnSelectionChanged = "Calendar1_SelectionChanged" <SelectedDayStyle BackColor = "#CCCCFF" Font-Bold ="True" /> <SelectorStyle BackColor="#FFCC66" /> <TodayDayStyle BackColor ="#FFCC66" ForeColor ="White"/> <OtherMonthDayStyle ForeColor="#CC9966" /> <NextPrevStyle Font-Size="9pt" ForeColor = "#FFFFCC"/> <DayHeaderStyle BackColor = "#FFCC66" Font-Bold="True" Height="1px"/><TitleStyle BackColor = "#990000" Font-Bold="True" Font-Size="9pt" ForeColor="#FFFFCC" /> </asp:Calendar>

The calendar web control includes the following features: One–month–at–a–time display, Day, week, or month selection, Date selection, Monthly, Programmatically displaying days.

Properties of Calendar Control

Property	Description
Caption	Caption of the calendar control.
CaptionAlign	Caption alignment.
CellPadding	Gets or sets the data–to–cell–border spacing.
CellSpacing	Sets cell spacing.
DayHeaderStyle	Gets the day's style characteristics.
DayNameFormat	Formats weekdays.
DayStyle	Gets the month days' style characteristics.
FirstDayOfWeek	Gets or sets the first column day.
NextMonthText	Gets or sets next month control text. > is default.
NextPrevFormat	Gets or sets the next/previous month format.
OtherMonthDayStyle	Gets the Calendar control's non–month days' style settings.
PrevMonthText	Gets or sets previous month control text. Default <
SelectedDate	Gets/sets the date.
SelectedDates	Gets the selected dates' DateTime objects.
SelectedDayStyle	Gets the date styles.
SelectionMode	Gets or sets the mode for selecting a day, week, or month.
SelectMonthText	Gets or sets the month selector's text.
SelectorStyle	Gets week and month column style characteristics.
SelectWeekText	Gets or sets the week selection column text.
ShowDayHeader	Gets or sets whether weekday headings are displayed.
ShowGridLines	Gets or sets gridlines' visibility.
ShowNextPrevMonth	Gets or sets whether next and previous month navigation elements are shown.
ShowTitle	Gets or sets the title section's visibility.
TitleFormat	Formats the title part.
Titlestyle	Get Calendar's title heading style properties.
TodayDayStyle	Gets today's Calendar date style characteristics.
TodaysDate	Gets today's date.
UseAccessibleHeader	Gets or sets a value indicating whether to render the table header th> HTML element for day headers.
VisibleDate	Gets or sets the month's date.
WeekendDayStyle	Gets Calendar's weekend style characteristics.

Following are the important events of the Calendar class :

Events of Calendar Control

Events	Description
SelectionChanged	A day, week, or month raises it.
DayRender	Each calendar data cell raises it.
VisibleMonthChanged	Change of month raises it.

Working of Calendar Control : A site can have a functional calendar that displays the months and days of the year by including just a calendar control with no code behind file. This type of calendar displays only the months and days of the year. Additionally, navigation to the following and prior months is possible.

Users can select a single day, a full week, or an entire month using the calendar functions on their devices. Using the SelectionMode property will allow you to accomplish this goal. The following examples show the values of this property :

Values of SelectionMode Property

Property	Description
Day	To choose just one particular day.
DayWeek	To choose only one day or the entire week.
DayWeekMonth	To choose a certain day, a particular week, or the entire month.
None	There is no option to choose from.

When DayWeekMonth is selected, an extra column with the > symbol appears for selecting the week, and a >> symbol appears to the left of the days name for selecting the month.

The following example shows how to select a date and display the date in a label :

Example : Calendar.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title>Untitled Page</title>

</head>

<body>

    <form id="form1" runat="server">

    <div>

        <p style="text–align: center">

        <b></b>
```

```
<asp:Label ID="Label1" runat="server" Font–Bold="True" Font–Names="Arial Black" Font–Size="Medium" ForeColor="#0066FF">Indian List of Holidays 2009</asp:Label><br /></b>
</p>
<asp:Calendar ID="Calendar1" runat="server" BackColor="#FFFFCC" BorderColor = "#FFCC66" BorderWidth="1px" DayNameFormat="Shortest" Font–Names="Verdana" Font–Size="8pt" ForeColor="#663399" ShowGridLines="True" OnDayRender="Calendar1_DayRender" OnSelectionChanged="Calendar1_SelectionChanged" OnVisibleMonthChanged = "Calendar1_VisibleMonthChanged">
<SelectedDayStyle BackColor="#CCCCFF" Font–Bold="True" />
<SelectorStyle BackColor="#FFCC66" />
<TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
<OtherMonthDayStyle ForeColor="#CC9966" />
<NextPrevStyle Font–Size="9pt" ForeColor="#FFFFCC" />
<DayHeaderStyle BackColor="#FFCC66" Font–Bold="True" Height="1px" />
<TitleStyle BackColor="#990000" Font–Bold="True" Font–Size="9pt" ForeColor = "#FFFFCC" />
</asp:Calendar>
<br /> <b></b> <asp:Label ID="LabelAction" runat="server"></asp:Label><br /></b>
</div>
</form>
</body>
</html>
```

Calendar.aspx.cs

```
using System;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Collections;
public partial class _Default : System.Web.UI.Page
{
```

```
Hashtable HolidayList;
protected void Page_Load(object sender, EventArgs e)
{
    HolidayList = Getholiday();
    Calendar1.Caption = "BAOU";
    Calendar1.FirstDayOfWeek = FirstDayOfWeek.Sunday;
    Calendar1.NextPrevFormat = NextPrevFormat.ShortMonth;
    Calendar1.TitleFormat = TitleFormat.Month;
    Calendar1.ShowGridLines = true;
    Calendar1.DayStyle.Height = new Unit(50);
    Calendar1.DayStyle.Width = new Unit(150);
    Calendar1.DayStyle.HorizontalAlign = HorizontalAlign.Center;
    Calendar1.DayStyle.VerticalAlign = VerticalAlign.Middle;
    Calendar1.OtherMonthDayStyle.BackColor = System.Drawing.Color.
    AliceBlue;
}
private Hashtable Getholiday()
{
    Hashtable holiday = new Hashtable();
    holiday["1/1/2022"] = "New Year";
    holiday["1/14/2022"] = "Makar Sakranti";
    holiday["1/26/2022"] = "Republic Day";
    holiday["3/1/2022"] = "Maha Shivaratri";
    holiday["3/18/2022"] = "Holi";
    holiday["4/2/2022"] = "Cheti Chand";
    holiday["4/10/2022"] = "Ram Navmi";
    holiday["4/14/2022"] = "Ambedkar Jayanti";
    holiday["4/13/2022"] = "Good Friday";
return holiday;
}
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{ LabelAction.Text = "Date changed to :" + Calendar1.SelectedDate.
ToShortDateString(); }
protected void Calendar1_VisibleMonthChanged(object sender, Month
ChangedEventArgs e)
{ LabelAction.Text = "Month changed to :" + e.NewDate.ToShort
DateString(); }
protected void Calendar1_DayRender(object sender, DayRenderEventArgs e)
{ if (HolidayList[e.Day.Date.ToShortDateString()] != null) {
```

```
Literal literal1 = new Literal();
literal1.Text = "<br/>";
e.Cell.Controls.Add(literal1);
Label label1 = new Label();
label1.Text = (string)HolidayList[e.Day.Date.ToShortDateString()];
label1.Font.Size = new FontUnit(FontSize.Small);
e.Cell.Controls.Add(label1);
} } }
```

3.21 Multiviews Control :

You are able to partition the content of a page into multiple groups and display only one of those groups at a time with the assistance of MultiView and View controls. Each View control is responsible for the administration of a single content group, and all of the View controls are consolidated into a single MultiView control.

Displaying a single instance of a View control at a time is the responsibility of the MultiView control. The view that is now being shown is referred to as the active view.

Control	Description	Syntax	Example
Multiview	The Multiview control displays or renders the particular view on the page in order to benefit the user.	<asp:MultView ID= "MultiView1" runat= "server"> </asp:MultiView>	
View	A number of different web server controls can be contained within the View control.	<asp:View ID= "View1" runat= "server"> </asp:View>	

On the other hand, the View control can never exist independently. If you tried to utilize it without anything else, you would get an error message. It is usually used in conjunction with a Multiview control in the sense that:

```
<asp:MultView ID= "MultiView1" runat= "server">
   <asp:View ID= "View1" runat= "server"> </asp:View>
</asp:MultiView>
```

Properties of Multiviews Control

Property	Description
Views	A collection of controls for the Views contained within the MultiView.
ActiveViewIndex	An index that starts at zero and indicates which view is currently being used. If there is not currently an active view, the index will be –1.

The CommandName attribute of the MultiView navigation button is linked to a MultiView field. When a button with CommandName NextView is clicked, the multiview navigates to the next view. It shows default property command names :

Properties of CommandNames

Property	Description
NextViewCommandName	View Next
PreviousViewCommandName	View Previous
SwitchViewByIDCommandName	SwitchViewByID
SwitchViewByIndexCommandName	SwitchViewByIndex

The following are the essential events of the multiview control :

Events of Multiview control

Event	Description
ActiveViewChanged	Decided to raise whenever a view is altered.
Activate	Decided to raise by the active view
Deactivate	Decided to raise by the passive viewpoint

Example : ShowMultiview.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind=" Default.aspx.cs" Inherits = "multiviewdemo._Default" %>

<!DOCTYPE html PUBLIC "–//W3C//DTD XHTML 1.0 Transitional// EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1–transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
    <head runat="server">
        <title>
            Untitled Page
        </title>
    </head>
    <body>
    <form id="form1" runat="server">
    <div><h2>MultiView and View Controls</h2>
    <asp:DropDownList ID="DropDownList1" runat="server" onselectedind exchanged= "DropDownList1_SelectedIndexChanged"></asp:Drop DownList><hr />
    <asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="2" onactiveviewchanged = "MultiView1_ActiveViewChanged" >
    <asp:View ID="View1" runat="server">
    <h3>This is view 1</h3><br />
    <asp:Button CommandName="NextView" ID="btnnext1" runat= "server" Text = "Go To Next" />
    <asp:Button CommandArgument="View3" CommandName="Switch ViewByID" ID="btnlast" runat="server" Text ="Go To Last" />
    </asp:View>
    <asp:View ID="View2" runat="server">
    <h3>This is view 2</h3>
```

```
<asp:Button CommandName="NextView" ID="btnnext2" runat=
"server" Text = "Go To Next" />
<asp:Button CommandName="PrevView" ID="btnprevious2" runat=
"server" Text = "Go To Previous View" />
</asp:View>
<asp:View ID="View3" runat="server">
<h3> This is view 3</h3>
<br /><asp:Calendar ID="Calender1" runat="server"></asp:Calendar>
<br />
<asp:Button CommandArgument="0" CommandName="SwitchViewBy
Index" ID="btnfirst" runat="server" Text = "Go To Next" />
<asp:Button CommandName="PrevView" ID="btnprevious" runat=
"server" Text = "Go To Previous View" />
</asp:View>
</asp:MultiView>
    </div>
</form>
</body>
</html>
```

3.22 Let Us Sum Up :

In this unit, we learned that the Control class derives all Windows .NET GUI controls. Control Classes represent use–case–specific control behaviour. Control classes can have use–case–specific behaviours. Control object behaviour directly affects use case execution.

.NET controls are components written with the.NET Framework that run in the managed environment of the.NET Common Language Runtime (CLR). Most client–side Windows programmes use.NET controls to encapsulate their user interface capabilities.

Web controls can enhance FORMs and pages. Web controls are HTML elements wrapped in ASP+ scripting tags. Web controls range from text boxes to grids and lists.

We learned that both the Control class and the Page class contain a Controls property that, when called, returns an instance of the ControlCollection class.

Since it is not inherited from the WebControl namespace and will not have significant functionality, the Literal Control, which is identical to the Label Control and is used to display static text on web pages, is referred to as a Label Control.

Labels are a type of C# control that are used to display text at a predetermined area on a web page. This text may then be used to add more descriptive text to a Form in order to provide the user with additional useful information. Collecting Data from Users will assign the job to a single user or to a group, and it will return the unique ID of the task item that holds the user's responses. This is done so that the workflow can search up this information at a later time by using the ID. Users are given the ability to enter

information within a box that supports wordwrapping as well as vertical and horizontal scrolling when using a control known as a multiline TextBox.

3.23 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : c **2 :** d

- **Check Your Progress 2 :**

  1 : b

- **Check Your Progress 3 :**

  1 : c

- **Check Your Progress 4 :**

  1 : c

3.24 Glossary :

1. **Controls :** It refers to the ability to manage, organize, or run something on a computer.
2. **Web :** It is an interconnected system of public webpages accessible through the Internet.
3. **ViewState :** View state is used automatically by the ASP.NET page framework to persist information that must be preserved between postbacks.

3.25 Assignment :

1. Explain the difference between Label Control and Literal Control.

3.26 Activities :

1. Use various list controls and find out the differences between all.

3.27 Case Study :

Examine which controls are used in our BAOU website.

3.28 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

Unit 04

STUDYING NAVIGATION CONTROLS

UNIT STRUCTURE

4.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About the Master Pages
- Navigation Controls
- TreeView Control
- Menu Control
- SiteMapPath Control

4.1 Introduction :

In this unit, we will understand about the use of Master Pages, Navigation Controls, TreeView Control, Menu Control and in last we will read about the SiteMapPath Control. Master Pages create a standard page layout. If you want all your website pages to have a three–column layout, you can construct it in a Master Page and apply it to various content pages. Master Pages can display common content across pages. Navigation Control is a combination of user interface controls allows users to construct a mental representation of the structures we design and then navigate them smoothly and reliably. These controls are the Information Architecture's user interface layer.

A clear navigation system helps users find pages and information on your site. It encourages visitors to remain, read your material, and have a great user experience, leading to higher sales and brand loyalty. A tree–view control displays a hierarchical list of elements, such as document headings, index entries, or disc files and directories. Each item has a label, optional bitmapped picture, and subitems.

The Menu control displays a menu in an ASP.NET Web page and is used with SiteMapDataSource to navigate a website. Menu control features: Menu items can be bound to hierarchical data sources via data binding. Sitemap Path control lets you navigate between website pages. It's a navigation control that shows the site's pages. This map names the website's pages.

4.2 Master Pages :

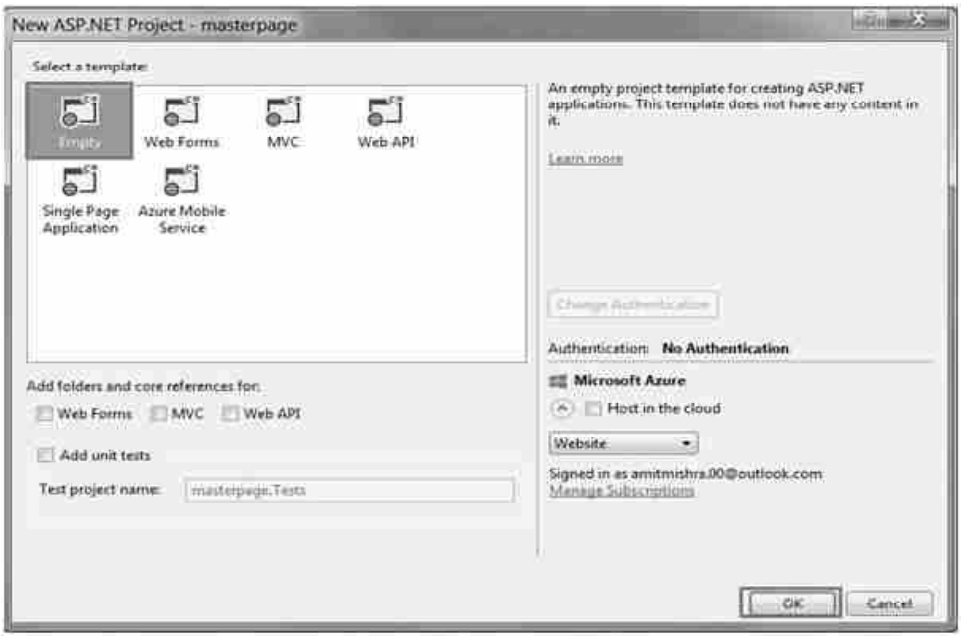
Making use of a Master Page allows for the production of a page format that is standard. If you want all of the pages on your website to have a layout that has three columns, for instance, you may only need to build the layout once in a Master Page and then apply it to multiple content pages. In addition to this, you may make use of something that is referred to as Master Pages so that the same content is displayed across multiple pages.

Let's get started by methodically designing the master page.

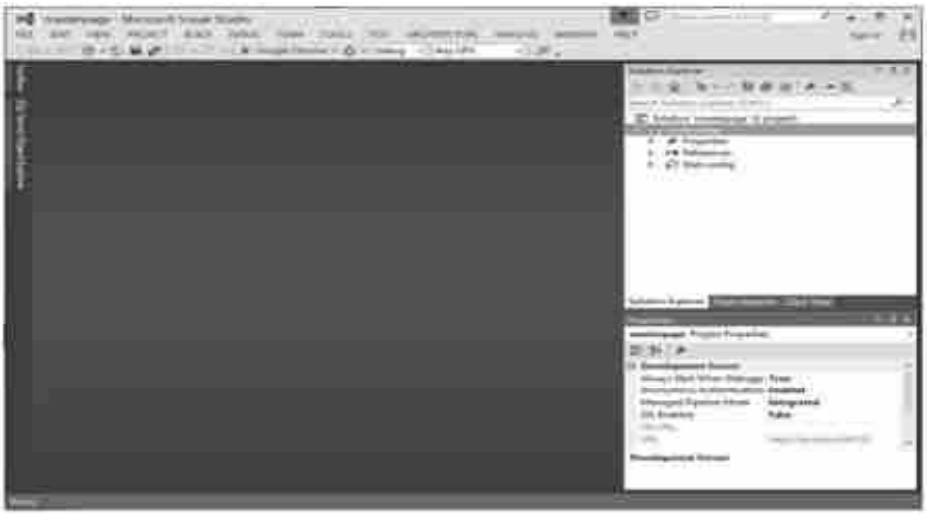
I Step : Open a new project in Visual Studio

New project-->Installed-->Visual C#-->Web-->ASP.NET Web Application

After clicking OK button in the Window, select Empty from the template.

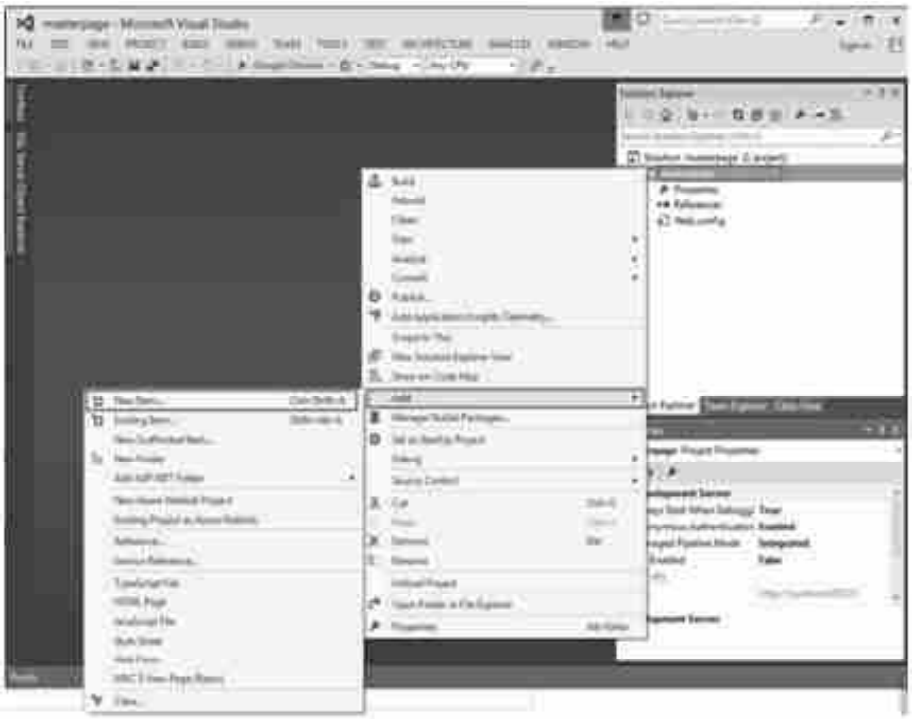


After clicking OK button, project "masterpage" opens. There will not be any file.



II Step : Add new item in our project by right click on the project in solution explorer.

Right click on Project-->Add-->New item



Once you have opened the new window that was generated by clicking on the new item, select Web Form then Web Forms Master Page.

After selecting the project that you want to add the'site1.master' master page to, click the 'Add' button.

Click on site1.master in the Solution Explorer.

III Step : Add new item in our project by right click on the project in solution explorer.

Construct the master page making use of HTML.

Example : HTML code for Master Page

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="
Site1.master.cs" Inherits="masterpage.Site1" %>
    <!DOCTYPE html>
    <html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Master Page Example</title>
    <link href="css/myStyle.css" rel="stylesheet" />
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <!DOCTYPE html>
<html>
<head>
    <title>Example: Master Page</title>
    <link rel="stylesheet" type="text/css" href="myStyle.css">
</head>
<body>
<header id="header">
<h1>BAOU Master Page</h1>
</header>
<nav id="nav">
```

```
    <ul>
        <li><a href="home.aspx">Home</a></li>
        <li><a href="#">About Us</a></li>
        <li><a href="#">Contact Us</a></li>
    </ul>
</nav>
<aside id="side">
    <h1>Study Material</h1>
    <a href="#"><p>BCA–501</p></a>
    <a href="#"><p>BCA–502</p></a>
    <a href="#">BCA–503</a>
</aside>
    <div id="con">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
    </div>
<footer id="footer">
    copyright @BAOU
</footer>
</body>
</html>
    <form id="form1" runat="server"> </form>
</body>
</html>
```

myStyle.css

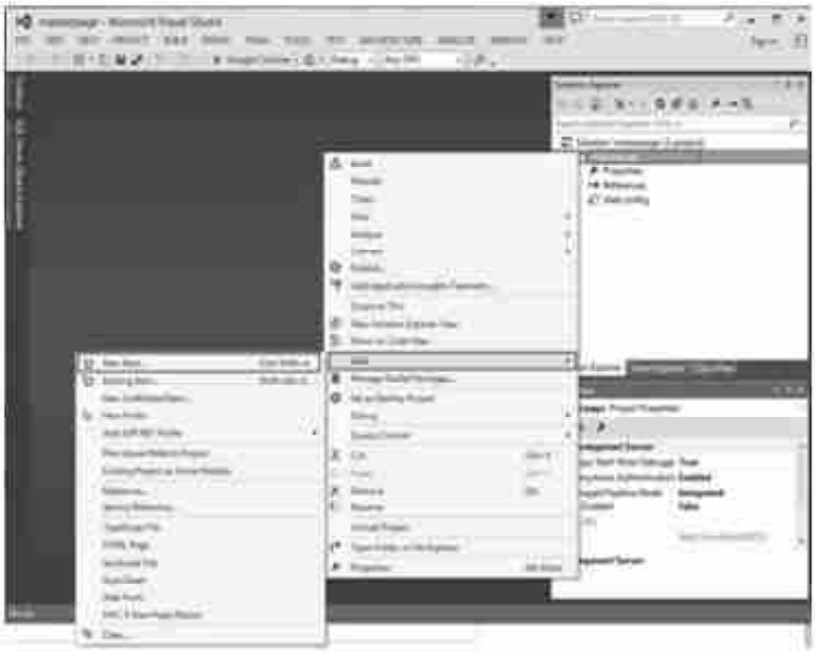
```
#header{
    color: "pink";
    text–align: center;
    font–size: 18px;
}
#nav{
    background–color:"pink";
    padding: 3px;
}
ul{
    list–style–type: none;
}
li a {
    color: grey;
```

```
font–size: 25px;
column–width: 5%;
    }
    li
    {
    display: inline;
    padding–left: 2px;
    column–width: 20px;
    }
    a{
text–decoration: none;
margin–left:20px
    }
    li a:hover{
        background–color: "green";
        color: "white";
        padding:1%;
        }
        #side{
        text–align: center;
        float: right;
        width: 10%;
        padding–bottom: 75%;
        background–color: "pink";
        }
        #footer{
        background–color: "grey";
        text–align:center;
        padding–bottom: 5%;
        font–size: 15px;
    }
        #con{
            border:dash;
            border–color:brown;
        }
```

Our design of webpage completed.

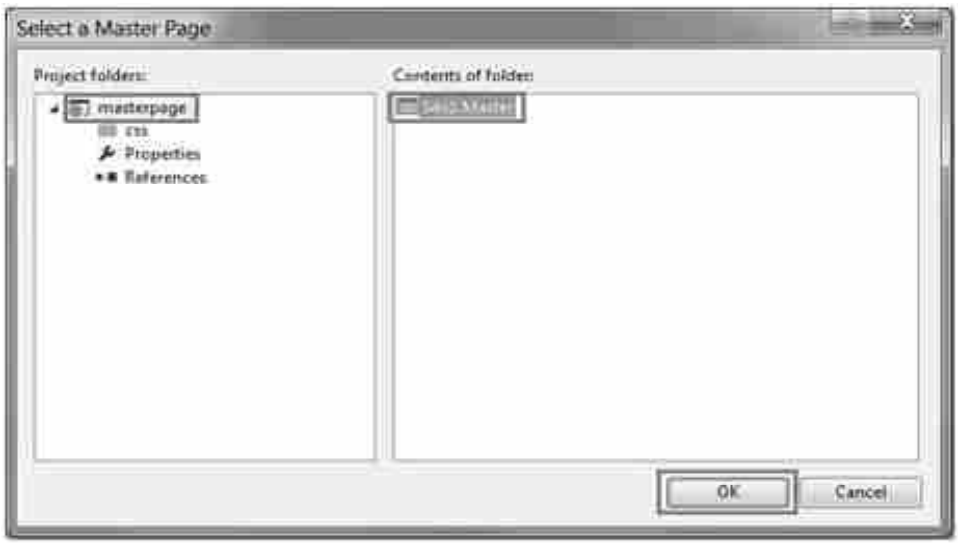
IV Step : Add a web form in our project by right clicking on the project

Right click on Project-->Add-->New item



Choose the Web form that corresponds with the master page.

After doing so, add the button labelled Window, then access the selected masterpage by going to site1.master, and then click the OK button.



Create the layout for our homepage now. Only the home page is being written about here.

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master"
AutoEventWireup="true" CodeBehind="home.aspx.cs" Inherits=
"masterpage.home" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head"
runat="server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlace
Holder1" runat="server">

    <h1>Home page</h1>

</asp:Content>
```

At last, our Master page has been crafted; proceed to construct and run the project.

4.3 Navigation Control :

The term "Navigation Control" refers to a menu that, in order to make it simpler to update and manage, can be saved in a file. This file is typically referred to as web.Sitemap, and it is kept in the directory that is considered to be the root of the web. There are three distinct kind of navigation controls that can be found in an ASP.NET application:

- TreeView Control
- Menu Control
- SiteMapPath Control

Some Namespaces which are used for above Navigation controls are given below :

- Using.System.Web.UI.WebControls.TreeView;
- Using.System.Web.UI.WebControls.Menu;
- Using.System.Web.UI.WebControls.SiteMapPath;

4.4 TreeView Control :

An additional navigation control in ASP.NET, TreeView organises data in a hierarchical list format and can be used to display the data in this format. When TreeView is presented for the very first time, it presents the user with an overview of all of its nodes. The user has the ability to control it by adjusting a variable known as ExpandDepth. It is possible to either explicitly provide the contents of the TreeView control within the control itself or bind them to an external source :

- XML file
- web.sitemap file
- Database table

Properties of TreeView Control

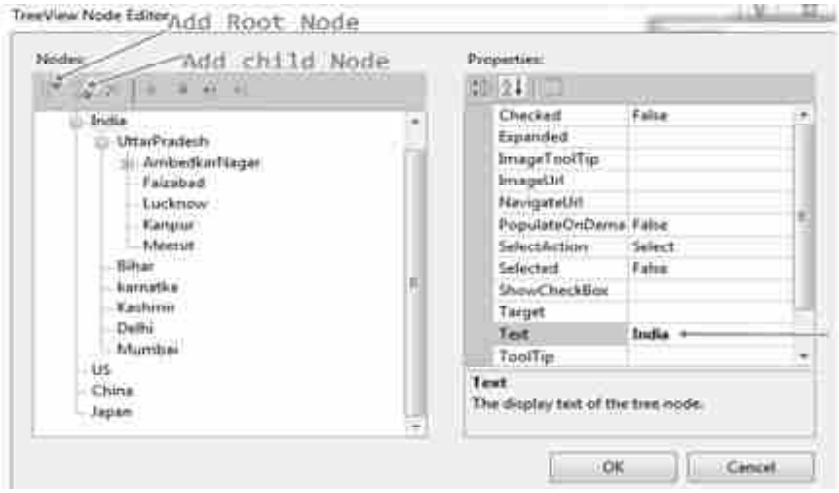
Property	Description
DataSourceID	This attribute is used to provide the data source that will be utilised when reading the data source from the sitemap file.
CssClass	The CSS class attribute that applies to the control can be specified by using this property.
ShowLines	The lines that will connect each unique item in the tree can be specified with the help of this property.
ExpandDepth	This property is used to specify the level at which items in the tree are expanded.

Creating a TreeView Control in ASP.NET with the corresponding procedure :

- **TreeView Node Editor Dialog Box :** The following is a list of the steps that need to be taken in order to generate the Tree structure on the page :

I Step : First open Visual Studio-->File-->New-->Website-->Select ASP.NET Empty Website -->OK-->Open solution explorer-->Add New Web Form-->Drag and Drop TreeView control from Toolbox as sown below :

II Step : Now, navigate to the properties of the TreeView control and click Click Nodes. In the following example, add the Root and child nodes :



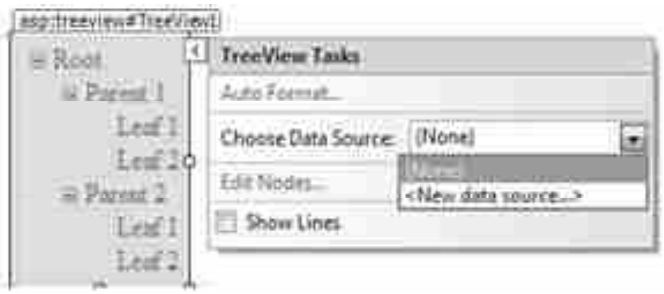
III Step : Now Run the Program(press F5). You will get the output like this :

- **TreeView based on XML Data :** The following is a list of the steps that need to be taken in order to generate the Tree structure on the page based on XML Data :

I Step : Now first add a Web Form and a XML File in the Solution Explorer-->now open the XML File and write the following codes as shown below-->now Click Save.

```
<?xml version="1.0" encoding="utf–8" ?>
<application>
  <homepage title="Country" value="default.aspx">
    <page title ="INDIA" value="default.aspx">
    <subpage title ="up" value="default.aspx"/>
    <subpage title ="delhi" value="default.aspx"/>
    <subpage title ="mumbai" value="default.aspx"/>
    <subpage title ="kolkata" value="default.aspx"/>
    </page>
    <page title ="US" value="default.aspx"/>
    <page title ="CHINA" value="default.aspx"/>
    <page title ="JAPAN" value="default.aspx"/>
    </homepage>
</application>
```

II Step : Now drag TreeView control and drop it on the Web Form --> Now Choose Data Source from TreeView control-->Select New data source as shown below :



III Step : Now choose XML File from the list as shown below and then -->OK.

IV Step : Now Browse your XML File as shown below-->OK.

V Step : Now click Edit TreeNode DataBindings..-->Select each page one by one -->click Add button -->set TextField = title from right side for each page-->click Apply as sown below :

VI Step : Now run the program. You will get the output like this :

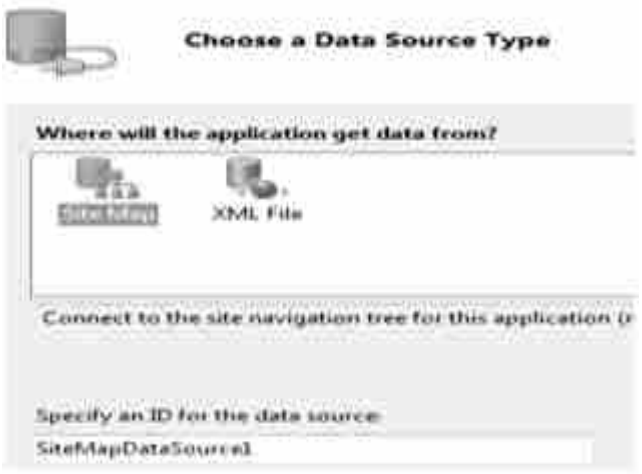
- **TreeView based on SiteMap Data :** The following is a list of the steps that need to be taken in order to generate the Tree structure on the page based on SiteMap Data:

I Step : Add a Web Form and a SiteMap in Solution Explorer as shown below-->Add

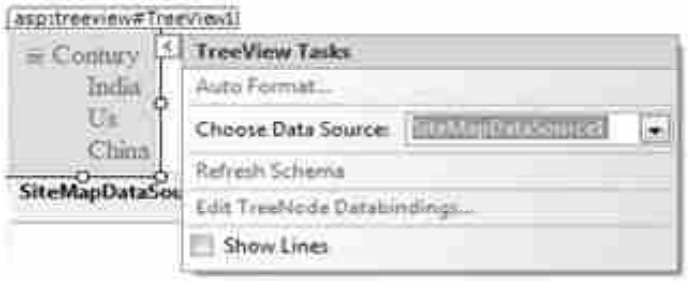
II Step : Open web.sitemap file and write the following codes, which are given below-->Save

```
<?xml version="1.0" encoding="utf–8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap–File–1.0" >
   <siteMapNode url="default.aspx" title="Contury" description="">
      <siteMapNode url="treeview1.aspx" title="India" description="" />
      <siteMapNode url="menu.aspx" title="Us" description="" />
      <siteMapNode url="menu1.aspx" title="China" description="" />
   </siteMapNode>
</siteMap>
```

III Step : Now drag and drop TreeView control on the Form-->Now choose Data Source-->select New data source-->Select SiteMap as sown below :



IV Step : Now click OK Button, you will see following output.



V Step : Now run the program. You will get the output like this:

❑ **Check Your Progress – 1 :**

1. _____ allows for the production of a page format that is standard.

   a. ImageMap b. SiteMap c. TreeView d. MasterPage

2. Which control is not used as a navigation control ?

   a. Button b. SiteMap c. TreeView d. MasterPage

4.5 Menu Control :

Another type of navigation control available in ASP.NET is called the Menu control, and it is the one responsible for displaying menus on web pages. In order to navigate the web site, you will need to utilize this control in conjunction with the SiteMapDataSource control. It presents two different kinds of menus: the static menu and the dynamic menu. In the menu Control, the Static menu will always be displayed; however, by default, only the menu items at the root levels will be shown. Only when the user moves the mouse pointer over the parent menu that is associated with the dynamic sub menu will the dynamic menu become visible.

Properties of Menu Control

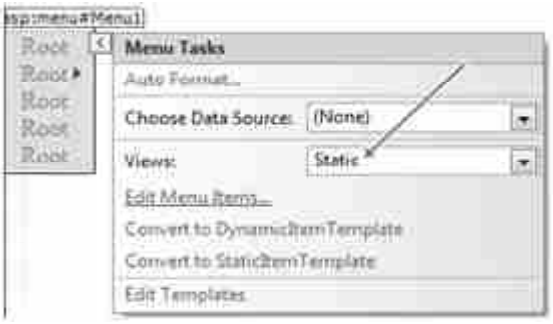
Property	Description
DataSourceID	With the help of this property, you can select the data source that will be utilised when the sitemap file is used as the data source.
CssClass	The CSS class attribute that applies to the control can be specified by using this property.
ImgeUrl	Utilizing this property allows one to select the image that will be shown next to the menu item.
Orientation	The alignment of menu controls can be specified with the help of this attribute, which is called "Orientation." It could be in a horizontal or vertical orientation.
Tooltip	This property allows you to customize the tooltip that appears when you move your mouse over an item in the menu.
Text	The wording that will be displayed in the menu can be specified by using this attribute.
NavigateUrl	When a menu item is clicked on, this attribute is utilized to indicate the destination location to which the user will be sent.
Target	Specifying the location of the target page can be done with this property. It may open in a new window or remain in the same window.
Value	Utilizing this property allows one to provide the one–of–a–kind id that should be utilized in server–side events.

In the example that was just presented, the <asp:Menu> control acts as a stand–in for a navigation menu that is generated by the server. The DataSourceId attribute of the control is what determines the data source for the control. It is connected to the <asp:SiteMapDataSource> control thanks to the id="nav1" attribute. The <ASP:SiteMapDataSource> control will automatically establish a connection to the sitemap file that is set as the default (web.sitemap).

The menu control can be utilized in either of these two ways.

- **Static menu :** It is utilized to display the parent menu items, as well as the sub menu items that correspond to those parent menu items, on the page. This indicates that it is the component that is responsible for displaying the full structure of the static menu.

I Step : Add a New Web Form in Solution Explorer -->Drag and Drop menu control on the Form--> now select Views = static as shown below :



II Step : Now click Edit Menu Items...-->Add parent and child nodes as shown below :

III Step : Now run the program, you will see following output.

Through the use of the site's static menu control, we are able to present the structure of the site both horizontally and vertically.

- **Dynamic menu :** It is utilized to display both the static menu and the dynamic menu on the website at the same time. It indicates that the menu will become visible on the site when the user moves the mouse pointer over it.

When the user moves the mouse pointer over the control, the data will immediately show on the site. Two different approaches will allow us to develop dynamic menu control :

- o Generate menu control using Xml Data source
- o Create a menu control by utilising SiteMap as the source data.

In this section, we will go over the steps necessary to add Menu and Sub Menu Links dynamically to our Web Application :

I Step : Create a database named Database.mdf. Add two tables in it first will be MainMenu and SubMenu. These will be look like this :

1. Table: MainMenu

Column Name	Data Type	Allow Nulls
MainMenuId	int	
MainMenu	nvarchar(MAX)	
FkContentId	nvarchar(MAX)	☑
MenuOrder	int	☑
IsActive	int	☑

1. Table: SubMenu

Column Name	Data Type	Allow Nulls
SubMenuId	int	
SubMenu	nvarchar(50)	
[Content]	nvarchar(MAX)	☑
MenuOrder	int	☑
ParentId	int	☑
IsActive	int	☑

II Step : Add ConnectionString in web.config. as given below. You will get the ConnectionString from the properties of Database.mdf.

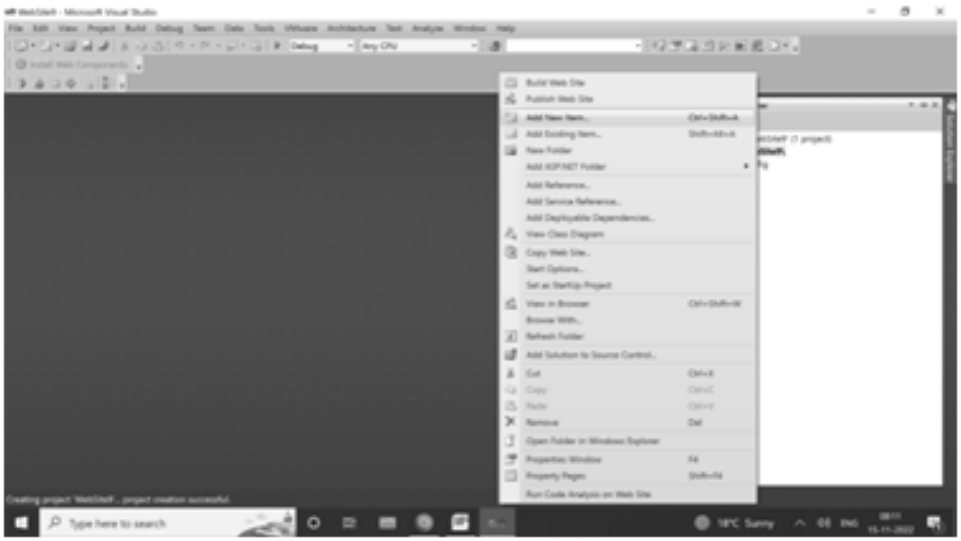
<connectionStrings>

<add name="ConnectionString" connectionString="Data Source=.; AttachDbFilename= DynamicallyControlCreationOfMenuSubmenuItems \App_Data\Database.mdf;User ID=**;Password=*****;Pooling=False" providerName=".NET Framework Data Provider for SQL Server"/>

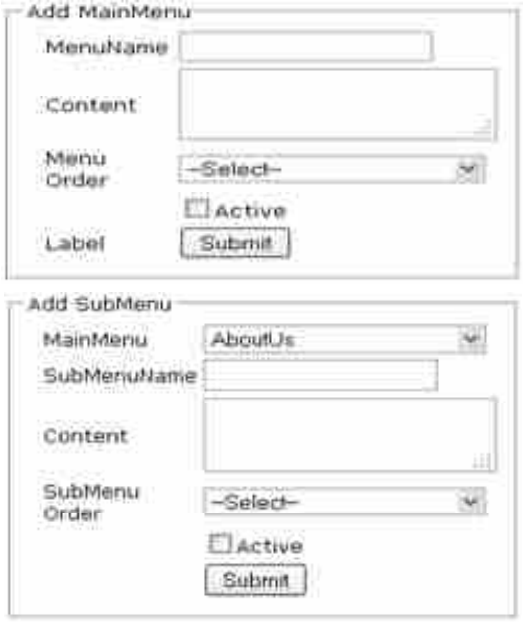
</connectionStrings>

III Step : Open Visual Studio -->File-->New Website

IV Step : Open Solution Explorer-->Add New item-->Web Form. Name it as CreateMenu.aspx



V Step : Create the page and incorporate the necessary controls into it. A user is able to enter a Menu name, content, order, and activate it, as is evident from the design of this layout. The same holds true for the sub menu, with the exception that the user will be required to choose a main menu.



VI Step : You can add items to the Main Menu by doing a double click on the submit button--> Now, write down the code in the following :

```
protected void btnMainmenu_Click(object sender, EventArgs e)
{
    int active;
    if (chkMActive.Checked)
    {
        active = 1;
    }
    else
    {
        active = 0;
    }
    SqlConnection conn = new SqlConnection(connection);
    string sql = "INSERT INTO MainMenu(MainMenu, MContent,
    MenuOrder, IsActive)VALUES('" + txtMainmenu.Text + "','" +
    txtMContent.Text + "'," + Convert.ToInt16 ( ddlMOrder.SelectedItem.
    Value) + "," + active + ")";
    SqlCommand cmd = new SqlCommand(sql, conn);
    conn.Open();
    cmd.CommandType = CommandType.Text;
    cmd.ExecuteNonQuery();
    conn.Close();
    FillddlMainMenu();
    Response.Redirect("SuccessMsg.aspx");
}
```

VII Step : Adding Submenus Items.

- In order to accomplish this, we must begin by filling out the dropdown on the Main Menu. The code can be found below.

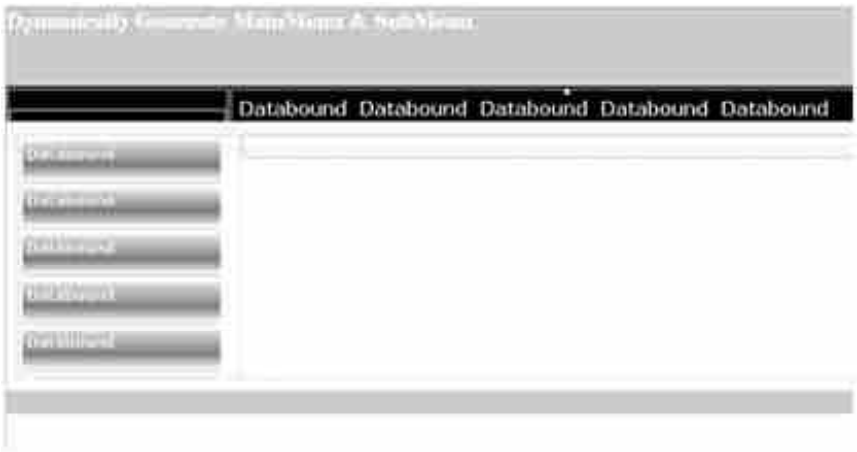
```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        FillddlMainMenu();
    }
}
public void FillddlMainMenu()
{
    SqlConnection conn = new SqlConnection(connection);
    DataSet DS = new DataSet();
    SqlDataAdapter da = new SqlDataAdapter();
```

```
SqlCommand cmd = new SqlCommand("Select MainMenuId,MainMenu
from MainMenu", conn);
da.SelectCommand = cmd;
da.Fill(DS);
ddlMainMenu.DataSource = DS;
ddlMainMenu.DataBind();
}
```

- Double click on Submit Button --> Write the below provided code in it

```
protected void btnSubmenu_Click(object sender, EventArgs e)
{
int active;
if (chkSActive.Checked)
{ active = 1; }
else
{ active = 0; }
SqlConnection conn = new SqlConnection(connection);
string sql = "INSERT INTO SubMenu(SubMenu, Content,
MenuOrder,ParentId, IsActive)VALUES('" + txtSubmenu.Text + "','"
+ txtSContent.Text + "'," + Convert.ToInt16(ddlSOrder.SelectedItem.
Value) + "," + Convert.ToInt16(ddlMainMenu.SelectedItem.Value) +
"," + active + ")";
SqlCommand cmd = new SqlCommand(sql, conn);
conn.Open();
cmd.CommandType = CommandType.Text;
cmd.ExecuteNonQuery();
conn.Close();
Response.Redirect("SuccessMsg.aspx");
}
```

VIII Step : To display the Menu Items at this time, we will need to add another page. Add two data lists for the submenus that are located horizontally under the main menu (Vertically).



IX Step : Now Create the code that will populate the Main Menu Items in the datalist so that it may be displayed as a menuBar. This code will be helpful in showing main menu items when the page loads.

```
protected void Page_Load(object sender, EventArgs e)
{
   FillMenu();
}
public void FillMenu()
{
   DataSet ds = new DataSet();
   SqlConnection conn = new SqlConnection(connection);
   SqlDataAdapter da = new SqlDataAdapter();
   SqlCommand cmd = new SqlCommand("Select * from MainMenu
   ORDER BY MenuOrder", conn);
   da.SelectCommand = cmd;
   da.Fill(ds);
   return ds;
   dlSubMenu.DataSource = ds1;
   dlSubMenu.DataBind();
}
```

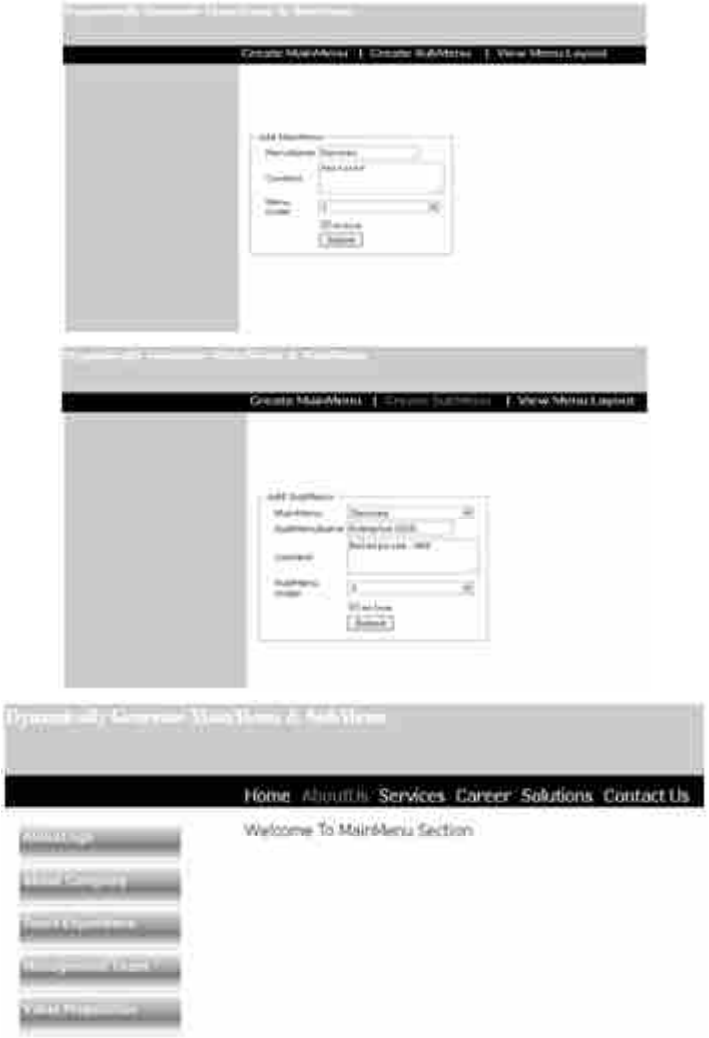
Now it is time to fill the submenus on the mainmenu link that was selected. On the datalist ItemCommand event, we are going to write some code.

X Step : datalist –>Properties –>Add ItemCommand Event.

```
protected void dlMenu_ItemCommand(object source, DataListCommand
EventArgs e)
{
   LinkButton lnkLink = e.Item.FindControl("lnkLink") as LinkButton;
   Label lblPageId = e.Item.FindControl("lblpageId") as Label;
   Session["Sublink"] = lblPageId.Text;
   if (lnkLink.Text != null)
   {
      Session["Choice"] = "Main";
      DataSet ds1 = new DataSet();
      SqlConnection conn = new SqlConnection(connection);
      SqlDataAdapter da = new SqlDataAdapter();
      SqlCommand cmd = new SqlCommand("Select * from SubMenu
      where IsActive=1 and ParentId='" + Convert.ToInt16(lblPageId.Text)
      + "' ORDER BY MenuOrder", conn);
      da.SelectCommand = cmd;
      da.Fill(ds1);
```

```
            dlSubMenu.DataSource = ds1;
            dlSubMenu.DataBind();
        }
    }
```

XI Step : Now run the program, you will see following output.



❑ **Check Your Progress – 2 :**

1. There are two kinds of Menu Control.

   a. Basic and Complicated b. Safety and Security

   c. Static and Dynamic d. None of these

4.6 SiteMapPath Control :

Site maps are a type of XML file that are employed primarily for the purpose of describing the logical structure of an online application. It specifies how each page in the web application should be laid out and how the pages should relate to one another. You have the ability to manage the navigation of your website in an effective manner by adding and removing pages from your site map whenever you see fit. The <sitemap> element serves as the file's root node and is denoted by the extension.sitemap. Site map files are denoted by the following :

Attributes of SiteMapPath Control

Property	Description
Title	It offers a written explanation of what the link is about.
Url	The location of the legitimate physical file is provided by it.
Description	It is used for the tooltip that appears on the link.

The SiteMapPath control illustrates the route used by the user to reach the current page. The path contains links that take you to the previous page that can be clicked on. The web is utilised by the sitemap path control. This control generates the navigation mechanism, which takes the form of a linear path and specifies the user's current position inside the navigation arrangement. The end user is given the ability to better understand his position in respect to the rest of the site.

Properties of SiteMapPath Control

Property	Description
PathSeparator	PathSeparator is a property that may be used to get or set the text of the out separator.
NodeStyle	This property is used to determine the appearance of all nodes that are going to be displayed.
RootNodeStyle	RootNodeStyle is the name of the property that determines the style applied to the node that is considered to be the absolute root.
PathDirection	This parameter is used to set the direction of the connections that are generated in the output.
CurrentNodeStyle	This attribute is responsible for setting the style of the node that represents the current page.
ShowToolTips	ShowToolTips is a property that may be used to configure the tooltip that is displayed for the control. Default value is true.
PathSeparatorStyle	This parameter is used to set the style of the path separator.

Example : The SiteMap File

```
<?xml version="1.0" encoding="ISO–8859–1" ?>
<siteMap>
   <siteMapNode title="Home" url="/aspnet/w3home.aspx">
      <siteMapNode title="Services" url="/aspnet/w3services.aspx">
         <siteMapNode title="Training" url="/aspnet/w3training.aspx"/>
         <siteMapNode title="Support" url="/aspnet/w3support.aspx"/>
      </siteMapNode>
   </siteMapNode>
</siteMap>
```

Creating a sitemap file according to these guidelines :

- The content of the XML file is required to be surrounded by a tag with the name <sitemap>.

- The <siteMap> tag is only permitted to have a single child node of type <siteMapNode> (the "home" page)
- There can be several child nodes associated with each <siteMapNode> (web pages)
- Every node in the sitemap has characteristics that specify the page title and URL.

Both the location of the sitemap file (which must be in the root directory of the web) and its attributes (which must be relative to the root directory) must be specified relative to the root directory. The SiteMapPath control on the web page must be implemented after completing a series of tasks. Which can be found down below :

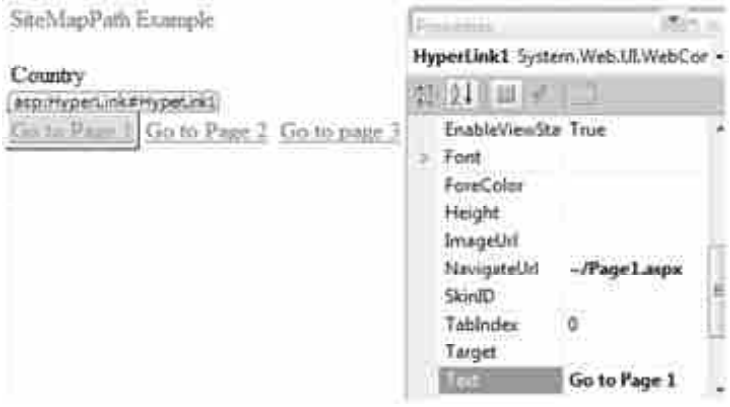
The following are some measures that need to be taken in order to put these ideas into practice on the site :

I Step : Add a Web Form in the Solution Explorer-->Drag and drop menu control on the Form --> choose Data Source -->Select XML File-->OK -->Browse XML File-->OK.

```
<?xml version="1.0" encoding="utf–8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap–File–1.0" >
   <siteMapNode url="SiteMap.aspx" title="Country" description="">
      <siteMapNode url="page1.aspx" title="India" description="" />
      <siteMapNode url="page2.aspx" title="China" description="" />
      <siteMapNode url="page3.aspx" title="US" description="" />
   </siteMapNode>
</siteMap>
```

II Step : Now, place the SiteMapPath control you just dragged and dropped on the web form (SiteMap.aspx). Now simply place the HyperLink control on the Form by dragging and dropping it, as seen below :

III Step : Add three more Web Forms (page1.aspx, page2.aspx, page3.aspx) in Solution Explorer-->Go properties of HyperLink Button control -->set NavigateUrl-->Write Text Information, as shown below :



IV Step : Now Go at page1.aspx -->drag and drop SiteMapPath control and HyperLink control on the Form as shown below-->Set the NavigateUrl of each HyperLink control as previously done.

V Step : Repeat same as IV Step in it for page2.aspx.

VI Step : Repeat same as IV Step in it for page3.aspx.

VII Step : Now run the program, you will see following output.

❑ **Check Your Progress – 3 :**

1. Which of these is not the property of SiteMapPath ?

   a. PathSeparator b. PathDirection c. ShowToolTips d. NavigateURL

2. Site maps are a type of _____ file.

   a. Data b. Word c. .cs d. XML

4.7 Let Us Sum Up :

In this unit we have learned about Master Pages, Navigation Controls, TreeView Control, Menu Control, and SiteMapPath Control. Master Pages layout pages. You can use a Master Page to create a three–column layout for all your website pages. Master Pages share material. Navigation Control is a set of user interface features that lets people visualise our structures and navigate them easily. Information Architecture's user interface is these controls.

Your site's navigation helps visitors find content. It keeps people on your site, stimulates reading, and improves user experience, increasing revenue and brand loyalty. Document headings, index entries, and disc files and directories

are displayed in a tree–view interface. Each item has a label, optional bitmapped picture, and subitems.

SiteMapDataSource and the ASP.NET Menu control traverse a website. Menu control features: Data binding lets menu items access hierarchical data sources. Sitemap Path control navigates website pages. It lists site pages. This map lists website pages.

4.8 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : d **2 :** a

- **Check Your Progress 2 :**

  1 : c

- **Check Your Progress 3 :**

  1 : d **2 :** d

4.9 Glossary :

1. **XML :** XML stands for eXtensible Markup Language
2. **Navigation :** Navigation involves determining the position, velocity, and attitude at a particular time.
3. **Menu :** A menu helps computer application users find information or perform a function.
4. **Node :** Nodes are connected by edges . Each node contains a value or data, and it may or may not have a child node.
5. **Root :** The first node of the tree is Root.

4.10 Assignment :

1. Explain the MasterPage Control in detail.

4.11 Activities :

1. Make a webpage using various Navigation Control: Menu Control, TreeView Control.

4.12 Case Study :

Study more about the various navigation controls in detail.

4.13 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

BLOCK SUMMARY :

In this block, you have learnt and understand about the basic of ASP.NET as a web development platform. The block gives an idea on the study and concept of ASP.NET frameworks for creating web applications. You have been well explained on the concepts of Web Controls.

The block detailed about the basic of ASP.NET web pages as main building block. The concept related to if statement and various other statements are also explained to you. The concept of various data types, operators explained in this block.

You got the knowledge of MasterPages and various navigation controls like siteMapPath, Menu, TreeView etc.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. What is Data Type ? List all data types.
2. Explain loop ? What is difference between while and do–while loop ?
3. Write a note on Application States ?
4. Write short note on switch statement ?

❖ **Long Questions :**

1. Write the execution process of Common Language Runtime ?
2. Write short note on Directives ?
3. Explain various objects in ASP.NET ?

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	1	2	3	4
No. of Hrs.				

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....

Dr. Babasaheb Ambedkar Open University Ahmedabad

BCAR-504

INTERNET PROGRAMMING (ASP.NET USING C#)

BLOCK 2 : VALIDATION CONTROL AND DATA SOURCE CONTROLS

VALIDATION CONTROL AND DATA SOURCE CONTROLS

Block Introduction :

Interacting with data-bound controls is the responsibility of a data source control. This control hides the complex data binding processes, supplies data for data-bound controls, and supports the execution of a variety of activities.

In this section, primary focuses will be the analysis and conceptualization of a wide variety of ASP.NET validation controls like RangeValidator, RequiredField Validator, RegularExpression Validator etc. These controls are widely used to give validations on the web controls of our webpages.

Asp.Net Details View enables adding SqlDataSource controls to pages with INSERT statements. Setting AutoGenerateInsertButton to true enables insert command. In Asp.Net, Bound field in GridView links a column to property in data source items to transmit data. It shows table data in database columns. This section covers grid view formatting with date, currency, and values. Grid Control and its visuals will be studied in the block. You'll understand Delete field links. This block introduces data binding and radio buttons. Create a custom column class and inserting a checkbox in gridview header template will also be covered. Gridview AutoGenerateColumns will be shown.

You will receive a suggestion on how to create a database. You will focus on learning and gaining an understanding of the SQLDataSource control throughout this section. You will also have the concept of the SELECT statement and the collection of Query Strings introduced to you at this time.

Block Objectives :

After learning this block, you will be able to understand :

- About Validator Class
- About Validation Controls
- About Data Base Systems
- About DataSource Web Controls
- About how to access data from the DataBase
- About how to display data on the WebPage
- About how to insert data from the WebPage to the DataBase

Block Structure :

Unit 5 : Studying other Web Controls and Validation Controls

Unit 6 : Working with Database Accessing and Displaying Data using Data Source Web Controls

Unit 7 : Working with Database

Unit
05

VALIDATION CONTROLS

UNIT STRUCTURE

5.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About the use of Validation Controls
- About BaseValidator Class
- About RequiredField Validator Control
- About Range Validator Control
- About RegularExpression Validator Control
- About Compare Validator Control
- About Custom Validator Control
- About Validation Summary Control

5.1 Introduction :

The process of validation is an essential component of any web application. The input from the user always needs to be checked before it is sent through the various layers of the application :

- Validation controls are typically used to in order to,
- Implement presentation logic.
- To validate user input data.

The validation process takes into account the data format, the data type, and the data range. There are two distinct types of validation :

- Client Side
- Serve Side

Client-side validation is beneficial, but its use requires us to rely on the support of individual browsers and scripting languages. Client-side validation is popular among consumers because it provides them with rapid feedback, making it a convenient validation method. The primary benefit is that it blocks a page from being posted back to the server until the client validation has been successfully carried out by the user.

From a developer's point of view, the server side is preferred since it cannot fail, it is independent of the browser, and it is not dependent on any scripting language.

You can utilize ASP.NET validation, which will guarantee validation on both the client and the server. It works on both ends; first, it will work on the validation of the client, and then it will work on the validation of the server. No matter what, server validation will always perform properly, regardless of whether or not client validation is carried out. Therefore, you have a check for the validity of safety. JavaScript was utilized for.client NET's scripting language. .NET utilizes the WebUIValidation.js file in order to perform client validation.

When developing ASP.NET Web sites that accept user input, the ability to verify that the information that users provide is accurate is an essential component. Validation controls are made available by ASP.NET. These controls offer a method that is not only simple but also highly effective for checking for mistakes and, if necessary, displaying alerts to the user.

Within ASP.NET, there are six distinct types of validation controls: RequiredFieldValidation Control, CompareValidator Control, RangeValidator Control, RegularExpressionValidator Control, CustomValidator Control and ValidationSummary.

5.2 BaseValidator Class :

The BaseValidator class is inherited by the validation control classes; as a result, the BaseValidator class's properties and methods are passed down to the validation control classes. Because of this, it would be beneficial to take a look :

Members	Description
ControlToValidate	Validate input control.
Display	Displays the error message.
EnableClientScript	Indicates client-side validation.
Enabled	Validator enable/disable.
ErrorMessage	String error.
Text	Validation error text.
IsValid	Validate control value.

SetFocusOnError	It specifies whether the focus should transfer to an invalid input control.
ValidationGroup	This control's validator group.
Validate()	This method revalidates and changes IsValid.

The BaseValidator class is responsible for implementing the fundamental aspects of all validation controls. Validation controls are used to validate user input that has been entered into an input control that is associated with the validation control. An error message is presented to the user by the validation control whenever the user inputs a value that does not pass validation. Because a validation control is distinct from an input control, you are free to place an error message anywhere on the page in relation to the control being validated. Validation controls are available in ASP.NET and can be used to perform a variety of different sorts of validation :

Validation Controls	**Description**
CompareValidator	Validates a value against another input, a constant, or a data type by setting the CompareValidator. Operator property to ValidationCompareOperator. DataTypeCheck.
CustomValidator	Validates a user-supplied custom validation routine.
RangeValidator	Validates whether a value is within a range of values.
RegularExpressionValidator	Validates a value using a regular expression.
RequiredFieldValidator	Validates that a value was entered in a required field.

Validation controls will always validate the input control on the server that they are associated with. Validation controls come with a full client-side implementation, which enables script-enabled web browsers to carry out client-side validation. The validation process is improved by client-side validation, which checks user input locally on the client computer before sending it to the server. Because of this, problems may be found and fixed on the client before the form is even sent to the server, which eliminates the need for the round trip of information that is required for server-side validation.

When validating diverse criteria, multiple validation controls can be combined with a single input control to achieve this. A TextBox control, for instance, is capable of having numerous validation controls applied to it. You can use a RequiredFieldValidator control to ensure that the user enters a value and a RangeValidator control to ensure that the value entered in the TextBox control is within a set range. You can also use a RangeValidator control to ensure that the value entered in the TextBox control is within a set range.

Many of the controls that are included with ASP.NET have the capability of posting data back to the server. Validation is carried out whenever one of these controls submits back to the server and has its CausesValidation attribute set to the true value. Some of these controls post back to the server only when the AutoPostBack property is set to true. The following controls are able to post data back to the server :

- System.Web.UI.WebControls.BulletedList
- System.Web.UI.WebControls.Button
- System.Web.UI.WebControls.CheckBox
- System.Web.UI.WebControls.CheckBoxList
- System.Web.UI.WebControls.DropDownList
- System.Web.UI.HtmlControls.HtmlButton
- System.Web.UI.HtmlControls.HtmlInputButton
- System.Web.UI.HtmlControls.HtmlInputImage
- System.Web.UI.WebControls.ImageButton
- System.Web.UI.WebControls.LinkButton
- System.Web.UI.WebControls.ListBox
- System.Web.UI.WebControls.RadioButtonList
- System.Web.UI.WebControls.TextBox

These controls each have a ValidationGroup property that, when set, validates just the validation controls within the specified group when the control initiates a post back to the server. When this property is set, only the validation controls within the specified group are validated. In order to group validation controls, you must first ensure that each validation control's ValidationGroup attribute is set to the same value.

Utilize the ControlToValidate attribute in order to associate an input control with a validation control in your application. Use the Text attribute of a validation control to specify the text that will be displayed in the control when validation fails. Utilizing a ValidationSummary control allows you to not only display a summary of all controls on the page that fail validation, but it also allows you to display the controls individually. Utilize the ErrorMessage attribute in order to provide the text that will be displayed in a ValidationSummary control. If you set the ErrorMessage property without setting the Text property, the value of the ErrorMessage property is also displayed in the validation control.

Before beginning any processing, you should ensure that the results of any server-side validation have been checked whenever validator controls are being used. The page calls the validator controls and aggregates the results of those calls into the page after a postback but before it calls your event functions. The property known as Page.IsValid. It is important that you verify, within your own code, that the Page. Prior to processing input, the IsValid property always returns true. Even though script-enabled browsers might stop a postback from happening on the client side if a validation check fails, you should still always check Page.IsValid in server code before processing verified data. This is because script-enabled browsers aren't always reliable.

Validation can also be performed by hand, if desired. Use the Page element to validate each validation control that is located on the page. Validate method. The Validate method of the control that is being verified allows for the validation of specific validation controls.

If you use the Page.IsValid property in a Page_Load method, you must first clearly call the Page.Validate method. As validation arises after the

Control.Load event for the page, but before the event handler for the Click or Command events, the Page.IsValid property is not reorganized until the Page.Validate method is called. As an alternate, you can place your code in the event handler for the Click or Command event as an alternative of the Page_Load method.

Validation controls are supported by only some of the Web server features. The following controls are examples of standard controls that can be validated :

- System.Web.UI.WebControls.DropDownList
- System.Web.UI.WebControls.FileUpload
- System.Web.UI.WebControls.ListBox
- System.Web.UI.WebControls.RadioButtonList
- System.Web.UI.WebControls.TextBox
- System.Web.UI.HtmlControls.HtmlInputFile
- System.Web.UI.HtmlControls.HtmlInputPassword
- System.Web.UI.HtmlControls.HtmlInputText
- System.Web.UI.HtmlControls.HtmlSelect
- System.Web.UI.HtmlControls.HtmlTextArea

It is necessary to apply the System.Web.UI.ValidationPropertyAttribute attribute to the control in order for the control to be validated when using input controls.

Make sure that the validator control and the control it is linked with are in the same panel whenever you use validator controls within an UpdatePanel control that inherit from BaseValidator. By making the SetFocusOnError attribute true, you will be able to direct the focus to the corresponding input control in the event that validation is unsuccessful. You need to override the following member when you inherit from the BaseValidator class: evaluateIsValid(). BaseValidator() sets a new instance of the BaseValidator class.

❑ **Check Your Progress – 1 :**

1. _____ class is inherited by the validation control classes ?

   a. Virtual b. BaseValidator

   c. Abstract d. None of these

For example :

public abstract class BaseValidator:System.Web.UI.WebControls.Label, System.Web.UI. IValidator

Example :

The following is an example of a form that all of the students in a school that is organised into four houses are going to have to fill out in order to pick a new president for the school. At this point, we validate the user input by making use of the validation controls. This is the form when viewed from the design perspective :


```
<form id="form1" runat="server">
    <table style="width: 66%;">
        <tr>
            <td class="style1" colspan="3" align="center">
            <asp:Label ID="lblmsg"
                Text="President Election Form : Choose your president"
                runat="server" />
            </td>
        </tr>
        <tr>
            <td class="style3">
                Candidate:
            </td>
            <td class="style2">
                <asp:DropDownList ID="ddlcandidate" runat="server" style=
                "width:239px">
                    <asp:ListItem>Please Choose a Candidate</asp:ListItem>
                    <asp:ListItem>M H Kabir</asp:ListItem>
                    <asp:ListItem>Steve Taylor</asp:ListItem>
                    <asp:ListItem>John Abraham</asp:ListItem>
                    <asp:ListItem>Venus Williams</asp:ListItem>
                </asp:DropDownList>
            </td>
            <td>
                <asp:RequiredFieldValidator ID="rfvcandidate"
                    runat="server" ControlToValidate ="ddlcandidate"
                    ErrorMessage="Please choose a candidate"
                    InitialValue="Please choose a candidate">
                </asp:RequiredFieldValidator>
```

```
    </td>
</tr>
<tr>
    <td class="style3">
        House:
    </td>
    <td class="style2">
        <asp:RadioButtonList ID="rblhouse" runat="server" Repeat
        Layout="Flow">
            <asp:ListItem>Red</asp:ListItem>
            <asp:ListItem>Blue</asp:ListItem>
            <asp:ListItem>Yellow</asp:ListItem>
            <asp:ListItem>Green</asp:ListItem>
        </asp:RadioButtonList>
    </td>
    <td>
        <asp:RequiredFieldValidator ID="rfvhouse" runat="server"
            ControlToValidate="rblhouse" ErrorMessage="Enter your
            house name" >
        </asp:RequiredFieldValidator>
        <br />
    </td>
</tr>
<tr>
    <td class="style3">
        Class:
    </td>
    <td class="style2">
        <asp:TextBox ID="txtclass" runat="server"></asp:TextBox>
    </td>
    <td>
        <asp:RangeValidator ID="rvclass"
            runat="server" ControlToValidate="txtclass"
            ErrorMessage="Enter your class (6 - 12)" MaximumValue
            ="12"
            MinimumValue="6" Type="Integer">
        </asp:RangeValidator>
    </td>
</tr>
<tr>
```

```
            <td class="style3">
                Email:
            </td>
            <td class="style2">
                <asp:TextBox ID="txtemail" runat="server" style="width: 250px">
                </asp:TextBox>
            </td>
            <td>
                <asp:RegularExpressionValidator ID="remail" runat="server"
                    ControlToValidate="txtemail" ErrorMessage="Enter your email"
                    ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
                </asp:RegularExpressionValidator>
            </td>
        </tr>
        <tr>
            <td class="style3" align="center" colspan="3">
                <asp:Button ID="btnsubmit" runat="server" onclick="btn submit_Click"
                    style="text-align: center" Text="Submit" style="width: 140px" />
            </td>
        </tr>
    </table>
    <asp:ValidationSummary ID="ValidationSummary1" runat="server"
        DisplayMode ="BulletList" ShowSummary ="true" HeaderText= "Errors:" />
</form>
```

The code behind the submit button :

```
protected void btnsubmit_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        lblmsg.Text = "Thank You";
    }
    else
    {
        lblmsg.Text = "Fill up all the fields";
    }
}
```

The validation control which we have used in this given example are explained one by one below.

5.3 RequiredFieldValidator Control :

The RequiredFieldValidator control makes certain that the required field does not contain any blanks by validating its contents. In most cases, it is connected to a text box so that input must be provided in the text box.

Make a particular input control a mandatory field by using this control. [Optional] The validation of the input control is considered to be invalid if, after leaving focus, the value of the control does not differ from the value stored in its InitialValue attribute.

One input control can have multiple validators connected with it at the same time. For instance, a RequiredFieldValidator may be used to validate the input to a control, and at the same time, a RangeValidator can be used to validate that the input falls within a certain data range. Both of these validators can be used in conjunction with one another.

When you use the RequiredFieldValidator control inside of an UpdatePanel control, you need to check to make sure that the validator control and the control it is connected with are located in the same panel. It initializes a new instance of the RequiredFieldValidator class with the RequiredFieldValidator() function.

The **syntax** of the RequiredFieldValidator Control is specified here :

```
<asp:RequiredFieldValidator ID="rfvUserName"
   runat="server" ControlToValidate ="txtUserName"
   ErrorMessage="Please enter the name of the User"
   InitialValue="Enter valid UserName">
</asp:RequiredFieldValidator>
```

Example :

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
   <title>RequiredField Validator Example</title>
<script runat="server">
   void ValidateBtn_Click(Object sender, EventArgs e)
   {
      if (Page.IsValid)
      {
         lblOutput.Text = "Required field is filled!";
      }
      else
      {
```

```
            lblOutput.Text = "Required field is empty!";
        }
    }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <h3>RequiredField Validator Example</h3>
        <table style="background-color:"Green"; padding:8">
            <tr valign="top">
                <td colspan="3">
                    <asp:Label ID="lblOutput"
                        Text="Fill in the required field below"
                        runat="server"
                        AssociatedControlID="txtUserName"/>
                    <br />
                </td>
            </tr>
            <tr>
                <td>
                    <asp:TextBox id="txtUserName"
                        runat="server"/>
                </td>
                <td>
                    <asp:RequiredFieldValidator id="RequiredFieldValidator2"
                        ControlToValidate="txtUserName"
                        Display="Static"
                        ErrorMessage="*"
                        runat="server"/>
                </td>
            </tr>
            <tr>
                <td></td>
                <td>
                    <asp:Button id="ValidateBtn"
                        Text="Validate"
                        OnClick="ValidateBtn_Click"
                        runat="server"/>
                </td>
```

```
            <td></td>
          </tr>
        </table>
      </form>
    </body>
    </html>
```

5.4 RangeValidator Control :

The RangeValidator control examines the value of an input control to determine whether or not it falls within a predetermined range.

When performing its validation function, the RangeValidator control relies on four critical features. The input control that has to be validated is stored in the ControlToValidate property. Both the lowest and maximum values of the valid range are given by the MinimumValue and MaximumValue attributes, respectively.

If the value that is supplied by the RangeValidator control's MaximumValue or MinimumValue property cannot be converted to the BaseCompareValidator that is specified, the RangeValidator control will throw an exception.

When you use the RangeValidator control inside of an UpdatePanel control, you need to check to see that the validator control and the control it is linked with are located in the same panel. A new instance of the RangeValidator class is created when you call the RangeValidator() function.

It possesses the following three distinct properties :

Properties	Description
Type	It identifies the type of data being stored. Currency, Date, Double, Integer, and String are the types of values that can be selected.
MinimumValue	It identifies the lowest possible value inside the range.
MaximumValue	It identifies the highest possible value inside the range.

The **syntax** of the RangeValidator control is as shown below:

```
<asp:RangeValidator ID="rvStandard" runat="server" ControlToValidate=
"txtStandard"
   ErrorMessage="Enter your Standard (1 - 12)" MaximumValue="12"
   MinimumValue="1" Type="Integer">
</asp:RangeValidator>
```

Example :

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head>
   <title>RangeValidator Example</title>
<script runat="server">
   void btnSubmit_Click(Object sender, EventArgs e)
   {
      if (Page.IsValid)
      {
         Label1.Text="Page is valid.";
      }
      else
      {
         Label1.Text="Page is not valid!!";
      }
    }
   </script>
</head>
<body>
   <form id="form1" runat="server">
      <h3>RangeValidator Example</h3>
      Enter your standard (1 – 12):
      <br />
      <asp:TextBox id="txtStandard"
         runat="server"/>
      <br />
      <asp:RangeValidator id="rvStandard"
         ControlToValidate="txtStandard"
         MinimumValue="1"
         MaximumValue="12"
         Type="Integer"
         EnableClientScript="false"
         Text="Standard should be in between 1 to 12!"
         runat="server"/>
      <br /><br />
      <asp:Label id="Label1"
         runat="server"/>
      <br /><br />
      <asp:Button id="btnSubmit"
         Text="Submit"
```

```
            OnClick="btnSubmit_Click"
            runat="server"/>
        </form>
    </body>
    </html>
```

❑ **Check Your Progress – 2 :**

1. Which one is not the property of RangeValidator ?

   a. RequiredFieldValidator b. Minimum Value

   c. Maximum Value d. All of these

5.5 RegularExpressionValidator Control :

A regular expression is used to describe a pattern, and the RegularExpressionValidator control determines whether or not the value of an input control matches that pattern. Checking for predictable sequences of characters, such as those found in email addresses, telephone numbers, and postal codes, is possible thanks to this sort of validation.

If the input control is blank, validation is considered to have been successful. Use an additional RequiredFieldValidator control in conjunction with the RegularExpressionValidator control if the related input control requires a value in order to function properly.

Validation occurs on both the server and the client, unless the user's browser does not support client-side validation or the user chooses to disable client-side validation in their browser settings (by setting the EnableClientScript property to false).

When you use the RegularExpressionValidator control inside of an UpdatePanel control, you need to ensure that both the validator control and the control it is connected with are located within the same panel.

A new instance of the RegularExpressionValidator class is created whenever the RegularExpressionValidator() function is called.

The RegularExpressionValidator provides the capability to validate the input text by performing a pattern match against a regular expression. The ValidationExpression property is where the regular expression is configured to be used.

The syntax constructs for regular expressions that are most frequently used are summarised in the following table :

Escape Characters	Description
\b	Identical to a backspace character.
\t	It corresponds to a tab.
\r	Identical to a carriage return in format.
\v	It corresponds to a vertical tab.
\f	A form feed that is a match.
\n	Suitable for a new line.
\	Escape character.

In addition to matching a single character, it is also possible to provide an entire class of characters that can be matched. These characters are referred to as metacharacters.

MetaCharacters	Description
.	Matches any character except \n.
[abcd]	Excludes any character in the set.
[^abcd]	Identical to a carriage return in format.
[2-7a-mA-M]	Matches any character specified in the range.
\w	Matches any alphanumeric character and underscore.
\W	Matches any non-word character.
\s	Matches whitespace characters like, space, tab, new line etc.
\S	Matches any non-whitespace character.
\d	Matches any decimal character.
\D	Matches any non-decimal character.

It was suggested that quantifiers be included so that the amount of times a character might appear could be specified :

Quantifiers	Description
*	Zero or more matches.
+	One or more matches.
?	Zero or one matches.
{N}	N matches.
{N,}	N or more matches.
{N,M}	Between N and M matches.

The **syntax** of the RegularExpressionValidator control is as shown below:

```
<asp:RegularExpressionValidator ID="string" runat="server" Error Message="string"
    ValidationExpression="string" ValidationGroup="string">
</asp:RegularExpressionValidator>
```

Example :

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>RegularExpressionValidator Example</title>
<script runat="server">
    void btnValidate_Click(Object sender, EventArgs e)
    {
```

```
        if (Page.IsValid)
        {
            lblOutput.Text = "Page is Valid.";
        }
        else
        {
            lblOutput.Text = "Page is InValid.";
        }
    }
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <h3>RegularExpressionValidator Example</h3>
        <table style="background-color:"Green"; padding:8">
            <tr valign="top">
                <td colspan="3">
                    <asp:Label ID="lblOutput"
                        Text="Enter a 5-digit ZIP Code"
                        runat="server"
                        AssociatedControlID="txtZIPCode"/>
                </td>
            </tr>

            <tr>
                <td colspan="3">
                    <b>Personal Information</b>
                </td>
            </tr>
            <tr>
                <td align="right">
                    Zip Code:
                </td>
                <td>
                    <asp:TextBox id="txtZIPCode"
                        runat="server"/>
                </td>
                <td>
                    <asp:RegularExpressionValidator id="revZIPCode"
```

```
                ControlToValidate="txtZIPCode"
                ValidationExpression="\d{5}"
                Display="Static"
                ErrorMessage="ZIP code must be 5 numeric digits"
                EnableClientScript="False"
                runat="server"/>
            </td>
        </tr>
        <tr>
            <td></td>
            <td>
                <asp:Button text="btnValidate"
                    OnClick="btnValidate_Click"
                    runat="server" />
            </td>
            <td></td>
        </tr>
    </table>
</form>
</body>
</html>
```

5.6 CompareValidator Control :

You can compare the value that the user enters into an input control, such as a TextBox control, with the value that is typed into another input control or with a constant value by using the CompareValidator control. If the value of the input control satisfies the requirements set forth by the Operator, ValueToCompare, and/or ControlToCompare attributes, the CompareValidator control is considered to have successfully validated the control.

You can also make use of the CompareValidator control to determine whether the value that was entered into an input control can be converted to the data type that was provided by the BaseCompareValidator.

Setting the ControlToValidate property will allow you to specify the input control that should be validated. Set the ControlToCompare property to specify the control you wish to compare the selected input control with, if you want to compare a certain input control with another input control.

If the value that was entered into the input control that was specified by the ControlToCompare property cannot be converted to the data type that was provided by the BaseCompareValidator, then the BaseCompareValidator will throw an error.

If the value that was entered into the input control that was being validated has the Type attribute, but that value can be transformed, then the input control that was being validated is regarded to be valid. As a result, you

need to additionally position a CompareValidator or RangeValidator control on the input control that is indicated by the ControlToCompare attribute.

You have the option of comparing the value of an input control to a constant value as an alternative to comparing the value of an input control with the value of another input control. Setting the ValueToCompare property will allow you to provide the value of the constant that will be compared.

An exception is generated whenever a successful conversion of the value that is specified by the BaseCompareValidator.Type property to the data type that is specified by the ValueToCompare property is not possible. Before you programmatically assign a value to the ValueToCompare property, you need to make sure that the value's data type has been checked.

Setting the ControlToCompare property as well as the ValueToCompare property at the same time is not allowed. You have the option of comparing the value of one input control to the value of another input control or to a value that is always the same. In the event that both properties are active, the ControlToCompare property will be given priority.

You can describe the kind of comparison you want to carry out by using the Operator property, such as "greater than," "equal to," and so on. In the event that you change the value of the Operator attribute to ValidationCompare Operator. The CompareValidator control ignores the ControlToCompare and ValueToCompare properties. Instead, it merely indicates whether the value that was entered into the input control can be converted to the data type that was specified by the BaseCompareValidator. DataTypeCheck is a property of the CompareValidator control.

The data types of both comparison values can be specified with the help of the Type attribute. Both of the values will first have their data types changed to this one automatically before the comparison procedure is carried out.

When you use the CompareValidator control inside of an UpdatePanel control, you need to ensure that both the validator control and the control it is connected with are located within the same panel.

A new instance of the CompareValidator class is created whenever the CompareValidator() function is called.

It possesses the following distinct properties :

Properties	Description
Type	It identifies the type of data being stored.
ControlToCompare	It specifies the value of the input control to compare with.
ValueToCompare	It specifies the constant value to compare with.
Operator	It specifies the comparison operator, the available values are : • Equal • NotEqual • GreaterThan • GreaterThanEqual • LessThan • LessThanEqual and • DataTypeCheck.

The **syntax** of the CompareValidator control is as shown below:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
   ErrorMessage="CompareValidator">
</asp:CompareValidator>
```

Example :

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
   <title>CompareValidator Example</title>
<script runat="server">
      void Button_Click(Object sender, EventArgs e)
      {
         if (Page.IsValid)
         {
            lblOutput.Text = "Result: Valid!";
         }
         else
         {
            lblOutput.Text = "Result: Not valid!";
         }
      }
      void Operator_Index_Changed(Object sender, EventArgs e)
      {
         Compare1.Operator = (ValidationCompareOperator) ListOperator.
         SelectedIndex;
         Compare1.Validate();
      }
         void Type_Index_Changed(Object sender, EventArgs e)
         {
            Compare1.Type = (ValidationDataType) ListType.Selected
            Index;
            Compare1.Validate();
         }
      </script>
   </head>
   <body>
      <form id="form1" runat="server">
```

```
<h3>CompareValidator Example</h3>
<br />
Enter a value in each textbox. Select a comparison operator<br />
and data type. Click "Validate" to compare values.
<table style="background-color:#eeeeee; padding:10">
  <tr valign="top">
    <td>
      <h5>String 1:</h5>
      <asp:TextBox id="TextBox1"
        runat="server"/>
    </td>
    <td>
      <h5>Comparison Operator:</h5>
      <asp:ListBox id="ListOperator"
        OnSelectedIndexChanged="Operator_Index_
        Changed"
        runat="server">
        <asp:ListItem Selected="True"Value="Equal">Equal
        </asp:ListItem>
        <asp:ListItem Value="NotEqual">NotEqual</asp:
        List Item>
        <asp:ListItem Value="GreaterThan">GreaterThan</
        asp:ListItem>
        <asp:ListItem Value="GreaterThanEqual">Greater
        ThanEqual</asp:ListItem>
        <asp:ListItem Value="LessThan">LessThan</asp:
        List Item>
        <asp:ListItem Value="LessThanEqual">LessThan
        Equal</asp:ListItem>
        <asp:ListItem Value="DataTypeCheck">DataType
        Check</asp:ListItem>
      </asp:ListBox>
    </td>
    <td>
      <h5>String 2:</h5>
      <asp:TextBox id="TextBox2"
        runat="server"/>
      <br />
      <asp:Button id="Button1"
        Text="Validate"
        OnClick="Button_Click"
        runat="server"/>
```

```
            </td>
        </tr>
        <tr>
            <td colspan="3" align="center">
                <h5>Data Type:</h5>
                <asp:ListBox id="ListType"
                    OnSelectedIndexChanged="Type_Index_Changed"
                    runat="server">
                    <asp:ListItem Selected="true" Value="String"
                    >String</asp:ListItem>
                    <asp:ListItem Value="Integer" >Integer</asp:List
                    Item>
                    <asp:ListItem Value="Double" >Double</asp:List
                    Item>
                    <asp:ListItem Value="Date" >Date</asp:ListItem>
                    <asp:ListItem Value="Currency" >Currency</asp:
                    ListItem>
                </asp:ListBox>
            </td>
        </tr>
    </table>
    <asp:CompareValidator id="Compare1"
        ControlToValidate="TextBox1"
        ControlToCompare="TextBox2"
        EnableClientScript="False"
        Type="String"
        runat="server"/>
    <br />
    <asp:Label id="lblOutput"
        Font-Names="verdana"
        Font-Size="10pt"
        runat="server"/>
  </form>
</body>
</html>
```

5.7 CustomValidator Control :

Make use of the CustomValidator control in order to supply an input control with a validation function that is user-defined. Because the CustomValidator control is a distinct control from the input control that it verifies, you will have complete command over the location at which the validation message appears.

Validation controls will always operate on the server to perform validation. They also offer a full client-side implementation, which enables script-enabled browsers (such Microsoft Internet Explorer 4.0 and subsequent versions) to carry out validation directly on the client. The validation process is improved by client-side validation, which checks user input locally on the client computer before sending it to the server. Because of this, problems may be found and fixed on the client before the form is even sent to the server, which eliminates the need for the round trip of information that is required for server-side validation.

You must supply a handler for the ServerValidate event, which is responsible for carrying out the validation, in order to develop a function that conducts server-side validation. Using the Value property of the ServerValidateEventArgs object, which is a parameter that is supplied into the event handler, you may have access to the string that is contained within the input control that is being validated. After the validation is complete, the result of the check is placed in the IsValid field of the ServerValidateEventArgs object.

Adding the server-side validation function that was discussed earlier is the first step in the process of creating a client-side validation function. The client-side validation script function needs to be included as the next step to the ASP.NET (.aspx) page.

If you're going to use JScript, the function has to be written in this form :

```
function ValidationFunctionName(source, arguments)
```

A reference to the element that was produced for the CustomValidator control can be found within the source argument. This enables you to exercise programmatic control over the tag, allowing you to do things like edit the InnerHtml attribute. The arguments parameter is an object that contains two attributes referred to respectively as Value and IsValid. You will be able to acquire the value of the control to validate using this parameter, and you will also be able to indicate whether or not the value is valid based on the custom validation method you have created.

You can define the name of the client-side validation script function that is connected with the CustomValidator control by making use of the property known as ClientValidationFunction. Since the script function is executed on the client, the function itself needs to be written in a language that is supported by the target browser. Some examples of such languages include VBScript and JScript.

When using the CustomValidator control within an UpdatePanel control, you need to ensure that both the validator control and the control with which it is related are located within the same panel.

In the same way that server-side validation operates, accessing the string from the input control that has to be validated is accomplished by using the Value property of the arguments parameter. By putting a value in the IsValid property of the arguments parameter, you can return the outcome of the validation.

Before beginning any processing, you should ensure that the results of any server-side validation have been checked whenever validator controls are being used. The page will contact the validator controls after a postback, but before it will call any of your event methods. It will then aggregate the results

of those calls into the Page.IsValid property. Before processing data, you should ensure that your own code checks that the Page.IsValid attribute returns the correct value (true). Even though script-enabled browsers might stop a postback from happening on the client side if a validation check fails, you should still always check Page.IsValid in server code before processing verified data. This is because script-enabled browsers aren't always reliable.

When validating diverse criteria, multiple validation controls can be combined with a single input control to achieve this. For instance, you are able to apply additional validation controls to a TextBox control if you want to give the user the ability to enter the quantity of things that they want to add to a shopping cart. You can use a CustomValidator control to guarantee that the value specified is less than the amount in inventory, and you can use a RequiredFieldValidator control to ensure that the user puts a value into the TextBox control. Both of these controls can be found in the Controls folder.

It is possible to utilise a CustomValidator control even if the ControlToValidate attribute has not been set. This is possible. This is a typical practise for validating a large number of input controls at once or validating input controls that cannot be used with validation controls, like the CheckBox control. In this particular scenario, the Value property of the arguments parameter that is sent to the event handler for the ServerValidate event as well as to the client-side validation function always contains an empty string (""). On the other hand, these validation functions are still called whenever it is necessary to determine validity on either the server or the client. In order to gain access to the value that has to be validated, you must first make a programmatic reference to the input control that you wish to validate and then retrieve the value from the property that is relevant. If you want to validate a CheckBox control on the server, for instance, you should avoid setting the ControlToValidate attribute of the validation control and instead use the following code as the handler for the ServerValidate event instead.

```
void ServerValidation (object source, ServerValidateEventArgs args)
{
   args.IsValid = (CheckBox1.Checked == true);
}
```

A new instance of the CustomValidator class is created whenever the CustomValidator() function is called.

The **syntax** of the CustomValidator control is as shown below:

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
   ClientValidationFunction=.cvf_func. ErrorMessage="CustomValidator">
</asp:CustomValidator>
```

Example :

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
```

```
    <title>CustomValidator ServerValidate Example</title>
<script runat="server">
    void btnValidate_OnClick(object sender, EventArgs e)
    {
        // Display whether the page passed validation.
        if (Page.IsValid)
        {
            Message.Text = "Page is valid.";
        }
        else
        {
            Message.Text = "Page is not valid!";
        }
    }
    void ServerValidation(object source, ServerValidateEventArgs args)
    {
        try
        {
            // Test whether the value entered into the text box is even.
            int i = int.Parse(args.Value);
            args.IsValid = ((i%2) == 0);
        }
        catch(Exception ex)
        {
            args.IsValid = false;
        }
    }
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <h3>CustomValidator ServerValidate Example</h3>
        <asp:Label id="Message"
            Text="Enter an even number:"
            Font-Names="Verdana"
            Font-Size="10pt"
            runat="server"
            AssociatedControlID="Text1"/>
```

```
<br />
<asp:TextBox id="Text1"
    runat="server" />
<asp:CustomValidator id="CustomValidator1"
    ControlToValidate="Text1"
    Display="Static"
    ErrorMessage="Not an even number!"
    ForeColor="green"
    Font-Names="verdana"
    Font-Size="10pt"
    OnServerValidate="ServerValidation"
    runat="server"/>
<br />
<asp:Button id="btnValidate"
    Text="Validate"
    OnClick="btnValidate_OnClick"
    runat="server"/>
</form>
</body>
</html>
```

The following sample of code serves as an example of how to construct a CustomValidator control on the client side.

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<html>
<head>
<script runat="server">
    void btnValidate_OnClick(object sender, EventArgs e)
    {
        // Display whether the page passed validation.
        if (Page.IsValid)
        {
            Message.Text = "Page is valid.";
        }
        else
        {
            Message.Text = "Page is not valid!";
        }
    }
    void ServerValidation(object source, ServerValidateEventArgs args)
    {
```

```
        try
        {
            // Test whether the value entered into the text box is even.
            int i = int.Parse(args.Value);
            args.IsValid = ((i%2) == 0);
        }
        catch(Exception ex)
        {
            args.IsValid = false;
        }
    }
    </script>
</head>
<body>
    <form id="Form1" runat="server">
        <h3>CustomValidator ServerValidate Example</h3>
        <asp:Label id="Message"
            Text="Enter an even number:"
            Font-Name="Verdana"
            Font-Size="10pt"
            runat="server"/>
        <p>
        <asp:TextBox id="Text1"
            runat="server" />
        <asp:CustomValidator id="CustomValidator1"
            ControlToValidate="Text1"
            ClientValidationFunction="ClientValidate"
            OnServerValidate="ServerValidation"
            Display="Static"
            ErrorMessage="Not an even number!"
            ForeColor="green"
            Font-Name="verdana"
            Font-Size="10pt"
            runat="server"/>
        <p>
        <asp:Button id="btnValidate"
            Text="Validate"
            OnClick="btnValidate_OnClick"
            runat="server"/>
```

```
    </form>
</body>
</html>
<script language="javascript">
    function ClientValidate(source, arguments)
    {
        if (arguments.Value % 2 == 0 ){
            arguments.IsValid = true;
        } else {
            arguments.IsValid = false;
        }
    }
</script>
```

❑ **Check Your Progress – 3 :**

1. Which property is these is different in CompareValidator as compare to another Validation Controls ?

   a. ControlToValidate b. ValueToCompare

   c. ControlToCompare d. Both b and c

5.8 ValidationSummary Control :

The ValidationSummary class is utilised to compile, in a single location, a summary of the error messages generated by all of the validators present on a Web page. If you assign the ValidationSummary control to a validation group on a Web page and set the ValidationGroup property, you will be able to compile a summary of the error messages generated by a collection of validators on the page.

Depending on what value is used for the DisplayMode property, the summary may be shown in the form of a list, a bulleted list, or a single paragraph. If the ShowSummary property and the ShowMessageBox property are both set to true, the summary will be displayed on the web page and in the message box, respectively. When you use the ValidationSummary control inside of an UpdatePanel control, you need to check to see that the validator control and the control with which it is linked are located in the same panel.

A new instance of the ValidationSummary class is created whenever the ValidationSummary() function is called. The error message is comprised of the two attributes shown below, both of which are inclusive of one another:

Properties	Description
ShowSummary	Displays the error messages in the format that was specified.
ShowMessageBox	A separate window that displays the error messages is displayed.

The **syntax** of the ValidationSummary control is as shown below:

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
    DisplayMode = "BulletList" ShowSummary = "true" HeaderText="Errors:"/>
```

Example :

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
   <title>ValidationSummary Sample</title>
</head>
<body>
   <h3>ValidationSummary Sample</h3>
   <br />
   <form id="form1" runat="server">
      <table cellpadding="10">
         <tr>
            <td>
               <table style="background-color:#eeeeee; padding:10">
                  <tr>
                     <td colspan="3">
                        <b>Credit Card Information</b>
                     </td>
                  </tr>
                  <tr>
                     <td align="right">
                        Card Type:
                     </td>
                     <td>
                        <asp:RadioButtonList id="RadioButtonList1"
                           RepeatLayout="Flow"
                           runat="server">
                        <asp:ListItem>MasterCard</asp:ListItem>
                        <asp:ListItem>Visa</asp:ListItem>
                     </asp:RadioButtonList>
                  </td>
                  <td align="center" rowspan="1">
                     <asp:RequiredFieldValidator
                        id="RequiredFieldValidator1"
                        ControlToValidate="RadioButtonList1"
                        ErrorMessage="Card Type. "
```

```
                Display="Static"
                InitialValue="" Width="100%" runat="server">
                *
                </asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td align="right">
                Card Number:
            </td>
            <td>
                <asp:TextBox id="TextBox1" runat="server" />
            </td>
            <td>
                <asp:RequiredFieldValidator
                    id="RequiredFieldValidator2"
                    ControlToValidate="TextBox1"
                    ErrorMessage="Card Number. "
                    Display="Static"
                    Width="100%" runat="server">
                *
                </asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td></td>
            <td>
                <asp:Button
                    id="Button1"
                    text="Validate"
                    runat="server" />
            </td>
            <td></td>
        </tr>
    </table>
</td>
<td valign="top">
    <table cellpadding="20">
        <tr>
```

```
                <td>
                    <asp:ValidationSummary
                        id="valSum"
                        DisplayMode="BulletList"
                        runat="server"
                        HeaderText="You must enter a value in the following fields:"
                        Font-Names="verdana"
                        Font-Size="12"/>
                    </td>
                </tr>
            </table>
        </td>
    </tr>
</table>
</form>
</body>
</html>
```

❑ **Check Your Progress – 4 :**

1. Which of the following validation is used to prevents a page from being post back to the server ?

   a. Client side validation b. Server side validation

   c. Both of these d. None of these

2. We use Validation Controls to :

   a. Enter data b. Validate Web Pages

   c. Enter Controls d. None of these

5.9 Let Us Sum Up :

Validation is crucial for any web project. Before being passed through the application's layers: user input must be verified, Validation controls implement presentation logic and validate user input.

Validation considers data format, type, and range. Two validations exist: Client and Server s ide. Client-side validation is helpful, but it relies on browsers and scripting languages. Client-side validation is popular with consumers because it delivers quick feedback. It stops a page from being posted to the server until the user completes client validation.

Developers favour the server side since it can't fail, is independent of the browser, and doesn't require scripting. ASP.NET validation ensures client and server validation. It validates the client and the server. Regardless of client validation, server validation will always work. So, safety is checked. .NET uses WebUIValidation.js to perform client validation.

Developing ASP.NET Web sites that take user input requires the ability to verify user input. ASP.NET has validation controls. RequiredFieldValidation,

CompareValidator, RangeValidator, RegularExpressionValidator, CustomValidator, and ValidationSummary are ASP.NET validation controls. These validation controls offer a simple, effective way to check for errors and warn the user, if necessary.

5.10 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : b

- **Check Your Progress 2 :**

  1 : a

- **Check Your Progress 3 :**

  1 : d

- **Check Your Progress 4 :**

  1 : a **2 :** b

5.11 Glossary :

1. **Class :** A class is a template definition of the method s and variable s in a particular kind of object.
2. **Validation :** It means checking the accuracy and quality of source data before using, importing or otherwise processing data.
3. **Range :** It ensures the values fall within a range limit.
4. **Compare :** The process of examining the similarities or differences of data or files and then listing them.
5. **Custom :** It provides a user-defined validation function for an input control.

5.12 Assignment :

1. Explain vaiour types of Validation Controls with suitable example.

5.13 Activities :

1. Develop a website and use comapare validator and range validator.

5.14 Case Study :

Study more about Validation Controls in detail.

5.15 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

Unit

06

WORKING WITH DATABASE, ACCESSING AND DISPLAYING DATA USING DATA SOURCE WEB CONTROLS

UNIT STRUCTURE

6.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About various Database Systems
- About About the Key Constraints
- About how to create Database

- About how to create tables in the Database
- About how to insert records in the Database tables
- About DataSource Controls
- About SqlDataSource Control
- About how to connect the database with our Web Application
- About the Select Statement Configuration
- About how to test the QueryString
- About the Query Behind the Code etc.

6.1 Introduction :

The complex steps involved in accessing data are concealed by ASP.NET, which also offers a significantly higher level of classes and objects via which the data may be accessed in an uncomplicated manner. The complexity of the connection, data retrieval, data querying, and data manipulation coding is hidden behind these classes.

The various ASP.NET control objects may communicate with the backend data source thanks to a technology called ADO.NET, which acts as a bridge between the two.

Access to and utilization of data from the following sources is made possible by ASP.NET :

- Databases
- XML documents
- Business Objects
- Flat files

The following databases will be displayed when you make your selection in the Solution Explorer: Access, SQL Server, Oracle, MySQL.

The main attributes of using database connectivity with ASP.NET are as follows: Object Navigator, Table Designer, Data Editor, Editors for Structured Query Language, Query Builder, Data Visualization, Database Diagram, SQL Profiler Code Manager, Search Database Objects, Search Table Data, Data Import and Export, SQL Profiler Code Manager, Search Database Objects, Search Table Data

6.2 Various Database Systems :

ASP.NET provides many options for storing, retrieving, and displaying data. This area focuses on relational databases. These are the options for database access in ASP.NET:

- Database Management Systems (DBMS)
- SQL Server Editions
- Object-Relational Mappers (ORM)
- Entity Framework Development Workflows
- LINQ versus SQL
- Accessing Data through a Web Service
- Working with Data in ASP.NET Web Forms Applications

- Working with Data in ASP.NET MVC applications
- Working with Data in ASP.NET Web Pages Applications

In this unit we will learn about DBMS and ORM in detail.

DataBase Management System (DBMS) : ADO.NET is the technology that enables a connection to be made to a relational database from within ASP.NET code that is executed on a server. Through the utilisation of data provider software, ADO.NET is able to interface with a database management system (DBMS) such as SQL Server or Oracle. You are able to connect to the following databases with the help of Microsoft data providers :

- SQL Server, including LocalDB and SQL Server Express.
- Compact SQL Server (SQL Server)
- Any database that is compatible with ODBC or OLEDB that can be used in a web application is considered a relational database.

There are various data sources available from third-party software suppliers that you can use. MySQL, SQLite, Oracle, and DB2 are a few examples of well-known relational databases for which data providers are readily available.

Choose SQL Server as your relational database for an ASP.NET web application if you don't have any specific requirements that require you to use a different option. The following are some of the many reasons why you should go with SQL Server :

- Microsoft provides support for the SQL Server database.
- SQL Server is compatible with a variety of other data access technologies developed by Microsoft, including the Entity Framework.
- Working with SQL Server is made easier with the integrated tools that come along with Visual Studio. You are able to construct databases with SQL Server Data Tools (SSDT), as well as manipulate schema and data, write and run scripts, debug databases and database updates, and deploy databases and database updates.
- Visual Studio comes equipped with web deployment options that make it easier to deploy SQL Server databases alongside web projects. SQL Server is the default database used for the ASP.NET membership database when using the web project templates for Visual Studio.

Object-Relational Mappers (ORM) : You can utilize ADO.NET directly to read data or alter existing data by using classes like as SqlCommand, SqlDataReader, SqlDataAdapter, and DataSet. The recommended alternative is to delegate the management of low-level code that interfaces with an ADO.NET data provider to an object-relational mapper (ORM) framework. One example of this type of framework is the Entity Framework. In order to use ADO.NET directly, you will need to manually compose and run SQL queries. In addition to this, you will need to build code that will convert the data from the format of the database into objects, properties, and collections that can be worked with in the code.

Compare the samples of code that you would write for an ORM with the samples of code that you would write to accomplish the same task using ADO.NET directly for a quick and easy method to see what an ORM can do for you. In the following school database example, there is a many-to-many link between the Instructor table, the Course table, and the CourseInstructor

association table. These three tables are referred to collectively as the Instructor table. You need to populate Instructor and Course objects that are comparable to the following in order to display a list of instructors and the courses they teach:

```
public class Instructor
{
    public string LastName { get; set; }
    public string FirstMidName { get; set; }
    public List<Course> Courses { get; set; }
}
public class Course
{
    public string Title { get; set; }
    public int Credits { get; set; }
}
```

When you use the Entity Framework ORM, the following is an example of the code that you would use to populate these classes :

```
public IEnumerable<Instructor> GetInstructors()
{
    List<Instructor> instructors;
    using (var db = new SchoolContext())
    {
        instructors = db.Instructors.Include("Courses").ToList();
    }
    return instructors;
}
```

And here is some code that performs the same purpose by making direct use of ADO.NET :

```
public IEnumerable<Instructor> GetInstructorsADONET()
{
var connString = ConfigurationManager.ConnectionStrings ["School Context"]. ConnectionString;
    List<Instructor> instructors = new List<Instructor>();
    using (var conn = new SqlConnection(connString))
    {
        conn.Open();
        var cmd = new SqlCommand("SELECT Person.PersonID, Person. LastName, Person.FirstName, Course.Title, Course.Credits " + "FROM Course INNER JOIN " +"CourseInstructor ON Course.CourseID = CourseInstructor.CourseID RIGHT OUTER JOIN " + "Person ON CourseInstructor.PersonID = Person.PersonID " + "WHERE (Person.Discriminator = 'Instructor') " + "ORDER BY Person.PersonID", conn);
```

```
            var reader = cmd.ExecuteReader();
            int currentPersonID = 0;
            Instructor currentInstructor = null;
            while (reader.Read())
            {
                var personID = Convert.ToInt32(reader["PersonID"]);
                if (personID != currentPersonID)
                {
                    currentPersonID = personID;
                    if (currentInstructor != null)
                    {
                        instructors.Add(currentInstructor);
                    }
                    currentInstructor = new Instructor();
                    currentInstructor.LastName = reader["LastName"].ToString();
                    currentInstructor.FirstMidName = reader["FirstName"].
                    ToString();
                }
                if (reader["Title"] != DBNull.Value)
                {
                    var course = new Course();
                    course.Title = reader["Title"].ToString();
                    course.Credits = Convert.ToInt32(reader["Credits"]);
                    currentInstructor.Courses.Add(course);
                }
            }
            if (currentInstructor != null)
            {
                instructors.Add(currentInstructor);
            }
            reader.Close();
            cmd.Dispose();
        }
        return instructors;
    }
```

When you don't utilize an ORM, you should be aware that not only does the amount of code that needs to be created, tested, and debugged significantly increase, but also the code that you write is database-specific. Using an ORM can make you more productive while also making it easier to manage your application. Because of these factors, it is recommended that an ORM be utilised wherever possible within an ASP.NET application that makes use of a relational database.

The following are some of the most popular object relationship managers (ORMs) that are compatible with ASP.NET :

- The primary object-relational mapper (ORM) that Microsoft offers for the.NET Framework is known as the ADO.NET Entity Framework.
- Microsoft offers a classic object relationship manager (ORM) called LINQ to SQL.
- NHibernate is an open-source object-relational mapping tool that works with the.NET Framework.

The ADO.NET Entity Framework is the easiest ORM framework to learn for an ASP.NET developer. We will learn more about it in the next unit.

❑ **Check Your Progress – 1 :**

1. _____ is the technology that enables a connection to be made to a relational database from within ASP.NET code that is executed on a server.

   a. ADO.NET b. AOD.NET

   c. ODA.NET d. None of these

2. The primary object-relational mapper (ORM) that Microsoft offers for the .NET Framework is known as the _____.

   a. ASP.NET Framework b. ADO.NET Entity Framework

   c. OLEDB d. None of these

6.3 Key Constraints :

Either when the table is being built with the CREATE TABLE statement or after the table has been created with the ALTER TABLE statement, constraints can be specified to be applied.

In order to set rules for the data included in a table, we can use SQL constraints. The kinds of data that can be entered into a table can be restricted with the use of constraints. The data in the table may now be relied upon to be accurate and precise as a result of this. Any time the constraint and the data action are found to be in conflict with one another, the action will be terminated.

Column-level constraints as well as table-level constraints are also possible. Constraints at the column level apply just to that column, while constraints at the table level apply to the entire table.

The following constraints are frequently applied while using SQL :

Constraints	Description
NOT NULL	Makes it such that a column can never have the value NULL in it.
UNIQUE	Ensures that each value in a column is distinct from the others.
PRIMARY KEY	A NOT NULL flag combined with a UNIQUE identifier. Specifies a one-of-a-kind identifier for each row in a table.
FOREIGN KEY	Prevents actions from taking place that would break linkages between tables.

CHECK	Ensures that the values included in a column meet the requirements of a particular condition
DEFAULT	Sets a default value for a column if no value is given
CREATE INDEX	Used for rapidly creating new records and retrieving existing ones from the database.

NOT NULL : By default, a column can carry NULL values. A column must not accept NULL values at any time if it is using the NOT NULL constraint. Because of this, a field is required to always have a value in it, which implies that you cannot change an existing record or create a new record without first adding a value to the field in question.

- **NOT NULL on CREATE TABLE :** When the "Persons" table is created, the SQL statement below assures that the "ID," "LastName," and "FirstName" columns will NOT accept NULL values :

```
CREATE TABLE Persons (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255) NOT NULL,
Age int
);
```

- **NOT NULL on ALTER TABLE :** Utilize the following SQL to set a NOT NULL constraint on the "Age" column of the "Persons" table after the table has already been created :

```
ALTER TABLE Persons
ALTER COLUMN Age int NOT NULL;
```

UNIQUE Constraint : The UNIQUE constraint makes sure that each value in a column is unique by requiring that it be so. A guarantee of an item's one-of-a-kind status can be obtained for a column or combination of columns by utilizing either the UNIQUE or PRIMARY KEY constraints. Every constraint that requires a PRIMARY KEY also requires a UNIQUE constraint.

On the other hand, you are only allowed one PRIMARY KEY constraint for each table, but you can have several UNIQUE constraints for each table.

- **UNIQUE Constraint on CREATE TABLE :** When the "Persons" table is created, the UNIQUE constraint will be applied to the "ID" column using the SQL statement that follows:

```
CREATE TABLE Persons (
ID int NOT NULL UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int
);
```

Use the following SQL syntax to name a UNIQUE constraint and to define a UNIQUE constraint on multiple columns at the same time :

```
CREATE TABLE Persons (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
CONSTRAINT UC_Person UNIQUE (ID,LastName)
);
```

- **UNIQUE Constraint on ALTER TABLE :** You can use the following SQL to add a UNIQUE constraint on the "ID" column even though the table has already been created :

  ```
  ALTER TABLE Persons
  ADD UNIQUE (ID);
  ```

  Use the following SQL syntax to name a UNIQUE constraint and to define a UNIQUE constraint on multiple columns at the same time :

  ```
  ALTER TABLE Persons
  ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);
  ```

- **DROP a UNIQUE Constraint :** Use the following SQL to drop a UNIQUE constraint from your database :

  ```
  ALTER TABLE Persons
  DROP CONSTRAINT UC_Person;
  ```

PRIMARY KEY Constraint : The PRIMARY KEY constraint ensures that each record in a table has its own distinct identifier. Primary keys are required to have UNIQUE values and are not allowed to have NULL values in them.

One primary key is the maximum number that can exist in a table; however, this main key might be comprised of a single column or several columns (fields).

- **PRIMARY KEY Constraint on CREATE TABLE :** When the "Persons" table is created, the "ID" column will have a PRIMARY KEY generated for it if the following SQL is executed :

  ```
  CREATE TABLE Persons (
      ID int NOT NULL PRIMARY KEY,
      LastName varchar(255) NOT NULL,
      FirstName varchar(255),
      Age int
  );
  ```

  Use the following SQL syntax to allow naming of a PRIMARY KEY constraint, as well as to define a PRIMARY KEY constraint on multiple columns :

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
);
```

There is just ONE PRIMARY KEY (PK Person) in the sample that was just presented. On the other hand, the value of the main key is constructed from TWO COLUMNS, which are referred to as "ID" and "LastName."

- **PRIMARY KEY Constraint on ALTER TABLE :** You can use the following SQL to add a PRIMARY KEY constraint on the "ID" column even though the table has already been created :

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

Use the following SQL syntax to allow naming of a PRIMARY KEY constraint, as well as to define a PRIMARY KEY constraint on multiple columns :

```
ALTER TABLE Persons
ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);
```

In order to use ALTER TABLE to add a primary key, the column(s) that will serve as the primary key must have already been defined to exclude NULL data (when the table was first created).

- **DROP a PRIMARY KEY Constraint :** Utilize the following statement in SQL to drop a PRIMARY KEY constraint:

```
ALTER TABLE Persons
DROP CONSTRAINT PK_Person;
```

FOREIGN KEY Constraint : It is possible to prohibit actions that would result in the destruction of linkages between tables by utilising the FOREIGN KEY constraint.

A field or combination of fields in one table can be referred to as a FOREIGN KEY if they contain references to the PRIMARY KEY in another table. The term "child table" refers to the table that contains the foreign key, while the term "referred table" or "parent table" refers to the table that contains the main key.

Take a look at the two tables that follow :

Persons Table

PersonID	LastName	FirstName	Age
1	Sharma	Ramesh	20
2	Jain	Harshita	25
3	Dixit	Hansraj	21

Orders Table

OrderID	OrderNumber	PersonID
1	23456	3
2	23455	2
3	23459	2
4	45678	1

Take note that the "Orders" table contains a "PersonID" column that is linked to the "Persons" database's "PersonID" field. The "PRIMARY KEY" for the "Persons" table is the "PersonID" column, which can be found in the "Persons" table. In the "Orders" table, the "PersonID" column serves as a FOREIGN KEY for the "Orders" table. Because the value in the foreign key column must be one of the values found in the parent table, the FOREIGN KEY constraint prohibits erroneous data from being placed into the column that contains the foreign key.

- **FOREIGN KEY Constraint on CREATE TABLE :** When the "Orders" table is created, the "PersonID" column will have a FOREIGN KEY defined for it if the following SQL is executed:

```
CREATE TABLE Orders (
    OrderID int NOT NULL PRIMARY KEY,
    OrderNumber int NOT NULL,
    PersonID int FOREIGN KEY REFERENCES Persons(PersonID)
);
```

Use the following SQL syntax to allow naming of a FOREIGN KEY constraint, as well as to define a FOREIGN KEY constraint on several columns :

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
    REFERENCES Persons(PersonID)
);
```

There is just ONE PRIMARY KEY (PK Person) in the sample that was just presented. On the other hand, the value of the main key is constructed from TWO COLUMNS, which are referred to as "ID" and "LastName."

- **FOREIGN KEY Constraint on ALTER TABLE :** You can use the following SQL to create a FOREIGN KEY constraint on the "PersonID" column of the "Orders" database even though that table has already been created :

```
ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

Use the following SQL syntax to allow naming of a FOREIGN KEY constraint, as well as to define a FOREIGN KEY constraint on several columns :

```
ALTER TABLE Orders
ADD CONSTRAINT FK_PersonOrder
FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

- **DROP a PRIMARY KEY Constraint :** The following statement in SQL must be used in order to drop a FOREIGN KEY constraint :

```
ALTER TABLE Orders
DROP CONSTRAINT FK_PersonOrder;
```

CHECK Constraint : It is possible to restrict the value range that can be entered into a column by utilizing the CHECK constraint. If you apply a CHECK constraint to a column, it will restrict the values that can be entered into that column to only those values you specify.

When a CHECK constraint is defined for a table, it can be used to restrict the values that can be entered into specific columns of the table based on the values of other columns in the row.

- **CHECK Constraint on CREATE TABLE :** The "Age" column will have a CHECK constraint applied to it after the following SQL statement has been executed on the "Persons" table. The CHECK constraint assures that a person's age is at least 18 years old and cannot be younger than that :

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
);
```

Use the following SQL syntax to allow for the naming of a CHECK constraint, as well as to define a CHECK constraint that applies to several columns :

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
);
```

- **CHECK Constraint on ALTER TABLE :** You can use the following SQL to set a CHECK constraint on the "Age" column even though the table has already been created :

```
ALTER TABLE Persons
ADD CHECK (Age>=18);
```

Use the following SQL syntax to allow for the naming of a CHECK constraint, as well as to define a CHECK constraint that applies to several columns :

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City = 'Sandnes');
```

- **DROP a CHECK Constraint :** Use the following SQL to drop a CHECK constraint from your database :

```
ALTER TABLE Persons
DROP CONSTRAINT CHK_PersonAge;
```

DEFAULT Constraint : A column's default value can be established with the use of the DEFAULT constraint. If no alternative value is supplied, the default value will be applied to each and every record that is created.

- **DEFAULT Constraint on CREATE TABLE :** When the "Persons" table is created, the following SQL will assign a value to the "City" column that will serve as the DEFAULT :

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
```

It is also possible to introduce system variables by utilizing the DEFAULT constraint in conjunction with functions such as GETDATE() :

```
CREATE TABLE Orders (
    ID int NOT NULL,
    OrderNumber int NOT NULL,
    OrderDate date DEFAULT GETDATE()
);
```

- **DEFAULT Constraint on ALTER TABLE :** You can use the following SQL to add a DEFAULT constraint on the "City" column even though the table has already been created :

```
ALTER TABLE Persons
ADD CONSTRAINT df_City
DEFAULT 'Sandnes' FOR City;
```

For Access :

```
ALTER TABLE Persons
ALTER COLUMN City SET DEFAULT 'Sandnes';
```

For Oracle :

```
ALTER TABLE Persons
MODIFY City DEFAULT 'Sandnes';
```

Use the following SQL syntax to allow for the naming of a CHECK constraint, as well as to define a CHECK constraint that applies to several columns :

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City = 'Sandnes');
```

- **DROP a DEFUALT Constraint :** Use the following SQL to drop a DEFAULT constraint from your database :

```
ALTER TABLE Persons
ALTER COLUMN City DROP DEFAULT;
```

CREATE INDEX : When creating indexes in tables, the CREATE INDEX statement is what's going to get you started. Indexes are utilised to make the process of retrieving data from a database significantly quicker than it would be otherwise. The users are unable to view the indexes; their sole purpose is to expedite the searching and querying processes.

When compared to updating a table that does not have indexes, updating a table that does contain indexes requires more time (because the indexes also need an update). Therefore, you should only establish indexes on the columns that will be searched against regularly.

- **CREATE INDEX Syntax :** Produces an index for use with a table. It is fine to enter the same value more than once :

```
CREATE INDEX index_name ON table_name (column1, column2, ...);
```

- **CREATE UNIQUE INDEX Syntax :** Creates a unique index on a table. It is forbidden to enter duplicate values :

```
CREATE UNIQUE INDEX index_name ON table_name (column1, column2, ...);
```

There are numerous databases, each of which has its own unique syntax for the creation of indexes. As a result, make sure that the syntax for your database's index creation is correct.

- **CREATE INDEX Example :** The following SQL line will establish an index on the "LastName" column of the "Persons" table, and it will give the index the name "idx lastname" :

```
CREATE INDEX idx_lastname ON Persons (LastName);
```

You can establish an index based on a combination of columns by listing the names of the columns between parenthesis and separating them with commas, as seen here :

```
CREATE INDEX idx_pname ON Persons (LastName, FirstName);
```

- **DROP INDEX Statement :** When you want to drop an index from a table, you can do so with the DROP INDEX statement. In SQL Server:

```
DROP INDEX table_name.index_name;
```

In Access :

```
DROP INDEX index_name ON table_name;
```

In Oracle :

```
DROP INDEX index_name;
```

❑ **Check Your Progress – 2 :**

1. _____ are required to have UNIQUE values and are not allowed to have NULL values in them.

   a. Foreign Key b. Primary Key c. Check d. None of these

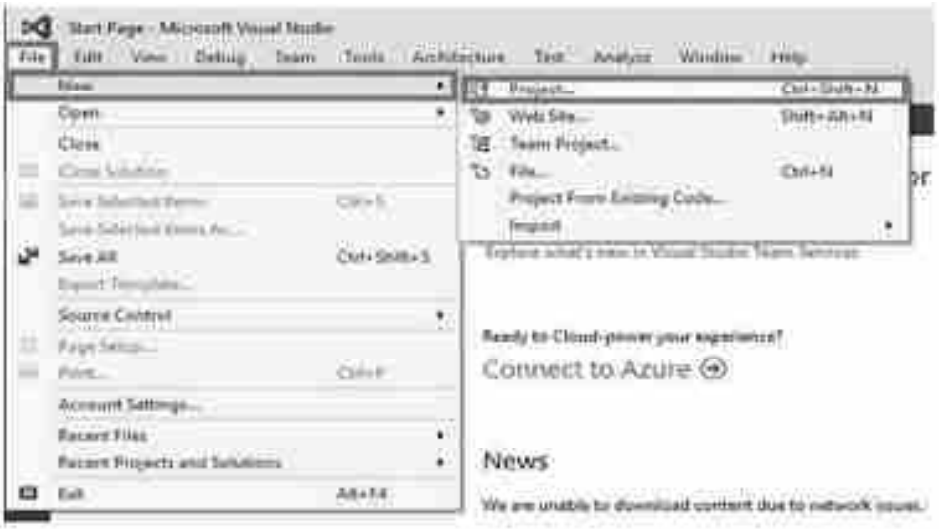
2. _____ ensures that each value in a column is distinct from the others.

   a. NOT NULL b. Foreign Key c. Check d. Unique Key

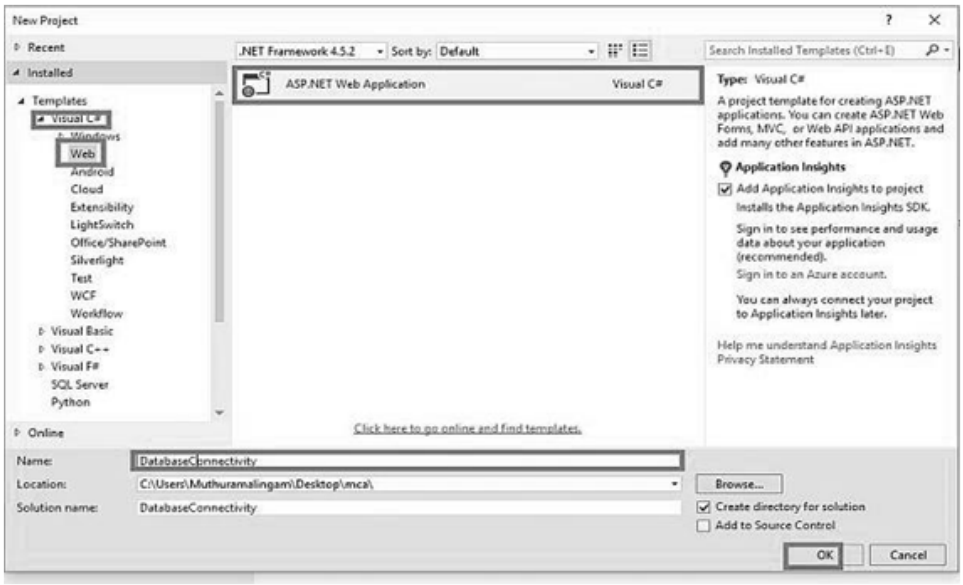
6.4 Creation of Database :

The steps that are outlined below should be carried out in order for you to successfully connect ASP.NET with a SQL database using C#.

Step I : Now, Open Visual Studio, go to the File >> New >> Project or use the shortcut key "Ctrl+Shift +N".



Step II : Here, select Visual C# >> Web >> ASP.NET Web Application. Finally, click "OK" button.



Step III : Here, you can select the template for your ASP.NET Application. We are choosing "Empty" here. Now, click OK button.

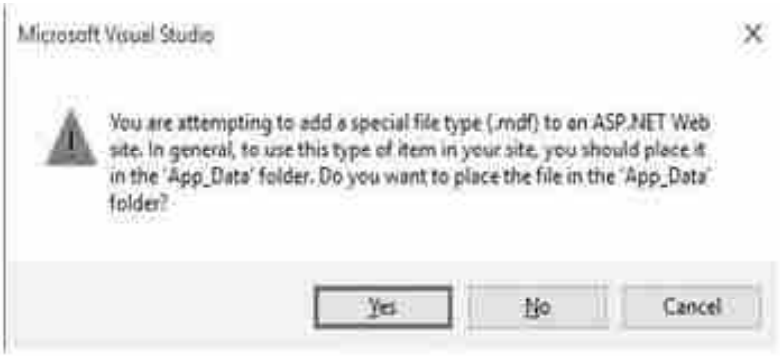
Step IV : Now, open the project and look for the Solution Explorer.

Step V : Here, open the default .aspx. If you want a Webform, you can add the page (Web Form). Add+New item (Ctrl+Shift+A).

Step VI : Now, we can create a login page, using ASP.NET code. You need to follow the drag and drop method. Here, we already created the login page.

Step VII : Now, open the project. If you want a SQL Server database, you can add the page (SQL Server database). Add+New item (Ctrl+Shift+A). Here, you can select Visual C# and choose SQL Server database. Afterwards, click "OK" button.

Step VIII : Here, open the new Window and click YES button.



Step IX : Now, add this to the database in our project.

The database named "RegisterDataBase" created. After that we will create table in this database. The process of Table creation is included in the next topic.

❑ **Check Your Progress – 3 :**

1. Which ASP.NET folder is used to store database files ?

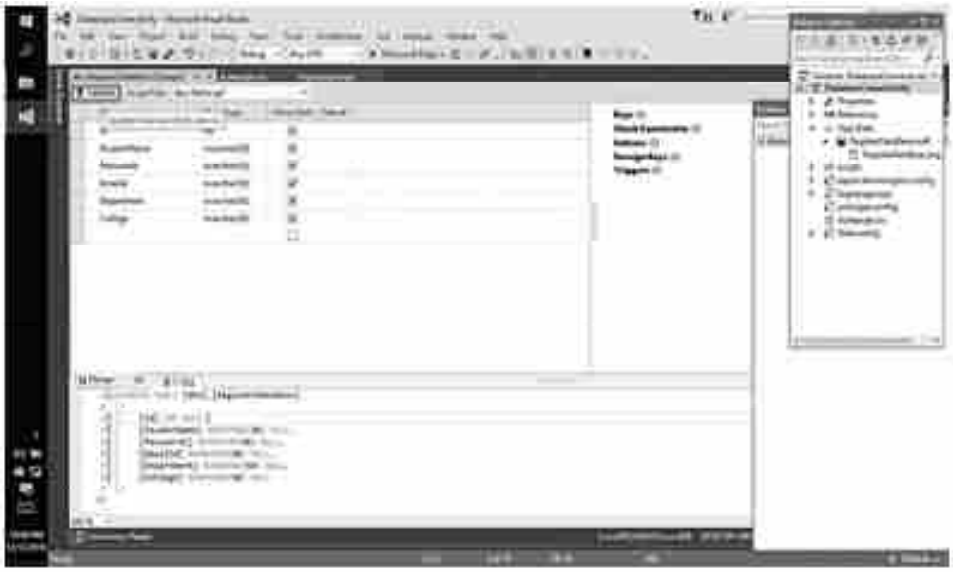
   a. App_Code b. App_Data

   c. Global Resources d. All of these

6.5 Tables :

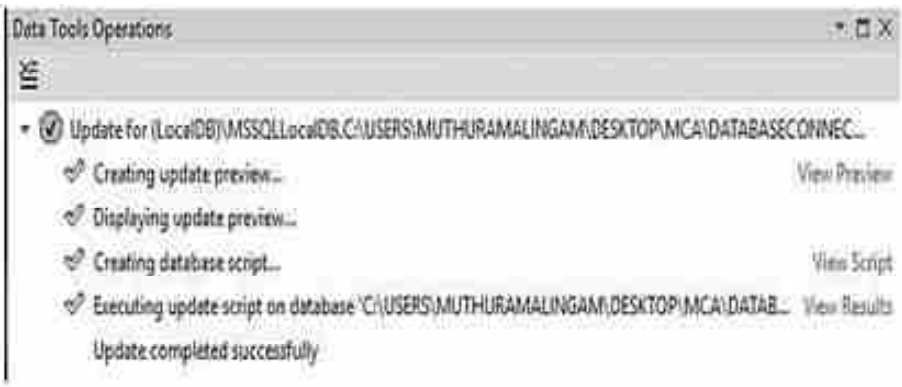
After database creation, we can create the table in it. We go to the Server Explorer and expand our dataase named "RegisterDataBase". We will right click on Tables and after that select Add New Table from there.

Now, open the new table and we can fill the data, which is like (studentname, password) and after that we click the Update as shown below :



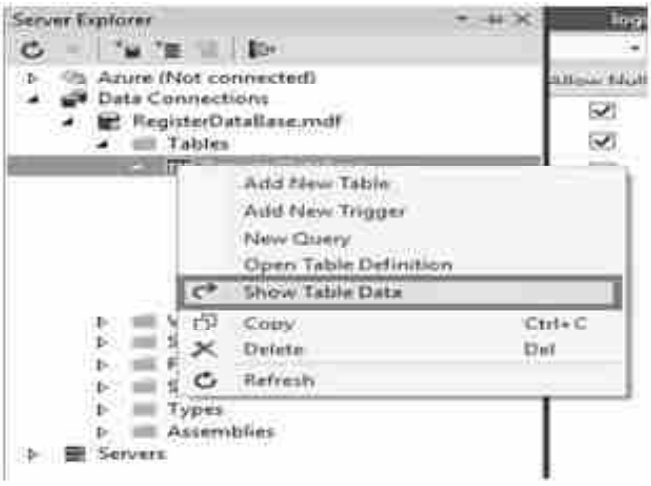
Here, click database in update and subsequently click update the database.

Here, the database is updated.



Here, the database and data are added.

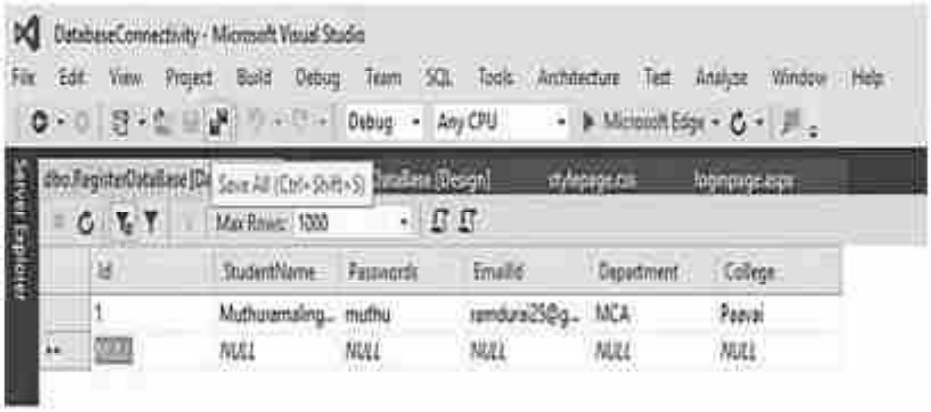
Now, we can click on the right click and click Show Table Data.



After creating Data Table, we can insert data in it. The data insertion in the table named RegisterDataBase is explained in next topic.

6.6 Records Insertion in Data Tables :

After creation of Data Table in the database we can insert the data into it. For data insertion we need to enter values as shown below in the data table :



6.7 Data Source Controls :

Interacting with the data-bound controls and hiding the sophisticated data binding operations is the responsibility of a data source control. These are the tools that supply data to the controls that are data bound and that support the execution of activities like as inserting, deleting, sorting, and updating data.

Each data source control encapsulates a distinct data provider, such as a relational database, an XML document, or a set of custom classes, and contributes to the following areas :

- Taking charge of the connection
- Selecting data
- Handling presentation-related tasks such as paging, caching, and so on.
- Manipulating data

Accessing data from SQL Server, ODBC or OLE DB servers, XML files, and business objects is made possible through the use of a wide variety of data source controls that are made available by ASP.NET.

These controls could be split into two different groups depending on the kind of data being monitored :

- Hierarchical data source controls
- Tables serve as the data source control.

The following are the data source controls that are utilized for hierarchical data :

- **The XMLDataSource :** It makes it possible to bind to XML files and strings with or without the presence of schema information.
- **SiteMapDataSource :** It makes it possible to bind to a provider who gives information about site maps.

The data source is used to retrieve the necessary information from the database and enter it into the appropriate fields of the controls. A table's rows can be automatically navigated through with the help of a tool called the binding navigator.

The data grid in C# is capable of connecting to the database and displaying all of the values from the table in a format similar to that of a grid. The various data view controls which are using to bind the data with the C# code we will learn into the next unit of this block.

Data Source Control manages the connection to the data, selection of data, and other jobs such as paging and caching of data etc.

Data Source Controls for Tabular Data

Data Source Controls	Description
SqlDataSource	It represents a connection to an ADO.NET data provider that returns SQL data, including data sources accessible via OLEDB and ODBC.
ObjectDataSource	It allows binding to a custom .Net business object that returns data.
LinqdataSource	It allows binding to the results of a Linq-to-SQL query (supported by ASP.NET 3.5 only).
AccessDataSource	It represents connection to a Microsoft Access database.

❑ **Check Your Progress – 4 :**

1. Which of these Data Source Controls is not for Tabular Data ?

   a. ObjectDataSource b. AccessDataSource

   c. SqlDataSource d. None of these

6.8 SqlDataSource Control :

A connection to a relational database, such as SQL Server or Oracle database, or data that is available by OLEDB or Open Database Connectivity is represented by the SqlDataSource control in an application (ODBC). The two critical properties ConnectionString and ProviderName are responsible for establishing a connection to the data. The basic syntax of the control can be found in the snippet of code that is presented below:

```
<asp:SqlDataSource runat="server" ID="MySqlSource"
    ProviderName='<%$ConnectionStrings:LocalNWind.ProviderName %>'
    ConnectionString='<%$ ConnectionStrings:LocalNWind %>'
    SelectionCommand= "SELECT * FROM EMPLOYEES" />
<asp:GridView ID="GridView1" runat="server" DataSourceID=
"MySqlSource" />
```

The numerous attributes (property groups) of the data source control are taken into consideration when configuring different data operations to be performed on the underlying data.

The associated sets of properties that make up the SqlDataSource control, which is responsible for providing the control's programming interface, are outlined in the table that can be found below :

Related sets of properties of the SqlDataSource control

Property Group	Description
DeleteCommand, DeleteParameters, DeleteCommandType	Gets or sets the SQL statement, parameters, and type for deleting rows in the underlying data.
FilterExpression, FilterParameters	Gets or sets the data filtering string and parameters.
InsertCommand, InsertParameters, InsertCommandType	Gets or sets the SQL statement, parameters, and type for inserting rows in the underlying database.
SelectCommand, SelectParameters, SelectCommandType	Gets or sets the SQL statement, parameters, and type for retrieving rows from the underlying database.
SortParameterName	Gets or sets the name of an input parameter that the command's stored procedure will use to sort data.
UpdateCommand, UpdateParameters, UpdateCommandType	Gets or sets the SQL statement, parameters, and type for updating rows in the underlying data store.

The following bit of code demonstrates an enabled data source control for the purpose of data manipulation :

```
<asp:SqlDataSource runat="server" ID= "MySqlSource"
    ProviderName='<%$ ConnectionStrings:LocalNWind.ProviderName %>'
    ConnectionString=' <%$ ConnectionStrings:LocalNWind %>'
    SelectCommand= "SELECT * FROM EMPLOYEES"
    UpdateCommand="UPDATE EMPLOYEES SET LASTNAME=@lame"
    DeleteCommand= "DELETE FROM EMPLOYEES WHERE
    EMPLOYEEID=@eid"
    FilterExpression= "EMPLOYEEID > 10">
    .....
</asp:SqlDataSource>
```

6.9 Data Connection :

The ASP.NET Database Connection may establish a connection with virtually any database, including Oracle, MongoDB, Microsoft SQL Server, and MySQL. As our database, we will be working with Microsoft SQL Server in this section. Database software developed by Microsoft that is completely free to use is called Microsoft SQL Server, and its Express Edition is very simple to download and set up. The following are some concepts that are typical of databases :

- **Connection :** When working with a database, the initial task is to create a link between the two systems. The following factors are required in order to successfully establish a connection :

- **Name of the database or the data source :** The name of the database that serves as the point of connection for this operation. You are only able to interact with a single database at a time.
- **Credentials :** It contains the username and password that are essential for establishing an ASP.NET Database Connection to the corresponding database. Credentials are necessary in order to authenticate to SQL Server, however Windows Authentication allows you to proceed without them.
- **Parameters that are optional :** You may also provide.NET with additional information regarding the manner in which it should be connected to the database in order to manage the data by specifying optional parameters.
- **Choosing certain data from within the database :** After the connection has been made, one can retrieve data from the database by issuing a SQL select statement against the database. In order to retrieve data from a particular table in the database, the SQL query must be used.
- **Inserting data into the database :** You are able to add new entries to the database, and it is required that the values for each row that are to be added to the database be provided in advance.
- **Updating data into the database :** It is also possible to make changes to previously entered records within the database. Each row in the database that needs to be changed should have its own set of fresh values that are given.
- **Getting rid of old information in the database :** There is also the option to remove records from the database.

We will connect to a database that has the name of Sampledb. The credentials that are needed to connect to the database are :

Username- dbtest and Password- sample123

To establish a connection, open your web application project and write the below SQL Server Connection String in Web.config file :

```
<connectionStrings>
    <add name="ConString" connectionString=" Data Source=WIN-50GP30FGO75;Initial Catalog=Demodb ;User ID=dbtest;Password= samplel23"/>
</connectionStrings>
```

Then, add System.Configuration Reference by right clicking on the project and click Add Reference option from the Context Menu. From the dialog box Add Reference, click on .Net Tab and look for System.Configuration assembly. Select the component and click OK.

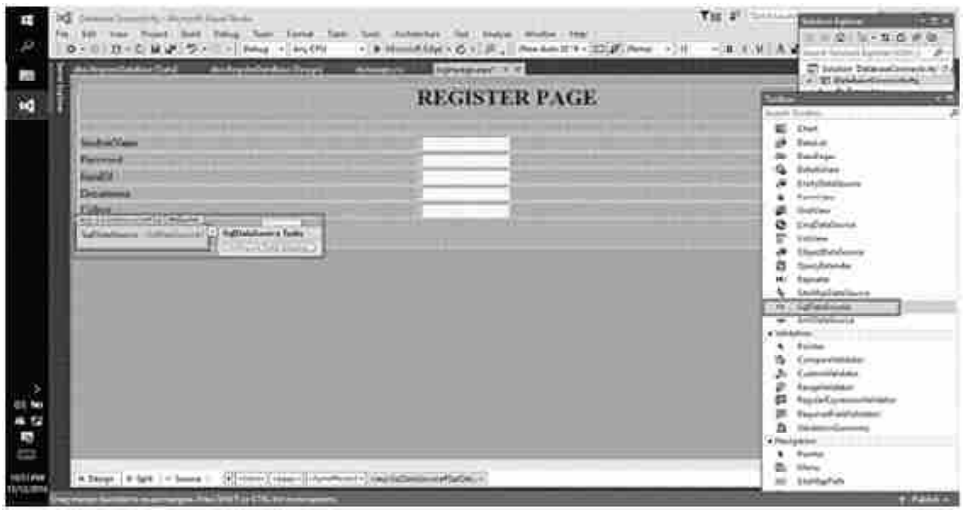
Once you have added the reference, it will appear in the References folder of the Solution Explorer. Now, write the behind code to get the connection string from web.config file :

```
using System;
using System.Configuration;
namespace DemoApplication
{
public partial class Demo  System.Web.UI.Page
      {
   protected void PageLoad(object sender, EventArgs e)
   {
string connetionString;
SqlConnection cnn;
connetionString = ConfigurationManager.ConnectionStrings["ConString"].
ConnectionString;
cnn = new SqlConnection(connetionString);
cnn.Open();
Response.Write("Connection Made");
conn.Close();
}
}
}
```

Here, the Data Source is the name of the server on which the database resides. Initial Catalog denotes the name of the database. User ID and Password are the credentials that are required. The output will be like this :

6.10 Configure Select Statement :

Now, we can add SQL Data Source. Drag and drop method. Here, click Configure Data Source to configure Select statement as shown below :



Now, we can choose our database and click NEXT button.

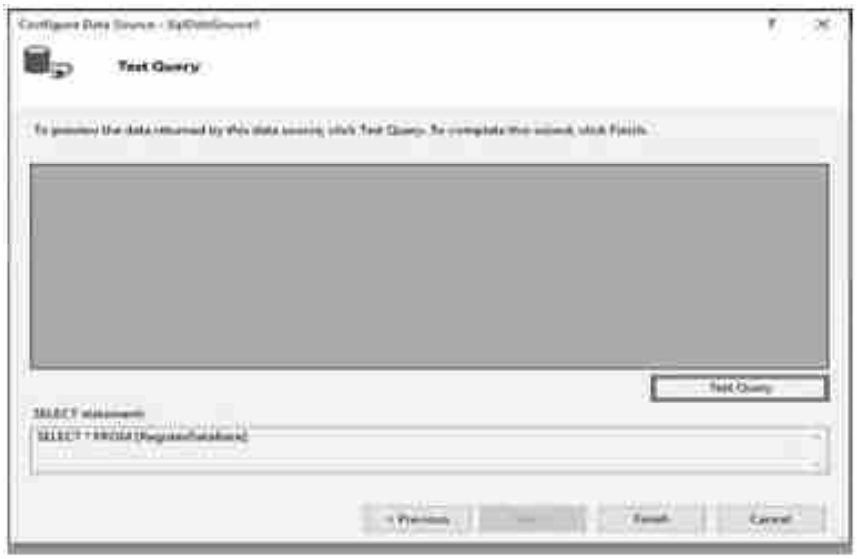
Now, you can select the ConnectionString and Click NEXT button.

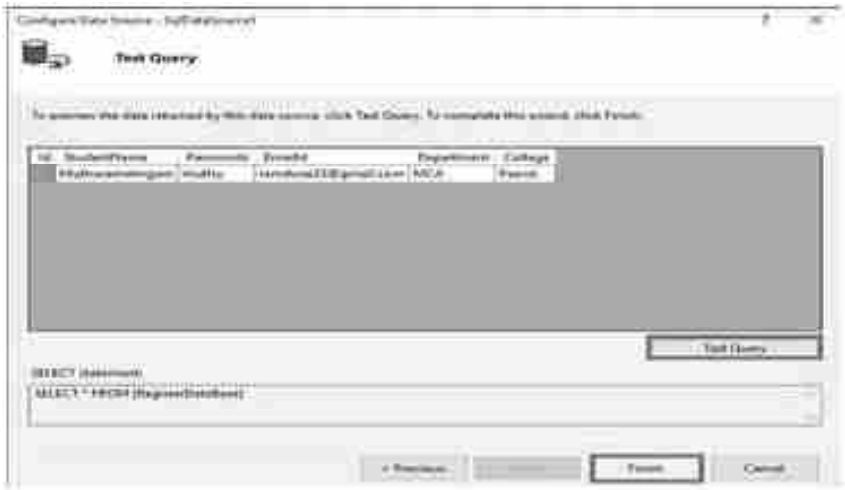
Now, we can choose Specify columns from a table or view and afterwards, click Next button.

This way we can configure Select Statement. After configuring the Select Statement we need to test the SQL Query. The process of testing query will be in the next topic.

6.11 Test The Query :

This final screen provides an opportunity to examine the results of the query configured from the previous step. Clicking the Test Query button executes the configured SELECT statement and displays the results in a grid.





At last, click on Finish Button.

6.12 Code Behind the Query :

After finishing the query testing process, we will go to the C# code page. We will write few code there.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Configuration;
namespace DatabaseConnectivity
{
    public partial class loginpage  System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
```

```
if(IsPostBack)
{
SqlConnection conn = new SqlConnection ( Configuration Manager. ConnectionStrings ["RegiConnectionString"]. ConnectionString);
    conn.Open();
    string checkuser = "select count(*) from RegisterDataBase where StudentName = '" + TextBox1.Text+"'";
    SqlCommand cmd = new SqlCommand(checkuser, conn);
    int temp = Convert.ToInt32(cmd.ExecuteScalar().ToString());
    if (temp == 1)
    {
        Response.Write("Student Already Exist");
    }
    conn.Close();
}
}
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        SqlConnection conn = new SqlConnection ( ConfigurationManager. ConnectionStrings["RegiConnection String"].ConnectionString);
        conn.Open();
string insertQuery = "insert into RegisterDataBase (StudentName, Passwords, EmailId, Department, College) values (@studentname, @passwords,@emailid,@department,@college) ";
        SqlCommand cmd = new SqlCommand(insertQuery, conn);
        cmd.Parameters.AddWithValue("@studentname", TextBox1. Text);
        cmd.Parameters.AddWithValue("@passwords", TextBox2. Text);
        cmd.Parameters.AddWithValue("@emailid", TextBox3.Text);
        cmd.Parameters.AddWithValue("@department", TextBox4. Text);
        cmd.Parameters.AddWithValue("@college", TextBox5.Text);
        cmd.ExecuteNonQuery();
        Response.Write("Student registeration Successfully!!!thank you");
        conn.Close();
    }
```

```
        catch (Exception ex)
        {
            Response.Write("error" + ex.ToString());
        }
    }
} }
```

Now, you can see the Loginpage.aspx code.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind=
"loginpage.aspx.cs" Inherits="DatabaseConnectivity.loginpage" %>
<!DOCTYPE html>
<html xmlns="http//www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link href="stylepage.css" type="text/css" rel="stylesheet" />
    <style type="text/css">
        .auto-style1 {
            width 100%;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div id="title">
    <h1>REGISTER PAGE</h1>
    </div>
        <div id ="teble"></div>
    <table class="auto-style1">
        <tr>
            <td>
                <aspLabel ID="Label1" runat="server" Text="StudentName">
                </aspLabel></td>
            <td>
                <aspTextBox ID="TextBox1" runat="server"></aspTextBox>
                </td>
        </tr>
        <tr>
            <td>
                <aspLabel ID="Label2" runat="server" Text="Password"></
                aspLabel></td>
            <td>
```

```
            <aspTextBox ID="TextBox2" runat="server"></aspTextBox>
            </td>
        </tr>
        <tr>
            <td>
                <aspLabel ID="Label3" runat="server" Text="EmailId"></
                aspLabel></td>
            <td>
                <aspTextBox ID="TextBox3" runat="server"></aspTextBox>
                </td>
        </tr>
        <tr>
            <td>
                <aspLabel ID="Label4" runat="server" Text="Department"></
                aspLabel></td>
            <td>
                <aspTextBox ID="TextBox4" runat="server"></aspTextBox>
                </td>
        </tr>
        <tr>
            <td>
                <aspLabel ID="Label5" runat="server" Text="College"></
                aspLabel></td>
            <td>
                <aspTextBox ID="TextBox5" runat="server"></aspTextBox>
                </td>
        </tr>
    </table>
    <div id="button">
        <aspButton ID="Button1" runat="server" Text="submit" OnClick=
        "Button1_Click" BackColor="Yellow" />
    </div>
        <div id="sim"></div>
        <aspSqlDataSource ID="SqlDataSource1" runat="server" Connection
        String="<%$ ConnectionStringsRegiConnectionString %>"
        SelectCommand="SELECT * FROM [RegisterDataBase]"></
        aspSqlDataSource>
        <div id="grid">
            <aspGridView ID="GridView1" runat="server" AllowPaging
            ="True" AllowSorting="True" AutoGenerateColumns="False"
            CellPadding="4" DataSourceID="SqlDataSource1" ForeColor=
            "#333333" GridLines="None">
```

```
<AlternatingRowStyle BackColor="White" ForeColor=
"#284775" />
<Columns>
<aspBoundField DataField="Id" HeaderText="Id"
SortExpression="Id" />
<aspBoundField DataField="StudentName" HeaderText=
"StudentName" SortExpression="StudentName" />
<aspBoundField DataField="Passwords" HeaderText=
"Passwords" SortExpression="Passwords" />
<aspBoundField DataField="EmailId" HeaderText=
"EmailId" SortExpression="EmailId" />
<aspBoundField DataField="Department" HeaderText=
"Department" SortExpression="Department" />
<aspBoundField DataField="College" HeaderText=
"College" SortExpression="College" />
</Columns>
<EditRowStyle BackColor="#999999" />
<FooterStyle BackColor="#5D7B9D" -Bold="True"
ForeColor="White" />
<HeaderStyle BackColor="#5D7B9D" -Bold="True"
ForeColor="White" />
<PagerStyle BackColor="#284775" ForeColor="White"
HorizontalAlign="Center" />
<RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
<SelectedRowStyle BackColor="#E2DED6" -Bold="True"
ForeColor="#333333" />
<SortedAscendingCellStyle BackColor="#E9E7E2" />
<SortedAscendingHeaderStyle BackColor="#506C8C" />
<SortedDescendingCellStyle BackColor="#FFFDF8" />
<SortedDescendingHeaderStyle BackColor="#6F8DAE" />
</aspGridView>
</div>
</form>
</body>
</html>
```

Here, we need to run any Browser and after a few minutes, we will get some output. Now, we can insert the data into our database.

6.13 Working with Sql Queries :

SQL has ten queries that account for three percent of the complete query set and can do ninety percent of the typical operations on databases using any SQL engine. Simply said, SQL is basically an interface that enables you to communicate with the database that is associated with your system through queries. To obtain data from your database, all we need to do is fire off some

straightforward SQL queries, and we won't even have to think about the internal d

C# has the ability to execute a select command written in SQL against a database. The 'SQL' statement can be utilised in order to retrieve data from a certain table within the database. The process of inserting data into the database can also be accomplished with C#, which can be used to add records to the database.

Utilizing the SqlCommand class would allow you to execute your command straight from within the C# environment. Using a SELECT SQL query and a SqlCommand object, it is possible to execute a C# function by passing a SQL-defined function as part of the query.

To query a SQL Server instance, an Azure SQL Database, an Azure SQL Managed Instance, or a database in Azure Synapse Analytics, you can make use of the mssql extension, which is one of the extensibility options that are supported by Visual Studio.atabase processes.

6.14 Where Clause :

When retrieving data from a single table or merging it with data from many tables, the SQL WHERE clause can be used to establish a condition that must be met. Only in the event that a particular condition is met does it retrieve a specific value from the table. The WHERE clause is utilized so that records can be filtered, and the retrieval of necessary records can take place.

Not only is it used in the SELECT statement, but the WHERE clause is also applied in the UPDATE and DELETE statements. Inserting one additional row into the database is the first step you need to do in order to comprehend why utilizing such a parameter is necessary.

The **syntax** of WHERE clause is :

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Take a look at the table named Student :

Student Table

ID	Name	Age
3	Ram	22
4	Raman	34
5	Shyam	21
6	Geeta	36
7	Ram	30

Now if we execute the query as SELECT * FROM Student WHERE Name = 'Ram'; The output will be :

Student Table

ID	Name	Age
3	Ram	22
7	Ram	30

So far, we have discovered that the usage of the name value alone in the WHERE clause does not allow us to retrieve a record that is truly unique. It is necessary to combine more than one condition in the WHERE clause, which is a straightforward process that can be accomplished by employing conditional keywords such as AND and OR. For instance, suppose we were to fire: SELECT * FROM Student WHERE Name = 'Ram' AND Age = 30; The output will be :

Student Table

ID	Name	Age
7	Ram	30

You can further narrow down your search by combining several conditions in the WHERE clause using the AND and OR operators. For instance, if we were to execute, SELECT * FROM Student WHERE Name = 'Ram' OR Age > 22. The output will be :

Student Table

ID	Name	Age
3	Ram	22
4	Raman	34
6	Geeta	36
7	Ram	30

To retrieve the rows you want, you can use the WHERE clause with a combination of different conditions, such as AND, OR, <, and >, or you can use each condition alone.

6.15 Order by :

When used in conjunction with the Select statement, the Order by clause is responsible for putting the retrieved data in sorted order, which sorts the data in descending order. It is the simplest of all queries and the one that is used the most. You only need to enter in the following: SELECT * FROM Student;

This query will produce an output that displays all of the rows that are currently.

Student Table

ID	Name	Age
3	Ram	22
4	Raman	34
5	Shyam	21
6	Geeta	36
7	Ram	30

Syntax of Order By is SELECT column-list|* from table-name order by column name;

We also have the option of incorporating an ORDER BY clause into our select statement if we want to display the results in a specific order. Take, for instance: SELECT * FROM Student ORDER BY Age;

Student Table

ID	Name	Age
5	Shyam	21
3	Ram	22
7	Ram	30
4	Raman	34
6	Geeta	36

The output is presented in sequential order, starting with the most recent. We are able to make use of DESC. If we want to organize the display in descending order, we need to add the keyword after the column name in the query.

6.16 Sorting in Ascending and Descending Order :

The output is arranged in increasing order of age. If we want the display to be arranged in descending order, we may add the DESC keyword after the column name in the query to accomplish this.

The result-set can be sorted either ascending or descending depending on whether or not the ORDER BY keyword was used. The records are sorted to be in ascending order when using the ORDER BY keyword as the default. Use the DESC keyword to sort the records in a descending order from highest to lowest. **Syntax** of sorting of records in Ascending and Descending order with ORDER BY Clause is

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Let's take an example of table "Employee":

Employee Table

EmpID	EmpName	EmpSalary
010	Shyam Sharma	21000
123	Ram Dubey	22500
211	Ram Chaudhary	30250
321	Raman Dixit	14580
654	Geeta Soni	12120

```
SELECT * from Employee ORDER BY by EmpSalary;
```

The output of Employee table will be in ascending order by the EmpSalary will look like this :

Employee Table

EmpID	EmpName	EmpSalary
654	Geeta Soni	12120
321	Raman Dixit	14580
010	Shyam Sharma	21000
123	Ram Dubey	22500
211	Ram Chaudhary	30250

Example of table "Employee" ORDER BY DESC is

SELECT * from Employee ORDER BY EmpSalary DESC;

The output of Employee table will be in descending order by the EmpSalary will be look like this :

Employee Table

EmpID	EmpName	EmpSalary
211	Ram Chaudhary	30250
123	Ram Dubey	22500
010	Shyam Sharma	21000
321	Raman Dixit	14580
654	Geeta Soni	12120

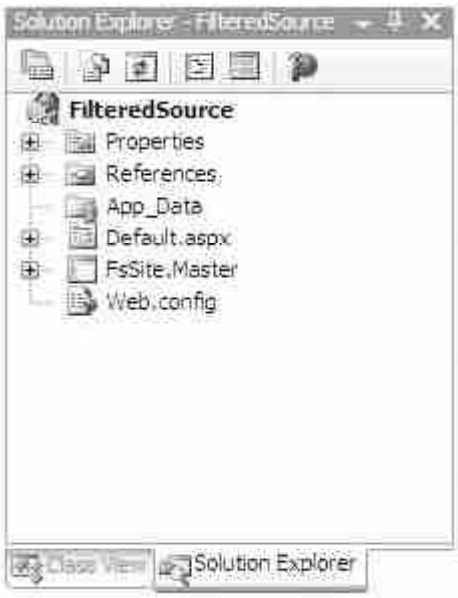
6.17 Filtering SqlDataSource Controls :

You are able to filter (sort, or select) the results of a query without having to execute the query again thanks to the SQLDataSource control. You are able to modify the data that is made available by a SQLDataSource control after a query has been executed by adding filtering to a SQLDataSource control. This can be done without having to return to the database. You need to configure the SQLDataSource control so that it stores its results in a cache and returns information in a dataset before you can utilize filtering. After doing so, you will have the opportunity to provide a filter expression that will be utilized by the RowFilter property of the DataView object that serves as the foundation for the SQLDataSource control.

The filter may make use of query strings, cookies, session variables, or other variables, as well as parameters that are determined by the values of another control. For instance, if a DropDownList control is used and it has a list of cities, you might utilize the chosen city from the DropDownList control as the filtering criteria for your search.

The data that is displayed can be limited by applying filter expressions to the data that was returned from the use of a SQL Data Source. The filter expression itself is quite similar in nature to a 'where' clause, but there is no need to generate a new query or stored procedure because the filter expression is applied to the data that was collected using a SQL Data Source. For instance, if a developer had a SQL Data Source that was set up to collaborate with a stored procedure that retrieved all of the accounts for a single zip code, the developer could use filter expressions to restrict the account data that was displayed based on a range of dates, a last name, or a specific account holder's name.

To get started, unzip the project that was provided for you, and then open the solution in the Visual Studio IDE. You should take note of the following files that are displayed in the solution explorer :



The answer is comprised of a solitary web application project that goes by the name "FilteredSource." In addition, the website in question features both a master page and a default web page. Master Page (FsSite.Master)

On the master page, there is only one table that is two rows high and two columns wide. The top two rows have been combined to create space for a banner, and there is a side bar on the left. A few of the controls for the hypertext links that allow navigation to other sites are located in the side bar. The homepage's default web page is displayed in the single content panel that can be found in the column to the right of the bottom row.

The code behind does not handle any data or carry out any specific functions, and as a result, there is no code to speak of; the HTML handles a few hyperlinks in the side bar and specifies the structure of the master page. The class starts off with the imports that are set by default (meaning that all of the imports are already set) :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace FilteredSource
{
public partial class FsSite : System.Web.UI.MasterPage {
protected void Page_Load(object sender, EventArgs e)
{
}
}
}
```

A GridView control and a few controls that are used to filter the data that is presented in that GridView are shown on this default web page. In

addition, there are two SQLDataSource controls on this page; one of them is utilized to populate the grid, and the other is utilized to populate a drop down list of cities. In this demonstration, both of the data source controls are bound to the sample Northwind database that is stored in SQL Server 2005. The second data source is the one that is filtered, and it is the one that is used to bind to the grid view, which is the primary method for displaying the data. The first data source control is used to populate a drop down list with a specific list of cities, and the other data source control is used to bind to the grid view. The class begins with the default imports.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

The next section contains the class declaration and default constructor and the namespace.

```
namespace FilteredSource
{
public partial class Default : System.Web.UI.Page {
```

The handling of the page load event is handled by the following portion of code. Store the filter expression in session, and then reload the page to reset the filter expression if it already exists. This will ensure that filtering is maintained between post backs.

```
protected void Page_Load(object sender, EventArgs e)
{
if(Session["FiltExp"] != null)
SQLDataSource1.FilterExpression = Session["FiltExp"].ToString();
}
```

The following piece of code is a handler for a drop-down list being selected and having its index modified. This drop down list is generated by a secondary SQLDataSource that has been set up to run a select distinct query against the City column of the employee table in the Northwind sample database. This query is run against the drop down list using the employee table. When a user chooses a city from the drop-down list, the handler modifies the SQLDataSource filter property to filter the SQLDataSource1 data source so that it only displays employees who live in the chosen city. This ensures that only relevant information is displayed. When you have finished configuring the expression, the filter expression will be saved to the session so that it can be reapplied to the data during subsequent post backs.

```
protected void DropDownList1_SelectedIndexChanged(object sender,
EventArgs e)
{
SQLDataSource1.FilterExpression = "city='" + DropDownList1.Selected
Value + "'";
```

```
Session["FiltExp"] = "city='" + DropDownList1.SelectedValue + "'";
}
```

This method simply demonstrates a different example of applying a filter to the same data source in order to modify the visible results of an identical set of data through the application of a filter expression. The following piece of code was used to handle the click event for the Show By Data button. The user can input a date into the demonstration web application, and then clicking on this button will filter the results to reveal only employees who were born after the date that the user entered.

```
protected void btnShowByDate_Click(object sender, EventArgs e)
{
if (!String.IsNullOrEmpty(txtDay.Text) andand
!String.IsNullOrEmpty(txtMonth.Text) andand
!String.IsNullOrEmpty(txtYear.Text))
{
DateTime dt = new DateTime(Convert.ToInt32(txtYear.Text),
Convert.ToInt32(txtMonth.Text),
Convert.ToInt32(txtDay.Text));
SQLDataSource1.FilterExpression = "BirthDate > #" + dt + "#";
Session["FiltExp"] = "BirthDate > #" + dt + "#";
}
}
```

When the show all button is clicked, the event handler simply sets the filter expression to null. Doing show will result in the redisplay of the entire result set without any filtering. The purpose of the final example use of filters in this demonstration is simply to demonstrate that they can be eliminated entirely.

```
protected void btnShowAll_Click(object sender, EventArgs e)
{
SQLDataSource1.FilterExpression = null;
Session["FiltExp"] = null;
```

6.18 Sorting SqlDataSource Controls :

Controls such as DataList, GridView, and DetailsView can be bound to SQL data that is retrieved from SQL databases such as Microsoft SQL Server and Microsoft Access through the use of the SQLDataSource control, which establishes a connection to a SQL database. The SQLDataSource control is not rendered on the page itself, so it is not visible to the user when the application is running. However, the SQLDataSource control is visible in the Web Designer in Visual Studio, and the web programmer can interact with it in the Design View of Visual Studio. The following are the data source controls that enable sorting :

- ObjectDataSource
- SQLDataSource
- AccessDataSource

Sorting is only possible when the SQLDataSource and AccessDataSource controls have their DataSourceMode attribute set to DataSet, as illustrated in the following example :

```
<asp:GridView ID="EmployeesGridView"
DataSourceID="EmployeesSQLDataSource"
DataKeyNames="EmpID"
AllowSorting="True"
RunAt="Server" />
<asp:SQLDataSource ID="EmployeesSQLDataSource"
SelectCommand="SELECT EmID, EmpName, EmpSalary FROM Employees"
Connectionstring="<%$ ConnectionStrings:NorthwindConnectionString %>"
Runat="server" />
<asp:GridView ID="EmployeesGridView"
DataSourceID="EmployeesSQLDataSource"
DataKeyNames="EmpID"
AllowSorting="True"
Runat="Server" />
<asp:SQLDataSource ID="EmployeesSQLDataSource"
SelectCommand="SELECT EmpID, EmpName, EmpSalary FROM Employees"
Connectionstring="<%$ConnectionStrings:NorthwindConnectionString %>" Runat="server" />
```

If the object that was returned by the SelectMethod was a DataSet, DataTable, or DataView object, then the ObjectDataSource control would have the ability to sort the data. In addition to that, the ObjectDataSource enables users to retrieve results from the data source in a sorted way. You can take advantage of the controls' sorting capabilities by utilising the SortParameterName property when working with the ObjectDataSource or SQLDataSource classes of data sources. You have the option of assigning the name of the parameter that contains a sort expression to the SortParameterName property. This expression is then sent to the data source control. The sort expression is a list of the fields, separated by commas, that should be sorted by (and optionally the DESC identifier to sort in descending order).

6.19 Studying The Sql Code :

SQL, which stands for "Structured Query Language," is a programming language that can edit and access data or information stored in a location known as a database. A number of tables are used in the construction of a SQL database. In a company, SQL tables would be used to section off and simplify the many

aspects of the operation, including the following: Table for the Customers, one for the Vendors, one for the Employees, etc.

Consider a straightforward statement written in English, such as "the company wants to pick all stock numbers from the pricing table where the price is more than 3500." Since we want to express this statement in SQL, we can write it as follows :

SELECT StockNumber FROM Prices WHERE Price > 3500

You are required to construct a database in order to work with SQL. As an illustration, the database of employees for a firm is required to be created in order to store information such as payroll details, job history, supervisor relationships, performance reviews, and other similar records. If this is the case, you will need to compile the information into a table that includes employees' various means of communication. The following is how the table will look: employee_id, first_name, last_name, street_address, city, state, zipcode, mobileno

In the beginning, you will need to input all of the data into the table by using the following command: SELECT * FROM personal_info;

The INSERT command in SQL needs to be used in order to insert the aforementioned information, for example: INSERT INTO personal_info values (123,'Neelkamal','Sharma','123 SilverOak','Bikaner','Rajasthan',313004, 9192190234)

These values match to the characteristics of the table in the order that they are mentioned.

6.20 Let Us Sum Up :

During the course of our studies for this lesson, we came to understand that a table can have a column, a mix of columns, or values that individually identify every row in the table. These columns are what is known as the primary key of the table, and they are responsible for ensuring the table's entity integrity.

In the Solution Explorer, you will need to select the option to "add database to new project" before you can establish a new project database. The Table, or the Data The information that is stored in the rows and columns of a table is considered to be part of the System. Data namespace in which you can add to, choose from, and iterate over the data that has been stored.

Data source controls interact with data-bound controls and hide complex data binding processes. Data binding processes are tools that provide data to data-bound controls. Data source controls support insertions, deletions, sorts, and updates. Data source controls also hide complex data binding processes. A connection to a relational database, such as SQL Server or Oracle database, or data that is available by OLEDB or Open Database Connectivity is represented by the SQLDataSource control in an application (ODBC).

When pulling information from a database, the SELECT statement is the one to utilize. When pulling information from a database, the SELECT statement is the one to utilize.

6.21 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : a 2 : b

- **Check Your Progress 2 :**

  1 : b **2 :** d

- **Check Your Progress 3 :**

  1 : b

- **Check Your Progress 4 :**

  1 : d

6.22 Glossary :

1. **Contraints :** Factors that impose restrictions and limitations on the system or actual limitations associated with the use of the system.
2. **SQL :** Structured Query Language
3. **ConnectionString :** A connection string is a string that specifies information about a data source and the means of connecting to it.
4. **DBMS :** Database Management System is essentially nothing more than a computerized data-keeping system.

6.23 Assignment :

1. What are the main Key Contraints for Database Connectivity ?

6.24 Activities :

1. Explain various data source controls in detail.

6.25 Case Study :

Study filtering sqlDataSource

6.26 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)?

Unit
07

WORKING WITH DATABASE

UNIT STRUCTURE

7.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About the CRUD(Create, Retrieve, Update and Delete) Operation on Data
- About GridView Control
- About DataList Control
- About DetailsView Control
- About ListView Control
- About FormView Control
- About SQLDataSource Control
- About AccessDataSource Control
- About Repeater Control
- About XmlDataSource Control
- About EntityDataSource Control

7.1 Introduction :

The ASP.NET Details View provides support for the addition of a SqlDataSource control to a page, and this control can be configured to include INSERT commands. In order to use the insert command, you will need to make sure that the AutoGenerateInsertButton property is set to the true value. When using Asp.GridView, Net's the Bound field links a column to a property in the data source items. This facilitates the back and forth movement of data. It is a column in the database where data from the table is stored, and it displays the data that is stored in the column of the table that is stored in the database. In this section, we will go through the fundamentals of formatting the grid view using date, currency, and value information. The research and concept of Grid Control, as well as its aesthetic features, will be the primary focus of this section. You will get an idea on Delete field appearing as links. During this section, you will be given the opportunity to discover and gain an understanding of the fundamentals of Data binding as well as the numerous radio buttons. You will also have the concept of creating a custom column class and inserting a checkbox in the gridview header template explained to you. The Gridview AutoGenerateColumns attribute will have its practical applications illustrated for you in this block.

7.2 CRUD (Create, Retrieve, Update and Delete) Operation on Data :

Create, Read, Update, and Delete, also known as CRUD, is an acronym that originates from the field of computer programming. It is an abbreviation that refers to the four functions that are thought to be required in order to develop an application that uses persistent data.

CREATE, READ, UPDATE, and DELETE are the actions denoted by the abbreviation CRUD. These terminologies represent the four key operations for creating and managing persistent data items, primarily in relational and NoSQL databases. These operations include: create, insert, update, and delete.

CRUD is an acronym that is routinely applied to a wide variety of database-related activities, including database design. Without CRUD processes, software engineers are unable to accomplish anything useful. REST, which stands for "representational state transfer," is a protocol that is used in the building of websites. It is a superset of the CRUD protocol, which is used for HTTP resources.

- **CREATE Operation :** The 'create' action in CRUD accomplishes exactly what one might expect from its name. To create an entry is what it means. It's possible that this entry is a user account, user information, a post, or a task. The POST method is the part of the HTTP protocol that is responsible for putting the CREATE operation into action. When working with a SQL database, "create" is synonymous with "insert." You can generate new data in a NoSQL database such as MongoDB by calling the insert() method.

- **READ Operation :** Getting access to the inputs or entries in the user interface (UI) is what the READ operation does. That is, actually observing it. Again, the entry may be anything-user information, posts on social media, or something else entirely, among other possibilities.

  This access may involve the user searching for previously made entries or receiving access to those entries immediately after they have been created. The user is given the ability to search for entries in order to remove those that aren't relevant to their needs.

  The GET method is the component of the HTTP protocol that makes the READ action possible.

  To read an entry in a SQL database, you must use the SELECT statement. You may read information from a NoSQL database such as MongoDB using the find() or findById() API.

- **UPDATE Operation :** UPDATE is the operation that enables you to make changes to data that has already been stored. To clarify, this refers to altering the data. In contrast to READ, the UPDATE operation modifies the previously stored data by adding to or removing from it.

  You can implement an update using either the PUT or the PATCH HTTP protocols, depending on what your requirements are. PUT and PATCH are both part of the HTTP standard.

  You should use the PUT command when you want to modify the complete entry, and you should use the PATCH command when you don't want to modify the full entry.

  You can update an entry in a SQL database by using the UPDATE command. Using the findByIdAndUpdate() method in a NoSQL database like MongoDB enables users to implement an update capability in their database.

- **DELETE Operation :** When an entry is deleted, it is removed from both the user interface and the database.

  The HTTP protocol that is used to carry out a DELETE action is referred to as DELETE. When deleting an entry from a SQL database, the

DELETE command is used. The delete operation can be implemented in a NoSQL database such as MongoDB through the use of the findByIdAndDelete() method.

- **Benefits of CRUD Operation :** Because of its superior efficiency, many programmers opt to make use of CRUD rather than write ad-hoc SQL statements in their code. When a stored procedure is invoked for the first time, its execution plan is saved in the procedure cache of SQL Server. This execution plan can then be utilised for any and all stored procedure applications.

  When a SQL statement is run through SQL Server, the relational engine performs a search of the procedure cache to determine whether or not an existing execution plan for that particular SQL statement is available. If one is found, the relational engine makes use of the existing plan to reduce the number of steps required for optimising, parsing, and recompiling the SQL statement.

In the event that an execution plan cannot be located, SQL Server will generate a brand new execution plan in order to process the query. Additionally, when you remove SQL statements from the application code, you can keep all of the SQL in the database while the client application simply contains stored procedure invocations. This is possible since the SQL statements are no longer in the application code. Using stored procedures is one way to reduce the amount of coupling between database tables.

In addition, protecting against SQL injection attacks is made easier by leveraging CRUD operations. If string concatenation is used to generate dynamic queries from user input data for any SQL Statements, then using stored procedures rather than string concatenation is required. This ensures that everything that is entered into a parameter is properly quoted.

❑ **Check Your Progress – 1 :**

1. CRUD stands for _____.

   a. Crawl, Reload, Upload and Download

   b. Create, Read, Update and Delete

   c. All of these

   d. None of these

7.3 GridView Control :

The values of a data source are shown in the form of a table by the GridView control. Each row in the table represents a record, and each column in the table represents a field. The following functionalities are supported by the GridView control :

- Connecting to data source controls like SqlDataSource, for example.
- Integrated capability for sorting
- Built-in update and delete capabilities.
- Built-in paging capabilities.
- Integrated capabilities for selecting individual rows.
- Access to the object model of the GridView via programming, allowing for the dynamic setting of properties, the handling of events, and so on.
- Multiple fields for the keys.
- Multiple data fields for each of the columns containing hyperlinks.
- The appearance can be changed using a variety of themes and styles.

Syntax: <asp:GridView ID="gridService" runat="server"> </ asp:GridView>

The following GridView operations are available to us to do :

- Bind data to GridView column
- Edit data in GridView
- Delete rows from the GridView and
- Update row information in the database.

Example :

HTML Code :

<h1>

Back | Service Master

</h1>

<asp:updatepanel id="UpdatePanelService" runat="server" updatemode= "Conditional">

<ContentTemplate>

<asp:GridView ID="gridService" runat="server" CssClass="EU_DataTable" AutoGenerateColumns="false" ShowFooter ="true"OnRowEditing = "gridService_RowEditing" onrowcreated = "gridService_RowCreated" onrowupdating= "gridService_RowUpdating">

```
<Columns>
<asp:TemplateField ItemStyle-Width="30px" HeaderText="SR.NO">
<ItemTemplate>
<asp:Label ID="lblID" runat="server" Text='<%#Eval("service_id")%>'></
asp:Label>
    </ItemTemplate>
</asp:TemplateField>
<asp:TemplateField ItemStyle-Width="600px" HeaderText="Service">
<ItemTemplate>
<asp:Label ID="lblService" runat="server" Text='<%#Eval("service_
name")%>'> </asp:Label>
    </ItemTemplate>
    <EditItemTemplate>
<asp:TextBox ID="txtService" runat="server" Text='<%#Eval("service_
name")%>'> </asp:TextBox>
    </EditItemTemplate>
    <FooterTemplate>
        <asp:TextBox ID="txtService" runat="server"></asp:TextBox>
    </FooterTemplate>
</asp:TemplateField>
<asp:TemplateField ItemStyle-Width="100px" HeaderText="Service
Photo">
    <ItemTemplate>
        <img src='<%# Eval("service_image")%>' alt='<%#
                Eval("service_image")%>' height="50px"
            width="50px" />
    </ItemTemplate>
    <EditItemTemplate>
        <asp:FileUpload ID="fuService" runat="server" />
    </EditItemTemplate>
    <FooterTemplate>
        <asp:FileUpload ID="fuService" runat="server" />
    </FooterTemplate>
</asp:TemplateField>
<asp:TemplateField>
    <ItemTemplate>
        <asp:LinkButton ID="lnkRemove" runat="server" Command
        Argument='<%#
                Eval("service_id")%>'
            OnClientClick="return confirm('Do you want to delete?')"
```

```
            Text="Delete"></asp:LinkButton>
        </ItemTemplate>
        <FooterTemplate>
            <asp:Button ID="btnAdd" runat="server" Text="Add" OnClick=
            "AddService"/>
        </FooterTemplate>
    </asp:TemplateField>
    <asp:CommandField ShowEditButton="True" />
</Columns>
</asp:GridView>
<asp:UpdateProgress ID="UpdateProgress1" runat="server"
    AssociatedUpdatePanelID="UpdatePanelService">
    <ProgressTemplate>
        Please wait image is getting uploaded....
    </ProgressTemplate>
</asp:UpdateProgress>
</ContentTemplate>
<Triggers>
</Triggers>
    </asp:updatepanel>
</div>
```

We can bind the GridView with database like this :

```
private void _BindService()
{
    try
    {
        List<BOService> service = dALService.Service.ToList();
            if (service.Count > 0 && service != null)
            {
                gridService.DataSource = service;
                    gridService.DataBind();
            }
        }
    catch (Exception)
    {
            throw;
        }
    }
```

We can edit the values in GridView like this :

```
protected void gridService_RowEditing(object sender, GridViewEditEvent
Args e)
{
    try
    {
        gridService.EditIndex = e.NewEditIndex;
    _BindService();
}
catch (Exception)
    {
        throw;
    }
}
```

We can update the GridView like this :

```
protected void gridService_RowUpdating(object sender, GridViewUpdate
EventArgs e)
{
try
    {
string servicename = ((TextBox)gridService.Rows[e.RowIndex].Find
Control("txtService")).Text;
string filePath=((FileUpload)gridService.Rows[e.RowIndex].FindControl
("fuService")).FileName;
        bOService.Service_name = servicename;
        bOService.Service_image = "../images/service/" + filePath;
            if (File.Exists(Server.MapPath("~/images/service/" + filePath)))
            {
                }
                else
                {
        ((FileUpload)gridService.FooterRow.FindControl("fuService")).
        SaveAs(Server.MapPath("~/images/service/" + filePath));
            }
            dALService.UpdateService(bOService);
                _BindService();
            }
                catch (Exception)
```

```
{
throw;
}
}
```

We can Add value in the GridView like this :

```
protected void AddService(object sender, EventArgs e)
{
try
{
string servicename = ((TextBox)gridService.FooterRow.FindControl
("txtService")).Text;
string filePath = ((FileUpload)gridService.FooterRow.FindControl
("fuService")).FileName;
bOService.Service_name = servicename;
bOService.Service_image = "../images/service/" + filePath;
if (File.Exists(Server.MapPath("~/images/service/" + filePath)))
{
}
else
    {
((FileUpload)gridService.FooterRow.FindControl("fuService")).SaveAs
(Server.MapPath("~/images/service/" + filePath));
}
((FileUpload)gridService.FooterRow.FindControl("fuService")).SaveAs
(Server.MapPath("~/images/service/" + filePath));
dALService.SetService(bOService);
_BindService();
}
catch (Exception)
    {
throw;
}
}
```

❑ **Check Your Progress – 2 :**

1. Which of these properties is not used to bind the data with the GridView Control ?

   a. ItemTemplate b. Color

   c. EditItemTemplate d. FooterTemplate

7.4 DataList Control :

Displaying and manipulating data in a web application is the responsibility of a Databound control known as DataList. It is a composite control that can combine other ASP.Net controls, and it is present in the form. Additionally, it has the ability to combine controls. The appearance of the DataList is determined by the template fields that it contains.

The DataList control provides support for the various template fields listed below :

- **Itemtemplate :** It renders itself in the browser as many rows present in the data source collection. It is responsible for specifying the Items that are currently present in the Datasource.
- **EditItemTemplate :** EditItemTemplate is a template that is used to grant the user the ability to update an item.
- **HeaderTemplate :** The data source collection can see the header text thanks to the template known as HeaderTemplate.
- **FooterTemplate :** FooterTemplate is a template that is utilised to show the data source collection the footer text.
- **ItemStyle :** ItemStyle is a tool that is utilised for the purpose of applying styles to an ItemTemplate.
- **EditStyle :** It is possible to apply styles to an EditItemTemplate using the EditStyle.
- **HeaderStyle :** It is possible to apply styles to a HeaderTemplate using a HeaderStyle.
- **FooterStyle :** FooterStyle is a component that is utilised in order to add styles to a FooterTemplate.

Properties of DataList Control

Properties	Description
RepeatDirection Horizontal Vertical(Default)	Used to set or get the Direction of Items being render. Horizontal: Items will render in Horizontal. Vertical: Items will render in Vertical.
RepeatColumns	Used to set number of columns display in DataList. Return type is int data type.
EditItemIndex	It is runtime property used to set an index value to change its template from ItemTemplate to EditItemTemplate. If it set to -1 no items will change its template.

Events of DataList Control

Events	Description
OnEditCommand	It will be raised when user clicks on a button (present in DataList) whose CommandNameis "Edit".
OnUpdateCommand	It will be raised when user clicks on a button (present in DataList) whose CommandNameis "Update".
OnCancelCommand	It will be raised when user clicks on a button (present in DataList) whose CommandNameis "Cancel".
OnDeleteCommand	It will be raised when user clicks on a button (present in DataList) whose CommandNameis "Delete".

The example that follows demonstrates how to use the DataList Control in conjunction with a bound picture and a FileUpload control :

Column Name	Datatype
EmpID	Int
EmpName	Varchar(50)
EmpEmailID	Varchar(50)
EmpMobile	Bigint
EmpImg	Varchar(50)

Default.aspx :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.
aspx.cs" Inherits="_Default" %>
<!DOCTYPE html>
<html
    xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
        <title></title>
        <style>
        .itemstyle {
            border-collapse: collapse;
            background-color: aquamarine;
            border: 1px solid red;
            border-radius: 8px;
            padding:4px;
        }
        tr {
            height:40px;
        }
        table {
            width: initial;
        }
    </style>
    </head>
    <body>
        <form id="form1" runat="server">
            <div>
                <asp:DataList ID="dl1" runat="server"
    RepeatDirection="Horizontal"
     RepeatColumns="3"
      OnEditCommand="dl1_EditCommand"
       OnCancelCommand="dl1_CancelCommand"
```

```
OnUpdateCommand="dl1_updateCommand"
 OnDeleteCommand="dl1_DeleteCommand" >
        <HeaderTemplate>
            <table>
                <tr>
                    <td>
                        <h1>Student Information</h1>
                    </td>
                </tr>
            </table>
        </HeaderTemplate>
        <ItemStyle CssClass="itemstyle" />
        <ItemTemplate>
            <table border="1">
                <tr>
                    <td>Employee ID:</td>
                    <td>
                    <asp:Label ID="lblempid" runat="server" Text='<%#
                    Eval("EmpID") %>'>
                        </asp:Label>
                    </td>
                    <td rowspan="5">
                        <asp:Image ID="empimg" runat="server"
                        ImageUrl='
    <%# "~/Images/" + Eval("EmpImage") %>' Height=
    "180px" Width="200px" />
                    </td>
                </tr>
                <tr>
                    <td>Employee Name:</td>
                    <td>
    <asp:Label ID="lblempname" runat="server" Text='<%# Eval
    ("EmpName") %>'>
                        </asp:Label>
                    </td>
                </tr>
                <tr>
                    <td>Employee EmailId:</td>
                    <td>
    <asp:Label ID="lblemailid" runat="server" Text='<%# Eval
    ("EmpEmailID") %>'>
```

```
                    </asp:Label>
                </td>
            </tr>
            <tr>
                <td>Employee Mobile Number</td>
                <td>
<asp:Label ID="lblmbnum" runat="server" Text='<%# Eval ("EmpMobile")  %>'>
                    </asp:Label>
                </td>
            </tr>
            <tr>
                <td colspan="3" style="text-align:left">
                    <asp:Button ID="btn1" runat="server" CommandName="edit" Text="Edit" />
                    <asp:Button ID="btn2" runat="server" Text ="Delete" CommandName="delete" />
                </td>
            </tr>
        </table>
    </ItemTemplate>
    <EditItemStyle CssClass="itemstyle" />
    <EditItemTemplate>
        <table border="1">
            <tr>
                <td>Employee Id:</td>
                <td>
<asp:TextBox ID="txtempid" ReadOnly="true" runat= "server" Text='<%# Eval("EmpID") %>'>
                    </asp:TextBox>
                </td>
                <td rowspan="5">
                    <asp:Image ID="empimg" runat="server" ImageUrl='
                        <%# "~/Images/" + Eval("EmpImage") %>' Height="180px" />
                        <br />
                        <asp:FileUpload ID="fu1" runat= "server" />
                </td>
```

```
                    </tr>
                    <tr>
                        <td>Employee Name</td>
                        <td>
                            <asp:TextBox ID="txtempname" runat
                            ="server" Text='<%# Eval("EmpName"
                            ) %>'>
                            </asp:TextBox>
                        </td>
                    </tr>
                    <tr>
                        <td>Employee EmailId</td>
                        <td>
<asp:TextBox ID="txtempmail" runat="server" Text='<%# Eval("EmpEmail
ID") %>'>
                            </asp:TextBox>
                        </td>
                    </tr>
                    <tr>
                        <td>Employee Mobile Number</td>
                        <td>
<asp:TextBox ID="txtmbnum" runat="server" Text='<%# Eval("Emp
Mobile") %>'>
                            </asp:TextBox>
                        </td>
                    </tr>
                    <tr>
                        <td colspan="3" style="text-align:left;">
<asp:Button ID="btn3" runat="server" CommandName="update" Text=
"update" />
<asp:Button ID="btn4" runat="server" CommandName="cancel" Text=
"Cancel" />
                        </td>
                    </tr>
                </table>
            </EditItemTemplate>
        </asp:DataList>
    </div>
</form>
</body>
</html>
```

Default.aspx.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Data;
using System.Web.Configuration;
public partial class _Default : System.Web.UI.Page
{
    SqlConnection con = new SqlConnection(WebConfigurationManager.
    ConnectionStrings ["myconnection"].ConnectionString);
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            Bind();
        }
    }
    public void Bind()
    {
        SqlCommand cmd = new SqlCommand("select * from Employee",
        con);
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        DataSet ds = new DataSet();
        da.Fill(ds, "Employee");
        dl1.DataSource = ds.Tables[0];
        dl1.DataBind();
    }
    protected void dl1_EditCommand(object sender, DataListCommand
    EventArgs e)
    {
        dl1.EditItemIndex = e.Item.ItemIndex;
        Bind();
    }
    protected void dl1_CancelCommand(object sender, DataListCommand
    EventArgs e)
```

```
{
    dl1.EditItemIndex = -1;
    Bind();
}
protected void dl1_updateCommand(object sender, DataListCommand
EventArgs e)
{
    int index = e.Item.ItemIndex;
int empid = Convert.ToInt32(((TextBox)dl1.Items[index].FindControl
("txtempid")).Text);
    string empname = ((TextBox)dl1.Items[index].FindControl("txtemp
    name")).Text;
    string empmail = ((TextBox)dl1.Items[index].FindControl("txtemp
    mail")).Text;
    long empmbnum = Convert.ToInt64(((TextBox)dl1.Items[index].Find
    Control ("txtmbnum")).Text);
    FileUpload fu = (FileUpload)dl1.Items[index].FindControl("fu1");
    // to update image name in data base and image in server, If he
    selectted new Image
    if (fu.HasFile)
    {
        string filepath = System.IO.Path.Combine(Server.MapPath("~/Images
        /"), fu.FileName);
            fu.SaveAs(filepath);
            SqlCommand fcmd = new SqlCommand();
            fcmd.CommandText = "update employee set EmpImage='"+fu.File
            Name+"'";
            fcmd.Connection = con;
            con.Open();
            fcmd.ExecuteNonQuery();
            con.Close();
        }
        SqlCommand cmd = new SqlCommand("update Employee set
        EmpName = '" + empname + "',EmpEmailID='" + empmail +
        "',EmpMobile =" + empmbnum + " where EmpID = " + empid
        + "", con);
        con.Open();
        int i = cmd.ExecuteNonQuery();
        con.Close();
        if (i == 1)
        {
```

```
            Response.Write("<script>alert('Successfully updated')</script>");
            dl1.EditItemIndex = -1;
            Bind();
        }
    }
    protected void dl1_DeleteCommand(object sender, DataListCommand
    EventArgs e)
    {
        int index = e.Item.ItemIndex;
        Label empid;
        empid = (Label)dl1.Items[index].FindControl("lblempid");
        SqlCommand cmd = new SqlCommand("delete from employee
        where EmpID = " + Convert.ToInt32(empid.Text) + "", con);
        con.Open();
        int res = cmd.ExecuteNonQuery();
        con.Close();
        if (res == 1)
        {
            Response.Write("<script>alert('Successfully deleted')</script>");
            Bind();
        }
    }
}
```

❑ **Check Your Progress – 3 :**

1. Which of these events are used with the DataList Control ?

   a. OnEditCommand b. OnUpdateCommand

   c. OnCancelCommand d. All of these

7.5 DetailsView Control :

We are only able to focus on a single data item at a time while utilising the DetailsView Control. The data elements, such as database records, can be displayed, edited, inserted, and deleted with the help of this control. In addition to that, it gives us the ability to page forward and backward through a collection of data elements. Each field is consistently rendered as its own row in a separate HTML table when using the DetailsView control.

An HTML table that presents the information contained within a single database record can be generated by a DetailsView control. Both declarative and programmatic methods of databinding are supported by the DetailsView.

Syntax: <asp:DetailsView ID="DetailsView1" runat="server"></asp:DetailsView>

Now, let's look at an example of how to bind data to DetailsView using ASP.NET. In this section, we will learn how to connect and show data in an ASP.NET C# DetailsView control that is sourced from a SQL Server database.

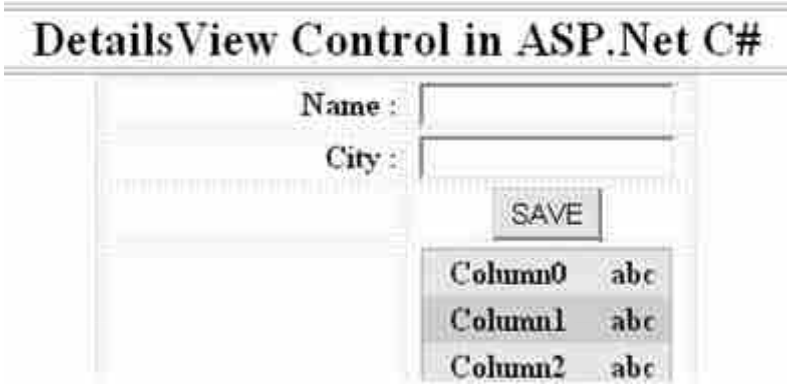
Example :

- Open the Visual Studio -> Create a new empty Web application.
- Create a New web page.
- Drag and drop DetailsView Control on web page from toolbox.
- Create New Database in SQL Server
- Make connection between Database and web application.
- Bind and Display data to DetailsView.

HTML Code :

```
<table>
<tr>
<td align="right">
Name :</td>
<td>
<asp:TextBox ID="txtname" runat="server"></asp:TextBox>
</td>
</tr>
<tr>
<td align="right">
City :</td>
<td>
<asp:TextBox ID="txtcity" runat="server"></asp:TextBox>
</td>
</tr>
<tr>
<td>
 </td>
```

```
<td>
<asp:Button ID="btnSave" runat="server" onclick="btnSave_Click"
Text="SAVE" />
</td>
</tr>
<tr>
<td>
 </td>
<td>
<asp:DetailsView ID="DetailsView1" runat="server"
BackColor="LightGoldenrodYellow" BorderColor="Tan" BorderWidth=
"1px"
CellPadding="2" DataKeyNames="ID" ForeColor="Black" GridLines=
"None"
Height="50px" onitemdeleting="DetailsView1_ItemDeleting"
onitemupdating="DetailsView1_ItemUpdating"
onmodechanging="DetailsView1_ModeChanging"
onpageindexchanging="DetailsView1_PageIndexChanging" Width=
"125px">
<FooterStyle BackColor="Tan" />
<PagerStyle BackColor="PaleGoldenrod" ForeColor="DarkSlateBlue"
HorizontalAlign="Center" />
<HeaderStyle BackColor="Tan" Font-Bold="True" />
<EditRowStyle BackColor="DarkSlateBlue" ForeColor="GhostWhite" />
<AlternatingRowStyle BackColor="PaleGoldenrod" />
</asp:DetailsView>
</td>
</tr>
</table>
```

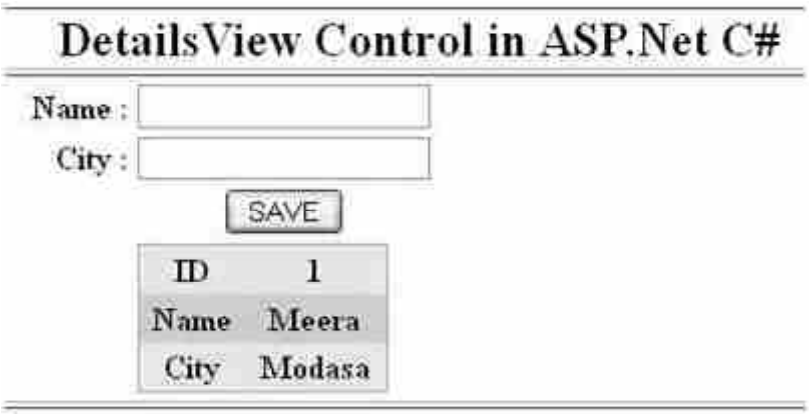


Now, once you have completed the following step to bind the DetailsView control from the Database. In this section, we begin by inserting a record into the database. Next, we get a record from the database and show it in the DetailsView control of an ASP.Net page. In order to insert and attach data to the DetailsView in ASP.net, write the following code on the SAVE button.

C# Source Code :

```
protected void btnSave_Click(object sender, EventArgs e)
{
SqlConnection SQLConn = new SqlConnection("Data Source= COMPUTER_1\\SQLEXPRESS;Initial Catalog=BOOK;Integrated Security=True");
SqlDataAdapter SQLAdapter = new SqlDataAdapter("insert into UserMst values ('"+  txtname.Text +"','"+ txtcity.Text +"')", SQLConn);
DataTable DT = new DataTable();
SQLAdapter.Fill(DT);
BindDetailsView();
}
private void BindDetailsView()
{
SqlConnection SQLConn = new SqlConnection("Data Source= COMPUTER_1\\SQLEXPRESS;Initial Catalog=BOOK;Integrated Security=True");
SqlDataAdapter SQLAdapter = new SqlDataAdapter("select * from UserMst", SQLConn);
DataTable DT = new DataTable();
SQLAdapter.Fill(DT);
DetailsView1.DataSource = DT;
DetailsView1.DataBind();
}
```

The Output of DetailsView control will look like this :



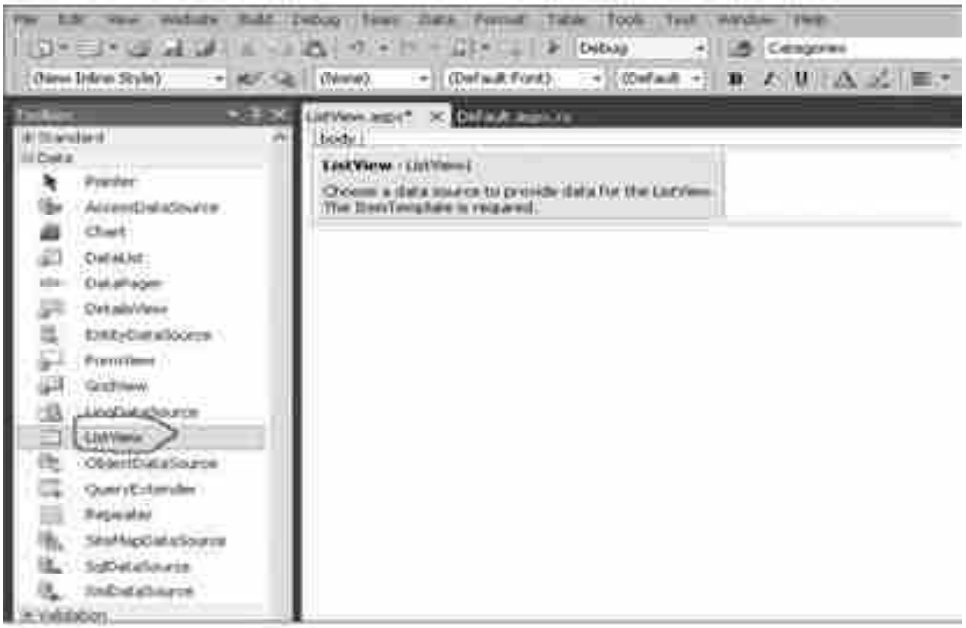
If you use the DetailsView control on your website, then it will only show a single record at a time. Because of this, you will need to use paging in order to display all of the records one at a time using the DetailsView control.

7.6 ListView Control :

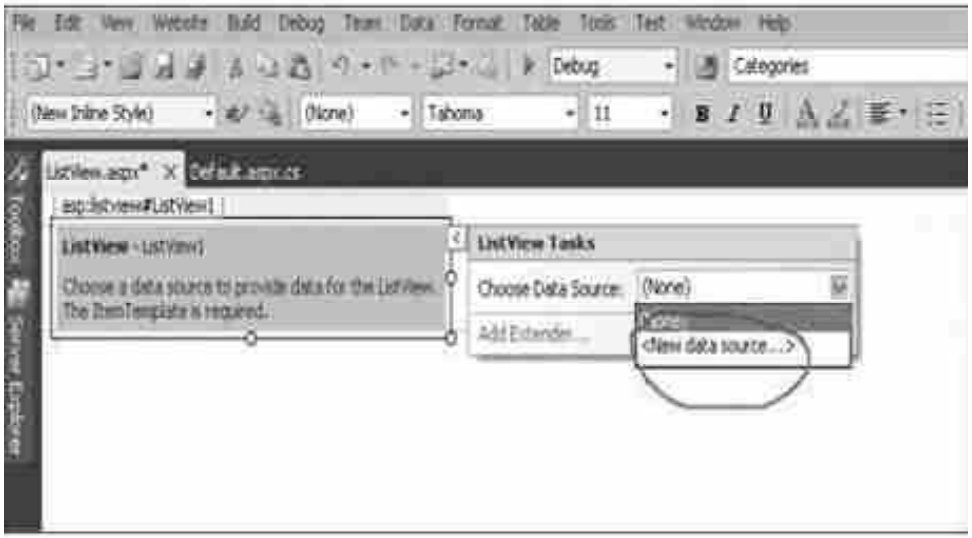
The ListView control presents the data in the form of columns and rows, and it enables the user to sort and page through the information. It is by far the most used data display control, and it is great for understanding how data display controls interact with data retrieval controls and code.

Note : Prior knowledge of SQL is required in order to interact with a database in any way, including creating, reading, updating, or deleting records. In this section, I will go over the following procedures to bind the ListView Control :

- **Data Access in ListView Control by SQL Server Express :** To begin, you will need to make a new web page; if you are unsure how to make a web page, refer back to the last tutorial for assistance. Now, navigate to the Design View and drag the ListView control from the Toolbox onto the design surface. You may do this by switching to the Design View.

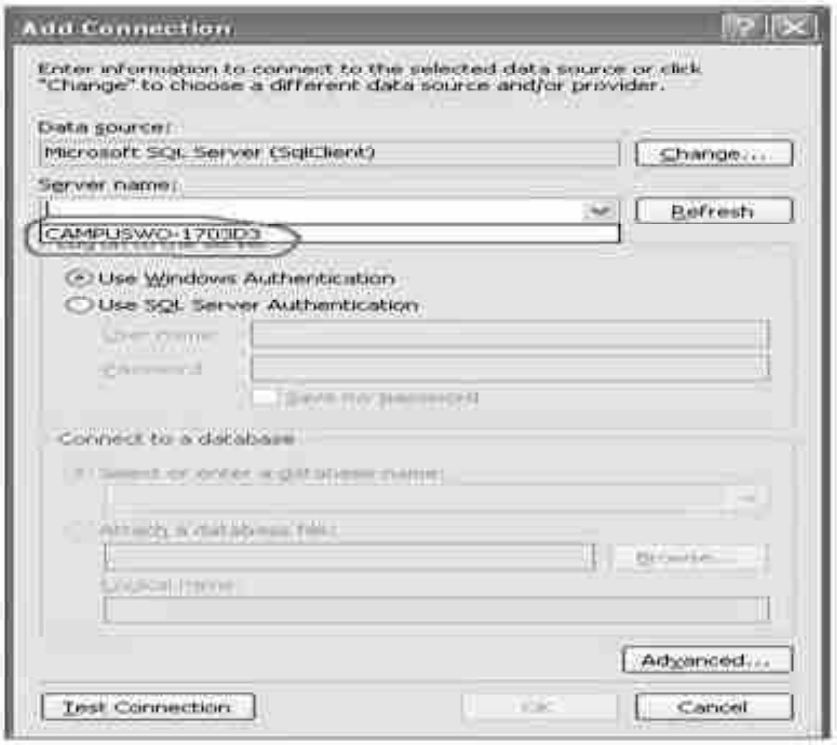


Now choose the data source over the ListView's right side.



You should now be looking at the Data Source Configuration Wizard. From here, you may select the type of Data Source you want to use, such as Access, SQL, Entity, LINQ, and so on. Then select the SQL Database that you have the most experience with. Specify an ID for the data source in the bottom box; the name is SqlDataSource1 by default, but you can opt to keep it that way or change it to anything else if you'd like.

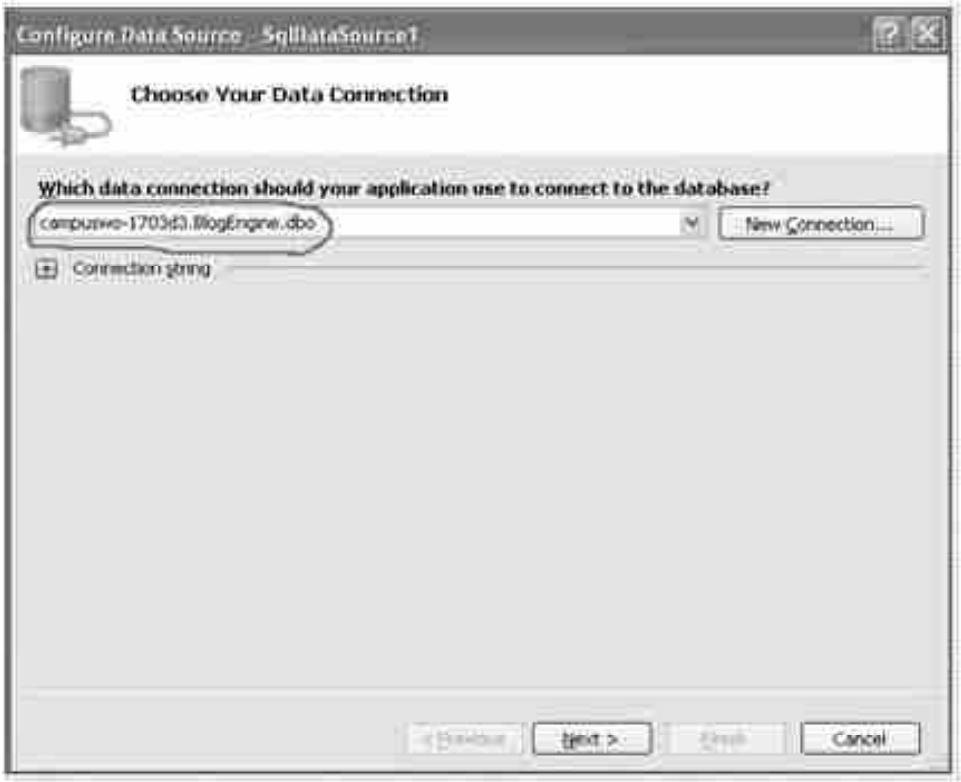
Now select the new Data Connection and Server name you want to use.



For instance, in the window labelled "Add Connection," the server's name is shown as CAMPUSWO-1703D3. Both Windows Authentication and SQL Server Authentication are available for selection here.

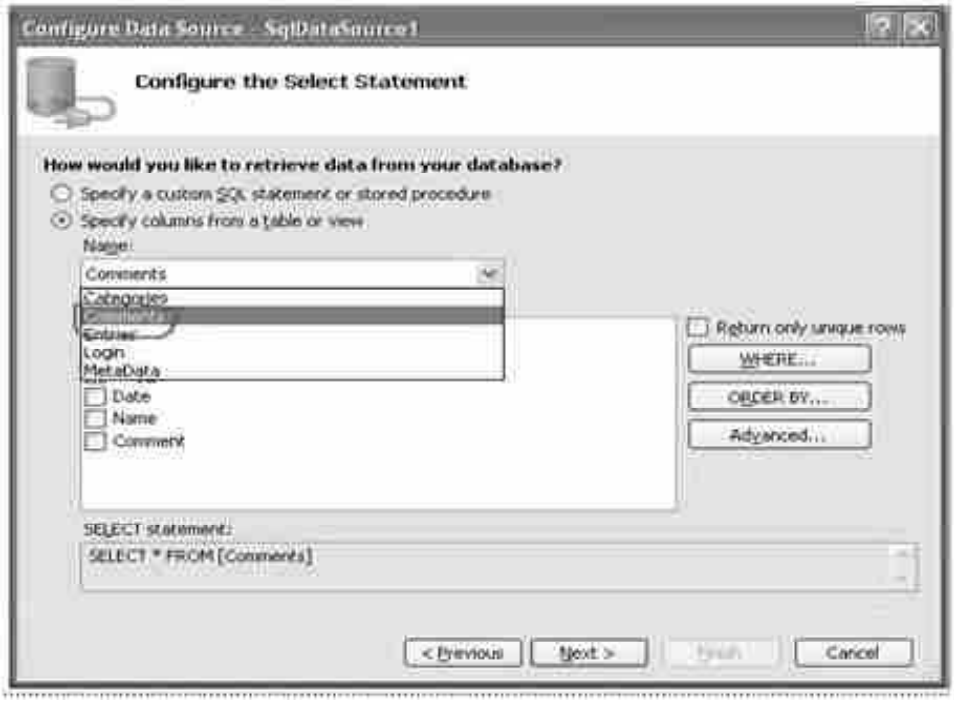
Choose the name of the database after you have authenticated. In upcoming tutorials, we will talk about another choice you have. Now is the time to check your connection; if all goes well, you can move on to the following step.

At this point, the name of your server appears alongside the database.

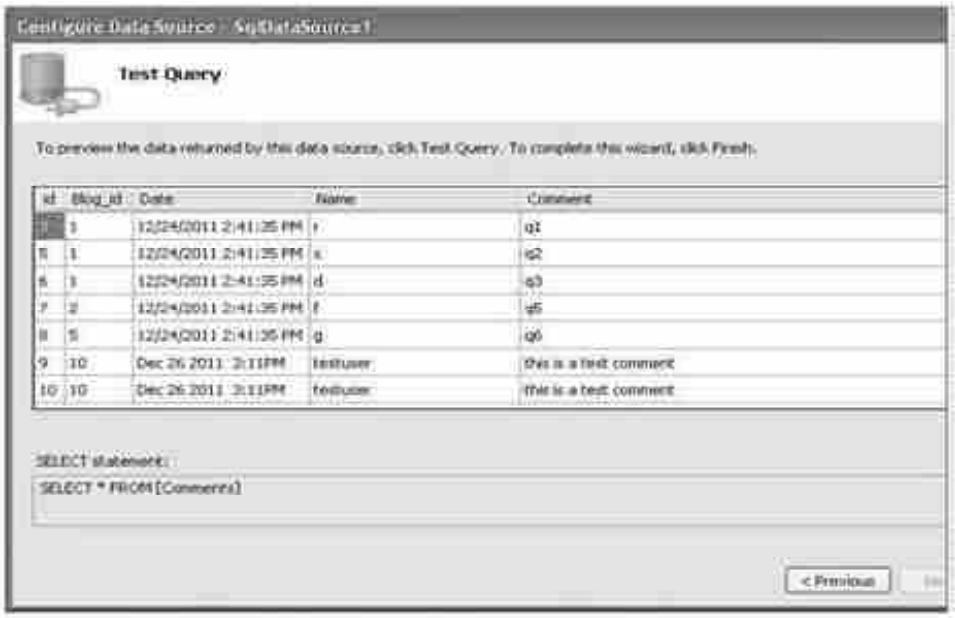


At this point, a Connection String for SQL Server Express should appear before you.

The database table or stored procedure from which the data will be retrieved should then be configured. For purposes of illustration, I've picked the Categories table. You have the option to order results by, or you may use more powerful SQL instructions.



Now put the query to the test and check out the results.



After finishing the test query, now see the HTML source file in "default.aspx" as below :

HTML Code :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ListView.
aspx.cs" Inherits="ListView"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head id="Head2" runat="server">

    <title></title>

</head>

<body>

    <form id="form2" runat="server">

    <div>

    <asp:ListView ID="ListView1" runat="server" DataSourceID=
    "SqlDataSource1">

        </asp:ListView>

        <asp:SqlDataSource ID="SqlDataSource1" runat="server"

            ConnectionString="<%$ ConnectionStrings:BlogEngine
            ConnectionString %>"

            SelectCommand="SELECT * FROM [Comments]"></asp:
            SqlDataSource>

    </div>

    </form>

</body>

</html>
```

However, the code that was just shown is not complete until a design and a data binding have been created for the ListView. Now that we have a table, we need to create a LayoutTemplate and an ItemTemplate in it, and

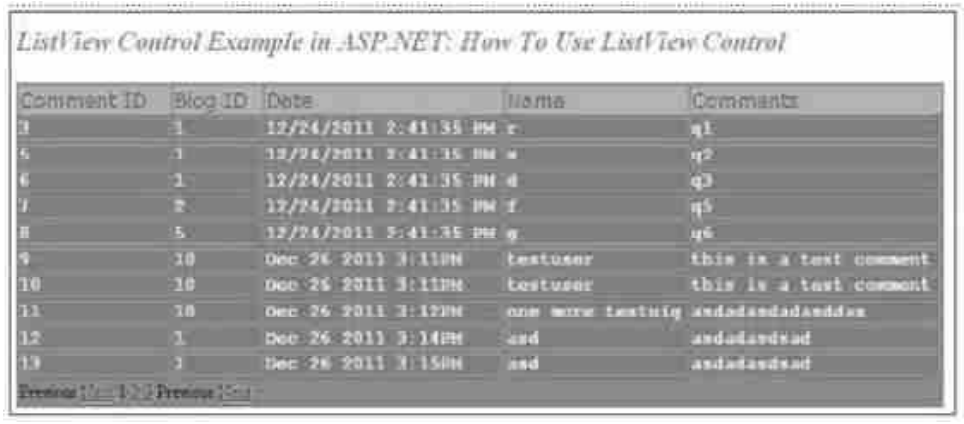
then we will tie those templates to the columns of the SQL table that is titled "Comments."

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ListView.aspx.cs" Inherits="ListView"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>ListView Control - ASP.NET Tutorials</title>
    <style type="text/css">
        .TableCSS
        {
            border-style:none;
            background-color:#3BA0D8;
            width: 850px;
        }
        .TableHeader
        {
            background-color:#66CCFF;
            color:#0066FF;
            font-size:large;
            font-family:Verdana;
            }
        .TableData
        {
            background-color:#82C13E;
            color:#fff;
            font-family:Courier New;
            font-size:medium;
            font-weight:bold;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <h2 style="color:#3BA0D8; font-style:italic;">ListView Control Example in ASP.NET: How To Use ListView Control</h2>
    <asp:ListView ID="ListView1" runat="server" DataSourceID = "SqlDataSource1">
```

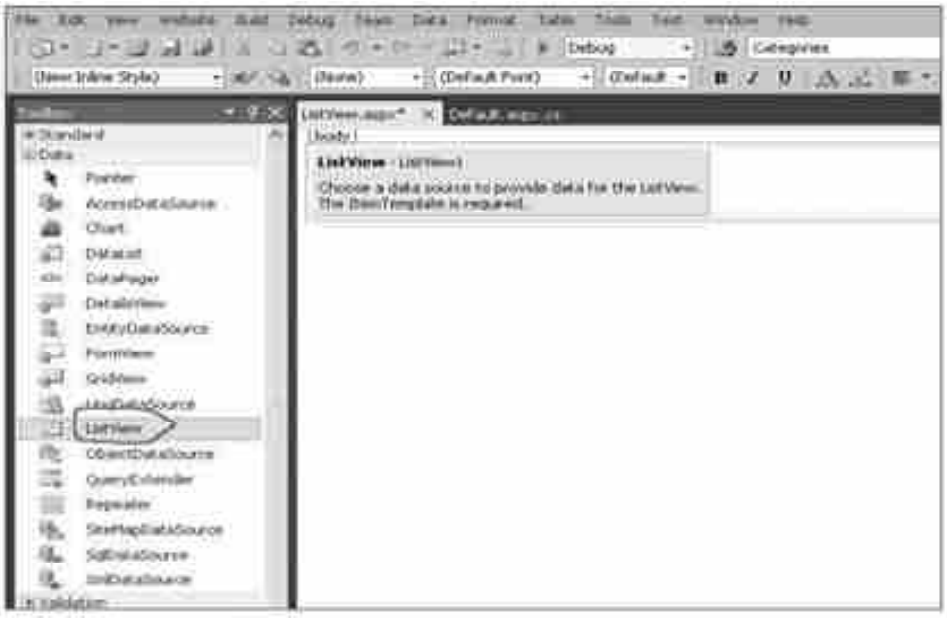
```
<LayoutTemplate>
  <table id="Table1" runat="server" class="TableCSS">
    <tr id="Tr1" runat="server" class="TableHeader">
      <td id="Td1" runat="server">Comment ID</td>
      <td id="Td2" runat="server">Blog ID</td>
      <td id="Td3" runat="server">Date</td>
      <td id="Td4" runat="server">Name</td>
      <td id="Td5" runat="server">Comments</td>
    </tr>
    <tr id="ItemPlaceholder" runat="server">
    </tr>
    <tr id="Tr2" runat="server">
      <td id="Td6" runat="server" colspan="2">
        <asp:DataPager ID="DataPager1" runat="server">
          <Fields>
            <asp:NextPreviousPagerField ButtonType
            ="Link" />
            <asp:NumericPagerField />
            <asp:NextPreviousPagerField ButtonType=
            "Link" />
          </Fields>
        </asp:DataPager>
      </td>
    </tr>
  </table>
</LayoutTemplate>
<ItemTemplate>
  <tr class="TableData">
    <td>
      <asp:Label ID="Label1" runat="server" Text='<%#
      Eval("id")%>'>
      </asp:Label>
    </td>
    <td>
  <asp:Label ID="Label2" runat="server" Text='<%# Eval
  ("Blog_id")%>'>
      </asp:Label>
    </td>
    <td>
    <asp:Label ID="Label3" runat="server" Text='<%# Eval
    ("Date")%>'>
```

```
                </asp:Label>
            </td>
            <td>
            <asp:Label ID="Label4" runat="server" Text='<%# Eval
            ("Name")%>'>
                </asp:Label>
            </td>
            <td>
        <asp:Label ID="Label5" runat="server" Text='<%# Eval
        ("Comment")%>'>
                </asp:Label>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString = " <%$ConnectionStrings:BlogEngineConnection
String%>" SelectCommand = "SELECT * FROM [Comments]"></
asp:SqlDataSource>
    </div>
    </form>
</body>
</html>
```

Now finally see the output in a browser by pressing F6 (Build) and F5 (Debug).



- **Data Access in ListView Control by Microsoft SQL Server :** Now that we have that out of the way, let's go on to learning how to populate a ListView using SQL Server queries and ASP.NET classes. First, in design view, drag a ListView control into place.



Source Code : You can view the code-behind files by pressing the F7 key at this time. This is where our ASP.NET classes and SQL Server queries will be written.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        SqlConnection con = new SqlConnection("data source=(local); database = BlogEngine; Integrated Security=true;");
        SqlDataAdapter da = new SqlDataAdapter("select * from Comments", con);
```

```
        DataSet ds = new DataSet();
        da.Fill(ds, "FillComent");
        ListView1.DataSource = ds.Tables["FillComent "];
        ListView1.DataBind();
    }
}
```

❑ **Check Your Progress – 4 :**

1. The _____ control presents the data in the form of columns and rows, and it enables the user to sort and page through the information.

   a. GridView b. DataView c. ListView d. All of these

7.7 FormView Control :

A new data-bound control called FormView has been introduced, and it is nothing more than a templated version of the DetailsView control. The most significant distinction between DetailsView and FormView is that the latter requires the user to specify the rendering template to be used for each individual item. The style properties of the table tag are used to implement its properties, such as BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, and Height, among other things.

Properties of FormView Control

Properties	Description
EditItemTemplate	The template that is applied whenever a record is being modified in any way.
InsertItemTemplate	The template that is used when a record is being created.
ItemTemplate	The template that is used to render the record to display only.

Events of FormView Control

Events	Description
ChangeMode	ReadOnly/Insert/Edit. Alter the working mode of the control so that it corresponds with the kind of FormViewMode that has been defined.
InsertItem	Utilized in the process of inserting the record into the database. When the DetailsView control is operating in the insert mode, you are required to use this method.
UpdateItem	Utilized for the purpose of updating the currently stored entry in the database. When the DetailsView control is in the edit mode, you are required to call this method.
DeleteItem	This function allows the current record to be removed from the database.

Example : Let us take an example of FormView, this is the form through which we insert the values into the database :

Now, we insert the values in it and click on insert button, like this :

After inserting the values, it will look like this, from here we can edit or delete the records from the database.

When we click on Edit Button, the form will look like this :

Now, change the values like this and click on update button :

When we click on Delete button to delete the particular selected record from the database, it will delete from there :

Let us take a look on the code for it.

HTML Code :

```
// FormView control
<asp:FormView ID="FormView1" runat="server" CellPadding="4" ForeColor="#333333"
    DataKeyNames="AutoID" DataSourceID="SqlDataSource1" Allow Paging="true">
    <FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor= "White" />
    <RowStyle BackColor="#EFF3FB" />
    <PagerStyle BackColor="#2461BF" ForeColor="White" Horizontal Align="Center" />
    <HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor= "White" />
    <ItemTemplate>
        <table border="1">
            <tr>
                <td>AutoID</td>
                <td><%# Eval("AutoID") %></td>
            </tr>
            <tr>
                <td>Name</td>
                <td><%# Eval("Name") %></td>
            </tr>
            <tr>
                <td>Address</td>
                <td><%# Eval("Address") %></td>
            </tr>
            <tr>
                <td>Phone</td>
                <td><%# Eval("Phone") %></td>
            </tr>
            <tr>
                <td>City</td>
                <td><%# Eval("City") %></td>
            </tr>
            <tr>
                <td> </td>
                <td>
            <asp:Button ID="btnEdit" runat="Server" CommandName="Edit" Text="Edit" />
```

```
        <asp:Button ID="btnInsert" runat="Server" CommandName="New"
        Text="New" />
    <asp:Button ID="btnDelete" runat="Server" CommandName="Delete"
    Text="Delete" OnClientClick="return confirm('Are you sure to
    Delete?');" />
            </td>
        </tr>
    </table>
    </ItemTemplate>
    <EditItemTemplate>
        <table border="1">
            <tr>
                <td>AutoID</td>
                <td><%# Eval("AutoID") %></td>
            </tr>
            <tr>
                <td>Name</td>
            <td>
<asp:TextBox ID="TextBox1" runat="Server" Text='<%# Bind("Name")
%>'> </asp:TextBox></td>
            </tr>
            <tr>
                <td>Address</td>
                <td>
<asp:TextBox ID="TextBox2" runat="Server" Text='<%# Bind("Address")
%>'> </asp:TextBox></td>
            </tr>
            <tr>
                <td>Phone</td>
                <td>
<asp:TextBox ID="TextBox3" runat="Server" Text='<%# Bind("Phone")
%>'> </asp:TextBox></td>
            </tr>
            <tr>
                <td>City</td>
            <td>
<asp:TextBox ID="TextBox4" runat="Server" Text='<%# Bind("City")
%>'> </asp:TextBox> </td>
            </tr>
            <tr>
                <td> </td><td>
```

```
<asp:Button ID="btnUpdate" runat="Server" CommandName="Update"
Text="Update" />
<asp:Button ID="Button1" runat="Server" CommandName="Cancel"
Text="Cancel" />
            </td></tr>
        </table>
    </EditItemTemplate>
    <InsertItemTemplate>
        <table border="1">
            <tr>
                <td>AutoID</td>
                <td><%# Eval("AutoID") %></td>
            </tr>
            <tr>
                <td>Name</td>
            <td>
<asp:TextBox ID="TextBox1" runat="Server" Text='<%# Bind("Name")
%>'> </asp:TextBox> </td>
            </tr>
            <tr>
                <td>Address</td>
                <td>
<asp:TextBox ID="TextBox2" runat="Server" Text='<%# Bind("Address")
%>'> </asp:TextBox></td>
            </tr>
            <tr>
                <td>Phone</td>
            <td>
<asp:TextBox ID="TextBox3" runat="Server" Text='<%# Bind("Phone")
%>'> </asp:TextBox></td>
            </tr>
            <tr>
                <td>City</td>
            <td>
<asp:TextBox ID="TextBox4" runat="Server" Text='<%# Bind("City")%>'>
</asp:TextBox></td>
            </tr>
            <tr>
                <td> </td>
            <td>
```

```
    <asp:Button ID="btnSave" runat="Server" CommandName="insert"
    Text="Insert" />
<asp:Button ID="Button1" runat="Server" CommandName="Cancel" Text=
"Cancel" />
            </td>
        </tr>
    </table>
    </InsertItemTemplate>
</asp:FormView>
    // SqlDataSource Control
<asp:SqlDataSource ID="SqlDataSource1" runat="server" Connection
String='<%$ ConnectionStrings:ConnStr %>'
    SelectCommand="Select * FROM SampleForTutorials ORDER BY
    [Name]"
    DeleteCommand="Delete FROM SampleForTutorials WHERE AutoID
    = @AutoID"
    UpdateCommand="UPDATE SampleForTutorials SET Name = @Name,
    Address = @Address, Phone = @Phone, City = @City WHERE
    AutoID = @AutoID"
    InsertCommand="INSERT INTO SampleForTutorials (Name, Address,
    Phone, City) VALUES (@Name, @Address, @Phone, @City)">
        <DeleteParameters>
            <asp:Parameter Name="AutoID" />
        </DeleteParameters>
        <UpdateParameters>
            <asp:Parameter Name="AutoID" Type="Int32" />
            <asp:Parameter Name="Name" Type="string" Size="50" />
            <asp:Parameter Name="Address" Type="string" Size="200" />
            <asp:Parameter Name="Phone"  Type="string" Size="50" />
            <asp:Parameter Name="City" Type="string" Size="20" />
        </UpdateParameters>
        <InsertParameters>
            <asp:Parameter Name="Name" Type="string" Size="50" />
            <asp:Parameter Name="Address" Type="string" Size="200" />
            <asp:Parameter Name="Phone" Type="string" Size="50" />
            <asp:Parameter Name="City" Type="string" Size="20" />
            <asp:Parameter Name="AutoID" Type="Int32" />
        </InsertParameters>
        </asp:SqlDataSource>
```

C# Source Code :

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class template_formview : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            if (FormView1.PageCount == 0)
                FormView1.DefaultMode = FormViewMode.Insert;
        }
        else
        {
            FormView1.DefaultMode = FormViewMode.ReadOnly;
        }
    }
}
```

❑ **Check Your Progress – 5 :**

1. Which event is not included in the FormView Control ?

   a. InsertItem b. UpdateItem c. DeleteItem d. None of these

7.8 SqlDataSource Control :

Connecting to a relational database can be accomplished with the use of a data source control known as the SqlDataSource control. You can even connect to Oracle or any other data source that is available through OLEDB and ODBC. Both of these protocols allow for such connections. Both the ConnectionString and the ProviderName properties are required in order to successfully connect to the database. The first one must specify the connection string, while the second one must specify the name of the provider. System is selected as the default supplier. Data. Connecting to a Sql Server database using SqlClient is what this term refers to.

Properties of Configuring Data Source

Properties	Description
InsertCommand, InsertParameters, InsertCommandType	Gets or sets the SQL Statement, parameter and type of command (text or stored procedure) to insert a record.
DeleteCommand, DeleteParameters, DeleteCommandType	Gets or sets the SQL Statement, parameters and type of the command (text or stored procedure) to delete a record.
UpdateCommand, UpdateParameters, UpdateCommandType	Gets or sets the SQL Statement, parameters and type of the command (text or stored procedure) to update a record.
SelectCommand, SelectParameters, SelectCommandType	Gets or sets the SQL Statement, parameters and type of the command (text or stored procedure) to select record(s).

Parameter Types of DataSource Controls

Properties	Description
ControlParameter	Gets the value from any public property of the server control.
FormParameter	Gets the value of any input field from the form.
QueryStringParameter	Gets the value from specified querystring.
Parameter	Gets the parameter value assigned in the code.

Other Properties of SqlDataSource Control

Properties	Description
ProviderName	Gets or sets the .NET ProviderName name.
DataSourceMode	DataSet/DataReader. Used to specify the data source mode. In case of DataReader paging is not supported in the target control (GridView etc.).
ConnectionString	Gets or sets the connection string to connect to the database.
ConflictDetection	CompareAllValues/OverWriteChanges. Used to determine how conflict will be handled in case of updation or deletion of the records.

Caching Properties of SqlDataSource Control

Properties	Description
EnableCaching	true/false. Used to indicate whether enable caching or not.
CacheDuration	Used to indicate number of seconds the data shoud be maintained in the cache.
CacheExpirationPolicy	Absolute/Sliding. Used to indicate if the cache policy is absolute or sliding. Absolute : The data is removed after specified duration. Sliding: The data is removed if it is not used for specified duration.
EnableCaching	true/false. Used to indicate whether enable caching or not.

Example : Let us use DetailsView Control with SqlDataSource Control to access data from the database.

Like this we can access data from the database and see on our DetailsView Control.

HTML Code :

```
// DetailsView Control
    <asp:DetailsView ID="DetailsView1" runat="Server" CellPadding="4"
ForeColor="#333333" GridLines="None"
    Width="100%" DataSourceID="SqlDataSource1" AllowPaging="True"
AutoGenerateRows = "True" DataKeyNames="AutoID">
    <FooterStyle BackColor="#5D7B9D" Font-Bold="True" Fore
    Color="White" />
    <CommandRowStyle BackColor="#E2DED6" Font-Bold="True" />
    <EditRowStyle BackColor="#999999" />
    <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
    <PagerStyle BackColor="#284775" ForeColor="White" Horizontal
    Align="Center" />
    <FieldHeaderStyle BackColor="#E9ECF1" Font-Bold="True" />
    <HeaderStyle BackColor="#5D7B9D" Font-Bold="True" Fore
    Color="White" />
    <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
</asp:DetailsView>
    // SqlDataSource Control
    <asp:SqlDataSource ID="SqlDataSource1" runat="server" Connection
String='<%$ ConnectionStrings:ConnStr %>'
    SelectCommand="Select * FROM SampleForTutorials ORDER BY
[Name]" DataSourceMode ="DataSet">
    </asp:SqlDataSource>
```

7.9 AccessDataSource Control :

The AccessDataSource control is a type of data source control that is utilised in the process of establishing a connection to an MS Access database. The ConnectionString property has been replaced by the DataFile property, which allows the user to provide the path to the MS-Access database. Because the AccessDataSource control is derived from the SqlDataSource control, the properties of AccessDataSource are nearly identical to those of SqlDataSource.

AccessDataSource Control works similar as the SqlDataSource Control.

Properties of Configuring Data Source

Properties	Description
InsertCommand, InsertParameters, InsertCommandType	Gets or sets the SQL Statement, parameter and type of command (text or stored procedure) to insert a record.
DeleteCommand, DeleteParameters, DeleteCommandType	Gets or sets the SQL Statement, parameters and type of the command (text or stored procedure) to delete a record.
UpdateCommand, UpdateParameters, UpdateCommandType	Gets or sets the SQL Statement, parameters and type of the command (text or stored procedure) to update a record.
SelectCommand, SelectParameters, SelectCommandType	Gets or sets the SQL Statement, parameters and type of the command (text or stored procedure) to select record(s).

Parameter Types of DataSource Controls

Properties	Description
ControlParameter	Gets the value from any public property of the server control.
FormParameter	Gets the value of any input field from the form.
QueryStringParameter	Gets the value from specified querystring.
Parameter	Gets the parameter value assigned in the code.

Other Properties of AccessDataSource Control

Properties	Description
ProviderName	Gets or sets the .NET ProviderName name.
DataSourceMode	DataSet/DataReader. Used to specify the data source mode. In case of DataReader paging is not supported in the target control (GridView etc.).
ConnectionString	Gets or sets the connection string to connect to the database.
ConflictDetection	CompareAllValues/OverWriteChanges. Used to determine how conflict will be handled in case of updation or deletion of the records.

Caching Properties of AccessDataSource Control

Properties	Description
EnableCaching	true/false. Used to indicate whether enable caching or not.
CacheDuration	Used to indicate number of seconds the data shoud be maintained in the cache.
CacheExpirationPolicy	Absolute/Sliding. Used to indicate if the cache policy is absolute or sliding. Absolute : The data is removed after specified duration. Sliding: The data is removed if it is not used for specified duration.
EnableCaching	true/false. Used to indicate whether enable caching or not.

HTML Code :

```
// DetailsView Control
<asp:DetailsView ID="DetailsView1" runat="Server" CellPadding="4" ForeColor= "#333333" GridLines="None" Width="100%" DataSource ID="AccessDataSource1" AllowPaging="True" AutoGenerateRows="True" DataKeyNames="pid">
    <FooterStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor= "White" />
    <CommandRowStyle BackColor="#E2DED6" Font-Bold="True" />
    <EditRowStyle BackColor="#999999" />
    <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
    <PagerStyle BackColor="#284775" ForeColor="White" Horizontal Align="Center" />
    <FieldHeaderStyle BackColor="#E9ECF1" Font-Bold="True" />
    <HeaderStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor= "White" />
    <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
</asp:DetailsView>
// AccessDataSource Control
<asp:AccessDataSource ID="AccessDataSource1" runat="Server" Data File="~/AnyDb.mdb"
SelectCommand="SELECT * FROM payments ORDER BY pdate DESC">
</asp:AccessDataSource>
```

7.10 Repeater Control :

The repeater control presents the data in a format chosen by the user for the layout, and it does so using that format. This setup may be horizontal or vertical, depending on your preference. This element only duplicates the HTML and ASP.NET controls that are located within a template block in order to perform the function that has been assigned to it. This is all that is needed in order for it to work.

Important Properties of Repeater Control

Properties	Description
AlternatingItemTemplate	The rendering of each alternate item will be defined using this template.
FooterTemplate	The template will describe how the footer should be rendered.
HeaderTemplate	A template that will describe how the header should be rendered.
Items	This function retrieves the collection of RepeaterItem objects.
ItemTemplate	To determine how objects are presented, a template is provided.
SeparatorTemplate	This template will determine how a separator will be rendered between the items in the list.

Example : Let us continue with the example which we discussed in previous topics. We can see the data like this after using Repeater Control.

17748	ROSA MARIA BERNAL MARTINEZ	PRIVADA CALLE 13 MANZANA 25 CASA 20	+527225590997	METEPEC
17749	ROSA MARIA BERNAL MARTINEZ	PRIVADA CALLE 13 MANZANA 25 CASA 20	+527225590997	METEPEC
17750	ROSA MARIA BERNAL MARTINEZ	PRIVADA CALLE 13 MANZANA 25 CASA 20	+527225590997	METEPEC

HTML Code :

```
// Repeater control
    <asp:Repeater ID="Repeater1" runat="server" DataSourceID=
    "SqlDataSource1">
    <HeaderTemplate>
       <h3>This is the Header of the Repeater Control</h3>
    </HeaderTemplate>
    <AlternatingItemTemplate>
       <table border="1" style="background-color:#c0c0c0;" width=
       "100%">
          <tr>
             <td style="width:10%;"><%# Eval("AutoID") %></td>
             <td style="width:25%;"><%# Eval("Name") %></td>
             <td style="width:40%;"><%# Eval("Address") %></td>
             <td style="width:10%;"><%# Eval("Phone") %></td>
             <td style="width:15%;"><%# Eval("City") %></td>
          </tr>
       </table>
    </AlternatingItemTemplate>
    <ItemTemplate>
       <table border="1" width="100%">
          <tr>
             <td style="width:10%;"><%# Eval("AutoID") %></td>
             <td style="width:25%;"><%# Eval("Name") %></td>
             <td style="width:40%;"><%# Eval("Address") %></td>
             <td style="width:10%;"><%# Eval("Phone") %></td>
             <td style="width:15%;"><%# Eval("City") %></td>
          </tr>
       </table>
```

```
</ItemTemplate></asp:Repeater>
// SqlDataSource control
<asp:SqlDataSource ID="SqlDataSource1" runat="server" Connection
String='<%$ ConnectionStrings:ConnStr %>'
SelectCommand="Select * FROM SampleForTutorials ORDER BY
[Name]">
</asp:SqlDataSource>
```

7.11 XmlDataSource Control :

The XmlDataSource control is a one-of-a-kind data source control that offers support for hierarchical as well as tabular representations of data. This support can be found in the control's properties. The user can access this functionality at their discretion. Access to this assistance will be made available to you through the characteristics of the control. In contrast to other DataSource Controls, this DataSource Control exclusively offers read-only method operations. Other DataSource Controls support both types of method operations. It is possible that other DataSource Controls will support either one of the two separate sorts of method operations.

Important Properties of XmlDataSource Control

Properties	Description
Data	Retrieves or modifies the blocks of text that make up the XML contents that will be bound to the target control.
DataFile	The path of the XML file that will be displayed and bound to the target controls.
XPath	This function retrieves an XPath query that can be applied to XML data.
Transform	A section of XSLT text that is applied to the data in order to convert it.
TransformFile	The path of the XSL file that defines the transformation that should be applied to the XML file.
EnableCaching	true/false. To enable caching or not to enable caching.
CacheDuration	The amount of time, in seconds, that data will be stored in the cache.
CacheExpirationPolicy	Absolute/Sliding. Application of cache policy to be carried out on. After a predetermined amount of time, the data is deleted. The information is discarded after a predetermined period of time if it is not used.

Example : We can see the data like this after using XmlDataSource Control.

HTML Code :

```
// TreeView Control
<asp:TreeView ID="TreeView1" runat="server" DataSourceID= "XmlDataSource1">
    <DataBindings>
        <asp:TreeNodeBinding TextField="#innertext" DataMember= "Name" />
        <asp:TreeNodeBinding TextField="#innertext" DataMember= "From" />
    </DataBindings>
    </asp:TreeView>
// XmlDataSource Control
<asp:XmlDataSource ID="XmlDataSource1" runat="Server" DataFile = " ~/tutorials /controls/controldata/XMLFile.xml" XPath="Writer Record/Writers"></asp: XmlDataSource>
// XML File
<WriterRecord>
    <Writers>
        <Name>Sheo Narayan</Name>
        <From>Hyderabad, India</From>
    </Writers>
    <Writers>
        <Name>Vijay Bhandari</Name>
        <From>Udaipur, India</From>
    </Writers>
    <Writers>
        <Name>Jayesh Shankara</Name>
        <From>Aurangabad, India</From>
    </Writers>
    <Writers>
```

```
        <Name>Dr. Sunita Narayan</Name>
        <From>Rajasthan, India</From>
    </Writers>
</WriterRecord>
```

7.12 EntityDataSource/ObjectDataSource Control :

The user is given the ability to design their very own specialised classes in order to bind data to the controls when they make use of the ObjectDataSource. This is done in order to accomplish the task. Before a class can be used as an ObjectDataSource, it needs to have its own methods for selecting data, adding new data, removing old data, and updating existing data. These methods need to provide the class with the capability to select data, delete data, insert data, and update data. Because ObjectDataSource does not permit the data to be updated in batches, the update method needs to be able to deal with the modification of a single record at a time.

Important Properties of EntityDataSource/ObjectDataSource Control

Properties	Description
InsertMethod, InsertParameters	This method either retrieves or sets the name of the method to be used to perform the insert operation.
UpdateMethod, UpdateParameters	This method either retrieves or sets the name of the method to be called in order to perform an update action.
DeleteMethod, DeleteParameters	This method either retrieves or sets the name of the method to be called in order to perform the delete operation.
SelectMethod, SelectParameters	This method either retrieves or sets the name of the method to be called in order to perform a select operation.
ConvertNullToDBNull	true/false. Used to indicate whether the method should convert a null parameter that was supplied to it to System.DBNull.
DataObjectTypeName	This function either retrieves or sets the name of the class that is utilised to carry out insert, update, delete, and select operations.
EnablePaging	true/false. If the procedure supports paging, then this value will be true.
FilterExpression, FilterParameters	This method either retrieves or sets the filter expression and filter parameter that are used to filter the select action.
SelectCountMethod	Provides access to or the ability to set the name of the technique that is used to count the records that have been selected (selected by SelectMethod method).
SortParameterName	This function either retrieves or sets the name of the input parameter that was used to sort the record that was selected.

StartRowIndex ParameterName	Only operates when the EnablePaging=true flag is set. This method either retrieves or sets the name of the parameter of the select method that is used as the starting record from which records are retrieved.
ConvertNullToDBNull	ConvertNullToDBNull true/false. Used to indicate whether the method should convert a null parameter that was supplied to it to System.DBNull.

Example : We can see the data like this after using EntityDataSource Control.

HTML Code :

```
// GridView Control
<asp:GridView ID="GridView1" DataSourceID="ObjectDataSource1" runat="server" CellPadding="4" ForeColor="#333333" GridLines="None" PageSize="10">
    <FooterStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
    <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
    <EditRowStyle BackColor="#999999" />
    <SelectedRowStyle BackColor="#E2DED6" Font-Bold="True" Fore Color="#333333" />
    <PagerStyle BackColor="#284775" ForeColor="White" Horizontal Align="Center" />
    <HeaderStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
    <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
</asp:GridView>
// ObjectDataSource Control
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
    TypeName="ObjectDataSourceSample" SelectMethod="Load">
    <SelectParameters>
<asp:ControlParameter ControlID="txtGreater" PropertyName="Text" Name= "greaterThan" Type="int32" />
```

```
    </SelectParameters>
</asp:ObjectDataSource>
// Parameter of SelectMethod
Show record WHERE AutoID greater than <br />
<asp:TextBox ID="txtGreater" runat="Server" Text="3" Columns="5"></
asp:TextBox>
<asp:CompareValidator ID="Cm" runat="server" ControlToValidate=
"txtGreater" Text="Numeric only" Display="Dynamic" Operator="Data
TypeCheck" Type="integer"></asp:CompareValidator>
<asp:Button ID="btnSubmit" runat="Server" Text="Submit" />
```

C# Source Code :

```
/// Load records
public DataSet Load(int greaterThan)
{
    DataSet dSet = new DataSet();
    string connStr = ConfigurationManager.ConnectionStrings
    ["ConectionString"]. ToString();
    using (SqlConnection conn = new SqlConnection(connStr))
    {
        conn.Open();
        SqlDataAdapter dAd = new SqlDataAdapter("SELECT * FROM
        Sample WHERE AutoID > "+ greaterThan +" ORDER BY
        Name", conn);
        dAd.Fill(dSet, "SampleTable");
    }
    return dSet; }
```

7.13 Let Us Sum Up :

In this unit, we have learned about ASP.NET Details View which supports adding a SqlDataSource control to a page with INSERT instructions. Asp.Net GridView's Boundfield related columns to data source properties which displays data from the table's database field. This section covered date, currency, and value formatting in the grid view. This unit mainly focused on Database Connectivity through various controls like ListView, DetailsView, DataList View etc. We have learned about Delete datafield links. This unit introduced data binding and radio buttons. Creating a custom column class and adding a checkbox to the gridview header template also covered in this unit. This unit showed how to use the Gridview AutoGenerateColumns attribute.

7.14 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : b

- **Check Your Progress 2 :**

  1 : b

- **Check Your Progress 3 :**

  1 : d

- **Check Your Progress 4 :**

  1 : c

- **Check Your Progress 5 :**

  1 : d

7.15 Glossary :

1. **Data :** Data is information that has been translated into a form that is efficient for movement or processing.
2. **Databind :** Data binding is the process that couples two data sources together and synchronizes them.
3. **Repeater Control :** The Repeater control is used to display a repeated list of items that are bound to the control.
4. **DataAdapter :** The DataAdapter serves as a bridge between a DataSet and a data source for retrieving and saving data.

7.16 Assignment :

1. Explain FormView Control in detail.

7.17 Activities :

1. Use the DetailsView control and make a website application to show Employee Details.

7.18 Case Study :

Generalised WHERE clause and discuss.

7.19 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

BLOCK SUMMARY :

In this block, we have learnt about various data validation controls in detail. You got knowledge about database connectivity and various database connection tools, like DataList, GridView, FormView, DetailsListView Control etc. in detail. We have learned about SQL Queries for CRUD Operations through Database Controls.

We have also learnt about various datasource controls, Key Constraints, Insertion of Records etc. The block gives an idea on the study and concept of Client side validation on browser and scripting language. You have been well explained on the concepts of SQLDataSource control.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Explain Client side and Server Side Validation Controls.
2. Write a short note on XMLDataSource Control.
3. Explain Custom Validation in detail.
4. Write short note on GridView Control ?

❖ **Long Questions :**

1. Explain CRUD Operationsin detail in ASP.NET ?
2. Explain various Validation Controls in detail with suitable example ?
3. Explain Data Connection and its factors ?

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	5	6	7
No. of Hrs.			

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....

Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-504

INTERNET PROGRAMMING (ASP.NET USING C#)

BLOCK 3 : WORKING WITH DATABASE

WORKING WITH DATABASE

Block Introduction :

The ASP.NET Details View offers support for the insertion of a SqlDataSource control to a page. This control can be configured to include INSERT instructions in the database statements it executes. In order to use the insert command, you will need to ensure that the AutoGenerateInsertButton property is set to the true value. If this is not the case, you will not be able to use the insert command. When working with Asp.GridView, the Bound field in Net ties a column to a property in the items being retrieved from the data source. The movement of data in both directions is made easier as a result of this. It is a column in the database that stores data from the table, and it displays the data that is stored in the column of the table that is stored in the database. The column in question is part of the table that is saved in the database. In the following section, we will cover the basics of structuring the grid view by making use of date, currency, and value information.

The investigation of Grid Control as a concept, in addition to its visual qualities, will function as the primary focal point of this block. You will develop an idea of how the Delete field's links ought to be displayed after doing so. During the course of this block, you will have the opportunity to learn about the principles of Data binding as well as the various radio buttons, and you will also get a grasp of these topics. You will also have the idea of making a personalised column class and incorporating a checkbox into the gridview header design explained to you as part of this process. In this block, you will see examples of how to put the Gridview AutoGenerateColumns attribute into practise in a variety of different contexts. In this block, you will also get the knowledge about the working of LINQ queries and the ADO.NET framework.

Block Objectives :

After learning this block, you will be able to understand :

- How to bind the data with the datacontrols
- How do we populate dynamic data using SqlDataSource Control
- How do we use LINQ queries with objects, with ADO.NET and with XML using LINQDataSource control and
- The use of ADO.NET Framework

Block Structure :

Unit 8 : Working with Dropdown Lists, Radio Lists, Radio Buttons, Checkboxes, Revisiting Data Binding

Unit 9 : LINQ Queries

Unit 10 : ADO.NET Framework

Unit

08

WORKING WITH DROPDOWN LISTS, RADIO BUTTONS, CHECKBOXES, REVISITING DATA BINDING

UNIT STRUCTURE

8.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About how to bind the data to a list web control
- About the benefits of populating the dynamic list using SqlDataSource
- About the filteration of data using Dropdownlist Control
- About the Checkoxlist Control
- About the selection of items from any list control
- About the selection of values from the list control
- About the Radiobuttonlist Control
- About RepeatDirection Control

- About the Checkboxfield
- About the Hyperlinkfield
- About the Imagefield
- About Sql Wild Cards
- About the Two-Way Data Binding

8.1 Introduction :

You are able to link any server control to simple attributes, collections, expressions, and/or methods when using the strong capability of data binding that ASP.NET makes available. When you utilise data binding, you have more flexibility when using data from a database or other means than when you do not use data binding.

A web server control known as the DropDownList is responsible for generating an HTML Select component when it is invoked. It gives us the ability to choose an alternative from the list that drops down. It is possible for it to hold any amount of objects. A tag that may be used to generate DropDownList for online applications is provided by ASP.NET.

The user has the option to pick many items at the same time by making use of the ASP.NET CheckBoxList web control, which is a web control that can be used to collate the items that can be checked. Using the functions for data binding, you are able to build this list of things that are contained in the CheckBoxList in a dynamic manner.

The RadioButtonList Control functions in the same way as the DropDownList Control, except instead of displaying a drop-down menu, it shows a list of radio buttons that can be arranged in a horizontal or vertical fashion. You are only able to choose one choice at a time from the RadioButtonList that has been provided. These choices are incompatible with one another.

The ImageField class is utilised by data-bound controls such as the GridView and the DetailsView in order to display an image in conjunction with each record that is shown.

8.2 Data Binding to A List Web Control :

The DataBind function is inherited by each and every ASP.NET online form control from its respective parent Control class. This endows each and every ASP.NET web form control with the innate potential to bind data to at least one of its attributes. This type of data binding is sometimes referred to as inline data binding or simple data binding.

Attaching any collection (item collection) that implements the IEnumerable interface, or the DataSet and DataTable classes, to the DataSource property of the control is what's meant by "simple data binding."

On the other hand, certain controls can make use of a DataSource control in order to bind records, lists, or columns of data into their own internal structure. The BaseDataBoundControl class is where these controls get their start in life. This type of data coupling is known as declarative data binding.

Controls that are data-bound require assistance in implementing capabilities such as sorting, paging, and altering data collections. The data source controls provide this assistance.

An abstract class called BaseDataBoundControl is inherited by two other abstract classes, and those classes are as follows :

- DataBoundControl
- HierarchicalDataBoundControl

The abstract DataBoundControl class is inherited by two further abstract classes, which are as follows :

- ListControl
- CompositeDataBoundControl

The following are examples of controls that are descended from the ListControl abstract class and are capable of doing simple data binding :

- BulletedList
- CheckBoxList
- DropDownList
- ListBox
- RadioButtonList

The CompositeDataBoundControl abstract class is the parent class from which all of the controls that are capable of declarative data binding (a more involved kind of data binding) are derived. These controls are the following :

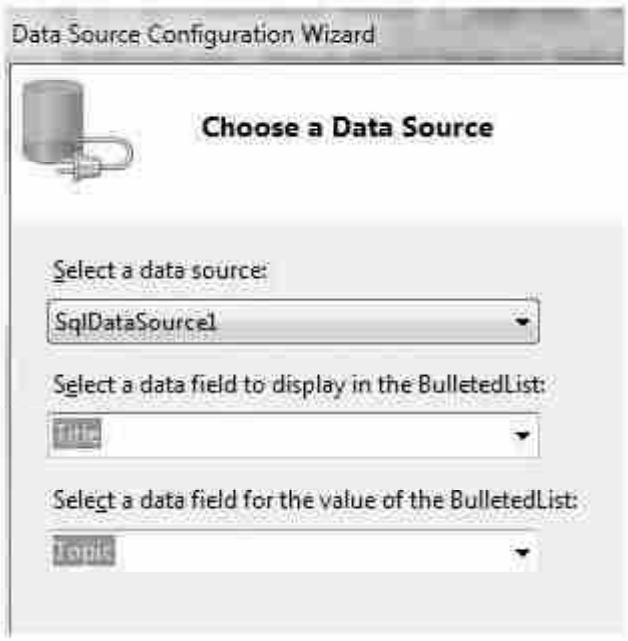
- DetailsView
- FormView
- GridView
- RecordList

Simple Data Binding : A read-only selection list is required for the simpler form of data binding. These controls have the ability to bind to a field or an array list from a database. The database or other data source provides selection lists with two values; one of these values is the value that is displayed by the list, and the other value is the value that is regarded to be the value matching to the display.

To further grasp the notion, let us examine a more concrete illustration. Make a website with a bulleted list that also contains a SqlDataSource control and upload it.

The following steps are required when selecting a data source for the bulleted list control :

- Selecting the control over the data source
- Choosing a field to display, also known as the data field, is the first step.
- Choosing a field for the value to be entered



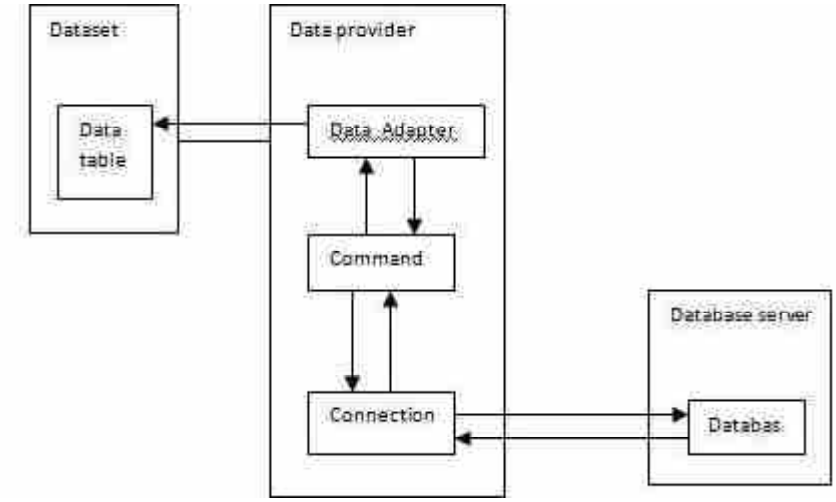
When the programme is being run, make sure that the complete title column is displayed and tied to the bulleted list.

Declarative Data Binding : When we were working with the GridView control in the previous tutorial, we actually made use of declarative data binding. The DetailsView, FormView, and RecordList controls are the others that belong to the category of composite data-bound controls and are capable of displaying and manipulating data in a tabular format.

In the following lesson, we will investigate the method that is used to manage databases, which is known as ADO.NET. The data binding, on the other hand, includes the following objects :

- A set of data that has been retrieved from a database and stored in this set.
- The data provider is responsible for obtaining data from the database through the execution of a command carried out over a connection.
- The data adapter is responsible for issuing the select statement that is stored in the command object. In addition, it is able to update the data in a database by issuing statements to insert, delete, and update data.

Relationship between the things that are data bindings :



Example :

Let us proceed with the steps listed below :

- Create a brand-new website as the first step. Create a new class and call it booklist by right-clicking on the name of the solution in the Solution Explorer, then selecting "Class" from the drop-down menu in the "Add Item" dialogue box that appears. It should be called booklist.cs.

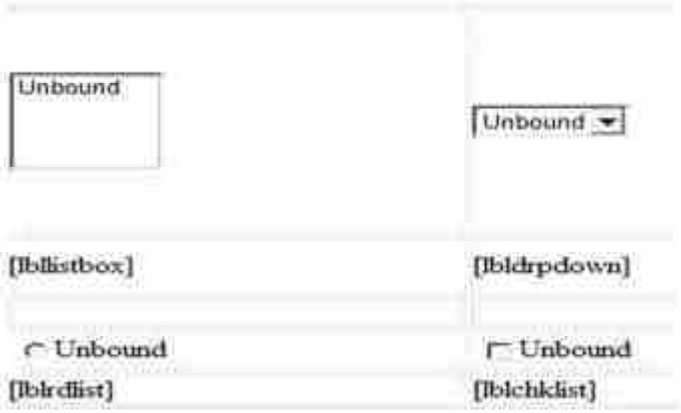
Booklist.cs Source Code :

```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

namespace databinding
{
    public class booklist
    {
        protected String bookname;
        protected String authorname;
        public booklist(String bname, String aname)
        {
            this.bookname = bname;
            this.authorname = aname;
```

```
        }
        public String Book
        {
            get
            {
                return this.bookname;
            }
            set
            {
                this.bookname = value;
            }
        }
        public String Author
        {
            get
            {
                return this.authorname;
            }
            set
            {
                this.authorname = value;
            }
        }
    }
}
```

- **Step two :** add four list controls to the page, including a list box control, a radio button list, a check box list, and a drop down list. Along with these list controls, add four labels to the page as well. When viewed in design mode, the page should seem as follows :



HTML Source Code :

```
<form id="form1" runat="server">
   <div>

      <table style="width: 559px">
         <tr>
            <td style="width: 228px; height: 157px;">
               <asp:ListBox ID="ListBox1" runat="server" AutoPost
               Back="True"
                  OnSelectedIndexChanged="ListBox1_SelectedIndex
                  Changed">
               </asp:ListBox>
            </td>

            <td style="height: 157px">
<asp:DropDownList ID="DropDownList1" runat="server" AutoPost
Back="True"
OnSelectedIndexChanged = "DropDownList1_SelectedIndexChanged">
               </asp:DropDownList>
            </td>
         </tr>
         <tr>
            <td style="width: 228px; height: 40px;">
               <asp:Label ID="lbllistbox" runat="server"></asp:Label>
            </td>
            <td style="height: 40px">
               <asp:Label ID="lbldrpdown" runat="server">
               </asp:Label>
            </td>
         </tr>
         <tr>
            <td style="width: 228px; height: 21px">
            </td>

            <td style="height: 21px">
            </td>
         </tr>
         <tr>
            <td style="width: 228px; height: 21px">
```

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server" Auto PostBack="True"
OnSelectedIndexChanged = "RadioButtonList1_SelectedIndexChanged">
</asp:RadioButtonList>
            </td>
            <td style="height: 21px">

<asp:CheckBoxList ID="CheckBoxList1" runat="server" AutoPostBack= "True"
OnSelectedIndexChanged="CheckBoxList1_SelectedIndexChanged">

                </asp:CheckBoxList>
            </td>
        </tr>
        <tr>

<td style="width: 228px; height: 21px"><asp:Label ID="lblrdlist" runat= "server">

                </asp:Label>
            </td>
            <td style="height: 21px">
                <asp:Label ID="lblchklist" runat="server"></asp:Label>
            </td>
        </tr>
    </table>
</div>
</form>
```

- **Step Three :** Lastly, you will need to write the following code behind the application's routines :

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        IList bklist = createbooklist();
        if (!this.IsPostBack)
        {
            this.ListBox1.DataSource = bklist;
            this.ListBox1.DataTextField = "Book";
            this.ListBox1.DataValueField = "Author";
```

```
        this.DropDownList1.DataSource = bklist;
        this.DropDownList1.DataTextField = "Book";
        this.DropDownList1.DataValueField = "Author";
        this.RadioButtonList1.DataSource = bklist;
        this.RadioButtonList1.DataTextField = "Book";
        this.RadioButtonList1.DataValueField = "Author";
        this.CheckBoxList1.DataSource = bklist;
        this.CheckBoxList1.DataTextField = "Book";
        this.CheckBoxList1.DataValueField = "Author";
        this.DataBind();
    }
}
protected IList createbooklist()
{
    ArrayList allbooks = new ArrayList();
    booklist bl;
    bl = new booklist("UNIX CONCEPTS", "SUMITABHA DAS");
    allbooks.Add(bl);
    bl = new booklist("PROGRAMMING IN C", "RICHI
    KERNIGHAN");
    allbooks.Add(bl);
    bl = new booklist("DATA STRUCTURE", "TANENBAUM");
    allbooks.Add(bl);
    bl = new booklist("NETWORKING CONCEPTS", "FOROUZAN");
    allbooks.Add(bl);
    bl = new booklist("PROGRAMMING IN C++", "B.
    STROUSTROUP");
    allbooks.Add(bl);
    bl = new booklist("ADVANCED JAVA", "SUMITABHA DAS");
    allbooks.Add(bl);
    return allbooks;
}
protected void ListBox1_SelectedIndexChanged(object sender,
EventArgs e)
{
    this.lbllistbox.Text = this.ListBox1.SelectedValue;
}
protected void DropDownList1_SelectedIndexChanged(object sender,
EventArgs e)
{
    this.lbldrpdown.Text = this.DropDownList1.SelectedValue;
```

```
    }
    protected void RadioButtonList1_SelectedIndexChanged(object sender,
    EventArgs e)
    {
        this.lblrdlist.Text = this.RadioButtonList1.SelectedValue;
    }
    protected void CheckBoxList1_SelectedIndexChanged(object sender,
    EventArgs e)
    {
        this.lblchklist.Text = this.CheckBoxList1.SelectedValue;
    }
}
```

❑ **Check Your Progress – 1 :**

1. The object which the data binding does not include is _____.

   a. Data Compiler b. DataSet

   c. DataAdapter d. DataProvider

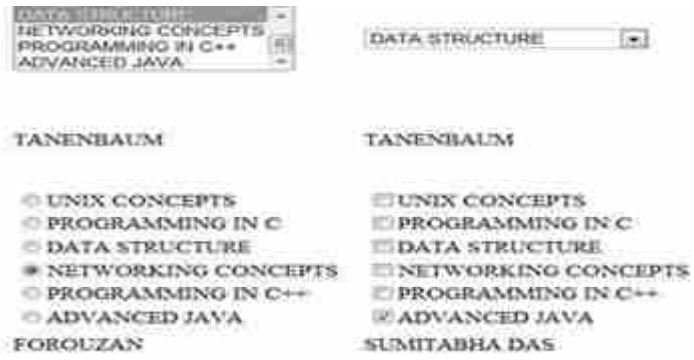
2. Which type of selection list is required for the simple data binding?

   a. Write only b. Read Write only

   c. Read Only d. None of these

Take into consideration the following :

- The bookname and authorname properties are the ones that make up the booklist class's set of attributes.
- This user-defined method is called createbooklist, and it is responsible for populating an array of booklist objects with the name allbooks.
- The Page Load event handler is responsible for generating a books list after the page has loaded. IList is a type that implements the IEnumerable interface and is capable of being attached to list controls. This particular list is of the type IList. The IList object known as 'bklist' is bound to the list controls when the page load event handler is called. The authorname property is taken into consideration to be the value, and the bookname property is to be shown as it is.
- If the user chooses a book before the page is launched, the name of the book will be selected and displayed by the list controls. On the other hand, the corresponding labels will display the author name, which is the value that corresponds to the selected index of the list control.



8.3 Benefits of Populating the List Dynamically Using SqlDataSource Selectedindexchanged :

SelectedIndexChanged is an event that is triggered whenever the Choose button on a row is pressed. However, this event does not take place until the GridView control has finished the select operation that it was performing when it was initiated. You will be given the opportunity to select an event-handling function that, once this event takes place, will execute a custom routine that you have created. You will have the ability to do so whenever it takes place thanks to this capability. You could, for example, change a status label so that it shows the row that is now selected. This would serve as an illustration.

The following snippet of code illustrates how to make use of the SelectedIndexChanged event that is made available by the GridView control. This event provides the opportunity to display the name of the consumer who is connected to the row that has been chosen for inspection.

```
<%@ Page language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
void CustomersGridView_SelectedIndexChanged(Object sender,
EventArgs e)
{
/*Get the currently selected row using the SelectedRow property.
GridViewRow row = CustomersGridView.SelectedRow; Display the first
name from the selected row. In this example, the third column (index
2) contains the first name.*/
MessageLabel.Text = "You selected " + row.Cells[2].Text + ".";
}
void CustomersGridView_SelectedIndexChanging(Object sender,
GridViewSelectEventArgs e)
{
/*Get the currently selected row. Because the SelectedIndexChanging
event occurs before the select operation in the GridView control, the
SelectedRow property cannot be used. Instead, use the Rows collection
and the NewSelectedIndex property of the e argument passed to this event
handler.*/
GridViewRow row = CustomersGridView.Rows[e.NewSelectedIndex];
/*You can cancel the select operation by using the Cancel property. For
this example, if the user selects a customer with the ID "ANATR", the
select operation is canceled and an error message is displayed*/
if (row.Cells[1].Text == "ANATR")
{
e.Cancel = true;
MessageLabel.Text = "You cannot select " + row.Cells[2].Text + ".";
}
```

```
}
</script>
<html xmlns="http://www.w3.org/1999/xhtml" > <head runat="server">
<title>GridView Select Example</title>
</head>
<body>
<form id="form1" runat="server">
<h3>GridView Select Example</h3>
<asp:gridview id="CustomersGridView" datasourceid="CustomersSource"
autogeneratecolumns="False" autogenerateselectbutton="True" selected index="1"
onselectedindexchanged="CustomersGridView_SelectedIndexChanged" onselectedindexchanging="CustomersGridView_SelectedIndexChanging" runat="server" DataKeyNames="CustomerID">
<Columns>
<asp:BoundField DataField="CustomerID" HeaderText="CustomerID" InsertVisible="False" ReadOnly="True" SortExpression="CustomerID" />
<asp:BoundField DataField="FirstName" HeaderText="FirstName" SortExpression = " FirstName" />
<asp:BoundField DataField="MiddleName" HeaderText="MiddleName" SortExpression = " MiddleName" />
<asp:BoundField DataField="LastName" HeaderText="LastName" SortExpression = " LastName" />
<asp:BoundField DataField="Phone" HeaderText="Phone" SortExpression="Phone" />
</Columns>
<selectedrowstyle backcolor="LightCyan" forecolor="DarkBlue" font-bold="true"/>
</asp:gridview>
<br/>
<asp:label id="MessageLabel" forecolor="Red" runat="server"/>
<!-- This example uses Microsoft SQL Server and connects  -->
<!-- to the sample database. Use an ASP.NET
<!-- expression to retrieve the connection string value  -->
<!-- from the Web.config file.
<asp:sqldatasource id="CustomersSource" Selectcommand="SELECT CustomerID, FirstName, MiddleName, LastName, Phone FROM SalesLT.Customer" connectionstring="<%$ConnectionStrings:Adventure WorksLTConnectionString %>" runat = "server"/>
</form>
</body>
</html>
```

8.4 Filtering Using Dropdownlist and Checkboxlists :

You have the ability to design a specialized column class that gives users a drop-down menu in place of the filter text box. This gives consumers with an additional option for entering filter criteria into the search. When the user picks an item from the drop-down menu, the grid can be programmed to filter the records automatically if the configuration is correct. The process of achieving this result can be broken down into the following steps:

- Construct a specialised column class that is an extension of the standard GridBoundColumn: When working with a custom column class, it is necessary to alter the SetupFilterControls method so that a DropDownList control can be used in place of the filter text box and image button. Adjust the settings of the DropDownList control to suit your needs. At a minimum, make sure that the AutoPostBack attribute is set to the True value. You should give the ID attribute a value that is only used once. Make the items list bindable. Include a handler for the SelectedIndexChanged event.
- To set or retrieve the value that is currently selected from the drop-down menu, you must override the methods that are named Get Current Filter Value From Control and Set Current Filter Value to Control.
- Get the GridFilteringItem of the grid, and in the SelectedIndexChanged event handler, call the GridFilteringItem's FireCommandEvent method to begin the process of applying a filter command.

Checkboxlists : A control that is comprised of a single control that groups together a collection of checkbox controls is referred to as a checkbox list control. Elements of checkable lists, each of which is represented on the screen as a separate "input type=checkbox" and "/input" element. Its style properties, such as BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, and Height, among others, are implemented using the style properties of the input> element. These attributes include: BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, and Height.

Properties of Checkboxlists and Dropdownlists

Properties	Description
SelectedValue	Gets the value of the first item that was selected.
SelectedIndex	This function either gets or sets the index of the first item that has been selected.
SelectedItem	Gets the first item that was picked.
TextAlign	Gets or sets the alignment of the text that is contained in the checkbox.
DataTextField	The name of the field in the data source that will be used to supply the item texts. (There is no need to set anything if you are directly adding items into .aspx page.)
DataValueField	The name of the field in the data source that will deliver the value for each item. (There is no need to set anything if you are directly adding items into .aspx page.)

DataSourceID	The identifier of the datasource component that will be used to supply data. (Only used when any DataSource component, such as SqlDataSource or AccessDataSource, is present on the page.)
DataSource	The datasource that is responsible for populating the checkboxlist box with items. (This is typically done when you are dynamically producing the items from the Database.)
AutoPostBack	true/false. When this condition is met, the form is automatically re-posted to the server whenever the user selects one of the checkboxes. Additionally, the OnSelectedIndexChanged function will be triggered.
AppendData BoundItems	true/false. If the condition is met, the item that was added statically (added from an.aspx page) is kept when things are added dynamically (from a code-behind file), otherwise the items are cleared.
OnSelected IndexChanged	The name of the method that is called whenever the user clicks on any of the checkboxes in the list. (Only activated when the AutoPostBack parameter is set to true.)
Items	Retrieves the collection of objects that are contained in the list.
RepeatLayout	table/flow. This function either retrieves or sets the layout of the checkboxes that are displayed on the page.
RepeatColumns	This property either retrieves or sets the number of columns that will be displayed when the control is produced.
RepeatDirection	Horizontal/Vertical. Gets or sets the value that indicates whether the control will be shown horizontally or vertically. This value may be gotten or set using either Get or Set.

When it is tied to a data source, the CheckBoxList control in ASP.NET 2.0 provides a collection of checkboxes that may be generated in a manner that is both dynamic and customisable. This is possible because of the control's ability to generate the checkboxes in response to the data source. This collection also has the ability to be personalised. In this post, we are going to investigate how to use a CheckboxList, as well as how to use ASP.NET and Javascript to pick or deselect all of the checkboxes that are contained within a CheckboxList.

❑ **Check Your Progress – 2 :**

1. The name of the field in the data source that will be used to supply the item texts is _____.

   a. DataValueField b. DataTextField

   c. Items d. None of these

2. Gets the value of the first item that was selected is _____.

   a. SelectedIndex b. SelectedValue

   c. SelectedItem d. All of these

8.5 Selecteditem :

Retrieves or sets the value of the item that is currently chosen.

Example :

The following piece of code serves as an illustration of how to make advantage of the SelectedIndexChanged event in order to locate and select an item from within a different ListBox control. This demonstration will show you how to know whether the selected item in a ListBox has been altered by utilising the SelectedIndexChanged event. The code then uses the SelectedItem property to read the item's text, and then it uses that text to execute the FindString function on a new ListBox, passing in the text that was returned by the SelectedItem property of the first ListBox. If the same item can also be found in the other ListBox, it will be chosen instead. In order to follow along with this example, a form will need to have two ListBox controls-named listbox1 and listbox2-added to it, and both of those ListBox controls will need to be populated with items that are the same. A connection must also be made between the event-handling method defined in the example and the SelectedIndexChanged event of listBox1 in order for the example to be valid.

C# Source Code :

```
private void listBox1_SelectedIndexChanged(object sender, System.
EventArgs e)
{
    // Get the currently selected item in the ListBox.
    string curItem = listBox1.SelectedItem.ToString();
    // Find the string in ListBox2.
    int index = listBox2.FindString(curItem);
    /*If the item was not found in ListBox 2 display a message box,
otherwise select it in ListBox2.*/
    if(index == -1)
        MessageBox.Show("Item is not available in ListBox2");
    else
        listBox2.SetSelected(index,true);
}
```

You can detect which item is selected in a normal ListBox by using this property to find out which item is selected in the ListBox. In the event that the SelectionMode property of the ListBox is set to either SelectionMode, the following behaviour will take place: MultiSimple or SelectionMode. If the ListBox is marked as MultiExtended, which denotes that it supports multiple selections, and if more than one item is picked from the list, then this property can return any item that has been chosen.

Utilizing the SelectedItems property will allow you to access a collection that contains all of the items that have been selected in a ListBox that has multiple selections. Utilizing the SelectedIndex property will allow you to acquire the index position of the item that is now chosen in the ListBox that you are working with. In addition, if you have a multiple-selection ListBox, you may utilise the SelectedIndices property to get a list of all the indexes that have been picked.

8.6 Selectedvalue :

This function either picks the item in the list control that contains the supplied value or gets the value of the item that is currently chosen in the list control. The value of the selected item in the list control. The default is an empty string ("").

Example :

The following example illustrates how to utilise the SelectedValue property of a ListBox control to select an item from within the control's list of options. It should be noted that the value of the selected item may also be retrieved with the help of this attribute.

Note : This particular example includes a text box that takes input from users, which presents a possible security risk. ASP.NET Web sites will, by default, perform validation to ensure that user input does not contain any HTML or script components.

Default.aspx Source Code :

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title> ListControl SelectedValue Example </title>
<script runat="server">
   void Button_Click(Object sender, EventArgs e)
   {
      // Perform this operation in a try-catch block in case the item is
      not found.
      try
      {
         List.SelectedValue = ItemTextBox.Text;
         MessageLabel.Text = "You selected " + List.SelectedValue + ".";
      }
      catch (Exception ex)
      {
         List.SelectedValue = null;
         MessageLabel.Text = "Item not found in ListBox control.";
      }
   }
   </script>
</head>
<body>
```

```
<form id="form1" runat="server">
    <h3> ListControl SelectedValue Example </h3>
    <asp:ListBox ID="List" runat="server">
        <asp:ListItem>Item 1</asp:ListItem>
        <asp:ListItem>Item 2</asp:ListItem>
        <asp:ListItem>Item 3</asp:ListItem>
        <asp:ListItem>Item 4</asp:ListItem>
    </asp:ListBox>
    <hr />
    Enter the value of the item to select: <br />
    <asp:TextBox ID="ItemTextBox" MaxLength="6" Text="Item 1"
    runat="server"/>
    <asp:Button ID="SelectButton" Text="Select Item" OnClick=
    "Button_Click" runat="server"/>
    <br /><br />
    <asp:Label ID="MessageLabel" runat="server"/>
</form>
</body>
</html>
```

This property is responsible for returning the Value property of the currently chosen ListItem. In order to identify the value of the item that is now chosen in the list control, the SelectedValue attribute is typically consulted. If there are numerous things that are selected, the value that is returned is that of the selected item that has the lowest index. In the event that no item is chosen, an empty string ("") will be returned.

Setting the SelectedValue property with the value of the item in the list control is another way to utilise the SelectedValue property to choose an item in the list. An ArgumentOutOfRangeException exception is generated whenever a postback operation is carried out after one in which the selected value is not present in the list of available values. The following illustration demonstrates how to detect an invalid value prior to doing a postback:

C# Source Code :

```
this.DropDownList1.Items.Add(new ListItem{ Text="Hello", Value="1"
});
if(DropDownList1.Items.FindByValue("2") != null) {
    Response.Write("Found");
}
```

8.7 Radiobuttonlist :

The RadioButtonList Control performs the same functions as the DropDownList Control, but instead of displaying a drop-down menu, it provides a list of radio buttons that can be arranged in a horizontal or vertical fashion. This control can also do the same functions as the DropDownList Control. This control, along with the DropDownList Control, is responsible for performing

the same functions. You will only be able to choose one of the available alternatives at a time from the RadioButtonList that has been made accessible to you because there is only room for one selection at a time. These two possibilities simply cannot coexist since they are incompatible with one another.

Important Properties of RadioButtonList Control

Properties	Description
RepeatLayout	Determines whether or not the radio buttons display in a table formatted in HTML.
RepeatColumns	It shows the amount of columns of radio buttons that are available.
RepeatDirection	The course that is repeated by each of the radio buttons. The value of RepeatDirection is set to vertical by default. Possible values are Horizontal and Vertical.

The following are some of the possible values :

- Table
- Flow
- OrderedList
- UnorderedList

Example :

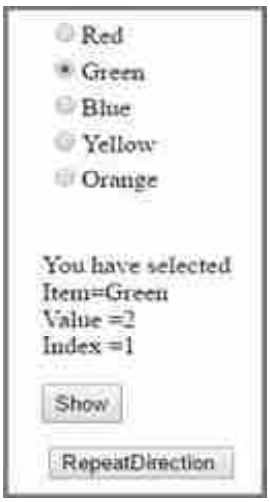
C# Source Code :

```
using System;
using System.Web.UI.WebControls;
public partial class ListControls : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = "You have selected </br> Item=" +
RadioButtonList1.SelectedItem.Text + "</br> Value =" +
RadioButtonList1.SelectedValue + "</br> Index =" +
RadioButtonList1.SelectedIndex ;
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        if (RadioButtonList1.RepeatDirection == RepeatDirection.Vertical)
        {
            RadioButtonList1.RepeatDirection = RepeatDirection.Horizontal;
        }
        else
```

```
        {
            RadioButtonList1.RepeatDirection = RepeatDirection.Vertical;
        }
    }
}
```

It is possible to add items to the RadioButtonList by using the item collection feature of the property window.

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server">
    <asp:ListItem Value="1">Red</asp:ListItem>
    <asp:ListItem Value="2">Green</asp:ListItem>
    <asp:ListItem Value="3">Blue</asp:ListItem>
    <asp:ListItem Value="4">Yellow</asp:ListItem>
    <asp:ListItem Value="5">Orange</asp:ListItem>
</asp:RadioButtonList>
```



When you select Horizontal as the repeat direction using the RepeatDirection button, the layout will be adjusted accordingly.

```
RadioButtonList1.RepeatDirection = RepeatDirection.Horizontal;
```

By default RepeatColumns value is zero. You are free to change this value based on the requirements of the application. Changing this value to three will result in the following changes to the output :

8.8 Repeatdirection :

This property allows you to get or set a value that determines whether the control shows in a vertical or horizontal orientation. One of the possible values for the RepeatDirection property. The Vertical orientation is the default.

Example :

The following snippet of code serves as an illustration of how to use the RepeatDirection property to designate that the display will be done in a vertical orientation.

Note : Because the following code samples are based on the single-file code model, it is possible that they will not function properly if they are copied and pasted directly into a code-behind file. Every code snippet needs to be pasted into a blank text file that has the .aspx extension.

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
   <title>CheckBoxList Example</title>
<script language="C#" runat="server">
   void Check_Clicked(Object sender, EventArgs e)
   {
      Message.Text = "Selected Item(s):<br /><br />";
      for (int i = 0; i < CheckBoxList1.Items.Count; i++)
      {
         if (CheckBoxList1.Items[i].Selected)
         Message.Text += CheckBoxList1.Items[i].Text + "<br />";
      }
   }
</script>
</head>
<body>
   <form id="form1" action="CheckBoxList.aspx" method="post" runat="server">
      <h3>CheckBoxList Example</h3>
      <asp:CheckBoxList id="CheckBoxList1" AutoPostBack="True" CellPadding="5"
      CellSpacing="5" RepeatColumns="2" RepeatDirection="Vertical" RepeatLayout="Flow"
      TextAlign="Right" OnSelectedIndexChanged="Check_Clicked" runat="server">
         <asp:ListItem>Item 1</asp:ListItem>
         <asp:ListItem>Item 2</asp:ListItem>
         <asp:ListItem>Item 3</asp:ListItem>
         <asp:ListItem>Item 4</asp:ListItem>
         <asp:ListItem>Item 5</asp:ListItem>
         <asp:ListItem>Item 6</asp:ListItem>
      </asp:CheckBoxList>
      <br /><br />
```

```
        <asp:label id="Message" runat="server"/>
    </form>
</body>
</html>
```

Example :

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
        <title> CheckBoxList RepeatDirection Example </title>
<script runat="server">
    void Check_Clicked(Object sender, EventArgs e)
        {
      Message.Text = "Selected Item(s):<br /><br />";
      // Iterate through the Items collection of the CheckBoxList
            // control and display the selected items.
              for (int i=0; i<checkboxlist1.Items.Count; i++)
              {
        if (checkboxlist1.Items[i].Selected)
              {
            Message.Text += checkboxlist1.Items[i].Text + "<br />";
        }
}     }
void Index_Change(Object sender, EventArgs e)
        {
      // Set the direction that the items are rendered in the
            // CheckBoxList control.
        checkboxlist1.RepeatDirection = (RepeatDirection)List.Selected
        Index;
        }
      </script>
</head>
<body>
    <form id="form1" runat="server">
      <h3> CheckBoxList RepeatDirection Example </h3>
      Select items from the CheckBoxList.
      <br /><br />
```

```
<asp:CheckBoxList id="checkboxlist1" AutoPostBack="True" Cell
Padding="5"
CellSpacing="5"  RepeatColumns="2" RepeatDirection="Vertical" Repeat
Layout = " Table" TextAlign="Right" OnSelectedIndexChanged="Check_
Clicked"                runat="server">
    <asp:ListItem>Item  1</asp:ListItem>
    <asp:ListItem>Item  2</asp:ListItem>
    <asp:ListItem>Item  3</asp:ListItem>
    <asp:ListItem>Item  4</asp:ListItem>
    <asp:ListItem>Item  5</asp:ListItem>
    <asp:ListItem>Item  6</asp:ListItem>
</asp:CheckBoxList>
<br /><br />
<asp:label id="Message" runat="server"/>
<hr />
Select the direction to render the CheckBoxList items.
<table cellpadding="5">
    <tr>
        <td>
            RepeatDirection:
        </td>
    </tr>
    <tr>
<td><asp:DropDownList id="List" AutoPostBack="True" OnSelected
IndexChanged = " Index_Change" runat="server">
                <asp:ListItem>Horizontal</asp:ListItem>
                <asp:ListItem Selected="True">Vertical</asp:ListItem>
            </asp:DropDownList>
        </td>
    </tr>
</table>
</form>
</body>
</html>
```

8.9 Checkboxfield :

When the user is restricted to selecting no more than one item at a time from the grid, a user interface consisting of a column of radio buttons is an appropriate choice. However, there can be situations in which we wish to give the user the option of selecting any number of items from the grid. Email clients that are web-based, for instance, will often present the list of mails along with a column containing checkboxes. The user has the ability to pick any number

of messages before carrying out a specific action, such as relocating the selected emails to a different folder or erasing them entirely.

Here we will learn how to add a column of checkboxes, as well as how to discover which checkboxes were selected once a postback has been completed. In specifically, we will construct an example that imitates the user interface of a web-based email client as nearly as possible. In this demonstration, we will use a paged GridView that contains a checkbox in each row to select individual products from the Products database table. When the button labelled "Delete Selected Products" is clicked, the products that have been selected are deleted.

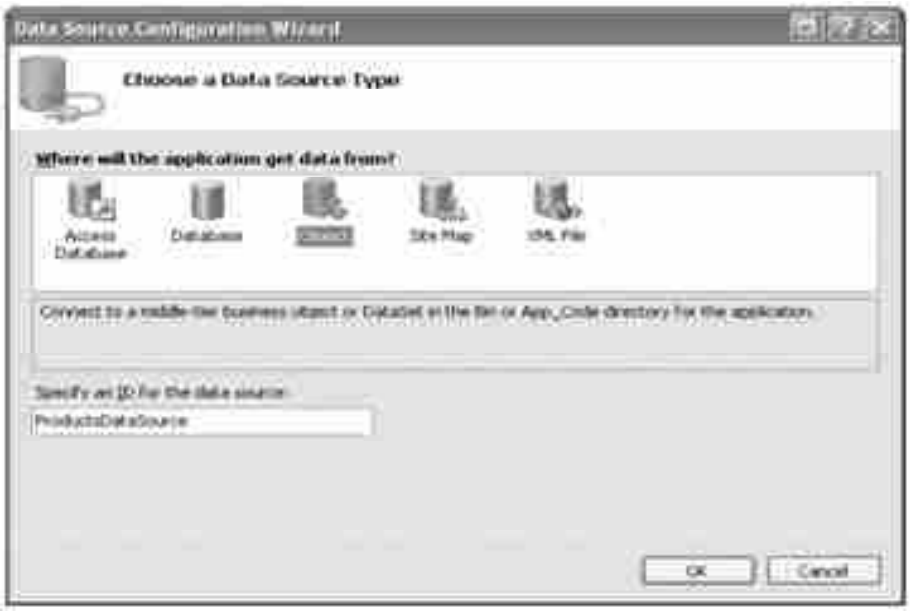
❑ **Check Your Progress – 3 :**

1. _____ property allows you to get or set a value that determines whether the control shows in a vertical or horizontal orientation.

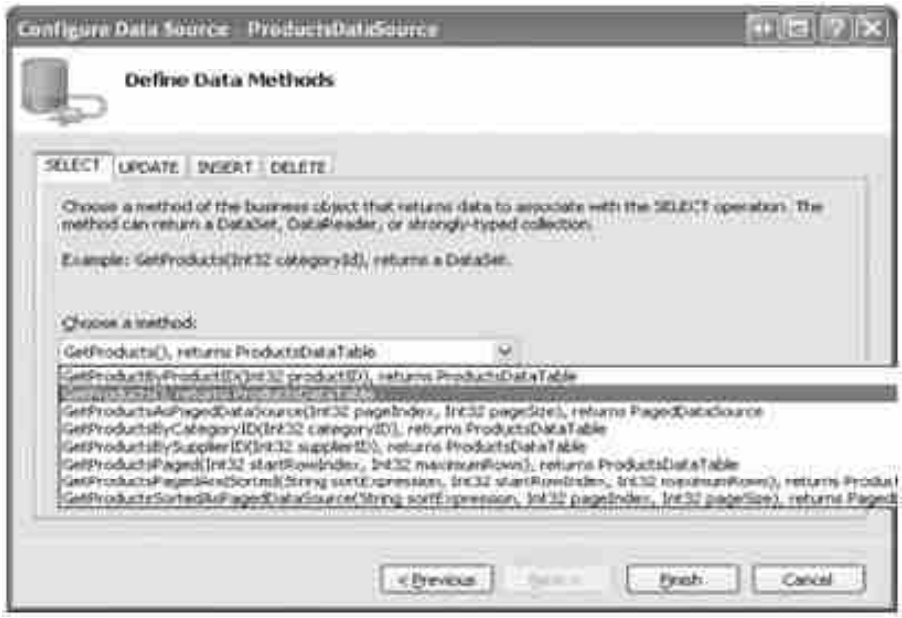
   a. CheckboxField b. RadiobuttonList

   c. Repeatdirection d. None of these

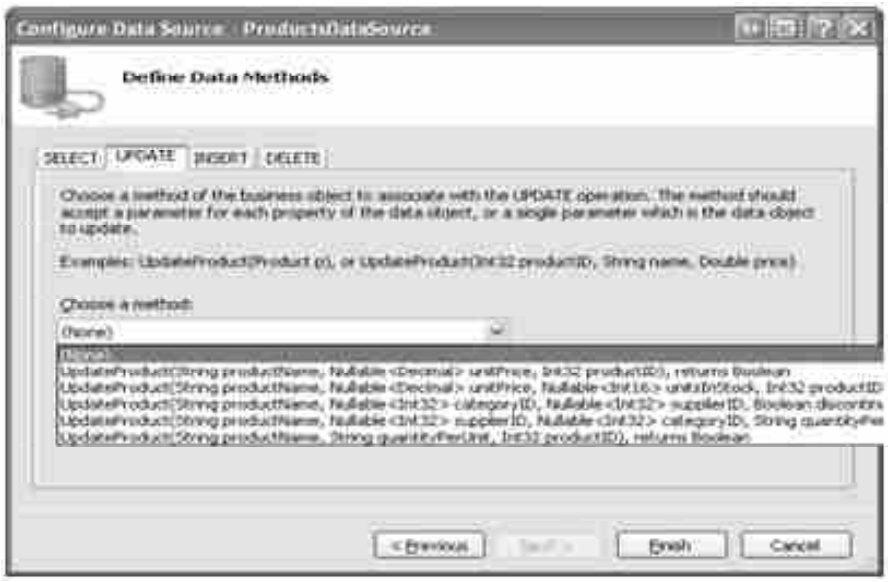
- **Including a GridView with Scrollable Pages that Lists Product Information :** Before we bother about adding a column of checkboxes, let's first concentrate on showing the products in a GridView that allows paging so that we can navigate through them more easily. To begin, navigate to the CheckBoxField.aspx file located within the EnhancedGridView folder. Next, drag a GridView from the Toolbox onto the Designer and change its ID to Products. Next, you'll want to select the option to tie the GridView to a brand-new ObjectDataSource that you've given the name ProductsDataSource. Get the data by calling the GetProducts() function when the ObjectDataSource has been configured to utilise the ProductsBLL class as its data source. Because only readers will be able to interact with this GridView, make sure that the drop-down lists in the UPDATE, INSERT, and DELETE tabs are set to (None).



Make sure to give the new object data source the name ProductsDataSource.



Adjust the settings of the ObjectDataSource so that it retrieves data using the GetProducts() method.



In the UPDATE, INSERT, and DELETE Tabs, make sure the Drop-Down Lists are set to the appropriate values (None).

Visual Studio will automatically generate BoundColumns and a CheckBoxColumn for the product-related data fields once the Configure Data Source wizard has been finished. Following the steps we took in the prior lesson, clear out all of the BoundFields save for ProductName, CategoryName, and UnitPrice, and then set the Product, Category, and Price properties for the HeaderText properties. Adjust the settings of the UnitPrice BoundField so that its value is displayed in the appropriate currency format. Additionally, ensure that the GridView is able to enable paging by selecting the checkbox located on the smart tag that bears the name "Enable Paging."

Let's not forget to include the user interface for removing the products that have been chosen. Add a Button Web control below the GridView, and give it the ID of DeleteSelectedProducts. The Text attribute of the Button Web control should read "Delete Selected Products." For the purpose of this demonstration, rather than actually removing products from the database, we will instead merely display a message that lists the products that were scheduled to be removed. In order to make room for this, a Label Web control should be added underneath the Button. Change its ID to DeleteResults, delete everything from its Text property, and make sure that both its Visible and EnableViewState properties are set to false.

Following the implementation of these modifications, the declarative markup for the GridView, ObjectDataSource, Button, and Labels should look something like this:

HTML Source Code :

```
<p>
<asp:GridView ID="Products" runat="server" AutoGenerateColumns=
"False"
DataKeyNames="ProductID" DataSourceID="ProductsDataSource"
AllowPaging="True" EnableViewState="False">
<Columns>
<asp:BoundField DataField="ProductName" HeaderText="Product"
   SortExpression="ProductName" />
<asp:BoundField DataField="CategoryName" HeaderText="Category"
   ReadOnly="True" SortExpression="CategoryName" />
<asp:BoundField DataField="UnitPrice" DataFormatString="{0:c}"
   HeaderText="Price" HtmlEncode="False"
   SortExpression="UnitPrice" />
</Columns>
</asp:GridView>
<asp:ObjectDataSource ID="ProductsDataSource" runat="server"
OldValuesParameterFormatString="original_{0}"
SelectMethod="GetProducts" TypeName="ProductsBLL">
```

```
</asp:ObjectDataSource>
</p>
<p>
<asp:Button ID="DeleteSelectedProducts" runat="server" Text="Delete Selected Products" />
</p>
<p>
<asp:Label ID="DeleteResults" runat="server" EnableViewState="False" Visible="False"></asp:Label>
</p>
```

At this point, you should be able to view the names of the first ten products, along with their categories and prices.

Product	Category	Price
Chai Tea	Beverages	$19.96
Chang	Beverages	$19.00
Aniseed Syrup	Condiments	$10.00
Chef Anton's Cajun Seasoning	Condiments	$26.62
Chef Anton's Gumbo Mix	Condiments	$21.35
Grandma's Boysenberry Spread	Condiments	$30.25
Uncle Bob's Organic Dried Pears	Produce	$30.00
Northwoods Cranberry Sauce	Condiments	$36.00
Mishi Kobe Niku	Meat/Poultry	$97.00
Ikura	Seafood	$31.00

1 2 3 4 5 ... >>

Delete Selected Products

- **Adding a Column of Checkboxes :** One may reason that because ASP.NET 2.0 comes with a CheckBoxField, it could be used to add a column of checkboxes to a GridView. However, this is not the case. Sadly, that is not the case, as the CheckBoxField is meant to be used in conjunction with a Boolean data field. In other words, in order to make use of the CheckBoxField, we need to define the underlying data field, the value of which is consulted in order to decide whether or not the rendered checkbox is checked. It is not possible for us to use the CheckBoxField to simply include a column of unchecked checkboxes in the form.

Instead, we need to add a TemplateField, and then add a CheckBox Web control to the ItemTemplate that we just created. You should go ahead and add a TemplateField to the Products GridView, and you should position it as the very first field on the far left. After clicking on the Edit Templates link from the smart tag of the GridView, dragged and dropped a CheckBox Web control from the Toolbox into the ItemTemplate to finish the process. Change the ID property of this CheckBox to read ProductSelector. Add a CheckBox Web Control Named ProductSelector to the TemplateFields ItemTemplate.

A checkbox has been added to each row as a result of the addition of the TemplateField and CheckBox Web control. Following the addition of the TemplateField and CheckBox, the final product will have this appearance.

	Product	Category	Price
☐	Chai Tea	Beverages	$19.96
☐	Chang	Beverages	$19.00
☐	Aniseed Syrup	Condiments	$10.00
☐	Chef Anton's Cajun Seasoning	Condiments	$26.62
☐	Chef Anton's Gumbo Mix	Condiments	$21.35
☐	Grandma's Boysenberry Spread	Condiments	$30.25
☐	Uncle Bob's Organic Dried Pears	Produce	$30.00
☐	Northwoods Cranberry Sauce	Condiments	$36.00
☐	Mishi Kobe Niku	Meat/Poultry	$97.00
☐	Ikura	Seafood	$31.00

1 2 3 4 5 ... >>

Delete Selected Products

- **Determining What Checkboxes Were Checked On Postback :** We currently have a column of checkboxes, but there is no way to determine which checkboxes were selected when the postback was performed. However, in order to delete the products that have been selected, we need to know which checkboxes have been selected after the Delete Selected Products button has been hit.

The Rows property of the GridView gives users access to the data rows contained within the GridView. We may access the CheckBox control programmatically, cycle through these rows, and then consult the Checked property of the CheckBox control to see whether or not the CheckBox has been selected.

Make sure the DeleteSelectedProducts Button on your website has an event handler attached to it. Click the event, then add the code in the following location :

C# Source Code :

```
protected void DeleteSelectedProducts_Click(object sender, EventArgs e)
{
    bool atLeastOneRowDeleted = false;
    // Iterate through the Products.Rows property
    foreach (GridViewRow row in Products.Rows)
    {
            // Access the CheckBox
            CheckBox cb = (CheckBox)row.FindControl("Product
            Selector");
            if (cb != null && cb.Checked)
            {
            // Delete row! (Well, not really...)
            atLeastOneRowDeleted = true;
            // First, get the ProductID for the selected row
            int productID = Convert.ToInt32 (Products.DataKeys [row .
            RowIndex].Value);
            // "Delete" the row
            DeleteResults.Text += string.Format(
            "This would have deleted ProductID {0}<br />", productID);
    }
    }
    // Show the Label if at least one row was deleted...
    DeleteResults.Visible = atLeastOneRowDeleted;
    }
```

The Rows property is responsible for returning a collection of GridViewRow instances, which are what the data rows of the GridView are composed of. This collection is iterated over by the foreach loop that we have here. Using the row.FindControl method, the CheckBox on each GridViewRow object may be accessed in a programatically driven manner ("controlID"). The value of the row's matching ProductID is received from the DataKeys collection if the CheckBox is selected to be checked. In this exercise, we simply show an informative message in the DeleteResults Label; however, in a real-world application, we would instead make a call to the ProductsBLL class's DeleteProduct(productID) method. This is because the DeleteProduct method is responsible for removing products from the database.

When this event handler was added, clicking the Delete Selected Products button now displays the ProductIDs of the products that have been selected. This was previously not the case.

When the Delete Selected Products button is selected, the ProductIDs of the products that have been chosen to be deleted are displayed.

- **Adding Check All and Uncheck All Buttons :** If a user wants to remove all of the products that are now displayed on the website, they must first choose all 10 of the checkboxes. The addition of a Check All button, which, when pressed, selects all of the checkboxes in the grid, is one of the ways in which we can help speed up this process. A button to uncheck all of the boxes would also be beneficial.

Include two Web Button controls on the page, and position them such that they are above the GridView. Change the ID of the first one to CheckAll and the Text property of that one to Check All. Then change the ID of the second one to UncheckAll and change the Text property of that one to Uncheck All.

Example :

```
<asp:Button ID="CheckAll" runat="server" Text="Check All" />
<asp:Button ID="UncheckAll" runat="server" Text="Uncheck All" />
```

The next step is to create a method in the code-behind class that will be called ToggleCheckState(checkState). This method, when it is called, will enumerate the Products GridView's Rows collection and set the Checked property of each CheckBox to the value of the checkState parameter that was passed in.

C# Source Code :

```
private void ToggleCheckState(bool checkState)
{
    // Iterate through the Products.Rows property
    foreach (GridViewRow row in Products.Rows)
    {
        // Access the CheckBox
        CheckBox cb = (CheckBox)row.FindControl("ProductSelector");
```

```
            if (cb != null)
                cb.Checked = checkState;
        }
    }
```

Next, you will need to develop event handlers for the Click action of the CheckAll and UncheckAll buttons. In the event handler for CheckAll, simply call ToggleCheckState with the value true. Similarly, in the handler for UncheckAll, call ToggleCheckState (false).

C# Source Code :

```
protected void CheckAll_Click(object sender, EventArgs e)
{
    ToggleCheckState(true);
}
protected void UncheckAll_Click(object sender, EventArgs e)
{
    ToggleCheckState(false);
}
```

A postback is generated whenever the Check All button in this code is clicked, and all of the checkboxes in the GridView are selected after it is done. Similarly, pressing the Uncheck All button will de-select all of the checkboxes. The following is what the screen will appear to look like once the Check All button has been selected :

Note : When showing a column of checkboxes, one method for choosing or deselecting all of the checkboxes at once is to use a checkbox that is located in the row that headers are displayed. In addition, the Check All / Uncheck All functionality as it is now implemented calls for a postback. However, the checkboxes can be checked or unchecked entirely through client-side script, which results in a more responsive and streamlined experience for the user. Checking All CheckBoxes in a GridView Using Client-Side Script and a Check All CheckBox is an article that explores the use of a header row checkbox for Check All and Uncheck All in greater detail. This article also has a discussion on the use of techniques that are specific to the client side.

❑ **Check Your Progress – 4 :**

1. The _____ property of the GridView gives users access to the data rows contained within the GridView.

   a. Columns b. Rows c. Table d. None of these

8.10 Hyperlinkfield :

Represents a field in a data-bound control that is shown as a hyperlink in the control's display.

Example :

The following piece of code serves as an example of how to use a GridView control's HyperLinkField object to display a column of static hyperlinks. The Text and NavigateUrl properties of the HyperLinkField object are responsible for assigning the identical navigation URL and caption to each each hyperlink contained within the object.

HTML Source Code :

```
<%@ Page language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
   <head runat="server">
      <title>HyperLinkField Example</title>
</head>
<body>
   <form id="form1" runat="server">
      <h3>HyperLinkField Example</h3>
      <!-- Populate the Columns collection declaratively. -->
      <!-- Set the HyperLinkField field column to a static      -->
      <!-- caption and URL.                -->
      <asp:gridview id="OrdersGridView" datasourceid="OrdersSqlData
      Source"
      autogeneratecolumns="false" runat="server">
         <columns>
         <asp:boundfield datafield="OrderID" headertext="OrderID"/>
            <asp:boundfield datafield="CustomerID" headertext=
            "Customer ID"/>
            <asp:boundfield datafield="OrderDate"  headertext="Order
            Date"
               dataformatstring="{0:d}" />
            <asp:hyperlinkfield text="Details..." navigateurl="~\details
            .aspx"
               headertext="Order Details" target="_blank" />
            </columns>
```

```
</asp:gridview>
<!-- This example uses Microsoft SQL Server and connects -->
<!-- to the Northwind sample database -->
<asp:sqldatasource id="OrdersSqlDataSource" selectcommand
="SELECT [OrderID], [CustomerID], [OrderDate] FROM
[Orders]" connectionstring ="server=localhost;database=
northwind;integrated security=SSPI" runat = "server">
</asp:sqldatasource>
</form>
</body>
</html>
```

The following piece of code serves as an illustration of how to tie a HyperLinkField object to the fields that are present in a data source. You can specify the fields to bind to the caption and the navigation URL of each hyperlink displayed in the HyperLinkField object by using the DataTextField and DataNavigateUrlFields properties, respectively. These properties are located on the HyperLinkField object.

HTML Source Code :

```
<%@ Page language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>HyperLinkField Example</title>
</head>
<body>
   <form id="form1" runat="server">
   <h3>HyperLinkField Example</h3>
   <asp:gridview id="OrdersGridView" datasourceid="OrdersSqlData
   Source"
      autogeneratecolumns="false" runat="server">
      <columns>
         <asp:boundfield datafield="OrderID" headertext="Order ID"/>
         <asp:boundfield datafield="ProductID" headertext="Product ID"/>
         <asp:hyperlinkfield datatextfield="UnitPrice" datatextformat
         string="{0:c}"
         datanavigateurlfields="ProductID" datanavigateurlformatstring=
         "~\details.aspx? ProductID={0}" headertext="Price" target="_
         blank" />
         <asp:boundfield datafield="Quantity" headertext="Quantity"/>
      </columns>
      </asp:gridview>
```

```
    <!-- This example uses Microsoft SQL Server and connects -->
    <!-- to the Northwind sample database.-->
    <asp:sqldatasource id="OrdersSqlDataSource" selectcommand=
    "SELECT [OrderID], [ProductID], [UnitPrice], [Quantity] FROM [Order
    Details]" connectionstring= "server=localhost;database=northwind;
    integrated security=SSPI" runat="server">
    </asp:sqldatasource> </form>
</body>
</html>
```

Properties of HyperLinkField Control

Properties	Description
AccessibleHeaderText	Gets or sets text that is rendered as the AbbreviatedText property value in some controls. (Inherited from DataControlField)
Control	Gets a reference to the data control that the DataControlField object is associated with. (Inherited from DataControlField)
ControlStyle	Gets the style of any Web server controls contained by the DataControlField object. (Inherited from DataControlField)
DataNavigateUrlFields	Gets or sets the names of the fields from the data source used to construct the URLs for the hyperlinks in the HyperLinkField object.
DataNavigateUrlFormatString	Gets or sets the string that specifies the format in which the URLs for the hyperlinks in a HyperLinkField object are rendered.
DataTextField	Gets or sets the name of the field from the data source containing the text to display for the hyperlink captions in the HyperLinkField object.
DataTextFormatString	Get or sets the string that specifies the format in which the hyperlink captions in a HyperLinkField object are displayed.
DesignMode	Gets a value indicating whether a data control field is currently viewed in a design-time environment. (Inherited from DataControlField)
FooterStyle	Gets or sets the style of the footer of the data control field. (Inherited from DataControlField)
FooterText	Gets or sets the text that is displayed in the footer item of a data control field. (Inherited from DataControlField)
HeaderImageUrl	Gets or sets the URL of an image that is displayed in the header item of a data control field. (Inherited from DataControlField)
HeaderStyle	Gets or sets the style of the header of the data control field. (Inherited from DataControlField)

HeaderText	Gets or sets the text that is displayed in the header item of a data control field. (Inherited from DataControlField)
InsertVisible	Gets a value indicating whether the DataControlField object is visible when its parent data-bound control is in insert mode. (Inherited from DataControlField)
IsTrackingViewState	Gets a value indicating whether the DataControlField object is saving changes to its view state. (Inherited from DataControlField)
ItemStyle	Gets the style of any text-based content displayed by a data control field. (Inherited from DataControlField)
NavigateUrl	Gets or sets the URL to navigate to when a hyperlink in a HyperLinkField object is clicked.
ShowHeader	Gets or sets a value indicating whether the header item of a data control field is rendered. (Inherited from DataControlField)
SortExpression	Gets or sets a sort expression that is used by a data source control to sort data. (Inherited from DataControlField)
Target	Gets or sets the target window or frame in which to display the Web page linked to when a hyperlink in a HyperLinkField object is clicked.
Text	Gets or sets the text to display for each hyperlink in the HyperLinkField object.
ValidateRequestMode	Gets or sets a value that specifies whether the control validates client input. (Inherited from DataControlField)
ViewState	Gets a dictionary of state information that allows you to save and restore the view state of a DataControlField object across multiple requests for the same page. (Inherited from DataControlField)
Visible	Gets or sets a value indicating whether a data control field is rendered. (Inherited from DataControlField)

Methods of HyperLinkField Control

Methods	Description
CloneField()	Creates a duplicate copy of the current DataControl Field-derived object. (Inherited from DataControlField)
CopyProperties (DataControlField)	Copies the properties of the current HyperLinkField object to the specified object.
CreateField()	Returns a new instance of the HyperLinkField class.

Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
DataControlRowState, Boolean)	Extracts the value of the data control field from the current table cell and adds the value to the specified IDictionary collection. (Inherited from DataControlField)
FormatDataNavigate UrlValue(Object[])	Formats the navigation URL using the format string specified by the DataNavigateUrlFormatString property.
FormatDataText Value(Object)	Formats the caption text using the format string specified by the DataTextFormatString property.
GetHashCode() Object)	Serves as the default hash function. (Inherited from
GetType() Object)	Gets the Type of the current instance. (Inherited from
Initialize (Boolean, Control)	Initializes the HyperLinkField object.
InitializeCell (DataControlFieldCell, DataControlCellType, DataControlRowState, Int32)	Initializes a cell in a HyperLinkField object.
LoadViewState(Object)	Restores the data source view's previously saved view state. (Inherited from DataControlField)
MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object)
OnFieldChanged()	Raises the FieldChanged event. (Inherited from DataControlField)
SaveViewState()	Saves the changes made to the DataControlField view state since the time the page was posted back to the server. (Inherited from DataControlField)
ToString()	Returns a string that represents this DataControlField object. (Inherited from DataControlField)
TrackViewState()	Causes the DataControlField object to track changes to its view state so they can be stored in the control's ViewState property and persisted across requests for the same page. (Inherited from DataControlField)
ValidateSupports Callback()	Indicates that the controls contained by the HyperLink Field object support callbacks.

8.11 Imagefield :

In a data-bound control, this represents a field that can also be shown as an image.

Example : In order to illustrate how to display an image using an ImageField object in a GridView control, the following example demonstrates how to do so.

HTML Source Code :

```
<%@ Page language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>ImageField Example</title>
</head>
<body>
   <form id="form1" runat="server">
   <h3>ImageField Example</h3>
   <asp:gridview id="EmployeesGrid" autogeneratecolumns="false" datasourceid = " EmployeeSource" runat="server">
   <columns>
   <asp:imagefield dataimageurlfield="PhotoPath" alternatetext= "Employee Photo" nulldisplaytext = "No image on file." headertext="Photo" readonly="true"/>
   <asp:boundfield datafield="FirstName" headertext="First Name"/>
   <asp:boundfield datafield="LastName" headertext="Last Name"/>
   </columns>
   </asp:gridview>
      <!-- This example uses Microsoft SQL Server and connects -->
      <!-- to the Northwind sample database. Use an ASP.NET  -->
      <!-- expression to retrieve the connection string value  -->
      <!-- from the Web.config file.  -->
<asp:sqldatasource id="EmployeeSource" selectcommand="Select [EmployeeID], [LastName], [FirstName], [PhotoPath] From [Employees]" connectionstring="<%$ ConnectionStrings:NorthWindConnectionString %>" runat="server"/>
</form>
</body>
</html>
```

Data-bound controls, such as GridView and DetailsView, make use of the ImageField class in order to display a picture in conjunction with each record that is shown in the control. Depending on the data-bound control in which it is used, the ImageField object will display its contents in a manner that is unique to that control. As an illustration, the DetailsView control displays an ImageField object in the form of a row, but the GridView control displays it in the form of a column.

In order to display images, you need to bind an ImageField object to a field in a data source that holds the URL of an image. This will allow you to display the image. Setting the DataImageUrlField property in the document

will accomplish this goal. The DataImageUrlFormatString property can be used to format the URL value in a variety of different ways. Alternate text may also be attached with each image if the user so chooses. When an image cannot be loaded or is not accessible, this text will be displayed in its place. This text will also be displayed as a ToolTip in browsers that have support for the ToolTips feature. You can choose which of the following approaches you want to use to indicate the alternate text for a picture that is being displayed:

- You can define alternate text that applies to all images by making use of the property known as AlternateText.
- Binding a field from a data source to the AlternateText property of each picture can be accomplished through the use of the DataAlternateTextField property. This gives you the ability to have a distinct substitute text shown alongside each image. When binding data, you have the option of formatting the alternate text by using the DataAlternateTextFormatString property. If you do so, the formatting will take effect immediately.

A null value for the URL associated with an image prevents it from being displayed in its entirety. By providing the NullImageUrl attribute, you will be able to display a different picture for fields that contain null values. You can display alternate text rather than an alternate picture by using the NullDisplayText attribute in place of an alternate image.

All fields that have been declared are going to be displayed in the data-bound control by default. By setting the Visible attribute of an ImageField object in a data-bound control to false, you can conceal that item from view.

You have the ability to personalise an ImageField object's header as well as its footer sections. By configuring the HeaderText or FooterText attributes, respectively, you can show a caption in the header or footer part of the document. Changing the value of the HeaderImageUrl property will cause the picture to be displayed in the header section rather than the text. By using the ShowHeader property of the ImageField object and setting it to false, the header area of the image can be concealed.

Note: Some data-bound controls, such as the GridView control, only allow the complete header part of the control to be shown or hidden when using the show or hide function. The ShowHeader property for an individual data-bound field is not supported by these data-bound controls. Use a data-bound control's ShowHeader property to display or conceal the whole header part of the control.

Properties of ImageField Control

Properties	Description
AccessibleHeaderText	Gets or sets text that is rendered as the AbbreviatedText property value in some controls. (Inherited from DataControlField)
AlternateText	Gets or sets the alternate text displayed for an image in the ImageField object.
Control	Gets a reference to the data control that the DataControlField object is associated with. (Inherited from DataControlField)

ControlStyle	Gets the style of any Web server controls contained by the DataControlField object. (Inherited from DataControlField)
ConvertEmptyStringToNull	Gets or sets a value indicating whether empty string ("") values are converted to null when the field values are returned from the data source.
DataAlternateTextField	Gets or sets the name of the field from the data source that contains the values to bind to the AlternateText property of each image in an ImageField object.
DataAlternateTextFormatString	Gets or sets the string that specifies the format in which the alternate text for each image in an ImageField object is rendered.
DataImageUrlField	Gets or sets the name of the field from the data source that contains the values to bind to the ImageUrl property of each image in an ImageField object.
DataImageUrlFormatString	Gets or sets the string that specifies the format in which the URL for each image in an ImageField object is rendered.
DesignMode	Gets a value indicating whether a data control field is currently viewed in a design-time environment. (Inherited from DataControlField)
FooterStyle	Gets or sets the style of the footer of the data control field. (Inherited from DataControlField)
FooterText	Gets or sets the text that is displayed in the footer item of a data control field. (Inherited from DataControlField)
HeaderImageUrl	Gets or sets the URL of an image that is displayed in the header item of a data control field. (Inherited from DataControlField)
HeaderStyle	Gets or sets the style of the header of the data control field. (Inherited from DataControlField)
HeaderText	Gets or sets the text that is displayed in the header item of a data control field.(Inherited from DataControlField)
InsertVisible	Gets a value indicating whether the DataControlField object is visible when its parent data-bound control is in insert mode.(Inherited from DataControlField)
IsTrackingViewState	Gets a value indicating whether the Data ControlField object is saving changes to its view state.(Inherited from DataControl Field)

ItemStyle	Gets the style of any text-based content displayed by a data control field. (Inherited from DataControlField)
NullDisplayText	Gets or sets the text to display in an ImageField object when the value of the field specified by the DataImageUrlField property is null.
NullImageUrl	Gets or sets the URL to an alternate image displayed in an ImageField object when the value of the field specified by the DataImageUrlField property is null.
ReadOnly	Gets or sets a value indicating whether the values of the field specified by the DataImageUrlField property can be modified in edit mode.
ShowHeader	Gets or sets a value indicating whether the header item of a data control field is rendered. (Inherited from DataControlField)
SortExpression	Gets or sets a sort expression that is used by a data source control to sort data. (Inherited from DataControlField)
ValidateRequestMode	Gets or sets a value that specifies whether the control validates client input. (Inherited from DataControlField)
ViewState	Gets a dictionary of state information that allows you to save and restore the view state of a DataControlField object across multiple requests for the same page. (Inherited from DataControlField)
Visible	Gets or sets a value indicating whether a data control field is rendered. (Inherited from DataControlField)

Methods of ImageField Control

Methods	Description
CloneField()	Creates a duplicate copy of the current DataControl Field-derived object. (Inherited from DataControlField)
CopyProperties (DataControlField)	Copies the properties of the current HyperLink Field object to the specified object.
CreateField()	Returns a new instance of the HyperLinkField class.
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)

ExtractValuesFromCell (IOrderedDictionary, DataControlFieldCell, DataControlRowState, Boolean)	Fills the specified IOrderedDictionary object with the values from the specified DataControlFieldCell object.
FormatImageUrlValue (Object)	Applies the format specified by the DataImageUrl FormatString property to a field value.
GetDesignTimeValue()	Retrieves the value used for a field's value when rendering the ImageField object in a designer.
GetFormattedAlternateText (Control)	Applies the format specified by the DataAlternate TextFormatString property to the alternate text value contained in the specified Control object.
GetHashCode()	Serves as the default hash function. (Inherited from Object)
GetType()	Gets the Type of the current instance. (Inherited from Object)
Initialize (Boolean, Control)	Initializes the HyperLinkField object.
InitializeCell (DataControlFieldCell, DataControlCellType, DataControlRowState, Int32)	Initializes a cell in a HyperLinkField object.
InitializeDataCell (DataControlFieldCell, DataControlRowState)	Initializes the specified DataControlFieldCell object with the specified row state.
LoadViewState(Object)	Restores the data source view's previously saved view state. (Inherited from DataControlField)
MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object)
OnFieldChanged()	Raises the FieldChanged event. (Inherited from DataControlField)
SaveViewState()	Saves the changes made to the DataControlField view state since the time the page was posted back to the server. (Inherited from DataControlField)
ToString()	Returns a string that represents this DataControl Field object. (Inherited from DataControlField)
TrackViewState()	Causes the DataControlField object to track changes to its view state so they can be stored in the control's ViewState property and persisted across requests for the same page. (Inherited from DataControlField)
ValidateSupportsCallback()	Indicates that the controls contained by the HyperLinkField object support callbacks.

8.12 Sql Wild Cards :

In order to change the value of one or more characters in a string, a wildcard character can be used. When use the LIKE operator, wildcard characters are required. Within a WHERE clause, the LIKE operator can be utilized to do a search for a certain pattern within a column.

WildCard Characters in SQL Server

Symbol	Description	Example
%	Represents zero or more characters	bl% finds bl, black, blue, and blob
_	Represents a single character	h_t finds hot, hat, and hit
[]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
^	Represents any character not in the brackets	h[^oa]t finds hit, but not hot and hat
–	Represents any single character within the specified range	c[a-b]t finds cat and cbt

The following is a selection of examples demonstrating several LIKE operators using the '%' and '_' wildcards :

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a__%'	Finds any values that starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"

- **Using the % Wildcard :**

Example : The following SQL statement will select all clients whose City names begin with "ber":

```
SELECT * FROM Customers
WHERE City LIKE 'ber%';
```

Number of Records: 3

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
14	Chop-suey Chinese	Yang Wang	Hauptstr. 29	Bern	3012	Switzerland
49	Magazzini Alimentari Riuniti	Giovanni Rovelli	Via Ludovico il Moro 22	Bergamo	24100	Italy

The following SQL statement will select all of the clients whose City matches the pattern "es" :

```
SELECT * FROM Customers
WHERE City LIKE '%es%';
```

Number of Records: 9

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
18	Du monde entier	Janine Labrune	67, rue des Cinquante Otages	Nantes	44000	France
26	France restauration	Carine Schmitt	54, rue Royale	Nantes	44000	France
38	Island Trading	Helen Bennett	Garden House Crowther Way	Cowes	PO31 7PJ	UK
40	La corne d'abondance	Daniel Tonini	67, avenue de l'Europe	Versailles	78000	France
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
64	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
88	Wellington Importadora	Paula Parente	Rua do Mercado, 12	Resende	08737-363	Brazil

- **Using the _ Wildcard :**

Example : The following SQL statement will pick all of the clients whose City name begins with any character and then includes the word "ondon":

```
SELECT * FROM Customers
WHERE City LIKE '_ondon';
```

Number of Records: 6

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
16	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London	WX1 6LT	UK
19	Eastern Connection	Ann Devon	35 King George	London	WX3 6FW	UK
53	North/South	Simon Crowther	South House 300 Queensbridge	London	SW7 1RZ	UK
72	Seven Seas Imports	Hari Kumar	90 Wadhurst Rd.	London	OX15 4NB	UK

The following SQL statement will select all customers whose City name begins with "L," is followed by any character, then "n," followed by any character, then "on," as shown below :

```
SELECT * FROM Customers
WHERE City LIKE 'L_n_on';
```

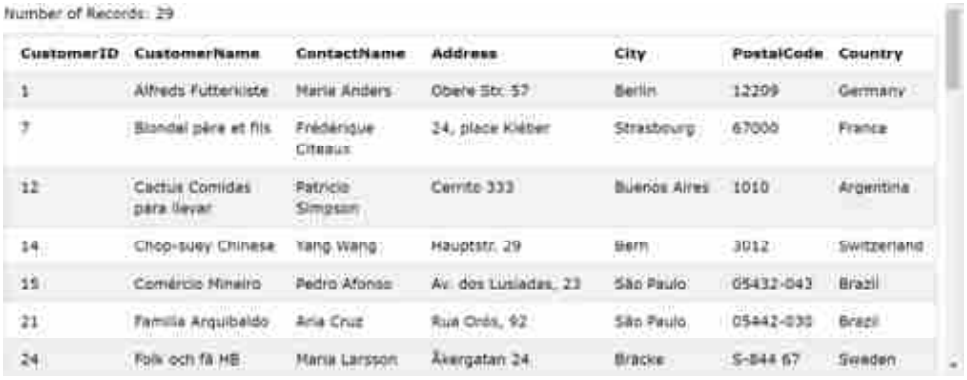
Number of Records: 6

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
16	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London	WX1 6LT	UK
19	Eastern Connection	Ann Devon	35 King George	London	WX3 6FW	UK
53	North/South	Simon Crowther	South House 300 Queensbridge	London	SW7 1RZ	UK
72	Seven Seas Imports	Hari Kumar	90 Wadhurst Rd.	London	OX15 4NB	UK

- **Using the [charlist] Wildcard :**

Example : All of the customers whose City names begin with "b," "s," or "p" will be selected by the following SQL statement:

```
SELECT * FROM Customers
WHERE City LIKE '[bsp]%';
```

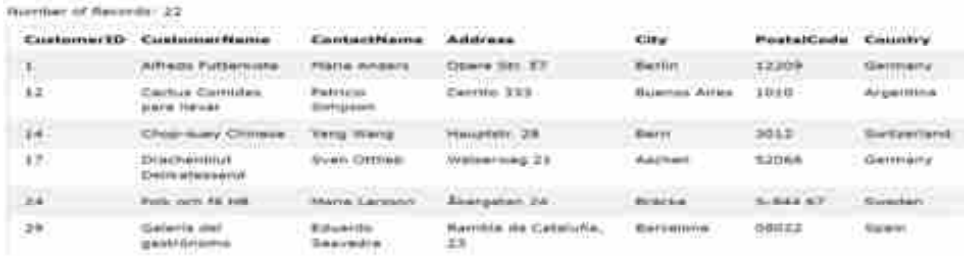


Number of Records: 29

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
14	Chop-suey Chinese	Yang Wang	Hauptstr. 29	Bern	3012	Switzerland
15	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	São Paulo	05432-043	Brazil
21	Família Arquibaldo	Aria Cruz	Rua Orós, 92	São Paulo	05442-030	Brazil
24	Folk och fä HB	Maria Larsson	Åvergatan 24	Bräcke	S-844 67	Sweden

The following SQL statement will pick all of the clients whose City names begin with "a," "b," or "c," respectively:

```
SELECT * FROM Customers
WHERE City LIKE '[a-c]%';
```



Number of Records: 22

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
14	Chop-suey Chinese	Yang Wang	Hauptstr. 29	Bern	3012	Switzerland
17	Drachenblut Delikatessend	Sven Ottlieb	Walserweg 21	Aachen	52066	Germany
24	Folk och fä HB	Maria Larsson	Åvergatan 24	Bräcke	S-844 67	Sweden
29	Galería del gastrónomo	Eduardo Saavedra	Rambla de Cataluña, 23	Barcelona	08022	Spain

- **Using the [!charlist] Wildcard :**

Example :

The two SQL commands that follow pick all clients whose City does not begin with "b," "s," or "p," respectively:

```
SELECT * FROM Customers
WHERE City LIKE '[!bsp]%';
```

OR

```
SELECT * FROM Customers
WHERE City NOT LIKE '[bsp]%';
```

Number of Records: 62

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany

8.13 Two–Way Data Binding :

Expressions that execute data binding employ the Eval and Bind functions in their construction. The delimiters %# and %> are what enclose these expressions in their respective brackets. The Eval function can be used to define one-way binding, which is also commonly referred to as read-only binding. Utilizing the Bind function is the best option for achieving two-way binding that permits updates. In addition to calling the Eval and Bind methods to

perform data binding in a data-binding expression, you can call any publicly scoped code within the %# and %> delimiters to execute that code and return a value while the page is being processed. This allows you to perform data binding in a data-binding expression without having to call the Eval and Bind methods. You won't have to explicitly call the Eval and Bind methods in order to conduct data binding if you take advantage of this feature.

When the DataBind method of a control or the Page class is called, all expressions requiring data binding will be parsed and resolved. This happens everytime the method is called. Because data-binding expressions are resolved automatically during the PreRender event of the control, it is not essential for you to manually activate the DataBind method for controls such as the GridView, DetailsView, and FormView controls. Instead, you can skip this step entirely. GridView is an illustration of one of these control examples.

Two Way data binding was developed as an alternative to the framework capabilities provided by the Page class in order to support a scheme such as built-in simple databinding, which does not require any support from the bound controls themselves. This is because built-in simple databinding does not require any support from bound controls themselves.

- Utilizing Something That Is Called the Eval Method
- Making use of what's known as the Bind Technique

Utilizing Something That Is Called the Eval Method : The Eval method is used to evaluate late-bound data expressions that are located in the templates of data-bound controls like as the GridView, DetailsView, and FormView controls. When the Eval method of the DataBinder object is called during the execution phase, it makes a reference to the data item that is presently active within the naming container. This happens when it calls the Eval method of the DataBinder object. In most circumstances, the name container is the smallest part of the data-bound control that stores a full record. This is because it is responsible for storing the record's name. One illustration of this would be a row in the control known as GridView. Due to the fact that this is the case, the Eval approach is the only one that may be used for binding inside of the templates of a data-bound control.

The name of a data field must be provided as an argument for the Eval method, and the method then returns a string that contains the value of the data field, which is derived from the record that is now selected in the data source. You have the option of supplying a second argument in order to specify a format for the string that is returned. This is done by specifying the format in which the string is to be provided. The syntax that is utilised for the string format parameter is the same syntax that is utilised for the Format function that is utilised for the String class.

Making use of what's known as the Bind Technique : The Eval method and the Bind method are somewhat comparable to one another; yet, there are important distinctions between the two systems. To be more specific, the Eval method as well as the Bind method. To retrieve the values that are kept in data-bound fields, you may either use the Eval method or the Bind method. Both of these methods are available. Since these two approaches are interchangeable, it is possible to accomplish this goal. On the other hand, the Bind method is utilised in circumstances in which the data itself can be modified, whilst the Eval method only makes it possible to obtain the data

that is at issue. Both methods, however, make it feasible to retrieve the data in question. The reason for this is that the Eval method merely makes it possible to access the data that is in concern. This is because the Bind method allows for modifications to be made to the data, which is the reason why this is the case.

8.14 Let Us Sum Up :

In this unit, we learned that ASP.NET web form controls inherit the DataBind method from their parent Control class, allowing them to bind data to one of their properties. Replacing filter text box with drop-down list creates custom column class. The dropdown list can filter records when the user picks an item.

RepeatColumns attribute controls editor item display direction. Most asp.net programmes require checkboxes in gridview header templates to check or uncheck all rows. If gridview AutoGenerateColumns property is True, gridview creates an AutoGenerateField object for every field in data source. If False, we must manually specify which column fields appear in gridview. ImageField is CCK-based picture upload field. Multiple photos per node, resolution limits, default images, and rich Views support are enhanced.

8.15 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : a **2 :** c

- **Check Your Progress 2 :**

  1 : b **2 :** b)

- **Check Your Progress 3 :**

  1 : c

- **Check Your Progress 4 :**

  1 : b

8.16 Glossary :

1. **Attributes :** An attribute is a changeable property or characteristic of some component of a program that can be set to different values.
2. **CCK :** Complimentary Code Keying.
3. **HyperLink :** An electronic link providing direct access from one distinctively marked place in a hypertext or hypermedia document to another in the same or a different document.
4. **Properties :** Properties are the settings of an object on a computer.

8.17 Assignment :

1. Explain Simple and Declarative Data Binding in detail.

8.18 Activities :

1. How do we use Eval Method in our Web Application ? Explain.

8.19 Case Study :

Study about the use of CheckboxList, RadiobuttonList and HyperLinkField in detail.

8.20 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

Unit
09

LINQ QUERIES

UNIT STRUCTURE

9.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About LINQ Queries
- About Query Operators
- About how we apply LINQ Query to Objects
- About how we apply LINQ Query to ADO.NET
- About how we apply LINQ Query to XML
- About LinqDataSourceControl

9.1 Introduction :

Object-oriented programming principles can be applied to relational data by utilising LINQ, which stands for Language Integrated Query. It does so without the need for any additional libraries by extending data capabilities directly into the programming languages C# and Visual Basic, providing a single programming model for accessing data from a variety of different types of data sources, and by doing so it eliminates the need for additional data access libraries. More information regarding Language-Integrated Query can be found by clicking here (LINQ).

Because it offers a single working model for a wide variety of data sources and formats, LINQ makes it significantly simpler to manage this issue. This model may be applied to the processing of data in a variety of ways.

When you use LINQ to do a query, you will always be interacting with objects in some capacity.

LINQ queries are never executed when the query variable is being formed; rather, they are always executed when the query variable is being iterated over. In other words, LINQ queries are never executed when the query variable is being formed. The term for this type of procedure is "delayed execution." When you need to cache the results of a query, you may also have a query execute rapidly, which is beneficial.

9.2 Introduction of LINQ Queries :

Language-Integrated Query, or LINQ, is the name given to a group of technologies that are founded on the principle of directly integrating query capabilities into the C# programming language. In the past, searches on data were typically written as uncomplicated strings, and there was no support for IntelliSense or type checking during the compilation process. In addition, you will need to become proficient in a distinct query language for each variety of data source :

- SQL databases,
- XML documents,
- Different Web services, among other things.

LINQ elevates the query to the same status as other first-class language constructs such as classes, methods, and events. Querying against strongly typed collections of objects is accomplished through the utilization of language keywords and conventional operators.

A consistent query experience can be obtained by utilising the LINQ family of technologies :

- Objects (LINQ to Objects) (LINQ to Objects),
- Relationship-based databases, LINQ to Structured Query Language, and
- XML (LINQ to XML) (LINQ to XML).

The query expression is the most "language-integrated" aspect of LINQ, and it is the part that is most visible to a developer who produces queries. Declarative query syntax is the format that query expressions are written in. You may do operations such as filtering, ordering, and grouping on data sources using query syntax, which requires a little amount of code on your part.

When querying and transforming data stored in SQL databases, ADO.NET Datasets, XML documents and streams, and.NET collections, you employ the same fundamental query expression patterns across all of these storage types.

You are able to build LINQ queries in C# for any language that supports IEnumerable or the generic IEnumerableT> interface are :

- SQL Server databases,
- XML documents,
- The ADO.NET Datasets, in addition to any group of objects.

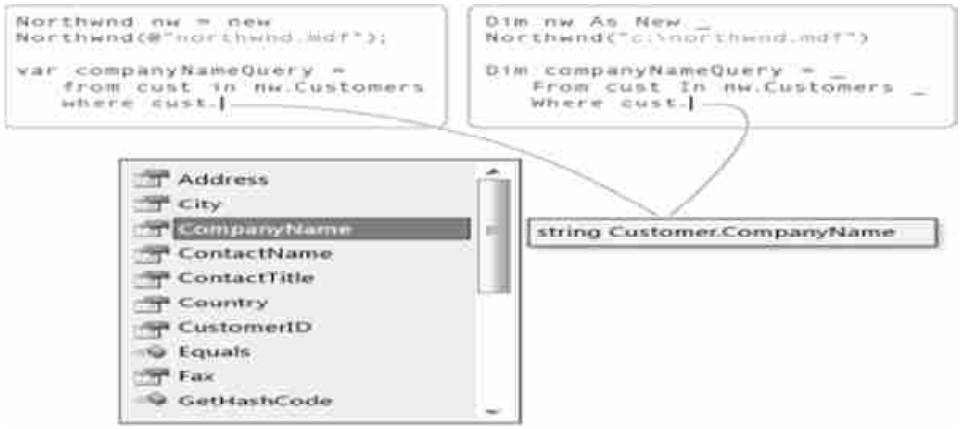
Support for the LINQ query language is also offered by third parties for a variety of database implementations and Web services.

Example : The operation is complete once a data source has been created, the query expression has been defined, and the query has been carried out using a foreach statement.

C# Source Code :

```
// Specify the data source.
int[] scores = { 97, 72, 88, 60 };
// Define the query expression.
IEnumerable<int> scoreQuery =
    from score in scores
    where score > 63
    select score;
// Execute the query.
foreach (int i in scoreQuery)
{
    Console.Write(i + " ");
}
// Output: 97 72 88
```

The following example is taken from Visual Studio and displays a LINQ query that has only been partially completed. The query is being run against a SQL Server database in both C# and Visual Basic, and it has full support for type checking and IntelliSense :



Some of the typical query operators that are used more frequently have their own specialized keyword syntax in the C# language, which makes it possible for them to be called as part of a query expression. The method-based equivalent of a query is known as a query expression, which is a different and more understandable approach of expressing a question. At the time of compilation, query expression clauses are transformed into calls to the respective query methods.

The following table provides a listing of the standard query operators that are accompanied by corresponding query expression clauses.

Method	Query Expression Syntax
Cast	Use an explicitly typed range variable, for example: from int i in numbers
GrouBy	group … by -or- group … by … into …
GroupJoin<TOuter,TInner,TKey, TResult>(IEnumerable<TOuter>, IEnumerable<TInner>, Func<TOuter, TKey>, Func<TInner,TKey>, Func<TOuter, IEnumerable<TInner>, TResult>)	join … in … on … equals … into …
Join<TOuter,TInner,TKey, TResult> (IEnumerable<TOuter>, IEnumerable <TInner>,Func<TOuter,TKey>, Func <TInner, TKey>, Func<TOuter, TInner,TResult>)	join … in … on … equals …
OrderBy<TSource,TKey>(IEnumerable <TSource>, Func<TSource,TKey>)	orderby
OrderByDescending<TSource, TKey>(IEnumerable<TSource>, Func<TSource,TKey>)	orderby … descending
Select	select
SelectMany	Multiple from clauses.
ThenBy<TSource,TKey> (IOrderedEnumerable<TSource>, Func<TSource,TKey>)	orderby …, …
ThenByDescending<TSource, TKey>(IOrderedEnumerable<TSource>, Func<TSource,TKey>)	orderby …, … descending
Where	where

9.3 Query Operators :

The LINQ pattern is formed by the usual query operators, which are also known as methods. The majority of these methods are designed to work with sequences, where a sequence is defined as an object whose type either implements the IEnumerable<T> interface or the IQueryable<T> interface. Query capabilities such as filtering, projection, aggregation, sorting, and more can be obtained through the use of the standard query operators.

There are two different sets of standard query operators that can be used with LINQ.

- The first set operates on objects of type IEnumerable<T>, and
- The second set operates on objects of type IQueryable<T>.

The Enumerable and Queryable classes each have their own set of static methods that are responsible for making up their respective sets. They are extension methods of the type that they act on, which is how they are defined. Either the syntax for calling static methods or the syntax for calling instance methods can be used to invoke extension methods.

In addition, IEnumerable<T> and IQueryable<T> aren't the only types that may be queried using conventional query operator methods. These methods can also be used on other types. Both of these kinds of methods are defined for the Enumerable type, and they both work on objects of the IEnumerable type.

Cast<TResult>(IEnumerable) and OfType<TResult>(IEnumerable) are two methods that enable you to query a non-parameterized or non-generic collection using the LINQ pattern.

These methods are called Cast and OfType, respectively. They accomplish this goal by constructing a collection of objects that is tightly typed. Cast<TResult>(IQueryable) and OfType<TResult>(IQueryable) are two methods that are defined by the Queryable class that are quite similar to one another and work on objects of the type IQueryable.

The timing of the execution of the standard query operators differs depending on whether they return a singleton value or a sequence of values. This is because the singleton value can be found only once. The methods that return a singleton value, such as Average and Sum, are the ones that are executed instantly. Methods that return a sequence postpone the execution of the query and instead return an object that can be iterated through.

The returned enumerable object for methods that act on in-memory collections, that is, methods that extend IEnumerable<T>, stores the arguments that were supplied to the method in the same order in which they were passed to the method. The logic of the query operator is applied whenever that object is enumerated, and the results of the query are the ones that are returned.

On the other hand, querying behaviour is not implemented in any way by methods that extend the IQueryable<T> interface. They construct an expression tree that is representative of the query that is going to be run. The processing of queries is taken care of by the source object, which is an IQueryable<T>. It is possible to link many calls to query methods together within a single query, which makes it possible for inquiries to become indefinitely complicated.

Example : The following example shows how the common query operators can be utilized to acquire information regarding a sequence.

```
string sentence = "the quick brown fox jumps over the lazy dog";
// Split the string into individual words to create a collection.
string[] words = sentence.Split(' ');
// Using query expression syntax.
var query = from word in words
            group word.ToUpper() by word.Length into gr
            orderby gr.Key
            select new { Length = gr.Key, Words = gr };
// Using method-based query syntax.
var query2 = words.GroupBy(w => w.Length, w => w.ToUpper()).
   Select(g => new { Length = g.Key, Words = g }).
   OrderBy(o => o.Length);
foreach (var obj in query)
```

```
{
    Console.WriteLine("Words of length {0}:", obj.Length);
    foreach (string word in obj.Words)
        Console.WriteLine(word);
}
// This code example produces the following output:
// Words of length 3:
// THE
// FOX
// THE
// DOG
// Words of length 4:
// OVER
// LAZY
// Words of length 5:
// QUICK
// BROWN
// JUMPS
```

9.4 LINQ to Objects :

The use of LINQ queries with any IEnumerable or IEnumerable<T> collection directly is what is meant by the term "LINQ to Objects." This means that there is no need to use an intermediate LINQ provider or API, such as LINQ to SQL or LINQ to XML.

Instead, LINQ queries can be used directly with the collection. You can query any enumerable collection with LINQ, including List<T>, Array, and Dictionary<TKey,TValue>, for example. The collection can either be user-defined or returned via a.NET API. Both options are possible.

A fresh strategy for dealing with collections is what LINQ to Objects introduces in the most fundamental sense. In the old method, in order to obtain data from a collection, you were required to create intricate foreach loops that detailed how to do so. The LINQ methodology requires you to develop code that makes declarative statements about the information you wish to get.

In addition, in comparison to conventional foreach loops, LINQ queries have the following three primary benefits:

- They are shorter and easier to read, particularly when filtering numerous conditions at the same time.
- They allow for sophisticated filtering, ordering, and grouping capabilities with a minimal amount of application code required.
- They require little to no change in order to be ported across to several different data sources.

In general, the more complicated the operation you wish to execute on the data, the greater value you will receive by utilizing LINQ rather than traditional iteration techniques. This is because LINQ allows you to do more sophisticated operations on larger amounts of data.

❑ **Check Your Progress – 1 :**

1. _____ is the name given to a group of technologies that are founded on the principle of directly integrating query capabilities into the C# programming language.

   a. SQL b. ORACLE c. LINQ d. None of these

2. The use of LINQ queries with any IEnumerable or IEnumerable<T> collection directly is what is meant by the term _____.

   a. LINQ to SQL b. LINQ to Objects

   c. LINQ to ADO.NET d. None of these

9.5 LINQ to ADO.NET :

You are able to query over any enumerable object in ADO.NET by using the Language-Integrated Query (LINQ) programming model, which is made possible by the LINQ to ADO.NET extension.

There are three individual technologies that fall within the ADO.NET Language-Integrated Query (LINQ) category :

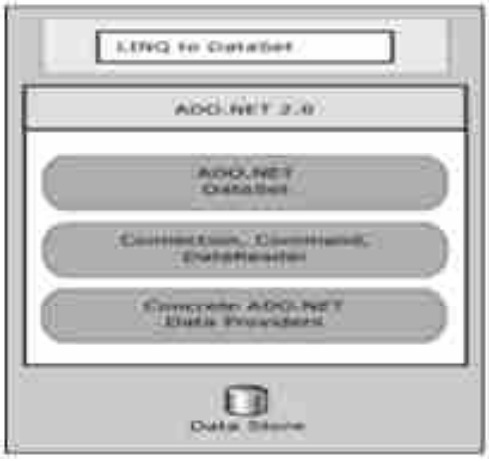
- LINQ to DataSet
- LINQ to SQL
- LINQ to Entities

You can directly query SQL Server database schemas using LINQ to SQL, and you can query an entity data model using LINQ to Entities. LINQ to DataSet offers more robust and efficient querying over the DataSet.

LINQ to DataSet : LINQ to DataSet enables querying cached data faster and easier. Instead of using a query language, LINQ to DataSet lets developers write queries in the programming language. Visual Studio developers can now query with compile-time syntax checking, static typing, and IntelliSense support.

LINQ to DataSet can also query consolidated data. This allows Web application middle-tier caching and querying locally aggregated data. Generic reporting, analysis, and business intelligence require this manipulation.

DataRowExtensions and DataTableExtensions extension methods expose LINQ to DataSet capability. LINQ to DataSet does not replace ADO.NET in application code; it builds on it. ADO.NET code will work in LINQ to DataSet. This diagram shows how LINQ to DataSet interacts with ADO.NET and the data store.



LINQ to SQL : LINQ to SQL is a component of the.NET Framework version 3.5 that offers a run-time infrastructure for managing relational data as objects. This infrastructure may be accessed through the use of a query language called LINQ.

Note that relational data is displayed as a collection of two-dimensional tables, also known as relations or flat files. Tables are connected to one another through the columns they share in common. In order to make effective use of LINQ to SQL, you need to have some prior experience with the fundamental concepts underlying relational databases.

LINQ to SQL is a mapping technique that takes the data model of a relational database and converts it to an object model that can be represented in the developer's preferred programming language. When the application is executed, LINQ to SQL takes the language-integrated queries included in the object model, converts them into SQL, and then delivers them to the database so that they may be run. When the database delivers the results, LINQ to SQL translates them back to objects that you may work with in your own programming language. Those objects can then be returned to the database.

In most cases, developers working with Visual Studio will use the Object Relational Designer, which offers a graphical user interface and can be used to construct a significant number of LINQ to SQL capabilities.

LINQ to Entities : Language-Integrated Query (LINQ) is supported by LINQ to Entities, which enables developers to write queries on the Entity Framework conceptual model using Visual Basic or Visual C#. LINQ is an acronym for Language-Integrated Query. Command tree queries are used to represent inquiries made against the Entity Framework. These queries are carried out against the object context. LINQ to Entities takes Language-Integrated Queries (LINQ) queries and translates them into command tree queries. It then performs the queries on the Entity Framework and provides objects that can be utilised by both LINQ and the Entity Framework.

The following is the procedure that must be followed in order to create and run a query using LINQ to Entities :

- Construct an instance of ObjectQuery<T> using ObjectContext as the starting point.
- Utilizing the ObjectQuery<T> instance in either C# or Visual Basic will allow you to compose a LINQ to Entities query.
- Convert the query operators and expressions that come standard with LINQ to command trees.

- Carry out the query against the data source using the command tree form of the query. Any exceptions that are thrown on the data source while the execution is taking place are sent straight up to the client.
- Send the results of the client's inquiry back to them.

Constructing an ObjectQuery Instance : The ObjectQuery<T> generic class is used to describe a query that returns a collection of zero or more typed entities. This collection can have any number of entries. Rather than being generated manually, an object query is often constructed from an already existing object context. Because of this, an object query will always belong to the same object context. This context supplies the necessary connection and metadata information for composing and running the query, which is required in order to do so. The IQueryable<T> generic interface is implemented by the ObjectQuery<T> generic class. This interface's builder methods make it possible to construct LINQ queries in an incremental fashion. Utilizing the var keyword in C# is another option for allowing the compiler to determine the type of.

Composing the Queries : Instances of the ObjectQuery<T> generic class provide the data source for LINQ to Entities queries. This class implements the generic IQueryable<T> interface. When you run a query, you will be asked to specify in great detail the information that you wish to obtain from the underlying data source. In addition, a query can indicate how the information that is retrieved should be sorted, organized, and formatted before it is returned. A query is saved as a variable while working with LINQ. This query variable does not perform any actions and does not return any data; all it does is hold the information about the query. Executing a query is required once it has been created in order to retrieve any data from the database.

The LINQ to Entities Querying Language There are two distinct syntaxes that can be utilized for composing inquiries; these are the query expression syntax and the method-based query syntax. Both C# 3.0 and Visual Basic 9.0 introduce brand new query syntaxes, namely query expression syntax and method-based query syntax.

Query Conversion : The LINQ query needs to be translated to a command tree representation before it can be used to conduct a LINQ to Entities query against the Entity Framework. This representation can then be used to execute the LINQ query against the Entity Framework.

The LINQ standard query operators (such as Select, Where, and GroupBy) and expressions (x > 10, Contact.LastName, and so on) are what make up a LINQ to Entities query. Instead of being specified by a class, LINQ operators are actually methods that are attached to a class. Expressions written in LINQ are free to include any data that is permitted by the System's kinds. Linq. Namespace for expressions and, by extension, everything that can be modelled using a lambda function. This is a superset of the expressions that are allowed by the Entity Framework, which are restricted by definition to operations that are allowed on the database and supported by ObjectQuery<T>. This is a superset of the expressions that are allowed by the Entity Framework.

The Entity Framework uses a unified type hierarchy to describe operators and expressions, which are then arranged in a command tree. This hierarchy is then used to construct queries. The query is carried out by the Entity Framework with the assistance of the command tree. During the process of converting the query, an exception will be thrown if the LINQ query cannot

be written as a command tree. There are two separate conversions that need to take place before LINQ queries can be converted to Entities queries: the conversion of the standard query operators and the conversion of the expressions.

Since expressions in LINQ to Entities are evaluated on the server in most cases, one should not anticipate the behavior of the expression to follow CLR semantics.

Query Execution : Following the creation of the LINQ query by the user, it is then translated into a representation that is compatible for use with the Entity Framework (in the form of command trees). Then, this representation will be used in conjunction with the data source after that. Each query expression, which is also referred to as a component of the query, is evaluated at the precise instant in time that the query is being executed. This evaluation can take place either on the client or on the server, depending on which option is selected for the query. This is comprised of terms that are utilized in the process of materializing results or visualizing the existence of entities.

Materialization : The process of delivering query results to the client in the form of CLR types is known as materialization. There is always a backing CLR type, defined by the user or by the Entity Framework, or created by the compiler when working with LINQ to Entities; query results data records are never returned while using this technology (anonymous types). The Entity Framework is responsible for all of the object's actual materialization. During the process of object materialization, exceptions will be thrown for any issues that arise as a consequence of the Entity Framework and the CLR not being able to properly map one another. The following are the typical formats in which query results are returned:

- A collection of zero or more entity objects that have been typed, or a projection of complicated types as they have been defined in the conceptual model.
- Entity Framework types that are supported by the Common Language Runtime (CLR).
- Inline collections.
- Persons who remain nameless.

❑ **Check Your Progress – 2 :**

1. There are _____ individual technologies that fall within the ADO.NET Language-Integrated Query (LINQ) category.

   a. Six b. Eight c. Three d. None of these

2. The process of delivering query results to the client in the form of CLR types is known as _____.

   a. Composing the Queries b. Materialization

   c. Query Execution d. None of these

9.6 LINQ to XML :

An in-memory XML programming interface is made available by the LINQ to XML library, which makes use of the Language-Integrated Query (LINQ) Framework from Microsoft. LINQ to XML is an XML programming interface that makes use of the features of.NET and is analogous to an updated and redesigned version of the Document Object Model (DOM).

In a variety of settings, the use of XML as a data formatting standard has seen widespread adoption. For instance, XML can be located in databases, on the World Wide Web, in configuration files, and in files associated with Microsoft Office Word.

The method of working with XML known as LINQ to XML has been brought up to date and given a new design. In addition to supporting LINQ query expressions, it offers the capabilities of the Document Object Model (DOM) for modifying documents while they are still in memory. In spite of the fact that the syntax of these query expressions is different from that of XPath, the functionality that they provide is equivalent.

LINQ to XML is intended for use by many different types of developers. By offering a query experience that is analogous to that of SQL, LINQ to XML makes XML more approachable for the type of developer who, on the whole, is just looking to get things done. Programmers can learn to develop concise and effective queries in the programming language of their choosing with only a little bit of study and practise.

LINQ to XML is a tool that can significantly boost the productivity of professional software developers. They may create less code that is more expressive, more compact, and more powerful thanks to the capabilities of LINQ to XML. They are able to use query expressions from a number of different data domains all at once.

LINQ to XML is an in-memory XML programming interface that is LINQ-enabled. It gives you the ability to work with XML from within the.NET programming languages.

When you use LINQ to XML, the XML document is loaded into memory just like it is when you use the Document Object Model (DOM). You have the ability to query and edit the document; once you have modified it, you have the option of saving it to a file or serializing it so that it can be sent over the internet. However, LINQ to XML is not the same as DOM in the following ways :

- It offers a new object model that is less cumbersome and simpler to manipulate than the previous one.
- It makes use of various language capabilities, specifically those found in C# and Visual Basic.

The combination of LINQ to XML with Language-Integrated Query is the most significant benefit offered by LINQ to XML (LINQ). Because of this integration, you will have the ability to write queries on the XML document that is stored in memory, allowing you to obtain groupings of elements and attributes. The querying capabilities of LINQ to XML are quite similar to those of XPath and XQuery in terms of their functionality, but not their syntax. The incorporation of LINQ into C# and Visual Basic results in enhanced type capabilities, enhanced testing at compile time, and enhanced support from the debugger.

The capability of LINQ to XML to use query results as parameters to XElement and XAttribute object constructors offers a sophisticated approach to the creation of XML trees. This capability is an additional advantage of LINQ to XML. This strategy, which is known as functional building, makes it possible for programmers to effortlessly alter XML trees from one shape

to another. An enhanced XML programming interface can be obtained through the use of LINQ to XML. You can do the following using LINQ to XML:

- XML can be loaded from either files or streams.
- XML can be serialized to either files or streams.
- Construct XML from scratch using the functional construction notation.
- Query XML using axes that are similar to XPath.
- Utilizing methods such as Add, Remove, ReplaceWith, and SetValue, you can manipulate the XML tree that is stored in memory.
- Validate XML trees using XSD.
- To change the form of XML trees from one configuration to another, you can use a combination of these features.

When it comes to the benefits of programming with LINQ to XML, one of the most significant advantages is that it is simple to generate XML trees.

❑ **Check Your Progress – 3 :**

1. LINQ query expressions, it offers the capabilities of the _____ for modifying documents while they are still in memory.

   a. Component Object Model (COM)

   b. Document Object Model (DOM)

   c. Both a and b

   d. None of these

2. Validate XML trees using _____.

   a. XSD b. XDD c. XDS d. None of these

9.7 LinqDataSourceControl :

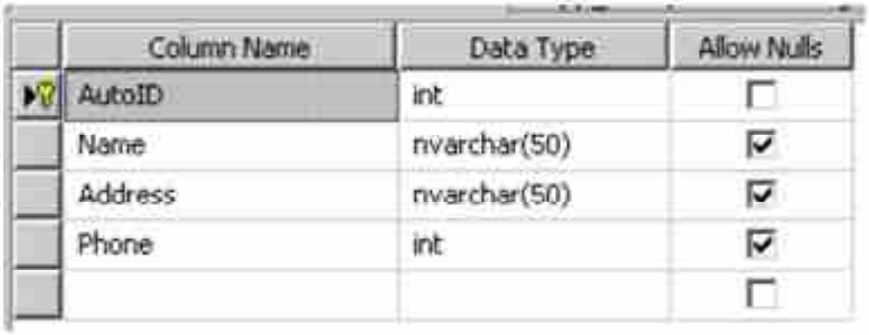
Setting property values in markup text gives you the ability to use LINQ in an ASP.NET Web page. This is made possible via the LinqDataSource control. The data instructions can be automatically generated by the LinqDataSource control thanks to the utilization of LINQ to SQL.

Syntax :

<asp:LinqDataSource ID="LinqDataSource1" runat="server" ContextTypeName =" DataClassesDataContext" Select= "new (VendorId, Vendor FName, VendorLName, VendorCity, VendorState, VendorCountry, PostedDate, VendorDescription)" TableName="Vendors" ></asp:LinqDataSource>

Create a new website with ASP.NET and link it to your database by using Server Explorer. This should be done first.

Select "Add New Item..." from the context menu that appears after you right-click the App Data folder. Choose a SQL Server Database, then click the Add button. Database Explorer should be used to investigate the tables in the database. Tables can be added by selecting the option from the context menu. Create the following fields in the same way as they are shown in the figure below.



Column Name	Data Type	Allow Nulls
AutoID	int	☐
Name	nvarchar(50)	☑
Address	nvarchar(50)	☑
Phone	int	☑
		☐

The next step is to construct some classes that use Linq to SQL. This could serve as the application's Data Access Layer if you want it to. To do this, pick Add ASP.NET Folder from the context menu that appears after you right-click the project folder in the solution explorer and choose the App Code option. Select Linq to SQL Classes using the context menu after right-clicking the App Code folder. You will discover that the App Code folder contains a DataClasses.dbml file after some time has passed. To open the DataClasses.dbml file in the Object Relational Designer (ORD) window, simply double-click on it. It is time to move on to the next step, which is to drag the Address table from Database Explorer into ORD. Move the Address table into the ORD window by going to Database Explorer and dragging it there. This will create two extra files in App Code linked to DataClasses.dbml entitled DataClasses.dbml.layout and DataClasses.designer.cs or.vb depending on what language you have selected. These files will be named DataClasses.dbml.layout and DataClasses.designer.cs or.vb respectively. Save DataClasses.dbml.

Up to this point, we have developed LINQ to SQL Classes and linked them to the appropriate database tables. When you open the DataClasses.designer.cs file, you will be able to view the many kinds of code that have been automatically generated to work with the Address database table.

Now that the LINQ to SQL Classes are installed, we can link our LinqDataSource control with them. In order to accomplish this, you will first need to generate a new.aspx page, then switch to the Design view, and then drag the LinqDataSource control from the data tab of the Toolbox. To configure the data source, open the smart tag of the LinqDataSource control and select the appropriate option. Choose DataClassesDataContext from the selection menu that appears in the Context Object list, then click the Next button. Now, from the dropdown list of tables, pick Addresses (Table<Address>), and then click the Finish button. Open the smart tag of the LinqDataSource control once more, and this time choose the checkboxes for enabling inserting, deleting, and updating data. This will make it possible to manipulate the data stored in the database.

Your LinqDataSource control can now be associated with any Data control, such as GridView or ListView, without causing any issues at all. Simply drag a GridView from the Data tab onto the LinqDataSource control, and after that, declare the DataSource attribute of the GridView that you just created. When you run your.aspx page, you will see that the records in your GridView are being fetched from the database and populated in an efficient and straightforward manner by using the LinqDataSource control. This will become apparent when you run your page.

AutoID	Name	Address1	Phone
1	Sheo Narayan	Cyberabad	5855656
6	Sunita Narayan	Hyderabad	45
8	Jay Shankar	Aurangabad	11444
9	Anil Kumar Pathak	Kothur	78

9.8 Let Us Sum Up :

In this unit, we have learned about the SQL databases, XML documents, different Web services, among other things. We have learned about the LINQ Queries and its benefits with ASP.NET. We got the knowledge about how LINQ is used with SQL, with objects, with XML and all.

We have learnt how can we fetch the data from the database using LINQdatasource with ADO.NET. We have read that how the this datasource connected to the data controls.

9.9 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : c 2 : b

- **Check Your Progress 2 :**

  1 : c 2 : b

- **Check Your Progress 3 :**

  1 : b 2 : a

9.10 Glossary :

1. **LINQ :** The name for a set of technologies based on the integration of query capabilities directly into the C# language.
2. **Objects :** Object is an entity that has state and behaviour.
3. **Execution :** To execute a program is to run the program in the computer, and, by implication, to start it to run.
4. **DOM :** ******

9.11 Assignment :

1. Explain LINQ to Entities in detail.

9.12 Activities :

1. Use LINQdatasource in your web application and try to fetch the data from the database.

9.13 Case Study :

Study Link to ADO.NET

9.14 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

Unit 10

ADO.NET FRAMEWORK

UNIT STRUCTURE

10.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About the concept of Data Modeling
- About the object services
- About Entity SQL Language
- About the entity client provider
- About the entity framework controls

10.1 Introduction :

It is a component of the whole. The link between applications and data sources can be established with the help of the Net Framework. SQL Server and XML are two examples of possible data sources. Classes are the building blocks of ADO.NET, and these classes allow users to connect, retrieve, insert, and remove data.

All of the ADO.NET classes are connected with the XML classes that are stored in the System.Xml.dll file. These classes can be found in the System.Data.dll file.

The.NET Framework data provider and the DataSet are the two primary components of ADO.NET that are utilized to gain access to and manipulate data. These components may be found in the DataSet.

Microsoft ActiveX Data Objects is what is meant when someone refers to ADO. With the help of Microsoft's ADO.NET, which is one of their Data Access Technologies, we are able to interface with a variety of data sources.

It is a component of the.NET Framework that helps to establish a connection between the.NET Application and a variety of data sources. This connection can be utilised for a variety of purposes. SQL Server, Oracle, MySQL, XML, and more formats are some examples of possible Data Sources.

ADO.NET is made up of a collection of predefined classes that may be used to connect to data sources, retrieve data, insert new data, update existing data, and remove data. This is referred to as performing CRUD operations. The System.Data.dll and System.Xml.dll files are extremely important to ADO.NET.

10.2 Concept of Data Modeling :

The process of establishing a data model for the purpose of the data that will be kept in a database is referred to as "data modelling," which is short for "data modelling." This data model is a conceptual representation of the Data objects, the rules, and the links between the various data elements.

The visual representation of data can be improved with the help of data modelling, which also ensures that business rules, regulatory compliances, and government policies are applied to the data. The consistency of naming conventions, default values, semantics, and security can all be ensured by using data models, in addition to the quality of the data itself.

An abstract model that organises the description of the data, the semantics of the data, and the consistency requirements of the data is referred to as the "Data Model." The data model places more of an emphasis on the types of data that are required and how they should be organised than it does on the types of operations that will be performed on the data. A Data Model can be thought of as the architectural plan that an architect uses to construct conceptual models and establish relationships between different data pieces.

There are two distinct categories of data modelling techniques :

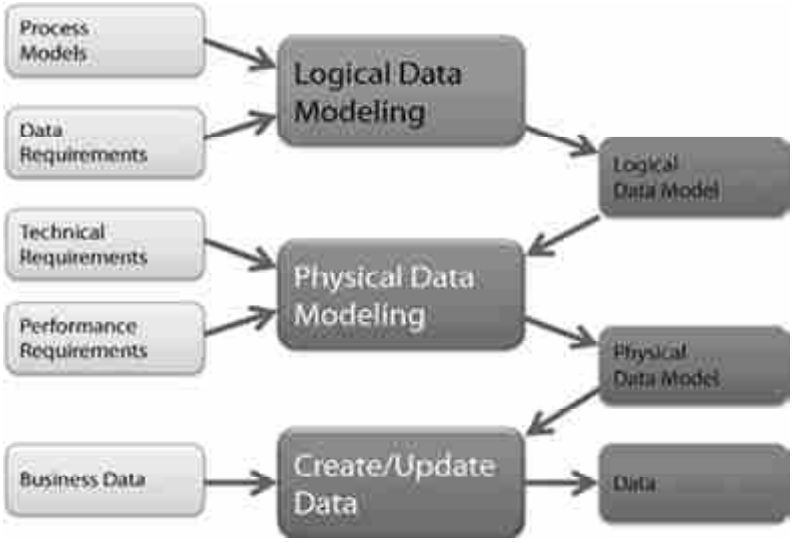
- UML (Unified Modelling Language)
- ER Model (Entity Relationship)

The following are the key objectives of utilizing data models :

- Ensures that all of the data objects that are necessary for the database are represented in an accurate manner. The omission of data will lead to the generation of inaccurate reports, which will in turn create the wrong results.
- Database design is aided conceptually, physically, and logically by the use of data models in the design process.
- The relational tables, primary and foreign keys, and stored procedures can all be better defined with the assistance of the Data Model framework.
- It offers a transparent representation of the data at its core and can be put to use by database developers in the process of creating a physical database.
- Finding data that is missing and data that is redundant can also be helpful.
- Even if the initial design of a data model requires a significant investment of work and time, in the long run, it will make it easier and less expensive to upgrade and maintain your IT infrastructure.

There are primarily three distinct varieties of data models, which are referred to as conceptual data models, logical data models, and physical data models. Each of these models serves a unique function. The data models are utilised to both represent the data and the manner in which it is kept in the database, as well as to establish the link between the various data pieces.

- **Conceptual Data Model :** The contents of the system are spelled out by this Data Model. In most cases, business stakeholders and data architects will work together to develop this model. The goal is to organise, categorise, and describe many business-related principles and concepts.
- **Logical Data Model :** Regardless of the DBMS that is used, this document defines HOW the system should be implemented. Data Architects and Business Analysts are often the ones responsible for developing this model. The objective is to create a detailed technical map of the rules and data structures involved.
- **Physical Data Model :** The Data Model that is being presented here specifies HOW the system will be implemented by making use of a certain DBMS. The DBA and developers are the ones who normally construct this model. The actual implementation of the database is the goal of this project.



Advantages of the data model :

- When building a data model, the primary objective is to ensure that the data objects provided by the functional team are appropriately represented.
- It is important that the data model be sufficiently thorough so that it may be used to the construction of the physical database.
- The information contained inside the data model can be utilized to define the relationship between tables, primary and foreign keys, as well as stored procedures, using the information contained within the data model.
- The Data Model facilitates communication both within and between businesses and their respective organizations.
- Data model helps to documents data mappings in ETL procedure
- Assist in identifying the appropriate data sources so that the model can be populated.

Disadvantages of Data Model :

- In order to construct a data model, one must first determine the physical features of the data that is stored.
- This is a navigational system that is responsible for the development and management of complicated applications. Because of this, having knowledge of the biographical truth is required.
- Even very minor alterations to the structure will necessitate modifications to the entire application.
- In DBMS, there is no standard language for the manipulation of data.

10.3 Object Services :

Entity Framework's Object Services service is a service that enables LINQ queries in addition to Entity SQL queries for all of the types that are defined in the Entity Data Model. LINQ is a query language that was developed by Microsoft. The data is sent returned to the data store in the form of objects by the Object Services, and this data includes any modifications that were done to the object. Due to the fact that Object Services enable both LINQ and Entity SQL queries, these services are able to perform data operations such as QUERY, INSERT, UPDATE, and DELETE. LINQ is a query language that is comparable to SQL in its functionality. In addition to these capabilities, this service is able to manage concurrency, bind objects to controls, and retain a record of any modifications that have been made.

- **Classes of Object Services :** The concept of Object service is implemented by classes in the System.Data.Objects and System.Data.Objects.DataClasses namespaces. Some classes that are used with Object Services are as follows :
  - o The ObjectContext Class
  - o The EntityConnection Class
  - o The MetadataWorkspace Class
  - o The ObjectStateManager Class

  The ObjectContext Class : Interacting with data represented as objects, which are instances of entity types described in a conceptual model, is done primarily through the use of the ObjectContext class. This class is the primary class that should be used.

  The EntityConnection Class : A connection to the database, represented by the EntityConnection object in this case.

  The MetadataWorkspace Class : In the form of an object called a MetadataWorkspace, the metadata provides a description of the model.

  The ObjectStateManager Class : An instance of the ObjectStateManager class that is responsible for managing objects that are cached.

When the Entity Data Model tools build the object layer that represents a conceptual model, the class that represents the EntityContainer for the model is derived from the ObjectContext. This ensures that the model always has a consistent look and feel.

- **Features of Object Services :** Object Services offers the following sorts for programming in Entity Framework :

- o Querying data as objects
- o Shaping Query results
- o Adding, changing, and deleting objects
- o Saving changes to the persisted datastore
- o Binding objects to control
- o Managing concurrency
- o Attaching and Detaching Objects
- o Serializing Objects
- o Managing Object identities and tracking changes
- o Manageing Connections
- o Using custom objects with an Entity Data Model (EDM)
- o Managing Transactions

10.4 Entity SQL Language :

A query language that is independent of storage and is quite similar to SQL is called Entity SQL. You are able to query entity data using Entity SQL, and the results can be shown either as objects or in a tabular format. When faced with any of the following scenarios, you should seriously consider utilizing Entity SQL :

- Whenever a query has to be created in a dynamic manner at run time. In this scenario, you should think about using the query builder methods of ObjectQueryT> rather than building an Entity SQL query string at run time. This is something you should consider.
- Whenever you want to include the definition of a query as a part of the model itself. In a data model, support is only provided for Entity SQL. Please refer to the QueryView Element for further details (MSL)
- When an EntityDataReader is used in conjunction with an EntityClient to return read-only entity data in the form of rowsets. Please refer to the EntityClient Provider for the Entity Framework page for any more details.
- If you are already an expert in SQL-based query languages, Entity SQL can feel like the most natural option for you to utilise.

10.5 Entity Client Provider :

Entity Framework programmes make use of a data provider known as the EntityClient provider in order to gain access to data that is outlined in a conceptual model.

EntityClient has access to the data source through the utilisation of many other.NET Framework data sources. When it needs to connect to a SQL Server database, for instance, EntityClient relies on the .NET Framework Data Provider for SQL Server (also known as SqlClient). he EntityClient provider is implemented in the System.Data.EntityClient namespace.

An EntityConnection is a connection that the Entity Framework provides to an underlying data provider and relational database. This connection allows the Entity Framework to build on top of storage-specific ADO.NET data providers. In order to generate an EntityConnection object, you need to make

a reference to a set of metadata that includes the required models and mapping, in addition to a storage-specific data provider name and connection string. This is required before you can begin building the object. Once the EntityConnection has been established, entities can be accessed using the classes that have been developed based on the conceptual model.

In the app.config file, you have the ability to specify a connection string. Additionally, the EntityConnectionStringBuilder class is a part of the System.Data. EntityClient package. By utilizing the class's various attributes and methods, developers are granted the ability to analyse and rebuild previously created connection strings, as well as create syntactically accurate connection strings programmatically using this class.

Entity SQL is a storage-independent variant of the SQL programming language that works directly with conceptual entity schemas and supports Entity Data Model notions like inheritance and relationships. The Entity SQL language is also known as ESQL. When executing an Entity SQL command against an entity model, the EntityCommand class is the one that gets used. You have the option of passing in a query text or the name of a stored procedure when you create new EntityCommand objects. The general Entity SQL queries are converted into storage-specific queries by the Entity Framework through its collaboration with storage-specific data providers.

The .NET Framework provides the following data providers that we can use in our application.

.NET Framework data providers

.NET Framework data provider	Description
.NET Framework Data Provider for SQL Server	It provides data access for Microsoft SQL Server. It requires the System.Data.SqlClient namespace.
.NET Framework Data Provider for OLE DB	It is used to connect with OLE DB. It requires the System.Data.OleDb namespace.
.NET Framework Data Provider for ODBC	It is used to connect to data sources by using ODBC. It requires the System.Data.Odbc namespace.
.NET Framework Data Provider for Oracle	It is used for Oracle data sources. It uses the System.Data.OracleClient namespace.
EntityClient Provider	It provides data access for Entity Data Model applications. It requires the System.Data.EntityClient namespace.
.NET Framework Data Provider for SQL Server Compact 4.0.	It provides data access for Microsoft SQL Server Compact 4.0. It requires the System.Data.SqlServerCe namespace.

.NET Framework Data Providers Objects : The following is a list of the primary objects that make up Data Providers.

Main objects of Data Providers

Object	Description
Connection	It is used to establish a connection to a specific data source.
Command	It is used to execute queries to perform database operations.
DataReader	It is used to read data from data source. The DbDataReader is a base class for all DataReader objects.
DataAdapter	It populates a DataSet and resolves updates with the data source. The base class for all DataAdapter objects is the DbDataAdapter class.

The data provider for SQL Server is a component that doesn't take up much space. Because it connects to SQL Server without going through any intermediate communication layers, the performance is significantly improved. In earlier versions, it had to communicate with the ODBC layer first before connecting to the SQL Server, which resulted in performance problems.

The classes that make up the .NET Framework Data Provider for SQL Server can be found in the System.Data.SqlClient namespace. Using the following code in our C# programme will allow us to incorporate this namespace in our project.

```
using System.Data.SqlClient;
```

This namespace contains the following important classes.

Class	Description
SqlConnection	It is used to create SQL Server connection. This class cannot be inherited.
SqlCommand	It is used to execute database queries. This class cannot be inherited.
SqlDataAdapter	It represents a set of data commands and a database connection that are used to fill the DataSet. This class cannot be inherited.
SqlDataReader	It is used to read rows from a SQL Server database. This class cannot be inherited.
SqlException	This class is used to throw SQL exceptions. It throws an exception when an error is occurred. This class cannot be inherited.

❑ **Check Your Progress – 1 :**

1. _____ is used to read rows from a SQL Server database.

   a. SqlCommand b. SqlDataReader

   c. SqlDataAdapter d. None of these

2. _____ is used to execute database queries. This class cannot be inherited.

   a. SqlException b. SqlConnection c. SqlAdapter d. SqlCommand

10.6 Entity Framework Controls :

The ADO.NET Entity Framework, sometimes known as EF, is an extremely effective object relational mapping (ORM) technology that may be found within Microsoft Visual Studio 2010. In the most recent iteration of ADO.NET Entity

Framework (EF), which provides functionality that is more advanced than that of EF 1.0, a large number of additional capabilities were included. Due to the fact that the version number of EF was modified to correspond with the version of the.NET framework, EF 1.0 is sometimes referred to as version 3.5.

EF uses multiple classes. The all-important ObjectContext is the gatekeeper for all change tracking and database access classes. EF tracks all associated entities. As additional entities are attached to ObjectContext, it can grow in memory and resource, making database persistence difficult.

Each logical series of operations should use a new ObjectContext (such a web form, MVC controller, or client-side view). A single ObjectContext with fewer actions will track fewer objects and preserve performance.

You need to create a new model before you can start working with data. You can do this by either working with an existing database or by creating a model that is completely empty.

Create a new "ADO.NET Entity Data Model" as the first step. Because the data is of the contact type, the name of the model in this particular illustration is "ContactModel.ddmx." The dialogue box labelled Choose Model Contents is brought into view.

Select Generate from database if you want to use an existing database in your work. Just move on to the following step. The dialogue box labelled Choose Your Data Connection will now appear.

Inside of the Visual Studio Server Explorer are all of the databases that you presently have setup, and those databases are found in the drop-down list. If the database you want is not one of the options in the list, you can add a new database by clicking the button labelled "New Connection."

When you look in the field labelled "Entity connection string," you will see a greater amount of information than you would in a standard ADO.NET connection string. There are three components that make up the entity connection string :

- **Metadata :** The real model can be accessed through the metadata. Your solution's model file is referred to as the file ContactModel.edmx is missing from the list that you provided. There are three files to choose from :
  - **ContactModel.csdl :** the conceptual model (or what you see in the entity design surface). This enumerates the classes that you collaborate with.
  - The storage model is located in **ContactModel.ssdl** (or the physical database model).
  - The mapping between the csdl and ssdl files is located in the **ContactModel.msl** file.
- The .emdx file is an XML file that stores all of the information that is contained in the other three files. These three files are generated and incorporated into your project whenever it is compiled.
- Provider: identifies the real ADO.NET provider that's being used.
- The term "Provider Connection String" refers to the standard "ADO.NET Connection String" that contains information that is unique to the provider.

The "Save entity connection settings in App.Config as" field is the final selection available in the dialogue box labelled "Choose your Data Connection." This field is where the connection string is saved. Make sure that the checkbox next to it is selected so that you are not forced to construct the connection string manually. The name that you give to the connection string will also serve as the name that is finally given to the ObjectContext class that you create.

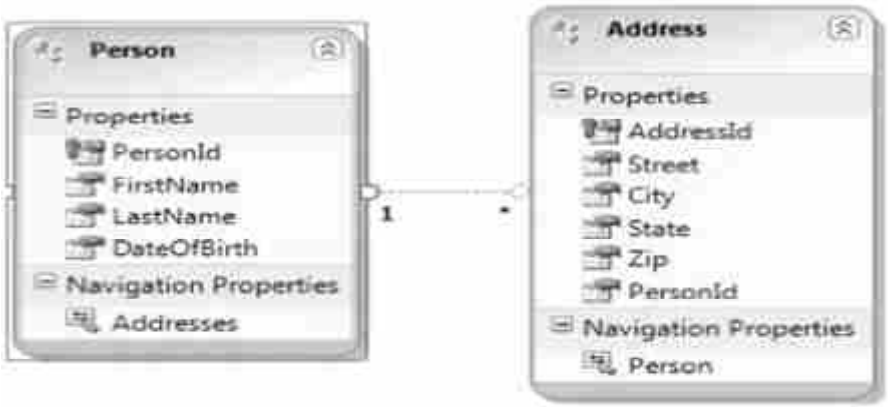
Click the Next button once you have successfully saved your connection string. The dialogue box labelled Choose your Database Objects will now appear.

Choose the tables, views, and stored procedures that you intend to employ in your query. For the purpose of this illustration, we will choose the Addresses and People tables.

You have the option to either pluralize or singularize your object names, as well as include foreign keys in the model, by selecting one of the two checkboxes at the bottom of the dialogue box. When working with entities, pluralizing and singularizing will assist in making the model make more sense from the perspective of the code. In this particular illustration, the name of the database table is "People." You would want the name of the entity to be in the single form, and it should be "Person." If you were going to be interacting with more than one individual, you should change this to "People."

The contents of the Model Namespace field at the bottom of the dialogue box are not visible because they are buried within the code that is created. Because you do not directly interact with it, changing this is not something that is recommended to you.

Simply hit the "Finish" button. You will then be transported to the model design surface, where the model that you have produced will be displayed for your inspection.



It is necessary to retrieve the Person and Address entities from the database. It is important to note that these are the singular forms of the table names. A one-to-many relationship between Person and Address is indicated by the lines linking the entities in the diagram. In other words, a single Person may have many Addresses, but an Address may only have a single Person associated with it. Navigation Properties can also be written in either the plural or the singular depending on the relationship. The navigation characteristics of an entity are the means through which you move between other entities. You can access the address data associated with a person by using the Addresses property of that person, and in the same way, you can access the person associated with an address by using the Person property that is attached to the address.

Visual Studio's properties pane allows you to make any changes to the model that you want to make. You can make these changes to anything that you desire.

You are able to refresh the model whenever you make modifications to your database by right-clicking on the model design surface and selecting the Update Model from Database option from the context menu that appears. A dialogue box that is quite similar to the one that allows you to choose your database objects opens. You will be able to add any items that are not presently

a part of the model, and any objects that are already a part of the model will have their state updated automatically.

To get started, you need to begin by adding a new ADO.NET Data Model to your project. In the Choose Model Contents dialogue box, pick Empty model, and then click the Finish button.

You will then be transported to a design surface that is currently vacant. To add an entity, right-click anywhere on the design surface, then go to Add > Entity from the context menu. The Add Entity dialogue box is brought up at this point.

In this particular illustration, the name of the Entity is specified as "Person," and the Entity Set field displays the plural as "People." The name that you will provide to the real thing is what is referred to as its Entity Name. Both the name of the collection in its plural form and the name of the table in your database will be the Entity Set. You have the option of changing it here if you do not want it to be in the plural form. If, on the other hand, you try to make modifications to the database directly and then update it, you can run into some issues. It is recommended that you stick with one design strategy for your database rather than switching between them. Either use Model First in a consistent manner throughout the entire process, or make changes to your database externally and then update the model. Because of this, ensuring that the model is consistent will be simpler.

❑ **Check Your Progress – 2 :**

1. Which one is the the conceptual model ?

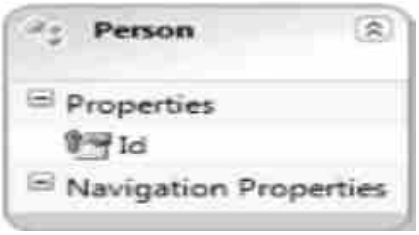
   a. .csdl b. .asad c. .aspx d. None of these

2. Which of the following is physical database model ?

   a. .csdl b. .asap c. .cmd d. .ssdl

If you wish the information that is now in the Property name box to be more descriptive, you can modify it, but it is not required by the design. If you do decide to alter it, make a note that you may in the future need to modify the titles of the associations you belong to.

To return to the model design surface with your newly created entity, click the OK button.



Simply right-click on the entity, and then select the "Add -> Scalar Property" option from the context menu that appears. To add a new property, you can also use the Insert key on your keyboard or the Enter key on most modern keyboards. In this particular illustration, a FirstName attribute was added, as shown here:

Click FirstName to view the Properties window :

After giving life to your entities, you will require a method to link them together. Adding foreign keys to the tables in SQL Server is a very straightforward process. When working with EF, you are required to make associations. To

accomplish this, right-click the object that contains the main key, and then select the Add -> Association option from the context menu. The Add Association dialogue box is brought up at this point.

The Add Association dialogue box lets you set the association type and endpoints on both sides. This example creates a one-to-many Person-Address relationship. Select associations from the Multiplicity drop-down lists. Navigate associations using navigation properties by using the Navigation Property checkboxes. Depending on the association, these are pluralized by default. You can rename these.

To add foreign key attributes to the non-primary key endpoint, check Add foreign key properties to the "Address" Entity. This box adds a PersonId to the address table. The additional key will be the primary key entity name and the primary key. The foreign key field is PersonId if the key is Id. PersonId would be the primary key and PersonPersonId the foreign key.

With a model, you may add data. Creating a person object creates a new person. After creating the object, add it to the context and run SaveChanges. Before storing, the person object needs address data.

```
using(EFRefCardEntities context = new EFRefCardEntities())
{
    Person person = new Person();
    person.FirstName = "Dane";
    person.LastName = "Morgridge";
    person.DateOfBirth = DateTime.Now;
    Address address = new Address();
    address.Street = "123 Somewhere";
    address.City = "Philadelphia";
```

```
    address.State = "PA";
    address.Zip = "12345";
    person.Addresses.Add(address);
    context.People.AddObject(person);
    context.SaveChanges();
}
```

Adding the address to the person object's Addresses collection adds it to the context. When SaveChanges is called, the context will inspect its tracked objects and decide what to do and in what order. Here, two new entities must be presented in order. Person must be added before Address due to a foreign key constraint. The EF arranges them.

SaveChanges is automatically transacted. If anything fails on insert, the whole system rolls back and throws an exception. Right-click the design surface and select Generate Database From Model.

❑ **Check Your Progress – 3 :**

1. It provides data access for Entity Data Model applications.
   a. .NET Framework Data Provider for SQL Server
   b. .NET Framework Data Provider for OLE DB
   c. .NET Framework Data Provider for ODBC
   d. .NET Framework Data Provider for Oracle
2. It is used to connect to data sources by using ODBC.
   a. .NET Framework Data Provider for OLE DB
   b. .NET Framework Data Provider for ODBC
   c. .NET Framework Data Provider for Oracle
   d. EntityClient Provider

Like adding from an existing database, a dialogue box opens to select the database location. Click Next to preview the created SQL file. Finish to create the file. The created SQL file just contains create statements and does not check your database for updates. Run this file against a compare database and sync the schemas with a SQL compare tool. Just add "_Compare" to the database name. This prevents accidental deletion of test data.

The database can retrieve data. LINQ or Entity SQL can query EF. To use the EF, you must know LINQ and its alternatives. Entity SQL, like TSQL, is run against the model instead of the database. Help is available online and in print.

A simple LINQ query returns all person data.

```
using (EFRefCardEntities context = new EFRefCardEntities())
{
    var people = from p in context.People
    select p;
    foreach (var person in people)
    {
```

```
            foreach (var address in person.Addresses)
            {
        }
    }
}
```

To show navigation, foreach through the persons collection and each person's addresses after querying the data. Two queries can be written:

```
var people = from p in context.People
    select p;
```

or

```
var people = context.People;
```

The second way is cleaner and recommended unless you need joins or other special LINQ operations.

Updates are straightforward. You can alter any entity or combination of entities after querying. SaveChanges on the ObjectContext saves entity modifications.

```
using(EFRefCardEntities context = new EFRefCardEntities())
{
        var people = context.People.Include("Addresses");
        foreach (var person in people)
        {
            person.DateOfBirth = DateTime.Now.AddYears(-30);
        }
            context.SaveChanges();
}
```

The above code updates the DateOfBirth to the current time minus 30 years, and the SaveChanges method persists the changes to the database. SaveChanges evaluates any entity change tracked by the ObjectContext. Use a single ObjectContext for a narrow logical set of operations because of this. It takes longer and uses more resources to assess more things.

To delete an entity, call DeleteObject on the ObjectSet:

```
context.People.DeleteObject(person);
context.SaveChanges();
```

Load the ObjectContext to delete an object. SaveChanges will erase the database. If you need to delete several objects, utilize a stored method because the ObjectContext must track them. Using a stored procedure is easier than the EF.

Delete IEnumerable or Ienumberable<T> entities carefully. Modifying the collection while enumerating will throw an error. Arraying your collection solves this.

10.7 Let Us Sum Up :

Throughout this unit, we have gained an understanding of the Data Modeling idea. We have learned about the several Services provided by objects as well as the Entities that are available through the SQL language.

Entity Client Providers and Entity Framework Controls are two topics that have been covered in this unit. LINQ Queries are something that we have gained knowledge of.

10.8 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : b **2 :** d

- **Check Your Progress 2 :**

  1 : a **2 :** d

- **Check Your Progress 3 :**

  1 : c **2 :** d)

10.9 Glossary :

1. **DataAdapter :** The DataAdapter serves as a bridge between a DataSet and a data source for retrieving and saving data.
2. **DataConnection :** Data Connection means any connection and/or communication between Devices by which data is either transmitted and/or received.
3. **ConnectionString :** a string that specifies information about a data source and the means of connecting to it.
4. **DataReader :** A DataReader parses a Tabular Data Stream from Microsoft SQL Server, and other methods of retrieving data from other sources.
5. **ExecuteNonQuery :** Enables you to execute SQL statements that INSERT, UPDATE, or DELETE data in a given data source.

10.10 Assignment :

1. Explain various .NET Framework Data Providers Objects.

10.11 Activities :

1. Explain concept of Data Modeling in detail with suitable example.

10.12 Case Study :

Study Entity Framework Control

10.12 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

BLOCK SUMMARY :

In this block, you have learnt and understand about various webserver controls which have strong capability of data binding that ASP.NET makes available. We have learnt about how do we used to generate DropDownList for online applications is provided by ASP.NET.

We have learnt the option to pick many items at the same time by making use of the ASP.NET CheckBoxList web control, which is a web control that can be used to collate the items that can be checked. We have learnt how do we use the functions for data binding, you are able to build this list of things that are contained in the CheckBoxList in a dynamic manner. We have learnt about the RadioButtonList Control functions in the same way as the DropDownList Control, except instead of displaying a drop-down menu, it showed a list of radio buttons that can be arranged in a horizontal or vertical fashion.

We have learnt about ImageField class that how it is utilised by data-bound controls such as the GridView and the DetailsView in order to display an image in conjunction with each record that is shown.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Write a short note on SQL WildCards.
2. Explain various Query Operators.
3. Write a note on LINQ to objects.
4. Write short note on Entity Client Provider ?

❖ **Long Questions :**

1. What are the benefits of Populating the List Dynamically
2. Explain functionality of LINQ to ADO.NET in detail.
3. Explain various databinding controls in detail ?

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	8	9	10
No. of Hrs.			

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....

Dr. Babasaheb Ambedkar Open University Ahmedabad

BCAR-504

INTERNET PROGRAMMING (ASP.NET USING C#)

BLOCK 4 : WORKING WITH USER

WORKING WITH USER

Block Introduction :

In most cases, only users who have successfully authenticated themselves will have access to restricted information on websites. It is standard practise for websites to provide a mechanism for users to log in and be authenticated, while also concealing information from users who browse the site anonymously. This approach is typical. Before we begin assigning responsibilities to users, we first determine which users already have certain roles and then check to see if any new users should be added. There are two distinct methods available for displaying information in this context: by role or by user. If the user functions with a small and defined list of users, the strategy of using administrative tools to register users might be advantageous. This is because it allows users to register themselves. You are now able to carry out precisely the same process that you accomplished earlier with the assistance of the Web Site Administration tool by utilising the Create User Wizard control, which is a component of ASP.NET.

In this section, we will delve deeper into the principles of the Web Site Administration tool as well as its operation to provide you with a better understanding of both topics. The limitation will focus on a wide range of applications of the ASP.NET concept and will require user websites to adhere to a predetermined set of criteria. You will acquire an understanding of logging into and out of websites while utilising the authenticated tools that will be offered to you, and you will obtain this understanding while working with these tools.

While working through this section, you will have the opportunity to acquire and gain an understanding of the fundamentals of password protection as well as the numerous actions that can be taken in order to recover a lost password. You will also have the chance to gain an understanding of the numerous actions that can be taken in order to recover a lost password. In addition to the process of registering as a user on the website, you will also be given the opportunity to select and create a new password for your account at that time. You will be provided with a comprehensive and hands-on demonstration about user websites.

Block Objectives :

After learning this block, you will be able to understand :

- About how to create a registration form
- About how to create a login form
- About how to recover User forgotten password
- About the Authorization and Authentication
- About the rules of accessing Webpages
- About the various themes of Master Pages

Block Structure :

Unit
11

WORKING WITH USER LOGINS

UNIT STRUCTURE

11.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About User Accounts and Membeeship
- About various Roles for users
- About access rules
- About creating new user accounts with the help of websites
- About the properties of email and Autogenerate Password
- About Website Login Control
- About all login actions performed by the websites

11.1 Introduction :

The ASP.NET login controls offer a dependable login method for ASP.NET Web applications, and they do so without requiring the developer to write any code. This makes the ASP.NET login controls extremely convenient. The ASP.NET membership and form authentication are immediately integrated with

the login controls. This behavior is the default. When it comes to the process of automating the user authentication that takes place on a website, this integration can prove to be of some aid.

Plain text communication is the method of communication that is utilized by the ASP.NET login controls and HTTP when it comes to exchanging information with one another. If the safety of your data is a concern of yours, you should transfer it using HTTPS, which includes the encryption that is given by SSL.

11.2 User Accounts, Membership :

The application services schema within a database, which added the tables, views, and stored procedures required by the SqlMembershipProvider and the SqlRoleProvider respectively. Because of this, the infrastructure has been developed that will be necessary for the remaining tutorials in this series. In this article, we will investigate how to establish new user accounts by utilising the Membership framework (by means of the SqlMembershipProvider). In this lesson, you will learn how to create new users both programmatically and by using the CreateUserWizard control that is integrated into ASP.NET.

In addition to learning how to create new user accounts, we will also need to learn how to update the demo website that we initially developed in the tutorial titled An Overview of Forms Authentication and which we later improved in the tutorial titled Forms Authentication Configuration and Advanced Topics. Users' credentials are checked against a list of predefined username and password combinations on the login page of our demonstration web application. In addition, the file Global.asax contains code that, for authenticated users, generates individualised instances of the objects IPrincipal and IIdentity. We are going to modify the login page so that it checks the credentials of users against the Membership framework, and then we are going to get rid of the custom principal and identity logic.

The Forms Authentication and Membership Checklist : Before integrating Membership functionality in your web application, you will first need to complete the following steps when using the Membership framework in conjunction with the SqlMembershipProvider in a forms-based authentication scenario :

- **Enable forms-based authentication.** Forms authentication is enabled by editing Web.config and setting the <authentication> element's mode attribute to Forms. With forms authentication enabled, each incoming request is examined for a forms authentication ticket, which, if present, identifies the requestor.
- **Add the application services schema to the appropriate database.** When using the SqlMembershipProvider we need to install the application services schema to a database. Usually this schema is added to the same database that holds the application's data model.
- **Customize the Web Application's Settings to reference the database from previous step.** There are two ways to configure the web application so that the SqlMembershipProvider would use the database selected in the previous step: by modifying the LocalSqlServer connection string name; or by adding a new registered provider to the list of Membership framework providers and customizing that new provider to use the database from previous step.

These three steps need to be completed before you can use the Membership class or the ASP.NET Login Web controls when developing a web application that makes use of the SqlMembershipProvider and forms-based authentication. This is the case even if you use forms-based authentication.

Example : Let's add new ASP.NET Pages to examine several Membership-related functions and capabilities. After that we will create a web.sitemap file.

Start by creating a new folder in the project named Membership. Next, add five new ASP.NET pages to the Membership folder, linking each page with the Site.master master page. Name the webpages will be: CreatingUser Accounts.aspx, UserBasedAuthorization.aspx, EnhancedCreateUserWizard.aspx, AdditionalUserInfo.aspx and Guestbook.aspx.

Now if you will see on the Solution Explorer, it will look like this :

At this time, both of these Content controls should be present on every page, one for each of the master page's ContentPlaceHolders: MainContent and LoginContent.

<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" Runat="Server">

</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="LoginContent" Runat="Server">

</asp:Content>

Depending on the user's authentication status, the default markup for the LoginContent ContentPlaceHolder displays either a link to log on to the site or a link to log off of the site. However, if you have the Content2 Content control present, it will take precedence over the markup that the master page normally uses.

On the other hand, we want to display the master page's default markup for the LoginContent ContentPlaceHolder on these five individual pages. Consequently, the declarative markup for the Content2 Content control should be removed. After you have finished doing so, the markup for each of the five pages should only have a single Content control.

In the same way that the Membership and Roles frameworks are built on top of the provider model, the Site Map framework is as well. The Site Map provider class is responsible for generating the in-memory structure that the SiteMap class uses from a persistent data store such as an XML file or a database table. This is the job that the Site Map provider class is supposed to do. The data for the site map is read from an XML file by the default Site Map provider that is packaged with the.NET Framework (XmlSiteMapProvider).

The default Site Map provider anticipates that the root directory will have an XML file with the name Web.sitemap that is correctly formatted. Because we are utilizing this default provider, we need to incorporate such a file and establish the structure of the site map utilizing the XML format that is acceptable. In the Solution Explorer, right-click on the project name, and then select the Add New Item option from the context menu. Choose to add a file of type Site Map with the name Web.sitemap from the corresponding dialogue box.

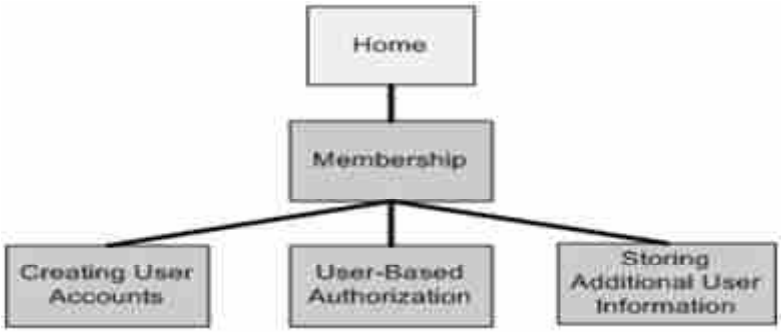
The website's structure is presented in the form of a hierarchy in the XML site map file. This hierarchical relationship is depicted in the XML file through the ancestry of the elements belonging to the siteMapNode node type. The Web.sitemap absolutely needs to begin with a <siteMap> parent node that is only going to have one single <siteMapNode> child. This <siteMapNode> element at the top-level represents the base of the hierarchy and can have any number of child nodes beneath it. Each <siteMapNode> element needs to have a title attribute, and it also has the option of adding 'url' and 'description' attributes, among others; the 'url' attribute of each non-empty siteMapNode element needs to be unique.

Copy and paste the XML code shown below into the Web.sitemap file :

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
    <siteMapNode url="~/Default.aspx" title="Home">
        <siteMapNode title="Membership">
            <siteMapNode url="~/Membership/CreatingUserAccounts.aspx"
title="Creating User Accounts" />
            <siteMapNode url="~/Membership/UserBasedAuthorization.
aspx" title="User-Based Authorization" />
            <siteMapNode url="~/Membership/Guestbook.aspx" title=
"Storing Additional User Information" />
```

```
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

It displays such type of hierarchy :



In order to generate this user interface, include the following declarative markup in the left column of the Site.master master page, which is currently occupied by the text "TODO: Menu will go here..."

```
<ul>
  <li>
<asp:HyperLink runat="server" ID="lnkHome" NavigateUrl="~/
Default.aspx"> Home </asp:HyperLink>
  </li>
  <asp:Repeater runat="server" ID="menu" DataSourceID="Site
  MapDataSource1">
    <ItemTemplate>
      <li>
        <asp:HyperLink ID="lnkMenuItem" runat="server"
          NavigateUrl='<%# Eval("Url") %>'><%# Eval("Title")
          %></asp:HyperLink>
        <asp:Repeater ID="submenu" runat="server" DataSource=
        "<%#
          ((SiteMapNode) Container.DataItem).ChildNodes %>">
          <HeaderTemplate>
            <ul>
          </HeaderTemplate>
          <ItemTemplate>
            <li>
              <asp:HyperLink ID="lnkMenuItem" runat="server"
              NavigateUrl='<%#
                Eval("Url") %>'><%# Eval("Title") %></asp:
                HyperLink>
            </li>
          </ItemTemplate>
          <FooterTemplate>
```

```
                    </ul>
                </FooterTemplate>
            </asp:Repeater>
        </li>
    </ItemTemplate>
</asp:Repeater>
</ul>
<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" Show
StartingNode= "false" />
```

In the preceding markup, a Repeater control with the name menu is bound to a SiteMapDataSource. This data source delivers the site map hierarchy that is specified in Web.sitemap. Due to the fact that the ShowStartingNode property of the SiteMapDataSource control is set to False, the control begins returning the site map's hierarchy with the nodes that are descendants of the "Home" node. Each of these nodes, which are now limited to "Membership," is displayed by the Repeater within a li element. After that, an additional, inner Repeater displays the children of the current node in a nested list that is not sorted.

Let's add a breadcrumb trail to each page, in addition to the list of links that is now located in the left column. A breadcrumb is a navigational user interface element that displays the user's current position within the site hierarchy in a short and easy to understand format. The SiteMapPath control makes use of the Site Map framework to ascertain the location of the current page in the site map. Once this information has been gathered, the control shows a breadcrumb trail based on its findings.

❑ **Check Your Progress – 1 :**

1. Forms authentication is enabled by editing _____.

   a. Global.asax b. Default.aspx c. Web.config d. None of these

2. The website's structure is presented in the form of a hierarchy in the _____.

   a. HTML site map file b. XML site map file

   c. XXHTML site map file d. None of these

To be more specific, add a element to the header element of the master page, and make sure the class attribute of the new span> element is set to "breadcrumb." (The "breadcrumb" class has its own rule, which may

be found in the Styles.css file.) After that, a SiteMapPath should be added to this brand-new element.

```
<div id="header">
    <span class="title">User Account Tutorials</span><br />
    <span class="breadcrumb">
        <asp:SiteMapPath ID="SiteMapPath1" runat="server"></asp:Site
        MapPath>
    </span>
</div>
```

The Breadcrumb Displays the Current Page and its Ancestors in the Site Map like this :

Home > Membership > Creating User Accounts

The authenticated user has the ability to have custom principal and identity objects associated with them. We were able to accomplish this by adding a handler to the PostAuthenticateRequest event of the application, which fires after the FormsAuthenticationModule has authenticated the user. This handler was placed in the Global.asax file. In this event handler, we substituted the CustomPrincipal and CustomIdentity objects for the GenericPrincipal and FormsIdentity objects that were added by the FormsAuthenticationModule.

Utilize the CreateUser method of the Membership class in order to establish a new user account within the Membership framework. This method accepts input from the user for the username, password, and any other user-related fields that may be present. After being called, it hands off the task of creating the new user account to the Membership provider that has been specified, and it then returns a MembershipUser object that represents the user account that was just made.

There are four possible overloads of the CreateUser method, and each one accepts a different number of input parameters :

- CreateUser(username, password)
- CreateUser(username, password, email)
- CreateUser(username, password, email, passwordQuestion, password Answer, isApproved, MembershipCreateStatus)
- CreateUser(username, password, email, passwordQuestion, password Answer, isApproved, providerUserKey, MembershipCreateStatus)

These four overloads each collect a different quantity of information, which is a key distinction between them. In the first overload, for example, all that is needed to create a new user account is the user's username and password, whereas in the second overload, it is also necessary to have the user's email address.

The reason for the existence of these overloads is that the information required to create a new user account is dependent on the configuration settings of the Membership provider.

The requiresQuestionAndAnswer setting is an example of one of these Membership provider configuration options that has an effect on the CreateUser overloads that can be used. When a new user account is created, a security question and answer must be provided if the requiresQuestionAndAnswer

property is configured to return true (this is the default behaviour). In the event that the user later has to reset or change their password, this information will be used. To be more specific, at that point in time, they are presented with the security question, and in order to reset or change their password, they are required to enter the proper response. Since of this, when either of the first two CreateUser overloads are called with the requiresQuestionAndAnswer parameter set to true, an error is thrown because the security question and answer are not present. When it comes to generating users programmatically, we will have no choice but to utilise one of the latter two overloads because the way our application is now set up, users are required to provide a security question and answer.

Let's develop a user interface together so that we can demonstrate how to utilise the CreateUser method. On this user interface, we will ask the user for their name, password, email address, and the answer to a security question that has been predefined. In the Membership folder, open the CreatingUserAccounts.aspx page and add the following Web controls to the Content control:

- A TextBox named Username
- A TextBox named Password, whose TextMode property is set to Password
- A TextBox named Email
- A Label named SecurityQuestion with its Text property cleared out
- A TextBox named SecurityAnswer
- A Button named CreateAccountButton whose Text property is set to "Create the User Account"
- A Label control named CreateAccountResults with its Text property cleared out

Add the Various Web Controls to the CreatingUserAccounts.aspx page and the screen will look like this after designing :

The SecurityQuestion Label and SecurityAnswer TextBox display and collect a pre-defined security question. Since the security question and answer are stored per user, each user can define their own. I chose a universal security question, "What is your favourite color?"

Add a passwordQuestion constant to the page's code-behind class and assign the security question. In the Page Load event handler, assign this constant to SecurityQuestion Label's Text property :

```
const string passwordQuestion = "What is your favorite color";
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
            SecurityQuestion.Text = passwordQuestion;
    }
```

Next, create an event handler for the CreateAccountButton's Click event and add the following code :

```
    protected void CreateAccountButton_Click(object sender, EventArgs e)
    {
        MembershipCreateStatus createStatus;
    MembershipUser newUser = Membership.CreateUser(Username.Text,
    Password.Text, Email.Text, passwordQuestion, SecurityAnswer.Text, true,
    out createStatus);
        switch (createStatus)
        {
            case MembershipCreateStatus.Success:
                CreateAccountResults.Text = "The user account was successfully
                created!";
                break;
            case MembershipCreateStatus.DuplicateUserName:
                CreateAccountResults.Text = "There already exists a user with
                this username.";
                break;
            case MembershipCreateStatus.DuplicateEmail:
                CreateAccountResults.Text = "There already exists a user with
                this email address.";
                break;
            case MembershipCreateStatus.InvalidEmail:
                CreateAccountResults.Text = "There email address you provided
                in invalid.";
                break;
            case MembershipCreateStatus.InvalidAnswer:
                CreateAccountResults.Text = "There security answer was
                invalid.";
                break;
            case MembershipCreateStatus.InvalidPassword:
```

```
                CreateAccountResults.Text = "The password you provided is
                invalid. It must be seven characters long and have at least one
                non-alphanumeric character.";
                break;
            default:
                CreateAccountResults.Text = "There was an unknown error; the
                user account was NOT created.";
                break;
            }
        }
```

Create a MembershipCreateStatus variable in the Click event handler. MembershipCreateStatus enumerates CreateUser status. The MembershipCreate Status instance will be set to Success if the user account is established successfully, or DuplicateUserName if the action fails due to a duplicate username. The CreateUser overload requires a MembershipCreateStatus instance as an out parameter. The CreateUser method sets this parameter to the right value, so we can check its value thereafter to see if the user account was created.

A switch statement outputs a message based on createStatus after running CreateUser. The above screen shows the output when a new user has successfully been created.

The above screenshot shows the output when the user account is not created successfully as the supplied password is Too Weak. The visitor entered a five-letter password, which does not meet the password strength requirements spelled out in the Membership provider's configuration settings.

The above screenshot shows the output when the user account is not created successfully as the visitor is attempting to create a user account with an existing username. The User Account is Not Created Because the Username is Already in Use.

Verify that the user accounts have been created after creating a few user accounts by listing the contents of the aspnet_Users and aspnet_Membership tables in the SecurityT.mdf database. You can do this after establishing a few user accounts. Tito and Bruce are the two users that have been added as a result of my use of the CreatingUserAccounts.aspx page as shown in the above figure.

Despite the fact that Bruce and Tito's account information can now be found in the Membership user store, we have not yet implemented the feature that will enable either of them to log on to the website. At the moment, Login.aspx verifies the user's credentials by comparing them to a predefined collection of username/password combinations; however, it does not verify the supplied credentials by comparing them to the Membership framework.

11.3 Putting Users into Various Roles :

Before we start giving users their roles, we check to see which users already belong to which roles. There are two ways to provide this information by role or by user.

In this particular scenario, the visitor can select a role, after which they will be shown all of the users who belong to that role. Alternatively, the visitor can be prompted to select a user, after which they will see all of the roles that have been assigned to "by user."

The "by role" view is vital in situations in which the visitor needs information about a specific set of users who belong to a particular role. The "by user" view is excellent in situations in which the visitor needs information about a specific user's roles.

Now develop an interface that is "by user," complete with a drop-down menu and checkboxes. The set of users that are now logged in to the system will be framed within the drop-down list, and the checkboxes will enumerate the roles. When a person is chosen from the drop-down list, the roles to which that user belongs are checked off. The user who is now viewing the page has the option to either check or uncheck the checkboxes in order to include or exclude the chosen user from the respective roles.

Developing a User Interface Driven "By User": Launch the UsersAndRoles.aspx page in your browser. Add a Label Web control at the top of the page and call it ActionStatus. The Text property of this control should be cleared out. Label can be used to provide feedback on activities that have been performed, displaying messages such as "User Tito has been added to the Administrators role," or "User Jisun has been removed from the Supervisors role." The "Roles" value should be used for the CssClass attribute of the Label in order to draw attention to these messages.

```
<p align="center">
<asp:Label ID="ActionStatus" runat="server" CssClass="Roles"></asp:
Label>
</p>
```

Add this in the CSS file:

```
.Roles
{
font-size: large;
color: Red;
}
```

Include a DropDownList on the page and ensure that its ID property is set to UserList. Additionally, ensure that the AutoPostBack property is set to True. Utilize DropDownList to compile a list of all of the users in the system, which will result in the collection of MembershipUser objects. Since we want DropDownList to display the UserName property of MembershipUser objects, we will also set the DataTextField and DataValueField properties of DropDownList to "UserName." Lastly, we will set DropDownList's DataSource property to "UserName."

Add a Repeater with the name UsersRoleList beneath the DropDownList. This Repeater will list all of the roles that are currently available in the system as a series of checkboxes. Declarative markup such as the following should be used to define the Repeater's ItemTemplate :

```
<asp:Repeater ID="UsersRoleList" runat="server">
<ItemTemplate>
<asp:CheckBox runat="server" ID="RoleCheckBox" AutoPostBack="true"
Text='<%# Container.DataItem %>' />
```


</ItemTemplate>

</asp:Repeater>

The ItemTemplate markup has a single CheckBox Web control that has been given the name RoleCheckBox. It has been determined that the AutoPostBack property of the CheckBox should be set to True, and the Text property should be tied to Container. DataItem. Because of this, the databinding syntax is referred to as Container. This string array is what we will be tying to the Repeater, therefore DataItem is necessary because the Roles framework returns the list of role names as a string array.

11.4 Access Rules :

This topic demonstrates how to define security permissions for folders and files by making use of a set of access rules for folders and files (the FileManagerSettingsPermissions .AccessRules collection). You are granted the ability to exercise control over the rule priority whenever you index rules within a collection. A higher index confers a higher priority on the corresponding regulation.

Set access permissions for individual files and folders by utilizing the FileManagerFolderAccessRule and FileManagerFileAccessRule elements, respectively. Folders can have access rules applied to their files and subfolders as well. The access rule properties that are accessible for files and folders are outlined in the table that can be found below.

Rule Properties	File Access Rule	Folder Access Rule	Description
Path	+	+	A path to which the rule is applied
Role	+	+	A role to which the rule is applied
Browse	+	+	Permission to view a file/folder
Download	+	-	Permission to download a file
Edit	+	+	Permission to edit files/folders
Edit Content	-	+	Permission to edit folder content (files within a folder)
Upload	-	+	Permission to upload files to a folder

Permissions are determined based on the Rights enumeration values listed below :

	Values	Description
Allow	Rights.Allow	The action is allowed within the access rule.
Deny	Rights.Deny	The action is denied within the access rule.
Default	Rights.Default	The action has the same permission as the current item (file or folder) parent element. It corresponds to the Rights.Allow permission if this value does not exist.

Example : The following are the steps to take to prevent users from changing any files other than JPG files :

```
<dx:ASPxFileManager ID="ASPxFileManager1" runat="server">
  <SettingsEditing AllowCreate="true" AllowDelete="true" AllowMove
="true" AllowRename="true" />
  <SettingsPermissions>
    <AccessRules>
      <dx:FileManagerFileAccessRule Edit="Deny" Path="*" />
      <dx:FileManagerFileAccessRule Edit="Allow" Path="*.jpg" />
    </AccessRules>
  </SettingsPermissions>
</dx:ASPxFileManager>
```

The following is a guide on how to stop users from exploring the 'Admin' folder :

```
<dx:ASPxFileManager ID="ASPxFileManager1" runat="server">
  <SettingsPermissions>
    <AccessRules>
      <dx:FileManagerFolderAccessRule Browse="Deny" Path=
      "Admin" />
    </AccessRules>
  </SettingsPermissions>
</dx:ASPxFileManager>
```

The following steps will show you how to stop users from altering the ReadOnly folder :

```
<dx:ASPxFileManager ID="ASPxFileManager1" runat="server">
  <SettingsEditing AllowCreate="true" AllowDelete="true"
AllowMove="true" AllowRename="true" />
  <SettingsPermissions>
    <AccessRules>
      <dx:FileManagerFolderAccessRule Edit="Deny" Path="Read
      Only" />
    </AccessRules>
  </SettingsPermissions>
</dx:ASPxFileManager>
```

The following steps will demonstrate how to stop users from uploading any folders other than the "UploadFolder" folder :

```
<dx:ASPxFileManager ID="ASPxFileManager1" runat="server">
  <SettingsPermissions>
    <AccessRules>
      <dx:FileManagerFolderAccessRule Upload="Deny" Path="" />
      <dx:FileManagerFolderAccessRule Upload="Allow" Path=
      "UploadFolder" />
```

```
        </AccessRules>
    </SettingsPermissions>
</dx:ASPxFileManager>
```

The following steps will restrict users who are not admins from altering files :

```
<dx:ASPxFileManager ID="ASPxFileManager1" runat="server">
<SettingsPermissions>
<AccessRules>
<dx:FileManagerFolderAccessRule Edit="Deny" Path="" />
<dx:FileManagerFolderAccessRule Edit="Allow" Path="" Role="Admin" />
</AccessRules>
</SettingsPermissions>
</dx:ASPxFileManager>
```

C# Source Code :

```
ASPxFileManager1.SettingsPermissions.Role = User.IsAdmin ? "Admin"
: string.Empty;
```

❑ **Check Your Progress – 2 :**

1. The action is denied within the access rule is _____.

   a. Allow b. Deny c. Declare d. None of these

2. The action has the same permission as the current item (file or folder) parent element is _____.

   a. Deny b. Allow c. Default d. None of these

11.5 Creating New User Accounts Via Websites :

You initiated the creation of a user with the help of the Web Site Administration tool in the previous section of this article. This strategy is excellent if you are operating with a small group of users who have well defined roles; nevertheless, on the majority of websites, users are permitted to sign up for accounts on their own. The CreateUserWizard control that is a part of ASP.NET carries out the same task as the Web Site Administration tool that you used earlier.

Let we demonstrate how to incorporate a feature into your website that gives visitors the ability to sign up for an account there. To get started, you will need to develop a page for people to register.

To Create a Page for Online Registration :

- In Solution Explorer, right-click the name of your website (http://localhost/membership), select the Add New Item menu option, and then specify the name of the new web form you want to create as Register.aspx.
- In the Register.aspx page, navigate to the View menu and select Design. Then, insert static text such as "Register" into the page. Format the text as Heading 1 by selecting Block Format from the drop-down list in the Formatting toolbar and applying the appropriate settings.

- To create a new user account, drag a CreateUserWizard control from the Login group of the Toolbox onto the page, as demonstrated in the accompanying figure :

- In the Properties window for the CreateUserWizard control, the ContinueDestinationPageUrlproperty to ~/Default.aspx.
- This configures the control so that when users click Continue after creating a user, the control returns to the home page.
- From the Standard group of the Toolbox, drag a HyperLink control onto the page. In the Properties window for the HyperLink control, set the Text property to Home and the NavigateUrl property to~/Default.aspx.
- Now I'll show you how to add a link to the home page that displays the registration page. For this, assume that you want to display the registration link only to users who are not logged in (Guest Users).

To Create a Registration Link on the Home Page :

- Open the Default.aspx page.
- Use your right mouse button to click on the LoginView control that was just added. Choose to Display the Smart Tag. To begin editing in the anonymous template, go to the LoginView Tasks panel and make a selection from the Views list box to activate editing in the anonymous template.
- Simply dragging a HyperLink control from the Standard group in the Toolbox into the anonymous template will do the trick. The Text property of the HyperLink control should be set to Register, and the NavigateUrl property should be set to ~/Register.aspx. Both of these settings can be found in the Properties window for the HyperLink control. Only users who are not currently logged in will see the Register option in their navigation menu.

To Test Registration :

- To run the website and display the Default.aspx page, press CTRL + F5 on your keyboard. Since you are not currently logged in, the page that allows you to register for an account is shown to you.
- To register, click the link provided. There is a representation of the registration page.
- As shown in the following figure, enter a new user name, a password, an e-mail address, and a security question and answer into the respective text boxes :

- Make sure the Create User button is selected. The user is presented with a confirmation message.
- To proceed, please click the Continue button. You have successfully logged in and have been taken to the homepage of the site. Take note that the link that used to read "Login" now reads "Logout," and also take note that the information that is displayed in the Login control comes from the LoggedInTemplate property and not the AnonymousTemplate property.
- Select the Logout link from the menu. When a visitor visits the page in anonymous mode, the page will display different information.
- To log in, click the appropriate link.
- Give the user account you just made some credentials by entering them here. You have successfully entered the system as a new user.
- Please close the window for the browser.

11.6 Autogeneratepassword & Required email Properties :

In this section, we will go over the steps necessary to generate random passwords using ASP.NET. The Random class in the.NET Framework can be used to create random integers for use in various applications. In order to produce random numbers, you first need to construct an instance of an object of the Random class. You are required to produce a website and add an additional page to the existing website. On the form, place three controls using the drag-and-drop method: two TextBoxes and one Button. One TextBox will display the length of the TextBox, while the second TextBox will display a random password of the length that was set when the Button control is clicked.

We will need to begin by developing a website.

- Go to Visual Studio 2010
- New-> Select a website application
- Click OK

Now add a new page to the website.

- Go to the Solution Explorer
- Right Click on the Project name
- Select add new item
- Add new web page and give it a name
- Click OK

Now, make a new website and use the drag-and-drop method to place two TextBox controls on the ASPX page. This is how the code for the TextBox looks:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="random number.aspx.cs" Inherits="randomnumber" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

    <div>

    Password Length:    <asp:TextBox ID="txtPassLength" runat="server"> </asp:TextBox>

    <br />

    Random Password: <asp:TextBox ID="txtpassword" runat="server"></asp:TextBox>

    <br />

    <br />

    <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Generate Password" />

    <br />

    </div>

    </form>

</body>

</html>
```

A Random class is available for use in ASP.Net, which allows for the generation of random numbers. Generate random numbers by first establishing an instance of an object belonging to the Random class. The following line of code can be used to create an instance of this class :

Random rand = new Random();

The variable rand is an instance of the Random Class in the preceding. Now, double-click the Button control, and add the code shown in the following table :

C# Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class randomnumber : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        string allowedChars = "";
        allowedChars = "a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,";
        allowedChars += "A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,
        W,X,Y,Z,";
        allowedChars += "1,2,3,4,5,6,7,8,9,0,!,@,#,$,%,&,?";
        char[] sep = { ',' };
        string[] arr = allowedChars.Split(sep);
        string passwordString = "";
        string temp = "";
        Random rand = new Random();
        for (int i = 0; i < Convert.ToInt32(txtPassLength.Text); i++)
        {
            temp = arr[rand.Next(0, arr.Length)];
            passwordString += temp;
        }
        txtpassword.Text = passwordString;
    }
}
```

Now you can test the programme by starting it up.

Now please tell me how long the password is :

Now, you can generate a password at random by clicking on the button.

The output seen above presents a password chosen at random.

Using ASP.NET and C# to send an email is covered in this unit's topic, which includes an explanation of several significant methods as well as notable errors.

Example : C# Source Code

```
using System.IO;
using System.Net;
using System.Net.Mail;
string to = "toaddress@gmail.com"; //To address
string from = "fromaddress@gmail.com"; //From address
MailMessage message = new MailMessage(from, to);
string mailbody = "In this unit we will learn how to send a email using Asp.Net & C#";
message.Subject = "Sending Email Using Asp.Net & C#";
message.Body = mailbody;
message.BodyEncoding = Encoding.UTF8;
message.IsBodyHtml = true;
SmtpClient client = new SmtpClient("smtp.gmail.com", 587); //Gmail smtp
System.Net.NetworkCredential basicCredential1 = new
System.Net.NetworkCredential("yourmail id", "Password");
client.EnableSsl = true;
client.UseDefaultCredentials = false;
client.Credentials = basicCredential1;
try
{
   client.Send(message);
}
catch (Exception ex)
{
   throw ex;
}
```

The following namespace is required to be added:

```
using System.Net;
using System.Net.Mail;
```

Simple Mail Transfer Protocol (SMTP) : A TCP/IP protocol known as Simple Mail Transfer Protocol (SMTP) is utilized in the process of sending and receiving electronic mail. SMTP is used by the vast majority of e-mail systems that are connected to the Internet to transfer messages from one server to another. After then, the mails can be retrieved from the server by using an email client using either POP or IMAP.

Here is a list of SMTP servers, along with their respective port numbers :

Sl.No	Mail Server	SMTP Server(Host)	Port Number
1	Gmail	smtp.gmail.com	587
2	Outlook	smtp.live.com	587
3	Yahoo Mail	smtp.mail.yahoo.com	465
4	Yahoo Mail Plus	plus.smtp.mail.yahoo.com	465
5	Hotmail	smtp.live.com	465
6	Office365.com	smtp.office365.com	587
7	zoho Mail	smtp.zoho.com	465

Properties of SMTP Class

Properties	Description
Host	Server URL for SMTP
EnableSsl	Check your host accepts SSL Connections (True or False).
Port	Port Number of the SMTP server
Credentials	Valid login credentials for the SMTP server (the email address and password).
UseDefaultCredentials	When we change the value of the UseDefault Credentials setting to True, we are telling the programme that we want to enable authentication that is based on the credentials of the account that is being used to send emails.

11.7 Logging to The Website Login Control :

The ASP.NET login controls offer a powerful login solution for ASP.NET Web applications that does not require any scripting on the developer's part. Login controls integrate with ASP.NET membership and forms authentication by default. This integration can assist in automating the user authentication process for a website. It gives you a completely functional user interface that asks the user for their username and password and provides a button labelled "Log In" so that you may access the system with those credentials. It verifies the user credentials against the membership API and encapsulates the basic form authentication functionality, such as redirecting back to the page that was originally requested in a restricted section of your application after a successful login.

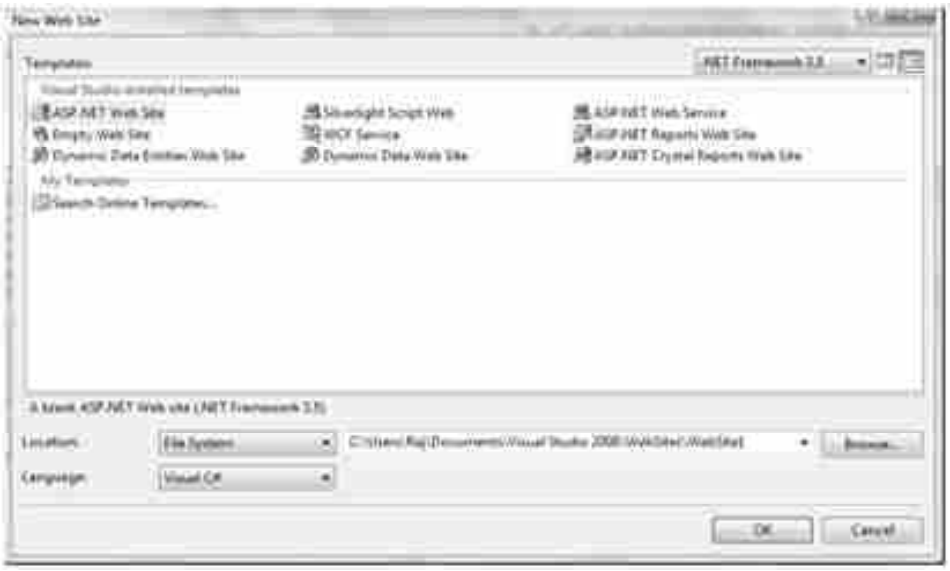
A user interface for user authentication is displayed via the Login control in your application. The Login control has text boxes for the user name and password, as well as a check box that allows users to indicate whether they want the server to store their identity using ASP.NET membership and automatically authenticate them the next time they visit the site. This check box is contained within the Login control.

The Login control has settings for customizing the display, the messages that are displayed, and the links that direct users to other pages where they can alter their password or recover a password that they have forgotten. The Login control can function independently as a control on a main or home page, or it can be used on a separate page that is specifically designated as a login

page. If you utilize the ASP.NET membership feature in conjunction with the Login control, you won't have to write any code to carry out authentication. You can, however, implement your own authentication logic by handling the Authenticate event of the Login control and adding new authentication code. This can be done if you wish to develop your own authentication logic.

Note: If the Method property of the ASP.NET Web page is changed from POST (which is the default) to GET, it is possible that the Login controls will no longer function properly.

- Start Microsoft Visual Studio 2008
- Create a new ASP.NET WebSite



Simply move the Login control from the Toolbox onto the page using drag and drop.

The membership API function Membership.ValidateUse() is called whenever the user clicks the Log In button, and the control is programmed to automatically validate the user name and password. If the validation was successful then FormAuthentication.redirectFromLoginPage() is invoked. The input that is sent to these methods by the control is affected by all of the options that are displayed on the UI of the LoginControl. When you tick the "Remember me next time" check box, for instance, the value true is sent to the createPresistentCookie parameter of the RedirectFromLoginPage() method. This ensures that you are remembered the next time you log in. Because of this, the FormAuthenticateModule generates a cookie that has staying power.

By default, there are three different Login Tasks.

- **Auto Format :** You can select preset schemes.
- **Change to the Template Format :** You are able to make changes to the content of Login Control.
- **Administer Website :** The Web Site Administration Tools, such as Security, Application, and Provider, can be configured by you.


```
<form id="form1" runat="server">
<div>
<asp:Login ID="Login1" runat="server" BackColor="#F7F7DE" Border
Color="#CCCC99" BorderStyle="Solid" BorderWidth="1px" Font-
Names="Verdana" Font-Size="10pt">
<TitleTextStyle BackColor="#6B696B" Font-Bold="True" ForeColor=
"#FFFFFF" />
</asp:Login>
</div>
</form>
```

You also have the ability to change the styles of LoginControl by using CSS. **For example :**

```
.LoginControl
{
    background-color:#F7F7DE;
    border-color:#CCCC99;
    border-style:solid;
    border-width:1px;
```

```
    font-family:Verdana;
    font-size:10px;
}
```

And finally, use some CSS to style control:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Login Control</title>
    <link href="StyleSheet.css" type="text/css" rel="Stylesheet" />
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Login ID="Login1" runat="server" CssClass="LoginControl">
            <TitleTextStyle BackColor="#6B696B" Font-Bold="True"
            ForeColor="#FFFFFF" />
        </asp:Login>
    </div>
    </form>
</body>
</html>
```

❑ **Check Your Progress – 3 :**

1. _____ protocol is utilized in the process of sending and receiving electronic mail..

   a. HTTP b. TCP c. SMTP d. IP

If you are running the page and if the CSS file is placed in a directory where anonymous access is denied, then add the following configuration for the CSS file to your web.config file.

```
<location path="StyleSheet.css">
<system.web>
<authorization>
<allow users="*"/>
</authorization>
</system.web>
</location>
```

You have the ability to add a number of hyperlinks to the Login control.

```
<asp:Login ID="Login1" runat="server" CssClass="LoginControl"
CreateUserText="Register"
CreateUserUrl="~/Register.aspx"
HelpPageText="Additional Help" HelpPageUrl="~/Help.aspx"
InstructionText="Please enter your user name and password for login.">
```

<TitleTextStyle BackColor="#6B696B" Font-Bold="True" ForeColor="#FFFFFF" />
</asp:Login>

C# Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!this.IsPostBack)
            ViewState["LoginErrors"] = 0;
    }
protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
    {
        if (YourValidationFunction(Login1.UserName, Login1.Password))
        {
            // e.Authenticated = true;
            Login1.Visible = false;
            MessageLabel.Text = "Successfully Logged In";
        }
        else
        {
            e.Authenticated = false;
        }
    }
```

```
protected void Login1_LoginError(object sender, EventArgs e)
{
    if (ViewState["LoginErrors"] == null)
        ViewState["LoginErrors"] = 0;
    int ErrorCount = (int)ViewState["LoginErrors"] + 1;
    ViewState["LoginErrors"] = ErrorCount;
    if ((ErrorCount > 3) && (Login1.PasswordRecoveryUrl != string.
    Empty))
        Response.Redirect(Login1.PasswordRecoveryUrl);
}
private bool YourValidationFunction(string UserName, string Password)
{
    bool boolReturnValue = false;
    string strConnection = "server=.;database=Vendor;uid=sa;pwd=
    wintellect;";
    SqlConnection sqlConnection = new SqlConnection(strConnection);
    String SQLQuery = "SELECT UserName, Password FROM Login";
    SqlCommand command = new SqlCommand(SQLQuery,
    sqlConnection);
    SqlDataReader Dr;
    sqlConnection.Open();
    Dr = command.ExecuteReader();
    while (Dr.Read())
    {
if ((UserName == Dr["UserName"].ToString()) & (Password == Dr
["Password"].ToString()))
        {
            boolReturnValue = true;
        }
        Dr.Close();
        return boolReturnValue;
    }
    return boolReturnValue;
}
}
```

If you enter an incorrect username and password, the following notice will appear on your screen :

If you provide the correct user name and password, your page will be redirected to the location of your choosing, and you will be given the option to display a message in the ErrorLabel box.

11.8 Logging Out :

When a user logs out of a computer system or a website, they are informing the computer or website that they want to end their login session. Logging out also closes the user's session on the computer or website. The action of logging out is often referred to as signing off and signing out. The term "log out" can be used as either a noun or an adjective to refer to the steps involved in signing out of an account. The action of signing off of an online account is referred to as "logging out," and the verb "log out" characterises this action.

Take, for instance, this sentence: The phrase "you must ensure that no one else can access your e-mail by logging out using your logout information" refers to both of these variants of the word.

The amount of time that passes from the moment a user logs in until it is time for them to exit their session is the duration of their login session. During this time, the user is able to carry out the actions that are specific to their account. There are two methods to log out of an account: the first is to use the log-out option that is supplied by an application or by the system; the second is to either turn off the computer or close the application without first logging out explicitly.

When a user's login session has been idle for an extended period of time, certain websites will automatically log that person out. There are certain applications that offer both an automatic logout and the ability to log out of multiple applications simultaneously.

C# Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
public partial class Logout : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {
        if (!Page.IsPostBack) {
            Session.Abandon();
            Session.Remove("username");
            Session.Remove("password");
            Response.Redirect("~/Login.aspx");
        }
    }
}
```

11.9 Logout Action Property :

Using the LoginStatus control, you can get or set a value that will determine what happens after a user logs out of a website. This value will dictate what action is taken.

```
[System.Web.UI.Themeable(false)]
public virtual System.Web.UI.WebControls.LogoutAction LogoutAction { get; set; }
```

Example : The following example of code sets the LogoutAction attribute to the value RedirectToLoginPage so that the user is taken to the login page once the logout action is performed.

```
<%@ Page Language="C#" AutoEventWireup="False" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>ASP.NET Example</title>
</head>
<body>
    <form id="form1" runat="server">
    <asp:LoginStatus id="LoginStatus1" runat="server"
        LogoutAction="RedirectToLoginPage" />
    </form>
</body>
</html>
```

When a user logs out of their account by using the LoginStatus control, the page that is displayed to the user is determined by the LogoutAction attribute. The activities that are carried out are outlined in the following table for each LogoutAction setting.

Actions taken on LogoutAction Values

LogoutAction Value	Action Taken
Redirect	Sends the user to the URL specified in the LogoutPageUrl property when this method is called. The user is taken to the login page specified in the application configuration settings if the LogoutPageUrl parameter does not include any information.
RedirectToLoginPage	Sends the user to the login page specified in the application configuration settings when this action is performed.
Refresh	Reloads the page that you are currently on.

Themes and style sheet themes are unable to change the value of this property.

11.10 Working with Login view and Login Name Web Controls :

LoginView Control : A user's authentication status and role membership are taken into consideration by the LoginView control in order to determine which content template should be displayed to that user.

You can use the LoginView control to display distinct controls for anonymous users and logged-in users by utilising templates known as <AnonymousTemplate> and <LoggedInTemplate>, which implement this capability, respectively.

[System.ComponentModel.Bindable(false)]

[System.Web.UI.Themeable(true)]

public class LoginView : System.Web.UI.Control, System.Web.UI. INamingContainer

Example : The following example of code demonstrates how to set templates for each of the three different template types that the LoginView control is capable of supporting.

```
<%@ Page Language="C#" %>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
    EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <script runat="server">
    </script>
    <html xmlns="http://www.w3.org/1999/xhtml" >
       <head runat="server">
       <title>ASP.NET Example</title>
    </head>
    <body>
          <form id="form1" runat="server">
             <p>
                <asp:LoginStatus id="LoginStatus1" runat="server"></
                asp:LoginStatus></p>
```

```
<p>
  <asp:LoginView id="LoginView1" runat="server">
    <AnonymousTemplate>
      Please log in for personalized information.
    </AnonymousTemplate>
    <LoggedInTemplate>
      Thanks for logging in
      <asp:LoginName id="LoginName1" runat="Server"></
      asp:LoginName>.
    </LoggedInTemplate>
    <RoleGroups>
      <asp:RoleGroup Roles="Admin">
        <ContentTemplate>
      <asp:LoginName id="LoginName2" runat="Server"></
      asp:LoginName>, you are logged in as an administrator.
        </ContentTemplate>
      </asp:RoleGroup>
    </RoleGroups>
  </asp:LoginView></p>
 </form>
</body>
</html>
```

Any website visitor who has not yet registered in will see the template that is stored in the AnonymousTemplate field when they browse the site. After a user has successfully logged in, the site will either display a template that is connected to one of the user's roles in the RoleGroups property or it will display the default template that has been specified in the LoggedInTemplate property.

Once you have assigned templates to any one of the following three template attributes of the LoginView class, the LoginView control will take care of managing the switching between the different templates :

- **AnonymousTemplate :** Users who have not yet authenticated themselves on the website will see the template specified by the AnonymousTemplate variable. Users who are currently logged in to their accounts will never be able to view this template.
- **LoggedInTemplate :** Users who have successfully signed in to the website but do not belong to any role groups that have predefined templates will see the default template specified by the LoggedInTemplate property.
- **RoleGroups :** Specifies the template to display to logged-in users who are members of roles that have defined role-group templates. This template will only be displayed if the user is logged in. In RoleGroup instances, content templates are linked to specific groups of roles to which they are assigned.

Example :

```
<asp:LoginView ID="LoginViewCtrl" runat="server">
<AnonymousTemplate>
<h2>You are anonymous</h2>
</AnonymousTemplate>
<LoggedInTemplate>
<h2>You are logged in</h2>
Submit your comment: <asp:TextBox runat="server" ID="CommentText" />
<br />
<asp:Button runat="server" ID="SubmitCommentAction" Text="Submit" />
</LoggedInTemplate>
<RoleGroups>
<asp:RoleGroup Roles="Admin">
<ContentTemplate>
<h2>Only Admins will see this</h2>
</ContentTemplate>
</asp:RoleGroup>
<asp:RoleGroup Roles="Supervisor">
<ContentTemplate>
<h2>This is for Supervisors!</h2>
</ContentTemplate>
</asp:RoleGroup>
<asp:RoleGroup Roles="Tester, Designer">
<ContentTemplate>
<h2>This is for web designers and testers</h2>
</ContentTemplate>
</asp:RoleGroup>
</RoleGroups> </asp:LoginView>
```

When searching for role-group templates, the order in which they are defined in the source is taken into consideration. The user is presented with the first role-group template that is a suitable match. If a user belongs to more than one role, the user's profile will be organized according to the first role-group template that matches any of the user's roles. If a single role has more than one template associated with it, only the template that was defined first will be used for that position.

LoginName Control : The value of the System.Web.UI.Page.User.Identity. Name property is displayed here.

```
[System.ComponentModel.Bindable(false)]
public class LoginName : System.Web.UI.WebControls.WebControl
```

Example : The following snippet of code serves as an illustration of how to incorporate the LoginName class into a page. When you click the button, the display will change formats.

```
<%@ Page Language="C#" autoeventwireup="False" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
   void Button1_Click(Object sender, EventArgs e) {
      LoginName1.FormatString = "Welcome to our Web site, {0}";
      Button1.Visible = false;
   }
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
   <title>ASP.NET Example</title>
</head>
<body>
   <form id="form1" runat="server">
      <p>
         <asp:LoginName id="LoginName1" runat="server"
            FormatString ="Welcome, {0}" />
      </p>
      <p>
         <asp:Button id="Button1" onclick="Button1_Click" runat=
         "server"
            Text="Change Format" />
      </p>
   </form>
</body>
</html>
```

The name that is stored in the User property of the Page class is what is displayed by the LoginName control when it is not changed. If there is no value assigned to the System.Web.UI.Page.User.Identity.Name property, the control will not be shown.

Setting the FormatString property will allow you to alter the text that is displayed by the LoginName control.

Note that the LoginName control cannot be used anywhere on a Web page that is not inside of a form> tag. To be more specific, the LoginName variable cannot be utilized to include the user's name in the title of a page.

11.11 Forgotten Password :

In the event that users forget their password, the website provides them with the ability to recover or reset their password. If the password is not hashed, then it is possible to recover it, which means that it can be given back to the user.

The real password is not stored in the membership system when a password is hashed instead of being stored. Instead, the system runs the password using a one-way technique known as a hashing algorithm, which generates a unique value for the password. This hash value is then saved by the system. The user's password can be tested multiple times during the login process using this approach, but the procedure cannot be inverted to produce the actual password. Because of this, the membership database will have a higher level of security because gaining access to the database will not result in the disclosure of any passwords.

The password is stored by the membership provider as a value that has been hashed by default. As a result, it will not be possible to restore the password. If a user has forgotten their password, the website must produce a new password and email it to the user. If a user has forgotten their password, the website must generate a new password and email it to the user. Your computer has to be able to connect to an SMTP (Simple Mail Transport Protocol) server in order for your website to be able to deliver email messages.

It is common known that users will regularly forget the passwords they use for their accounts. At this point, I'll demonstrate how to incorporate a page for password recovery into your website so that users can access your site without having to reset their passwords. There are two methods for recovering lost passwords:

- You have the ability to send people the password that they choose (or that you created for them when you set up the site). When selecting this option, the website will be required to store the password using reversible encryption.
- You have the ability to email users a new password, which they will be able to alter using the page titled Change Password that you developed earlier. If the website keeps passwords using a method of encryption that is not reversible, such as hashing, then selecting this option can be helpful.

The ASP.NET membership system protects users' credentials by hashing them by default. This ensures that the passwords cannot be retrieved and used by unauthorised parties. Because of this, when they reach this topic of the unit, the users of your website will receive a brand new password.

Now we will get the process of writing code to recover a forgotten password in ASP.NET using C# from the following example.

Example :

Before we start implementing this first step, construct one table in our database called UserInfo like it's shown below.

Column Name	Data Type	Allow Nulls
UserId	int (Set Identity=true)	No
UserName	varchar(50)	Yes
Email	varchar(125)	Yes
Location	varchar(50)	Yes

Once the table design is finished, insert some dummy data for the sake of demonstration, as seen below.

UserId	UserName	Password	Email	Location
1	SureshDasari	abcdef	administrator@aspdotnet-suresh.com	Chennai
2	MaheshDasari	defghd	Mahesh@gmail.com	Nuzividu
3	MadhavSai	abcdef	Maddy@hotmail.com	Nagpur
4	MahendraD	abcdef	mahendra@gmail.com	Guntur
5	PrasanthiDonthi	abcdef	prasanthi@gmail.com	Chennai

Now that we are going to use our Gmail account credentials to send mail, the first thing we need to do is enable the POP enable option in our Gmail account. In order to do this, we need to open our Gmail account and navigate to Settings ---> Forwarding and POP/IMAP. Now that we have done that, we can move on to the next step.

After that, make sure that our ASPX page looks something like this.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
<title>Code to recover forgot password in asp.net using C# and VB.NET</
title>
<style type="text/css">
.Button
{
background-color :#FF5A00;
color: #FFFFFF;
font-weight: bold;
margin-right: 2px;
padding: 4px 20px 4px 21px;
}
</style>
</head>
<body>
<form id="form1" runat="server">
<div>
<table cellspacing="2" cellpadding="2" border="0">
<tr><td></td><td><b>Forgot Password Example</b></td></tr>
<tr><td><b>Enter Your Email:</b></td><td><asp:TextBox ID="txtEmail"
runat="server" /></td></tr>
<tr><td></td><td><asp:button ID="btnSubmit" Text="Submit"
runat="server" onclick="btnSubmit_Click" CssClass="Button"/></td></
tr>
<tr><td colspan="2" style=" color:red"><asp:Label ID="lbltxt"
runat="server"/></td></tr>
</table>
</div>
```

```
</form>
</body>
</html>
```

Following that, include the following namespaces in your codebehind::

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Net.Mail;
```

C# Source Code : For Submit Button

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
try
{
DataSet ds = new DataSet();
using (SqlConnection con = new SqlConnection("Data Source=Suresh Dasari;Integrated Security=true;Initial Catalog=MySampleDB"))
{
con.Open();
SqlCommand cmd = new SqlCommand("SELECT UserName,Password FROM UserInfo Where Email= '" + txtEmail.Text.Trim() + "'", con);
SqlDataAdapter da = new SqlDataAdapter(cmd);
da.Fill(ds);
con.Close();
}
if(ds.Tables[0].Rows.Count>0)
{
MailMessage Msg = new MailMessage();
// Sender e-mail address.
Msg.From = new MailAddress(txtEmail.Text);
// Recipient e-mail address.
Msg.To.Add(txtEmail.Text);
Msg.Subject = "Your Password Details";
Msg.Body = "Hi, <br/>Please check your Login Detailss<br/><br/>Your Username: " + ds.Tables[0].Rows[0]["UserName"] + "<br/><br/>Your Password: " + ds.Tables[0].Rows[0]["Password"] + "<br/><br/>";
Msg.IsBodyHtml = true;
// your remote SMTP server IP.
SmtpClient smtp = new SmtpClient();
smtp.Host = "smtp.gmail.com";
smtp.Port = 587;
```

```
smtp.Credentials = new System.Net.NetworkCredential ("yourusername@gmail.com", "yourpassword");
smtp.EnableSsl = true;
smtp.Send(Msg);
//Msg = null;
lbltxt.Text = "Your Password Details Sent to your mail";
// Clear the textbox valuess
txtEmail.Text = "";
}
else
{
lbltxt.Text = "The Email you entered not exists.";
}
}
catch (Exception ex)
{
Console.WriteLine("{0} Exception caught.", ex);
}
}
}
```

11.12 Change Password Control :

The registered user is required to log in using the credentials that were provided to them (user name and password). Following a successful login, a link to change your password will be displayed. If the user wants to change his or her password, they must first click on this link, which will take them to a new page. Once there, they must input their current password, their new password, and their confirmation password before hitting the Update button.

Example : The database table's structure will be like this :

Column Name	Data Type	Allow Nulls
UserName	varchar(50)	☐
Password	varchar(50)	☐

HTML Source Code for Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="Password_Change_in_asp.net._Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
```

```
<div>
<asp:Label ID="Label1" runat="server" Text="Name" Font-Bold="True"
    Width="100px" BackColor="#FFFF66" ForeColor="#FF3300"></
    asp:Label>
<asp:TextBox ID="TextBox_user_name" runat="server"
ForeColor="#993300" Width="100px"></asp:TextBox><br />
<asp:Label ID="Label2" runat="server" Text="Password" Font-
Bold="True"
    Width="100px" BackColor="#FFFF66" ForeColor="#FF3300"></
    asp:Label>
<asp:TextBox ID="TextBox_password" runat="server"
ForeColor="#CC6600"
    TextMode="Password" Width="100px"></asp:TextBox><br />
<asp:Button ID="btn_login" runat="server" Text="Login" Font-
Bold="True"
    BackColor="#CCFF99" onclick="btn_login_Click"  /><br />
<asp:Label ID="lb1" runat="server" Font-Bold="True" ForeColor=
"#FF3300"> </asp:Label>
</div>
</form>
</body>
</html>
```

C# Source Code for Default.aspx

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Data.SqlClient;
namespace Password_Change_in_asp.net
{
    public partial class _Default : System.Web.UI.Page
        {
```

```
string strConnString = ConfigurationManager.ConnectionStrings
["ConnectionString"] .ConnectionString;
        string str = null;
        SqlCommand com;
        protected void btn_login_Click(object sender, EventArgs e)
        {
            object obj = null;
            SqlConnection con = new SqlConnection(strConnString);
            con.Open();
                Session["UserName"] = TextBox_user_name.Text;
            str = "select count(*) from login where UserName=@UserName
and Password =@Password";
            com = new SqlCommand(str, con);
            com.CommandType = CommandType.Text;
            com.Parameters.AddWithValue("@UserName", Session
            ["UserName"]);
            com.Parameters.AddWithValue("@Password", TextBox_
            password.Text);
            obj = com.ExecuteScalar();
            if ((int)(obj) != 0)
            {
                Response.Redirect("Welcome.aspx");
            }
            else
            {
                lb1.Text = "Invalid Username and Password";
            }
            con.Close();
        }
    }
}
```

HTML Code for Welcome.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind=
"Welcome.aspx.cs" Inherits="Password_Change_in_asp.net.Welcome" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
```

```
<body>
    <form id="form1" runat="server">
    <div>
    <asp:Label ID="lb1" runat="server" Font-Bold="True" ForeColor=
    "#FF3300"> </asp:Label><br />
    </div>
    <asp:LinkButton ID="lnk_changepassword" runat="server"
        onclick="lnk_changepassword_Click">Change Password</asp:
        LinkButton>
    </form>
</body>
</html>
```

C# Source Code for Welcome.aspx

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
namespace Password_Change_in_asp.net
{
    public partial class Welcome : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            lb1.Text = "WELCOME :: " + Session["UserName"];
        }
            protected void lnk_changepassword_Click(object sender,
            EventArgs e)
        {
            Response.Redirect("Changepassword.aspx");
        }
    }
}
```

HTML Code for Changepassword.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind=
"Changepassword.aspx.cs"  Inherits="Password_Change_in_asp.net.
Changepassword" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

   <title>Untitled Page</title>

</head>

<body>

   <form id="form1" runat="server">

   <div>

      <asp:Label ID="Label1" runat="server" Text="Current password"
      Width="120px"

         Font-Bold="True" ForeColor="#996633"></asp:Label>

         <asp:TextBox ID="txt_cpassword" runat="server" TextMode=
         "Password"> </asp:TextBox>

      <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat=
      "server"

         ControlToValidate="txt_cpassword"

         ErrorMessage="Please enter Current password"></asp:Required
         FieldValidator>

      <br />

      <asp:Label ID="Label2" runat="server" Text="New password"
      Width="120px"

         Font-Bold="True" ForeColor="#996633"></asp:Label>

         <asp:TextBox ID="txt_npassword" runat="server" TextMode=
         "Password"> </asp:TextBox>

      <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat=
      "server"

         ControlToValidate="txt_npassword" ErrorMessage="Please enter
         New password"> </asp:RequiredFieldValidator>

      <br />

         <asp:Label ID="Label3" runat="server" Text="Confirm
         password" Width="120px"

         Font-Bold="True" ForeColor="#996633"></asp:Label>

            <asp:TextBox ID="txt_ccpassword" runat="server"
         TextMode="Password">  </asp:TextBox>

         <asp:RequiredFieldValidator ID="RequiredFieldValidator3"
         runat="server"

         ControlToValidate="txt_ccpassword"
```

```
            ErrorMessage="Please enter Confirm  password"></asp:Required
            FieldValidator>
            <asp:CompareValidator ID="CompareValidator1" runat="server"
            ControlToCompare="txt_npassword"  ControlToValidate="txt_
            ccpassword"
            ErrorMessage="Password Mismatch"></asp:CompareValidator>
    </div>
    <asp:Button ID="btn_update" runat="server" Font-Bold="True"
BackColor="#CCFF99" onclick="btn_update_Click" Text="Update" />
    <asp:Label ID="lbl_msg" Font-Bold="True" BackColor="#FFFF66"
ForeColor= "#FF3300" runat="server" Text=""></asp:Label><br />
    <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/
Default.aspx"> Login </asp:HyperLink>
    </form>
</body>
</html>
```

C# Source Code for Changepassword.aspx

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Data.SqlClient;
namespace Password_Change_in_asp.net
{
    public partial class Changepassword : System.Web.UI.Page
    {
        string strConnString = ConfigurationManager.ConnectionStrings
["ConnectionString"] .ConnectionString;
        string str = null;
        SqlCommand com;
        byte up;
        protected void btn_update_Click(object sender, EventArgs e)
        {
```

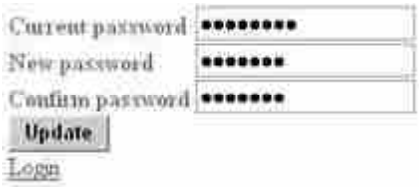
```
            SqlConnection con = new SqlConnection(strConnString);
            con.Open();
            str = "select * from login ";
            com = new SqlCommand(str, con);
            SqlDataReader reader = com.ExecuteReader();
            while (reader.Read())
            {
                if (txt_cpassword.Text == reader["Password"].ToString())
                {
                    up = 1;
                }
            }
            reader.Close();
            con.Close();
            if (up == 1)
            {
                con.Open();
            str = "update login set Password=@Password where UserName='"
            + Session["UserName"].ToString()+ "'";
                com = new SqlCommand(str, con);
                com.Parameters.Add(new SqlParameter("@Password",
                SqlDbType.VarChar, 50));
                com.Parameters["@Password"].Value = txt_npassword.Text;
                com.ExecuteNonQuery();
                con.Close();
                lbl_msg.Text = "Password changed Successfully";
            }
            else
            {
                lbl_msg.Text = "Please enter correct Current password";
            }
        }
    }
}
```

The output will be looked like this:

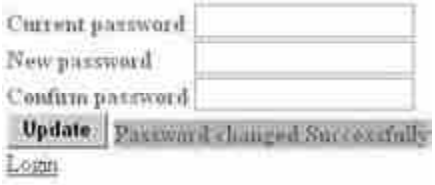
When I will enter username and password, it will be like this :

After we will login successfully, the screen will look like this :

When we change the current password from a new password, it will look like this :



After changing password successfully, the output screen will look like this :



11.13 Let Us Sum Up :

In this unit, we have learnt about how to create and manage User Accounts. Also learnt how do we provide Membership to the users and provide various Roles to the users. Learnt how do we can set the access rules for the users.

We have also learnt how do we create new User Accounts Via Websites. Wh have also learnt how to recover password, how to generatepassword automatically & the properties of Required email.

We have also got the knowledge from this unit about the Website Login Controls, how to Logging Out and the Properties of Logout action. We have also learnt how to work with Login view and Login Name Web Controls. We have learnt how to change password Control.

11.14 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : c **2 :** b

- **Check Your Progress 2 :**

  1 : b **2 :** c

- **Check Your Progress 3 :**

  1 : c

11.15 Glossary :

1. **Protocol :** A protocol is an established set of rules that determine how data is transmitted between different devices in the same network.
2. **SMTP :** Simple Mail Transfer Protocol. SMTP is used to send and receive email.
3. **Role :** A role is a database object that groups together one or more privileges and can be assigned to users.

4. **Anonymous :** Anonymous means keeping a user's name and identity concealed through various applications.

11.16 Assignment :

1. Explain Logging In Control in detail.

11.17 Activities :

1. Create a webapplication and use Login Control in it and provide roles to the users.

11.18 Case Study :

Study about the User Registration and apply it into a web application.

11.19 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

Unit 12

INTORDUCTION OF MASTER PAGES

UNIT STRUCTURE

12.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About Importance of Master Pages and Themes
- About Master Pages and Themes Creation
- About Configuration of Master Pages and Themes
- About Content Modification of the Master Page and Theme
- About Properties and Methods of Master Pages and Themes

12.1 Introduction :

A document-style report might have a master page, which is a predefined collection of formatting that is used to prepare the individual sections of the report. You have the ability to specify a master page within a template. This master page can have a header element, a footer element, and layout properties like orientation and borders.

When creating a document, using master pages helps ensure that each subsequent page looks the same. Page headers, page footers, margin and column guides, and any other components that appear on numerous pages throughout your document should all be included on your project's master pages.

Your website's pages and controls can take on a variety of looks thanks to ASP.NET themes, which are collections of properties that influence how they look. A theme may contain cascading style sheet files (.css files), graphics, and skin files. Skin files are used to define property settings for ASP.NET Web server controls. Other file types that may be included in a theme are cascading style sheet files and graphics.

To apply a theme, you can use either the Theme or StyleSheetTheme attribute of the @ Page directive, or you can set the pages Element (ASP.NET Settings Schema) element in the application configuration file. Both of these methods are described in more detail below. The StyleSheetTheme attribute is the only way for Visual Web Developer to graphically depict the various themes that can be applied.

12.2 Importance of Master Pages and Themes :

Master pages offer functionality that traditionally has been created by developers by repeatedly copying existing code, text, and control elements; utilizing framesets; utilizing include files for common elements; utilizing ASP.NET user controls; and so on. Master pages offer this functionality instead. The following is a list of advantages that come with using master pages :

- They enable you to concentrate the functionality that is shared across all of your pages into a single location, allowing you to make changes to that location alone.
- They simplify the process of creating a single set of controls and code and applying the resulting controls and code to multiple pages. For instance, you may make a menu that appears on all pages by utilizing the controls that are located on the master page.
- They do this by enabling you to modify the way in which the placeholder controls are rendered, which in turn provides you with granular control over the layout of the final page.
- They give an object model that enables you to personalize the master page based on the content pages that you have created individually.

The overall appearance of a website is determined by its theme. It is a group of files that together determine how a page will appear to the user. It is possible for it to contain skin files, CSS files, and picture files.

We define themes inside of a specialized subdirectory called App Themes. The actual themes are specified under this folder's one or more corresponding subfolders, which have names like Theme1, Theme2, etc. The theme attribute is applied late in the page's life cycle, which means that it effectively overrides any customizations you may have made for individual controls on your page.

- Because themes can contain CSS files, image files, and skins, you are able to alter the colours, fonts, positions, and images simply by applying the themes that you want.
- You are free to use an unlimited number of themes, and you can switch between them at any time by modifying a single attribute in the web.config file or on an individual aspx page. In addition, you have the ability to programmatically swap between themes.
- By letting the themes be set programmatically, you give your users a fast and simple option to customise the page to their preferences and preferences alone.
- Themes give users with vision problems the opportunity to select a theme with a high contrast and a large text size, which improves the usability of a website by making it easier for users with vision problems to navigate the website.

12.3 Master Pages and Themes Creation :

A master page is a page that other pages derive their structure and functionality from. ASP.NET makes it very simple to create a master page for your website. Let's get started by methodically designing the master page.

Open new project in visual studio :

After clicking OK button in the Window, select Empty

After clicking OK button, project "masterpage" opens but no file is there :

Add new file in to our project. Add the master page into our project. Right click Project->Add->New item

After clicking on new item, Window will open, select Web Form->Web Forms Master Page :

After clicking the add button, master page 'site1.master' adds to our project. Click on site1.master into Solution Explorer

Design the master page, using HTML.

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site1.master.cs" Inherits="masterpage.Site1" %>
    <!DOCTYPE html>
    <html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>c# corner</title>
    <link href="css/my.css" rel="stylesheet" />
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <!DOCTYPE html>
<html>
<head>
    <title>my layout</title>
    <link rel="stylesheet" type="text/css" href="my.css">
</head>
<body>
<header id="header">
<h1>c# corner</h1>
</header>
<nav id="nav">
    <ul>
        <li><a href="home.aspx">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Article</a></li>
        <li><a href="#">Contact</a></li>
    </ul>
</nav>
<aside id="side">
    <h1>news</h1>
    <a href="#"><p>creating html website</p></a>
    <a href="#"><p>learn css</p></a>
    <a href="#">learn c#</a>
</aside>
    <div id="con">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
            </asp:ContentPlaceHolder>
```

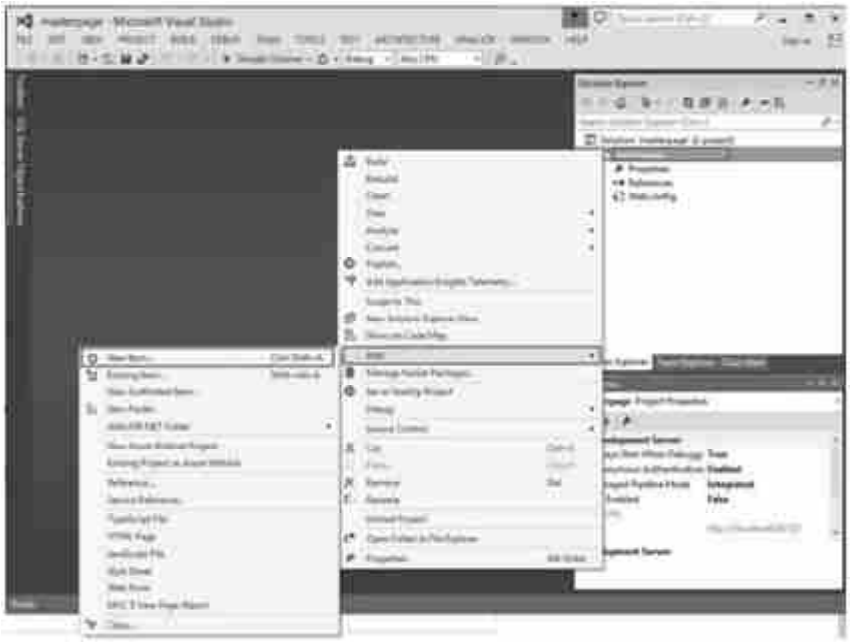
```
    </div>
    <footer id="footer">
    copyright @c# corner
</footer>
</body>
</html>
    <form id="form1" runat="server">
        </form>
</body>
</html>
#header{
    color: #247BA0;
    text-align: center;
    font-size: 20px;
}
#nav{
    background-color:#FF1654;
    padding: 5px;
}
ul{
    list-style-type: none;
}
li a {
    color: #F1FAEE;
font-size: 30px;
column-width: 5%;
    }
    li
    {
    display: inline;
    padding-left: 2px;
    column-width: 20px;
    }
    a{
text-decoration: none;
margin-left:20px
    }
    li a:hover{
        background-color: #F3FFBD;
        color: #FF1654;
```

```
padding:1%;
}
#side{
text-align: center;
float: right;
width: 15%;
padding-bottom: 79%;
background-color: #F1FAEE;
}
#article{
background-color: #EEF5DB;
padding: 10px;
padding-bottom: 75%;
}
#footer{
background-color: #C7EFCF;
text-align:center;
padding-bottom: 5%;
font-size: 20px;
}
#con{
    border:double;
    border-color:burlywood;
}
```

Our master page is designed. Move to the next step.

Step 4 : Add web form in to our project.

Right click on the project->Add->New item



Select Web form with the master page.

After clicking on that, add the button Window, open the selected masterpage->site1.master and click OK.

Now, design our homepage. Here, we write home page only,

Home.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup= "true" CodeBehind="home.aspx.cs" Inherits= "masterpage.home" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">

</asp:Content>

<asp:Content ID="Content2"ContentPlaceHolderID="ContentPlace Holder1" runat="server">

    <h1>Home page</h1>

</asp:Content>
```

Finally, our Master page is created; build and run the project. The master page looks aas shown in the picture :

A collection of property settings that enables you to create the visual appearance of pages and controls is referred to as a theme. It is possible to give all of the pages in a web application the same overall appearance. Themes are constructed from a variety of components, including server control skins, CSS files, and additional resources. To demonstrate how to construct a theme, consider the following scenario: you have access to a skin and a style sheet.

Within the project for your website, separate folders are used to define the themes. For theme creation we need to follow the following steps :

- Right-clicking the name of the Web site project in the Solution Explorer will bring up the Add ASP.NET Folder menu. From there, select Theme.
- A new themes folder with the name Theme1 is added, and at the same time, the App Themes folder is created automatically.
- Select the new Theme1 folder, then click the Rename button on the context menu that appears. After typing Blue, hit the Enter key.
- After giving the new Blue folder a right-click, select Add New Item from the context menu.
- Select Skin File from the submenu of the Add New Item dialogue box, and give the file the name default.skin. Click Add.
- Right-click the newly created Blue folder in Solution Explorer, then select the Add New Item option from the context menu.
- Choose Style Sheet from the drop-down menu in the "Add New Item" dialogue box. You should call the default style sheet default.css. Click Add.
- After that, an empty CSS file and a server control skin file are used to build the initial theme. You will soon be editing these files, but first you need to create a page that has some HTML and a control on it so that the theme can be applied to it.

❑ **Check Your Progress – 1 :**

1. We define themes inside of a specialized subdirectory is _____.

   a. App b. App Themes c. Data Themes d. None of these

2. A _____ is a page that other pages derive their structure and functionality from.

   a. Master Page b. SiteMapPath c. Theme d. None of these

3. Themes give users with vision problems the opportunity to select a theme with a low contrast and a small text size

a. True b. False c. None of these

12.4 Configuration of Master Pages and Themes :

It's not always easy to use an ASP.Net master page in conjunction with a theme and CSS. 2005 was the year that themes and master pages were made available. During that time, there were hundreds of articles published that discussed them. However, since then, a tremendous deal of change has occurred in the process that we use to construct websites. It is high time that this issue was discussed again. In this essay, we'll go through the fundamentals of the topic. After that, we will add stylesheets specifically for Internet Explorer as well as printing.

Let's say I want to create a website, but I don't want to use .Net. Instead, I want to use only HTML and CSS. The heading of my HTML document could look something like this :

```
<head>
 <title>My Title</title>
 <meta http-equiv="content-type" content="text/html;charset=utf-8" />

 <link rel="stylesheet" href="css/style.css" type="text/css" media="screen, projection" />
 <link rel="stylesheet" href="css/print.css" type="text/css" media="print" />

 <!--[if IE]>
  <link rel="stylesheet" href="css/ie.css" type="text/css" media="screen, projection" />
 <![endif]-->
</head>
```

Please take note that I have linked in three different CSS stylesheets: style.css, print.css, and ie.css.

- The primary CSS file that my website uses is called style.css. When my webpage is viewed on a computer screen or projector, the "media" element specifies that this CSS comes into play and applies the style.
- My second CSS file, which is named print.css and has a media tag of "print," indicates that the file contains specialised CSS that is only applicable when the page in question is printed. When the page is printed, I can make this section invisible by adding some CSS code that hides the header, footer, adverts, and so on. When it comes to printing, I am able to define a variety of typefaces and font colours.
- The very last CSS file, ie.css, provides special formatting instructions that are only relevant when my website is accessed in Internet Explorer. The comment block includes a link to the CSS file that can be accessed using that link. This supplemental CSS file will be included by Internet Explorer. It will be disregarded by other browsers. This file will include any specific CSS hacks that are required in order for my page to show appropriately in Internet Explorer.

My problem is that I need a Master Page that will use style.css as the primary CSS file instead of the default one. When I am working on Internet Explorer, I also want to incorporate the ie.css file. When I am printing, I want to use the print.css file. In addition to this, whenever I look at a page in Visual Studio's designer view, I want the programme to apply the style.css file for me. This will give me a better picture of how my page will appear once it

is finished being developed. The basic of it is I am going to give you a very brief overview of the fundamentals. There are literally hundreds of articles that can be found online that can teach you how to make a master page and theme. I'm going to keep this brief and only bring up a few points.

First, I made a website for my business. I made a new folder and gave it the name App Themes. I put a second folder that I've titled BlueTheme inside of this one. That is the concept of this website, if you will. If I desired to make use of more themes, I might make use of additional folders such as GreenTheme, ModernTheme, and so on. In addition to that, I included a Master Page that is simply titled Basic.Master.

I have indicated in the Web.config file for this website that I want to make use of the BlueTheme theme as my overall theme. The Web.config file already contained the pages node when we started working on it. Simply adding the attribute styleSheetTheme=BlueTheme was all that was required of me.

```
<pages styleSheetTheme="BlueTheme">
  <controls>
    <add tagPrefix="asp" namespace="System.Web.UI" assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKe
    <add tagPrefix="asp" namespace="System.Web.UI.WebControls" assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutr
  </controls>
</pages>
```

Please take into consideration that I might have simply said theme = BlueTheme instead of styleSheetTheme = BlueTheme. The primary distinction between these two options is whether or not the styles derived from the theme are applied first, and then whether or not the styles added to a specific page have the ability to override those styles.

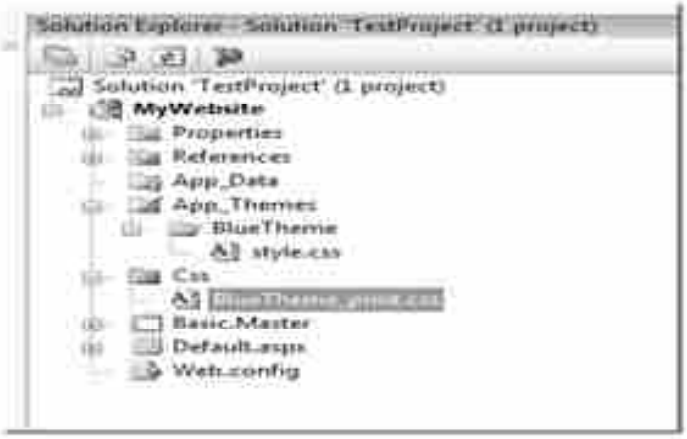
When adding Style.css, It's easy to include the style.css file. Basically, all I do is drop the file into the BlueTheme folder. Since I've already told the Web.config file that I want to use BlueTheme, any CSS files that I place in the BlueTheme folder will have their styles automatically linked in and applied to the site. It is not necessary for me to make any changes to the Master Page.

Now, when I look at the Master Page in Visual Studio's designer view, it displays me (roughly) what my page looks like after applying style.css. This information is quite helpful.

This would allow me to incorporate several CSS files; all I would need to do is add them to the BlueTheme folder. The BlueTheme folder has a bunch of CSS files, and those files will all be linked in and applied automatically.

When adding Print.css, It is not possible for me to just place my print.css file inside of the BlueTheme folder. Even if I am not printing, all of the CSS files that are located in the BlueTheme folder will be linked in and applied. When it comes to printing, I simply want the print.css file to be used as a style guide. It's imperative that I move my CSS file to a different location.

The name of the new folder I've added to my project is Css. My print CSS file has been placed inside of this folder for your convenience. I changed the name of my file from print.css to BlueTheme print.css so that if I decide to add other themes in the future, I won't end up getting confused.



It is now necessary for me to include a link to the following CSS file in my Master Page :

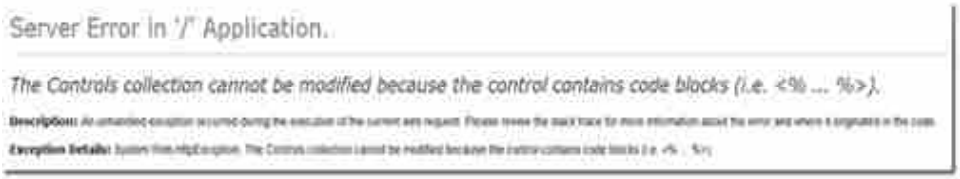
Take note that I did not firmly embed the location of the CSS file in the code. Instead, I utilized the %# Page.ResolveUrl()%> directive to link in the file during the execution of the programme. In order for this to operate, the head tag needs to include the runat="server" attribute. In the end, I put

one line of code to the Master Page's code behind, and that line simply says Page. Header. DataBind(); .

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace MyWebsite
{
    public partial class Basic : System.Web.UI.MasterPage
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            // Binds the Page.ResolveUrl code in the master page head.
            Page.Header.DataBind();
        }
    }
}
```

Please take note that you must use the %# Page.ResolveUrl()%>, not <%= Page.ResolveUrl()%> . In other words, replace = with #. This is a simple error to commit, but it has led to the frustration of a great number of programmers, as evidenced by the posts they have made on. Net forums asking for help. If you enter the incorrect information, you will receive the following error :



When adding ie.css, similar to the situation with the print CSS, I am unable to simply include the ie.css file in the BlueTheme folder. Regardless of the web browser that I have open, each of the CSS files contained within the BlueTheme folder will be linked in and applied. Only the ie.css file should be applied while using Internet Explorer, per my instructions. It's imperative that I move my CSS file to a different location. I will include it in the CSS file that is used for printing :

After that, I edit my Master Page to include a link to the file. I make sure to add the %# Page.ResolveUrl()%> code that was previously described.

There are many more ways to accomplish what I have shown here than just the ones I've detailed. There are additional approaches to completing the task, each of which has a unique set of benefits and drawbacks.

❑ **Check Your Progress – 2 :**

1. A collection of property settings that enables you to create the visual appearance of pages and controls is referred to as a _____.

   a. WebPage b. Theme c. MasterPage d. All of these

2. What should we use to resolve problem of embedding the CSS file stored at different location ?

   a. %# Page.ResolveUrl()%> b. %= Page.ResolveUrl()%>

   c. %% Page.ResolveUrl()%> d. None of these

12.5 Let Us Sum Up :

Throughout the course of this unit, we have gained an understanding of the MasterPages and the role that they play in our online applications. Through this, we have gained knowledge regarding the production of Master Pages as well as the application of themes to web pages.

We have been familiar with the process of configuring Master Pages and Themes throughout the course of this unit. We have researched the process of changing the content of the master page as well as the content of the themes. In addition, we have investigated and gained knowledge on the attributes of Master Pages as well as Events of Master Pages and themes.

12.6 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1 : b **2 :** a **3 :** b

❑ **Check Your Progress 2 :**

1 : b **2 :** b

12.7 Glossary :

1. **CSS :** CSS (Cascading Style Sheets) is the core technology for building and designing Web pages.
2. **Theme :** a theme is a preset package containing graphical appearance and functionality details.
3. **MasterPage :** A master page is a defined set of formatting that is applied to the sections of your document-style report.

4. **Skins :** It is visual style or a custom graphical appearance preset package achieved by the use of a graphical user interface (GUI) that can be applied to specific computer software, operating system, and websites to suit the purpose, topic, or tastes of different users.

12.8 Assignment :

1. Explain the configuration of MasterPage and Themes.

12.9 Activities :

1. Create a Master Page, three web pages. Open those three web pages in the Master Page.

12.10 Case Study :

Visit the web sites and try to find out what are the Master Pages and which type of themes they used in their websites.

12.11 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

Unit
13

FILES AND STREAMS

UNIT STRUCTURE

13.0 Learning Objectives :

After learning this unit, you will be able to understand :

- About System.IO namespace
- About Working with Drives, Directories, Files
- About Properties and ACL
- About Directory Class
- About DriveInfo Class
- About DirectoryInfo Class
- About how to Create, Copy Directory and Subdirectories
- About how to Retrieve Files from a Directory
- About Files: Creating, Copying, Reading, Appending, Renaming, Compressing

13.1 Introduction :

Both the file and stream I/O, which stands for "input/output," refers to the process of moving data into or out of a storage media. The System.IO

namespaces in.NET contain types that enable reading and writing on data streams and files in both a synchronous and an asynchronous manner. These operations can be performed synchronously or asynchronously. In addition, these namespaces provide types that may compress and decompress files, as well as types that can facilitate communication via pipes and serial ports.

A file is a collection of bytes that is arranged and given a name, and it has storage that is durable. When working with files, you will be working with directory paths, disc storage, as well as the names of files and directories. A stream, on the other hand, is a series of bytes that can be used to read from and write to a backing store, which can be any one of several different storage mediums (for example, discs or memory). There are many different forms of streams than file streams, such as network streams, memory streams, and pipe streams. Just as there are many different kinds of backup stores besides discs, there are also many different kinds of streams besides file streams.

13.2 Introducing System.IO Namespace :

Classes, structures, delegates, and enumerations pertaining to input/output operations are contained within the System.IO namespace. These classes can be used to read data from files or data streams as well as write data to those locations. Additionally, it has classes for supporting files and directories.

The classes that can be found in the System.IO namespace are listed below.

Class	Description
BinaryReader	It is used to read primitive data types as binary values in a specific encoding.
BinaryWriter	It is used to write primitive types in binary to a stream.
BufferedStream	It is used to add a buffering layer to read and write operations on another stream. It is a sealed class.
Directory	It is used to expose static methods for creating, moving and enumerating through directories and subdirectories. It is a sealed class.
DirectoryInfo	It is used to expose instance methods for creating, moving and enumerating through directories and subdirectories. It is a sealed class.
DirectoryNotFoundException	It is used to handle exception related to the file or directory cannot be found.
DriveInfo	It is used to access the information on a drive.
DriveNotFoundException	It is used to handle drive not found exception.
EndOfStreamException	It is used to handle end of stream exception.

ErrorEventArgs	It provides data for the FileSystemWatcher. Error event.
File	This class provides static methods for the creation, copying, deletion, moving and opening of a single file.
FileFormatException	It is used to handle file format exception.
FileInfo	It is used to provide properties and instance methods for the creation, copying, deletion, moving and opening of files.
FileLoadException	It is used to handle file load exception.
FileNotFoundException	It is used to handle file load exception.
FileNotFoundException	It is used to handle file not found exception.
FileStream	It provides a Stream for a file, supporting both synchronous and asynchronous read and write operations.
FileSystemEventArgs	It provides data for the directory events.
FileSystemInfo	It provides the base class for both FileInfo and DirectoryInfo objects.
FileSystemWatcher	It listens to the file system change notifications and raises events when a directory or file in a directory, changes.
InternalBufferOverflowException	This class is used to handle internal buffer overflow exception.
InvalidDataException	It is used to handle invalid data exception.
IODescriptionAttribute	It sets the description visual designers can display when referencing an event, extender or property.
IOException	It is an exception class that handles I/O errors.
MemoryStream	It is used to create a stream whose backing store is memory.
Path	It performs operations on String instances that contain file or directory path information.
PathTooLongException	It is an exception class and used to handle path too long exception.
PipeException	This exception class is used to handle pipe related exception.
RenamedEventArgs	It is used to provide data for the Renamed event.
Stream	It is used to provide a generic view of a sequence of bytes. It is an abstract class.

StreamReader	It is used to implement a TextReader that reads characters from a byte stream.
StringReader	It is used to implement a TextReader that reads from a string.
StringWriter	It is used to implement a TextWriter for writing information to a string. The information is stored in an underlying StringBuilder.
TextReader	This class is used to represent a reader that can read a sequential series of characters.
TextWriter	This class is used to represent a writer that can write a sequential series of characters.
UnmanagedMemoryAccessor	It is used to provide random access to unmanaged blocks of memory from managed code.
UnmanagedMemoryStream	It is used to get access to unmanaged blocks of memory from managed code.

13.3 Working with Drives, Directories, Files :

By utilizing the following classes in.NET, you will have access to the information contained within the file system :

- System.IO.FileInfo
- System.IO.DirectoryInfo
- System.IO.DriveInfo
- System.IO.Directory
- System.IO.File

Both the FileInfo and DirectoryInfo classes provide properties that expose many of the file attributes that are supported by the NTFS file system. These properties are used to represent a file or directory, respectively. In addition to that, they include procedures for opening, shutting, relocating, and deleting folders and files. You can make instances of these classes by providing the function Object() { [native code] } with a string that contains the name of the file, folder, or disc that you want to work with:

System.IO.DriveInfo di = new System.IO.DriveInfo(@"C:\");

The names of files, folders, and drives can also be obtained through the use of calls to

- DirectoryInfo.GetDirectories,
- DirectoryInfo.GetFiles, and
- DriveInfo.RootDirectory.

The System.IO.Directory and System.IO.File classes provide static methods for retrieving information about directories and files.

❑ **Check Your Progress – 1 :**

1. _____ exception class is used to handle pipe related exception..

   a. TextReader b. TextWriter c. Path d. PipeException

2. The _____ classes provide static methods for retrieving information about directories and files.

   a. System.IO.Directory b. System.IO.File

   c. Both a and b d. None of these

Example : The following example demonstrates several different approaches of gaining access to information regarding files and directories.

```
class FileSysInfo
{
   static void Main()
   {
      // You can also use System.Environment.GetLogicalDrives to
      // obtain names of all logical drives on the computer.
      System.IO.DriveInfo di = new System.IO.DriveInfo(@"C:\");
      Console.WriteLine(di.TotalFreeSpace);
      Console.WriteLine(di.VolumeLabel);
      // Get the root directory and print out some information about it.
      System.IO.DirectoryInfo dirInfo = di.RootDirectory;
      Console.WriteLine(dirInfo.Attributes.ToString());
      // Get the files in the directory and print out some information
      about them.
      System.IO.FileInfo[] fileNames = dirInfo.GetFiles("*.*");
      foreach (System.IO.FileInfo fi in fileNames)
      {
         Console.WriteLine("{0}: {1}: {2}", fi.Name, fi.LastAccessTime,
         fi.Length);
      }
      // Get the subdirectories directly that is under the root.
      // See "How to: Iterate Through a Directory Tree" for an example
      of how to
      // iterate through an entire tree.
      System.IO.DirectoryInfo[] dirInfos = dirInfo.GetDirectories("*.*");
      foreach (System.IO.DirectoryInfo d in dirInfos)
      {
         Console.WriteLine(d.Name);
      }
      // The Directory and File classes provide several static methods
      // fo accessing files and directories.
```

```
// Get the current application directory.
string currentDirName = System.IO.Directory.GetCurrentDirectory();
Console.WriteLine(currentDirName);
// Get an array of file names as strings rather than FileInfo objects.
// Use this method when storage space is an issue, and when you might
// hold on to the file name reference for a while before you try to access
// the file.
string[] files = System.IO.Directory.GetFiles(currentDirName, "*.txt");
foreach (string s in files)
{
    // Create the FileInfo object only when needed to ensure
    // the information is as current as possible.
    System.IO.FileInfo fi = null;
    try
    {
        fi = new System.IO.FileInfo(s);
    }
    catch (System.IO.FileNotFoundException e)
    {
        // To inform the user and continue is
        // sufficient for this demonstration.
        // Your application may require different behavior.
        Console.WriteLine(e.Message);
        continue;
    }
    Console.WriteLine("{0} : {1}",fi.Name, fi.Directory);
}
// Change the directory. In this case, first check to see
// whether it already exists, and create it if it does not.
// If this is not appropriate for your application, you can
// handle the System.IO.IOException that will be raised if the
// directory cannot be found.
if (!System.IO.Directory.Exists(@"C:\Users\Public\TestFolder\"))
{
    System.IO.Directory.CreateDirectory(@"C:\Users\Public\Test
    Folder\");
}
```

```
        System.IO.Directory.SetCurrentDirectory(@"C:\Users\Public\Test
        Folder\");
        currentDirName = System.IO.Directory.GetCurrentDirectory();
        Console.WriteLine(currentDirName);
        // Keep the console window open in debug mode.
        Console.WriteLine("Press any key to exit.");
        Console.ReadKey();
    }
}
```

13.4 Properties and ACL :

Get the FileSecurity or DirectorySecurity object from the file or directory you want to edit in order to add or remove entries from an Access Control List (ACL). Make the necessary changes to the object, and then reapply those changes to the file or directory.

Add or remove an ACL entry from a file : These are the following steps to be followed:

- Make a call to the File.GetAccessControl method to obtain a FileSecurity object that stores the active ACL entries of a file. This object can then be used to restrict access to the file.
- You can modify the FileSecurity object that was returned from step 1 by adding or removing ACL elements.
- Send the FileSecurity object to the File.SetAccessControl method in order for the changes to take effect.

Add or remove an ACL entry from a directory : These are the following steps to be followed:

- Make a call to the Directory.GetAccessControl method to obtain a DirectorySecurity object that stores the active ACL entries of a directory. This object can then be used to restrict access to the directory.
- You can modify the DirectorySecurity object that was returned from step 1 by adding or removing ACL elements.
- Send the DirectorySecurity object to the Directory.SetAccessControl method in order for the changes to take effect.

Example : In order to successfully run this example, you need to log in using a legitimate user or group account. In this example, a File object is used. It is recommended that the FileInfo, Directory, and DirectoryInfo classes all use the same technique.

```
using System;
using System.IO;
using System.Security.AccessControl;
namespace FileSystemExample
{
    class FileExample
    {
```

```
public static void Main()
{
    try
    {
        string fileName = "test.xml";
        Console.WriteLine("Adding access control entry for "+
        fileName);
        // Add the access control entry to the file.
        AddFileSecurity(fileName, @"DomainName\AccountName",
            FileSystemRights.ReadData, AccessControlType.Allow);
        Console.WriteLine("Removing access control entry from "+
        fileName);
        // Remove the access control entry from the file.
        RemoveFileSecurity(fileName, @"DomainName\Account
        Name",
            FileSystemRights.ReadData, AccessControlType.Allow);
        Console.WriteLine("Done.");
    }
    catch (Exception e)
    {
        Console.WriteLine(e);
    }
}
// Adds an ACL entry on the specified file for the specified account.
public static void AddFileSecurity(string fileName, string account,
    FileSystemRights rights, AccessControlType controlType)
{
    // Get a FileSecurity object that represents the
    // current security settings.
    FileSecurity fSecurity = File.GetAccessControl(fileName);
    // Add the FileSystemAccessRule to the security settings.
    fSecurity.AddAccessRule(new FileSystemAccessRule(account,
    rights, controlType));
    // Set the new access settings.
    File.SetAccessControl(fileName, fSecurity);
}
// Removes an ACL entry on the specified file for the specified
account.
public static void RemoveFileSecurity(string fileName, string
account,
    FileSystemRights rights, AccessControlType controlType)
```

```
{
    // Get a FileSecurity object that represents the current security settings.
    FileSecurity fSecurity = File.GetAccessControl(fileName);
    // Remove the FileSystemAccessRule from the security settings.
fSecurity.RemoveAccessRule(new FileSystemAccessRule(account, rights, controlType));
    // Set the new access settings.
    File.SetAccessControl(fileName, fSecurity);
} } }
```

❑ **Check Your Progress – 2 :**

1. ACL stands for _____.

   a. Admission Control Library b. Access Control Locator

   c. Access Control List d. None of these

2. The FileInfo and DirectoryInfo classes provide properties that expose many of the file attributes that are supported by the _____.

   a. NFST file system b. NTFS file system

   c. NEFT file system d. None of these

13.5 Directory Class :

It makes available static methods for generating and traversing through directories and subdirectories, as well as for enumerating them. It is not possible to inherit from this class.

```
public static class Directory
```

Example : The following example demonstrates how to extract all of the text files contained within a directory and then relocate those files to a different directory. After the files have been relocated, a record of them in the previous directory will no longer be there.

```
using System;
using System.IO;
namespace ConsoleApplication
{
   class Program
   {
      static void Main(string[] args)
      {
         string sourceDirectory = @"C:\current";
         string archiveDirectory = @"C:\archive";
         try
         {
            var txtFiles = Directory.EnumerateFiles(sourceDirectory, "*.txt");
            foreach (string currentFile in txtFiles)
```

```
                {
                    string fileName = currentFile.Substring(sourceDirectory.
                    Length + 1);
                    Directory.Move(currentFile, Path.Combine(archive
                    Directory, fileName));
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

This example shows how to use the EnumerateFiles method to extract a collection of text files from a directory, and then how to utilise that collection in a query to locate all of the lines that contain the word "Example." The example may be found below.

```
using System;
using System.IO;
using System.Linq;
namespace ConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            string archiveDirectory = @"C:\archive";
                var files = from retrievedFile in Directory.EnumerateFiles
                (archiveDirectory, "*.txt", SearchOption.AllDirectories)
                        from line in File.ReadLines(retrievedFile)
                        where line.Contains("Example")
                        select new
                        {
                            File = retrievedFile,
                            Line = line
                        };
                foreach (var f in files)
                {
                    Console.WriteLine("{0} contains {1}", f.File, f.Line);
```

```
            }
            Console.WriteLine("{0} lines found.", files.Count().
            ToString());
        }
    }
}
```

The next example will show you how to transfer a directory along with all of its files to a another directory. After being relocated, the previous directory can no longer be found in its previous location.

```
using System;
using System.IO;
namespace ConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            string sourceDirectory = @"C:\source";
            string destinationDirectory = @"C:\destination";
            try
            {
                Directory.Move(sourceDirectory, destinationDirectory);
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Utilize the Directory class for common tasks like copying, moving, renaming, creating, and deleting directories. Other examples of these tasks include.

- Use one of the CreateDirectory methods to create a directory on your computer.
- Use one of the Erase ways to delete a directory from your computer.
- Use the GetCurrentDirectory or SetCurrentDirectory method in an application if you want to get or set the directory that is now active in that app.
- SetLastAccessTime and SetCreationTime are two examples of methods that can be used to manipulate DateTime information relating to the

creation, access, and writing of a directory. Other examples are GetCreationTime and GetLastAccessTime.

All of the methods in the Directory class, including the static ones, go through a series of security tests. If you intend to reuse an object more than once, you should think about employing the appropriate instance method of DirectoryInfo rather than performing the security check. This is because the check will not always be required.

If you are only going to be operating on one directory-related task, it is possible that using a static Directory method rather than the matching DirectoryInfo instance method will be more time and resource efficient. The path to the directory that you are currently modifying is required by the majority of Directory methods.

When it comes to members that accept a path, the path can either refer to a file or a directory depending on the context. For a server and share name, you have the option of utilizing either a full path, a relative path, or a Universal Naming Convention (UNC) path. For instance, the following courses of action are all appropriate options :

- "c:\\MyDir" in C#, or "c:\MyDir" in Visual Basic.
- "MyDir\\MySubdir" in C#, or "MyDir\MySubDir" in Visual Basic.
- "\\\\MyServer\\MyShare" in C#, or "\\MyServer\MyShare" in Visual Basic.

Every user has complete read and write access to new directories by default, and this setting can't be changed. However, in order to access pre-existing directories, the application needs to have the appropriate security.

Place the directory separator character at the very end of the path string when you want to require rights for a directory and all of its subdirectories. (For instance, entering "C:\Temp\" in the path bar allows access to "C:\Temp\" as well as all of its subdirectories.) Place a period at the end of the path string when you want to restrict permission requests to a certain directory. (For instance, "C:\Temp\." only provides access to C:\Temp\; it does not extend to the directory's subdirectories.)

Any combination of literal characters and the two wildcard characters, * and?, can be used as the search string in members that take a searchPattern argument. This option is not sensitive to regular expressions in any way.

13.6 DriveInfo Class :

The DriveInfo Class allows access to various pieces of information regarding a drive.

public sealed class DriveInfo : System.Runtime.Serialization.ISerializable

Example : The following snippet of code is an example of how the DriveInfo class can be used to display information about all of the drives that are currently connected to the active system.

```
using System;
using System.IO;
class Test
{
   public static void Main()
```

```
    {
        DriveInfo[] allDrives = DriveInfo.GetDrives();
        foreach (DriveInfo d in allDrives)
        {
            Console.WriteLine("Drive {0}", d.Name);
            Console.WriteLine("  Drive type: {0}", d.DriveType);
            if (d.IsReady == true)
            {
                Console.WriteLine("  Volume label: {0}", d.VolumeLabel);
                Console.WriteLine("  File system: {0}", d.DriveFormat);
                Console.WriteLine(" Available space to current user:{0, 15}
                bytes",
                    d.AvailableFreeSpace);
                Console.WriteLine("  Total available space: {0, 15} bytes",
                    d.TotalFreeSpace);
                Console.WriteLine("  Total size of drive: {0, 15} bytes ",
                    d.TotalSize);
            }
        }
    }
}/*
This code produces output similar to the following :
Drive A:\
    Drive type: Removable
Drive C:\
    Drive type: Fixed
    Volume label:
    File system: FAT32
    Available space to current user:      4770430976 bytes
    Total available space:                4770430976 bytes
    Total size of drive:                  10731683840 bytes
Drive D:\
    Drive type: Fixed
    Volume label:
    File system: NTFS
    Available space to current user:     15114977280 bytes
    Total available space:               15114977280 bytes
    Total size of drive:                 25958948864 bytes
Drive E:\
    Drive type: CDRom
```

The actual output of this code will vary based on machine and the permissions

granted to the user executing it.*/

This class represents a drive and has methods and properties that can be used to query for information about drives. DriveInfo can be used to discover what kind of drives are installed, as well as what drives are now available. You may also run a query to find out the total capacity of the drive as well as the amount of free space that is currently accessible.

❑ **Check Your Progress – 3 :**

1. Use the _____ or _____ method in an application if you want to get or set the directory.

   a. GetCurrentDirectory b. CurrentDirectory

   c. SetCurrentDirectory d. Both a and c

13.7 DirectoryInfo Class :

It provides access to instance methods for creating and navigating directories and subdirectories, as well as enumerating their contents. It is not possible to inherit from this class.

public sealed class DirectoryInfo : System.IO.FileSystemInfo

Example : The following example provides a demonstration of several of the most important components that make up the DirectoryInfo class.

```
using System;
using System.IO;
class Test
{
   public static void Main()
   {
      // Specify the directories you want to manipulate.
      DirectoryInfo di = new DirectoryInfo(@"c:\MyDir");
      try
      {
         // Determine whether the directory exists.
         if (di.Exists)
         {
            // Indicate that the directory already exists.
            Console.WriteLine("That path exists already.");
            return;
         }
         // Try to create the directory.
         di.Create();
         Console.WriteLine("The directory was created successfully.");
         // Delete the directory.
```

```
            di.Delete();
            Console.WriteLine("The directory was deleted successfully.");
        }
        catch (Exception e)
        {
            Console.WriteLine("The process failed: {0}", e.ToString());
        }
        finally {}
    }
}
```

The following example shows how to duplicate the contents of a directory as well as the directory itself.

```
using System;
using System.IO;
class CopyDir
{
    public static void CopyAll(DirectoryInfo source, DirectoryInfo target)
    {
        if (source.FullName.ToLower() == target.FullName.ToLower())
        {
            return;
        }
        // Check if the target directory exists, if not, create it.
        if (Directory.Exists(target.FullName) == false)
        {
            Directory.CreateDirectory(target.FullName);
        }
        // Copy each file into it's new directory.
        foreach (FileInfo fi in source.GetFiles())
        {
            Console.WriteLine(@"Copying {0}\{1}", target.FullName,
            fi.Name);
            fi.CopyTo(Path.Combine(target.ToString(), fi.Name), true);
        }
        // Copy each subdirectory using recursion.
        foreach (DirectoryInfo diSourceSubDir in source.GetDirectories())
        {
    DirectoryInfo nextTargetSubDir = target.CreateSubdirectory (diSource
    SubDir.Name) ;
            CopyAll(diSourceSubDir, nextTargetSubDir);
```

```
        }
    }
    public static void Main()
    {
        string sourceDirectory = @"c:\sourceDirectory";
        string targetDirectory = @"c:\targetDirectory";
        DirectoryInfo diSource = new DirectoryInfo(sourceDirectory);
        DirectoryInfo diTarget = new DirectoryInfo(targetDirectory);
        CopyAll(diSource, diTarget);
    }
    // Output will vary based on the contents of the source directory.
}
```

Use the DirectoryInfo class to perform common tasks like copying, moving, renaming, creating, and deleting directories. Other common tasks include creating new directories.

If you want to reuse an object more than once, you should think about utilizing the instance method of the DirectoryInfo class rather than the corresponding static methods of the Directory class. This is because a security check is not necessarily required to be performed by the instance method.

Members that accept a path can use that path to refer to either a file or just a directory if they so want. The path that is supplied can either refer to a relative path or a path based on the Universal Naming Convention (UNC) for a server and share name. For instance, the following courses of action are all appropriate options:

- "c:\\MyDir" in C#, or "c:\MyDir" in Visual Basic.
- "MyDir\\MySubdir" in C#, or "MyDir\MySubDir" in Visual Basic.
- "\\\\MyServer\\MyShare" in C#, or "\\MyServer\MyShare" in Visual Basic.

Every user has complete read and write access to new directories by default, and this setting can't be changed.

13.8 Creating, Copying Directory and Subdirectories :

When you are in charge of a community website where users can upload articles, forum messages, and other content, you are running a community website. When a user registers in a website, a folder may be generated at run time. After that, the data may be placed automatically within the user's own directory rather than the root directory or any other location. This may occur when data is uploaded or when a query is asked for. You are required to import the namespace located at this location:

Example :

CreateDirectory.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile= "Create Directory.aspx.cs" Inherits="CreateDirectory" %>

<%@ Register assembly="Telerik.Web.UI" namespace="Telerik.Web.UI" tagprefix="telerik" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style type="text/css">
        .style1
        {
            width: 230px;
        }
        .style2
        {
            width: 169px;
        }
        .style3
        {
            width: 230px;
            height: 26px;
        }
        .style4
        {
            width: 169px;
            height: 26px;
        }
        .style5
        {
            height: 26px;
        }
        .style6
        {
            width: 230px;
            height: 29px;
        }
        .style7
        {
            width: 169px;
            height: 29px;
        }
```

```
        .style8
        {
            height: 29px;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <h3>Create Directory</h3>
        <table style="width: 47%;">
            <tr>
                <td class="style1" style="font-weight: bold">
                    Enter File Name to Create</td>
                <td class="style2">
                    <asp:TextBox ID="txtname" runat="server"></asp:
                    TextBox>
                </td>
                <td>
                    </td>
            </tr>
            <tr>
                <td class="style3">
                </td>
                <td class="style4">
                    <telerik:RadButton ID="RadButton1" runat="server" Font-
                    Bold="True"
                        onclick="RadButton1_Click" Text="Create Directory">
                    </telerik:RadButton>
                </td>
                <td class="style5">
                    <asp:Label ID="Sucesslbl" runat="server" ForeColor=
                    "Red"></asp:Label>
                </td>
            </tr>
            <tr>
                <td class="style6" style="font-weight: bold">
                    Enter File Name to Delete</td>
```

```
            <td class="style7">
                <asp:TextBox ID="deletetxt" runat="server"></asp:
                TextBox>
            </td>
            <td class="style8">
            </td>
        </tr>
        <tr>
            <td class="style1">
            </td>
            <td class="style2">
                <telerik:RadButton ID="RadButton2" runat="server" Font-
                Bold="True"
                    onclick="RadButton2_Click" Text="Delete Directory">
                </telerik:RadButton>
            </td>
            <td>
                <asp:Label ID="Deletelbl" runat="server" ForeColor=
                "Red"></asp:Label>
            </td>
        </tr>
    </table>
</div>
</form>
</body>
</html>
```

CreateDirectory.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;        // Add This Namespace
public partial class CreateDirectory : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void RadButton1_Click(object sender, EventArgs e)
```

```
    {
        string path = @"D:\" + txtname.Text;  // Give the specific path
        if (!(Directory.Exists(path)))
        {
            Directory.CreateDirectory(path);
            Sucesslbl.Text = "Directory Created Successfully";
        }
        else
        {
            Sucesslbl.Text = "Already Directory Exits With Same Name";
        }
    }
    protected void RadButton2_Click(object sender, EventArgs e)
    {
        string path = @"D:\" + deletetxt.Text;
        if (Directory.Exists(path))
        {
            DeleteDirectory(path);
        }
        else
        {
            Deletelbl.Text = "Directory not exits";
        }
    }
    private void DeleteDirectory(string path)
    {
        // Delete all files from the Directory
        foreach (string filename in Directory.GetFiles(path))
        {
            File.Delete(filename);
        }
        // Check all child Directories and delete files
        foreach (string subfolder in Directory.GetDirectories(path))
        {
            DeleteDirectory(subfolder);
        }
        Directory.Delete(path);
        Deletelbl.Text = "Directory deleted successfully";
    }
}
```

The next example copies subdirectories by instructing the CopyDirectory method to perform recursive copying by making the recursive parameter of the method true. The CopyDirectory method copies subdirectories in a recursive manner by calling itself on each individual subdirectory until there are no more subdirectories left to copy.

```
using System.IO;
CopyDirectory(@".\", @".\copytest", true);
static void CopyDirectory(string sourceDir, string destinationDir, bool recursive)
{
    // Get information about the source directory
    var dir = new DirectoryInfo(sourceDir);
    // Check if the source directory exists
    if (!dir.Exists)
        throw new DirectoryNotFoundException($"Source directory not found: {dir.FullName}");
    // Cache directories before we start copying
    DirectoryInfo[] dirs = dir.GetDirectories();
    // Create the destination directory
    Directory.CreateDirectory(destinationDir);
    // Get the files in the source directory and copy to the destination directory
    foreach (FileInfo file in dir.GetFiles())
    {
        string targetFilePath = Path.Combine(destinationDir, file.Name);
        file.CopyTo(targetFilePath);
    }
    // If recursive and copying subdirectories, recursively call this method
    if (recursive)
    {
        foreach (DirectoryInfo subDir in dirs)
        {
            string newDestinationDir = Path.Combine(destinationDir, subDir.Name);
            CopyDirectory(subDir.FullName, newDestinationDir, true);
        }
    }
}
```

13.9 Retrieving Files from a Directory :

In this section, we will learn how to retrieve files from a folder or directory using C# in ASP.NET and bind them to a Gridview. It is necessary to write the code in the format indicated below in order for Gridview to display files from a folder.

If you want to check it out in a more detailed example, put down the code below.

HTML Source Code

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Get files from folder & bind to gridview in c#.net</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:Button ID="btnGetFiles" Text="Get Files From Folder" runat="server"
onclick = "btnGetFiles_Click" />
<asp:GridView ID="gvDetails" CellPadding="5" runat="server"
AutoGenerateColumns = "false">
<Columns>
<asp:BoundField DataField="Text" HeaderText="FileName" />
</Columns>
<HeaderStyle BackColor="#df5015" Font-Bold="true" ForeColor="White"
/>
</asp:GridView>
</div>
</form>
</body>
</html>
```

Example : C# Source Code

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Web.UI.WebControls;
/ insert files in folder
protected void btnGetFiles_Click(object sender, EventArgs e)
{
BindGridview();
}
// Bind Data to Gridview
protected void BindGridview()
```

```
{
string[] filesPath = Directory.GetFiles(Server.MapPath("~/SampleFiles/"));
List<ListItem> files = new List<ListItem>();
foreach (string path in filesPath)
{
files.Add(new ListItem(Path.GetFileName(path)));
}
gvDetails.DataSource = files;
gvDetails.DataBind();
}
```

13.10 Files : Creating, Copying, Reading, Appending, Renaming, Compressing :

The following examples demonstrate how to perform synchronous file and folder operations such as copying, moving, and deleting by utilizing the:

- System.IO.File,
- System.IO.Directory,
- System.IO.FileInfo, and
- System.IO.DirectoryInfo classes from the System.IO namespace.

Use System.IO. FileSystemWatcher will supply events that will let you calculate the progress when working on several files, and it will do this for you. Using platform invoke to call the appropriate file-related routines in the Windows Shell is an additional option that can be utilized.

Example : The subsequent example demonstrates how to copy both files and directories.

```
// Simple synchronous file copy operations with no user interface.
// To run this sample, first create the following directories and files:
// C:\Users\Public\TestFolder
// C:\Users\Public\TestFolder\test.txt
// C:\Users\Public\TestFolder\SubDir\test.txt
public class SimpleFileCopy
{
    static void Main()
    {
        string fileName = "test.txt";
        string sourcePath = @"C:\Users\Public\TestFolder";
        string targetPath =  @"C:\Users\Public\TestFolder\SubDir";
        // Use Path class to manipulate file and directory paths.
        string sourceFile = System.IO.Path.Combine(sourcePath, fileName);
        string destFile = System.IO.Path.Combine(targetPath, fileName);
        // To copy a folder's contents to a new location:
```

```
            // Create a new target folder.
            // If the directory already exists, this method does not create a
            new directory.
            System.IO.Directory.CreateDirectory(targetPath);
            // To copy a file to another location and
            // overwrite the destination file if it already exists.
            System.IO.File.Copy(sourceFile, destFile, true);
            // To copy all the files in one directory to another directory.
            // Get the files in the source folder. (To recursively iterate through
            // all subfolders under the current directory, see
            // "How to: Iterate Through a Directory Tree.")
            // Note: Check for target path was performed previously
            //       in this code example.
            if (System.IO.Directory.Exists(sourcePath))
            {
                string[] files = System.IO.Directory.GetFiles(sourcePath);
                // Copy the files and overwrite destination files if they already
                exist.
                foreach (string s in files)
                {
                    // Use static Path methods to extract only the file name from
                    the path.
                    fileName = System.IO.Path.GetFileName(s);
                    destFile = System.IO.Path.Combine(targetPath, fileName);
                    System.IO.File.Copy(s, destFile, true);
                }
            }
            else
            {
                Console.WriteLine("Source path does not exist!");
            }
            // Keep console window open in debug mode.
            Console.WriteLine("Press any key to exit.");
            Console.ReadKey();
        }
    }
```

The following example demonstrates how to relocate files and folders in your computer.

```
// Simple synchronous file move operations with no user interface.
public class SimpleFileMove
{
    static void Main()
    {
        string sourceFile = @"C:\Users\Public\public\test.txt";
        string destinationFile = @"C:\Users\Public\private\test.txt";
        // To move a file or folder to a new location:
        System.IO.File.Move(sourceFile, destinationFile);
        // To move an entire directory. To programmatically modify or
        combine
        // path strings, use the System.IO.Path class.
    System.IO.Directory.Move(@"C:\Users\Public\public\test\",@"
    C:\Users\ Public\private");
    }
}
```

The following example demonstrates how to delete files and directories in our computer.

```
// Simple synchronous file deletion operations with no user interface.
// To run this sample, create the following files on your drive:
// C:\Users\Public\DeleteTest\test1.txt
// C:\Users\Public\DeleteTest\test2.txt
// C:\Users\Public\DeleteTest\SubDir\test2.txt
public class SimpleFileDelete
{
    static void Main()
    {
        // Delete a file by using File class static method...
        if(System.IO.File.Exists(@"C:\Users\Public\DeleteTest\test.txt"))
        {
            // Use a try block to catch IOExceptions, to
            // handle the case of the file already being
            // opened by another process.
            try
            {
                System.IO.File.Delete(@"C:\Users\Public\DeleteTest\test.
                txt");
            }
            catch (System.IO.IOException e)
            {
```

```
            Console.WriteLine(e.Message);
            return;
        }
    }
    // ...or by using FileInfo instance method.
System.IO.FileInfo fi = new System.IO.FileInfo(@"C:\Users\Public
\Delete Test\test2.txt");
    try
    {
        fi.Delete();
    }
    catch (System.IO.IOException e)
    {
        Console.WriteLine(e.Message);
    }
    // Delete a directory. Must be writable or empty.
    try
    {
        System.IO.Directory.Delete(@"C:\Users\Public\DeleteTest");
    }
    catch (System.IO.IOException e)
    {
        Console.WriteLine(e.Message);
    }
    // Delete a directory and all subdirectories with Directory static
    method...
    if(System.IO.Directory.Exists(@"C:\Users\Public\DeleteTest"))
    {
    try
    {
        System.IO.Directory.Delete(@"C:\Users\Public\DeleteTest", true);
    }
    catch (System.IO.IOException e)
    {
        Console.WriteLine(e.Message);
    }
}
    // ...or with DirectoryInfo instance method.
System.IO.DirectoryInfo di = new System.IO.DirectoryInfo (@ "C:\Users
\Public\public");
```

```
            // Delete this dir and all subdirs.
            try
            {
                di.Delete(true);
            }
                catch (System.IO.IOException e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
```

13.11 Let Us Sum Up :

Here in this unit we have learnt about the filestreams. We studied about the System.IO Namespace. We have studied about the working with Drives using ASP.NET, also studied about the Directories, Files.

We have learnt about the properties of ACL to secure files and directories. We got the knowledge of various classes like Directory Class, DriveInfo Class, DirectoryInfo Class. We got to know how to create directory, how to copy the contents of one directory to another and about the subdirectories.We have also studied about how do we retrieve Files from a Directory. We have also learnt how to create new file, how to copy the contents of one file to another file, how do we read the contents of files in ASP.NET using C#. We got the knowledge about to append the multiple files. We can also rename the created filed through ASP.NET. Through ASP.NET we can even compress the files using suitable properties and methods for it.

13.12 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1 : d **2 :** c

- **Check Your Progress 2 :**

  1 : c **2 :** b

- **Check Your Progress 3 :**

  1 : d

13.13 Glossary :

1 **File :** A file is an object on a computer that stores data, information, settings, or commands used with a computer program.

2. **Directory :** It is a file system cataloging structure which contains references to other computer files, and possibly other directories.

3. **Drive :** A computer device that stores and retrieves information, data, files, programs, etc., from a disk.

4. **System.IO :** Provides properties and instance methods for the creation, copying, deletion, moving, and opening of files, and aids in the creation of FileStream objects.

13.14 Assignment :

1. Explain various classes included in the System.IO Namespace.

13.15 Activities :

1. Explain the working of Drives, Files and Directories.

13.16 Case Study :

Create a web application and try to use System.IO classes in it for creating 2 files. Add few content in a file and copy that content into another file.

13.17 Further Readings :

1. Anne Boehm, Murachs, ASP.NET Web Programming with VB 2008, Mike Murach and Associates
2. Stephen Walther, Data Access in ASP.NET Framework, 2007, Sams Publishing
3. Israel B. Ocbina, Mastering VB.NET and C#, 2004, Cyberocbina
4. Kogent Learning Solutions Inc.. ASP.NET 4.5 Black Book, 2013, Dreamtech
5. ASP.Net 4 Unleashed (English, Electronic book text, Walther Stephen)

Unit
14

APPLICATIONS

UNIT STRUCTURE

14.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Step-by-step instructions on how to develop an application on the ASP.NET framework utilising the SQL Server database as the data source.
- Detailed instructions on how to make use of any one of our many online applications so that you can successfully add a new entry to the database.
- Instructions on how to use our ASP.NET web apps to do the four tasks of "Create," "Read," "Update," and "Delete" (CRUD)
- With regard to the methods by which our website application and our database might be linked to one another online.
- Detailed instructions on how to link data to datacontrols like the GridView are provided here.
- With regard to the methods that can be utilised in order to dynamically populate the dropdownlists with data taken from the database tables.

14.1 Introduction :

The process of developing web applications begins with the incorporation of web forms into an application that will ultimately be utilised by users. This is the first phase in the process of designing web applications. The first stage of the design process has now been entered into the procedure. After that, the controls are added to the forms, and in the end, the computer responds to how the users interact with the controls that they have been given based on how they have interacted with the controls that they have been given. This occurs because the computer remembers how the users have interacted with the controls that they have been given. It is usual practise to work in a development environment that makes use of the Visual Studio.NET platform when developing a web application. This is because Visual Studio.NET is the most widely used development platform. This is due to the fact that Visual Studio.NET is the platform for development that is utilised the most. This is because Visual Studio.NET is the platform for development that is utilised the majority of the time. The reason for this is given in the previous sentence. It is possible to use it to construct a wide range of online apps, web services that allow

access to data, and it can be used to make use of both of these things. It is also possible to use it to make use of both of these things. It is also possible to use it to make advantage of both of these things, which is a useful feature. An additional benefit is the fact that it can be utilised to make use of each of these different things, which is a key advantage. In addition to that, by making use of it during the design process for either of these items, it is feasible to include it in the process somehow. Active Server Pages (ASP.NET) is an environment for compiled programming that can be utilised in the process of creating web pages. This environment can be found on Microsoft's ASP.NET platform. This environment can be discovered by looking for it under the Active Server Pages hood. It was constructed from the ground up using the.NET framework as its foundation. Active Server Pages is what the "ASP.NET" acronym refers to, which is also what the full name of the technology stands for. The abbreviation ASP.NET stands for Active Server Pages.

14.2 Employee Application :

New hires will have access to fill out the web form with their personal information by using this web application, which will be made available to them. After this information has been entered, a new database table with the heading "Employee" will be created and saved to store it. This table will include the information that was just entered. When the Employee table is established, these are the fields that will be included in it; once the table is built, the information pertaining to freshly recruited employees will be stored in it using these fields.:

- EmpCode,
- Name,
- Address,
- DeptCode,
- StateCode and
- CityCode.

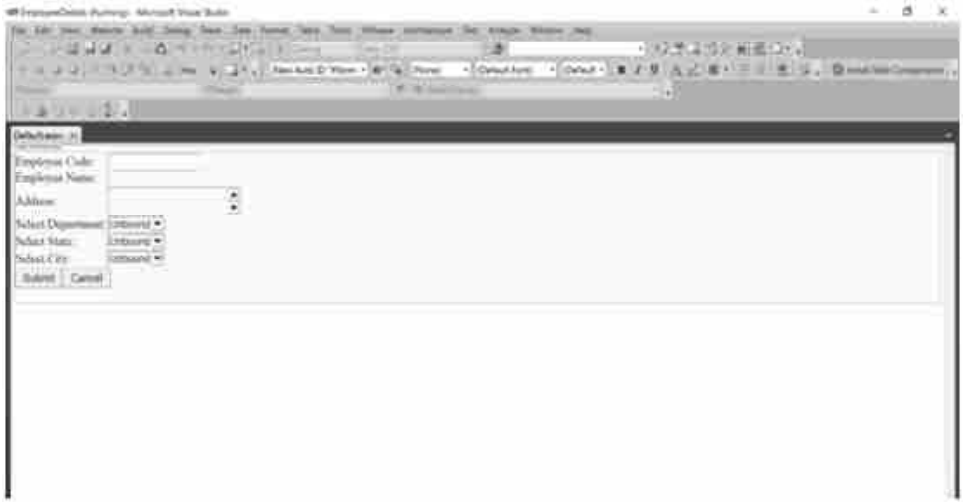
The EmpCode, which should be formed automatically and have the form E001, E002,... E010, and so on and so forth, should be generated. immediately after the user pressed the insert button on the keyboard in order to put anything into the system. immediately after, or as soon as something was added to the system by the user, whichever comes first.

Depending on which of the following options the employee chooses, we are able to retrieve codes.:

- state,
- city, and
- department

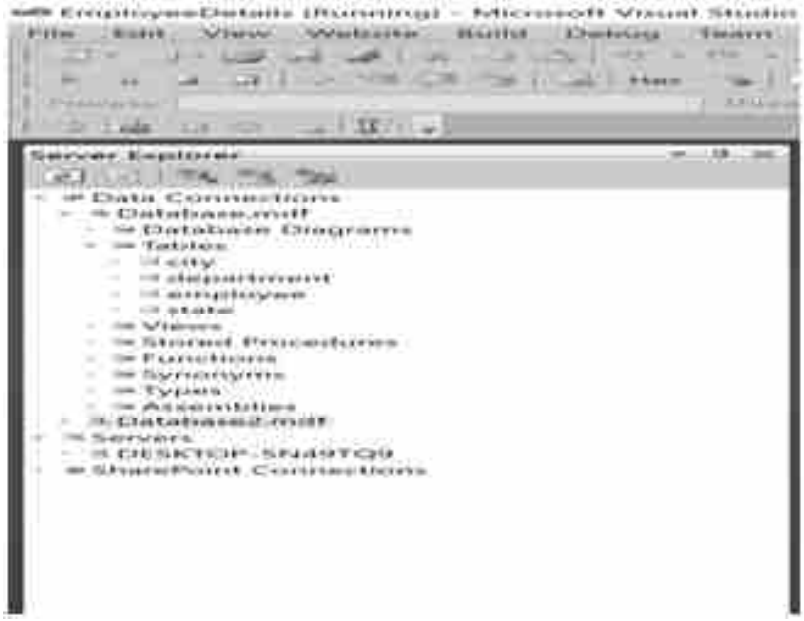
We will be utilizing the DropDownList control so that we can make our selections for the Department, City, and State. A list of the departments, cities, and states will be displayed for the user to look through whenever this control is utilized. This list will be compiled with the help of the data tables that have been labelled with the headings of department, city, and state in the order that those headings occur in the tables. Once we put the code into action, the dropdownlist controls that we employ will already have their various possibilities pre-populated.

To add all of these particulars to the Employee table, you need do nothing more than click the "Insert" button an extra time.



Database : Database.mdf

Tables : city, department, employee and state



Database : Database.mdf

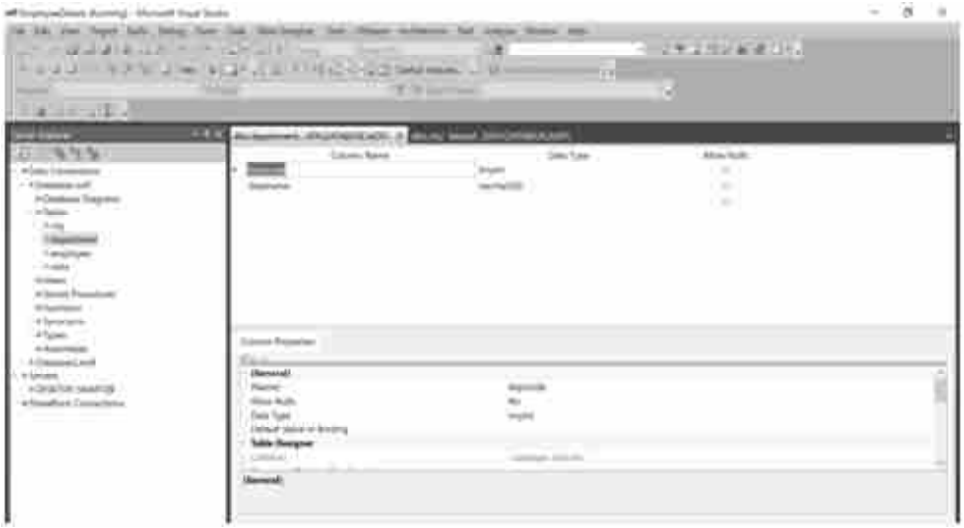
Table : city

Fields : citycode, cityname and sid (i.e. state ID)

Database : Database.mdf

Table : depatment

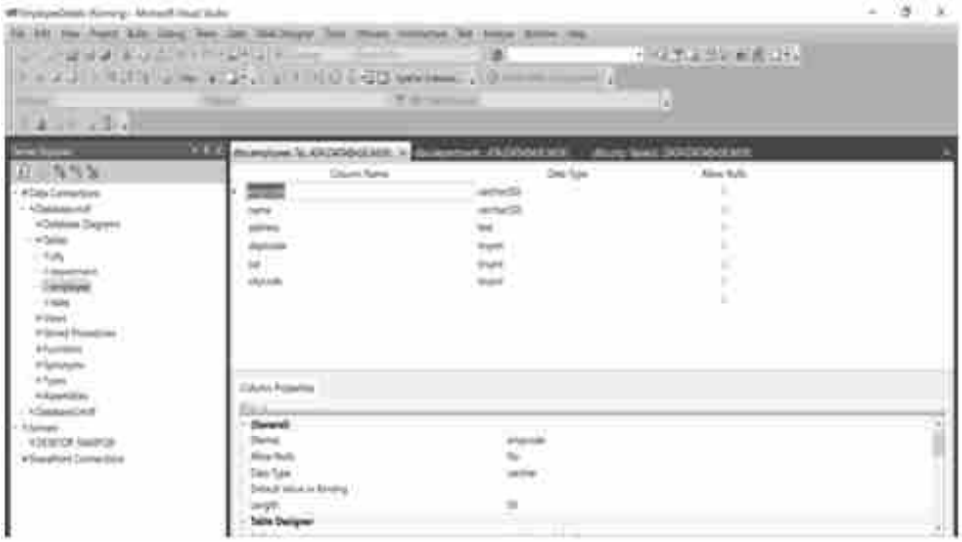
Fields : deptcode, deptname



Database : Database.mdf

Table : employee

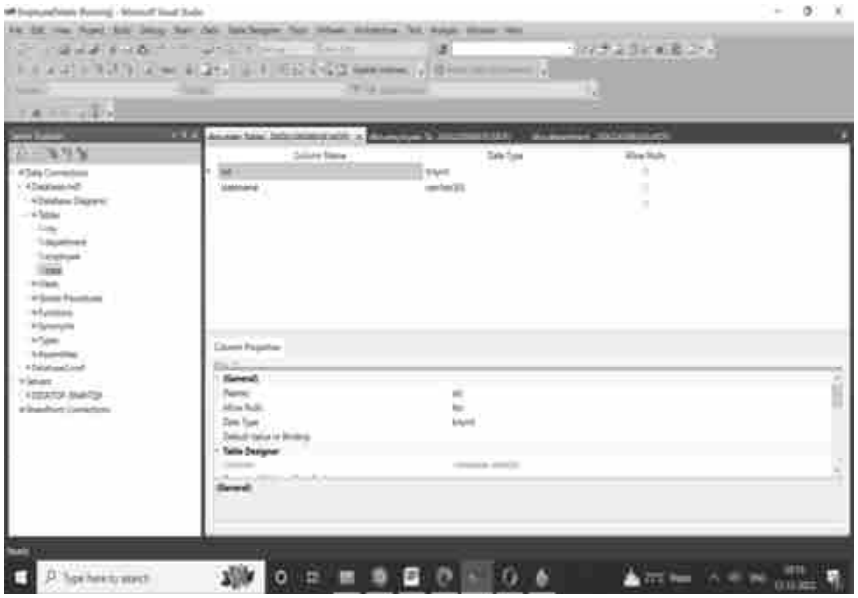
Fields : empcode, name, address, deptcode, sid and citycode



Database : Database.mdf

Table : state

Fields : sid and statename



When we have finished entering all of the employee details and have pressed the Insert Button, the information will be saved in the database, and a message will be displayed at the top of the screen, as seen in the screen below : [Insert Button]

Following the completion of the data insertion, we can now open the employee table and examine the data. The details of the newly received employee will be written down in the table.

This is the designing code, and the source code, often known as the "Code behind," is supplied for your convenience. You should give it a shot on your own personal computer and check to see if the data is being inserted correctly into the database.

Design Code :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

<style type="text/css"">

        .style1{width:100;}

        .style2{}

    </style>

</head>

<body>

    <form id="form1" runat="server">

    <table class="style1">

        <tr>

            <td class="style2">

                Employee Code:

            </td>

            <td>

                <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

            </td>

        </tr>

        <tr>

            <td class="style2">

                Employee Name:

            </td>

            <td>

                <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>

            </td>

        </tr>

        <tr>

            <td class="style2">
```

```
            Address :
        </td>
        <td>
            <asp:TextBox ID="TextBox3" runat="server" Height="63px"
TextMode="MultiLine"></asp:TextBox>
        </td>
    </tr>
    <tr>
        <td class="style2">
            Select Department:
        </td>
        <td>
            <asp:DropDownList ID="DropDownList1" runat="server">

            </asp:DropDownList>
        </td>
    </tr>
    <tr>
        <td class="style2">
            Select State:
        </td>
        <td>
            <asp:DropDownList ID="DropDownList2" runat="server"
                onselectedindexchanged="DropDownList2_SelectedIndex
                Changed">

            </asp:DropDownList>
        </td>
    </tr>
    <tr>
        <td class="style2">
            Select City:
        </td>
        <td>
            <asp:DropDownList ID="DropDownList3" runat="server">

            </asp:DropDownList>
        </td>
    </tr>
    <tr>
```

```
            <td class="style2" colspan="2">
                <asp:Button ID="Button1" runat="server" Text="Submit"
onclick="Button1_Click" />
                <asp:Button ID="Button2" runat="server" Text="Cancel" />
            </td>
        </tr>
    </table>
    <div>
            </div>
    </form>
</body>
</html>
```

C# Source Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;

public partial class _Default : System.Web.UI.Page
{
    SqlConnection cn = new SqlConnection("Data Source=
.\\SQLEXPRESS;AttachDbFilename=C:\\Users\\BAPS\\Documents\\Visual
Studio 2010\\WebSites\\EmployeeDetails\\App_Data\\Database.mdf;
Integrated Security=True;User Instance=True");
    SqlCommand cmd;
    SqlDataAdapter da = new SqlDataAdapter();
    DataSet ds = new DataSet();

    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            cn.Open();
            DropDownList1.Items.Clear();
            String str = "Select * from department";
            cmd = new SqlCommand(str, cn);
```

```
        SqlDataReader dr = cmd.ExecuteReader();
        while (dr.Read())
        {
            DropDownList1.Items.Add(dr[1].ToString());
        }
        cn.Close();
        DropDownList2.Items.Clear();
        cn.Open();
        String str2 = "Select * from state";
        cmd = new SqlCommand(str2, cn);
        SqlDataReader dr2 = cmd.ExecuteReader();
        while (dr2.Read())
        {
            DropDownList2.Items.Add(dr2[1].ToString());
        }
        cn.Close();
        cn.Open();
        DropDownList3.Items.Clear();
        String str3 = "SELECT * FROM  city WHERE sid = 1";
        cmd = new SqlCommand(str3, cn);
        SqlDataReader dr3 = cmd.ExecuteReader();
        while (dr3.Read())
        {
            DropDownList3.Items.Add(dr3[1].ToString());
        }
        cn.Close();
    }
}
protected void DropDownList2_SelectedIndexChanged(object sender,
EventArgs e)
{
    cn.Open();
    DropDownList3.Items.Clear();
    String str = "Select * from city c, state s where c.sid=s.sid and
s.name='" + DropDownList2.SelectedItem.ToString() + "'";
    cmd = new SqlCommand(str, cn);
    SqlDataReader dr = cmd.ExecuteReader();
    while (dr.Read())
    {
        DropDownList3.Items.Add(dr[1].ToString());
```

```
    }
    cn.Close();
}

protected void Button1_Click(object sender, EventArgs e)
{
    cn.Open();
    Int32 sid, citycode, deptcode;
    String str = "Select citycode from city where cityname='" +
DropDownList3.SelectedItem + "'";
    cmd = new SqlCommand(str, cn);
    citycode = Convert.ToInt32(cmd.ExecuteScalar());
    cn.Close();
    cn.Open();
    String str2 = "Select deptcode from department where deptname='"
+ DropDownList1.SelectedItem + "'";
    cmd = new SqlCommand(str2, cn);
    deptcode = Convert.ToInt32(cmd.ExecuteScalar());
    cn.Close();
    cn.Open();
    String str3 = "Select sid from state where statename='" + DropDown
List2.SelectedItem + "'";
    cmd = new SqlCommand(str3, cn);
    sid = Convert.ToInt32(cmd.ExecuteScalar());
    cn.Close();
    cn.Open();
    String empcode = "";
    String str4 = "Select max(empcode) from employee";
    cmd = new SqlCommand(str4, cn);
    empcode = Convert.ToString(cmd.ExecuteScalar());
    cn.Close();
    if (empcode == "")
    {
        empcode = "E001";
    }
    else
    {
        Int32 newcode;
        newcode = Convert.ToInt32(empcode.Substring(1));
        newcode += 1;
```

```
        if (newcode.ToString().Length < 2)
    {
    empcode = "E00" + newcode;
    }
    else
    {
        empcode = "E0" + newcode;
        }
    }
    cn.Open();
    String str5 = "Insert into employee values('" + empcode + "','" +
TextBox2.Text + "','" + TextBox3.Text + "','" + deptcode + "','" + citycode
+ "','" + sid + "')";
    cmd = new SqlCommand(str5, cn);
    Response.Write(str5);
    cmd.ExecuteNonQuery();
    cn.Close();
    Response.Write("Record saved successfully");
    }
}
```

14.3 CRUD Operation Application :

4.3

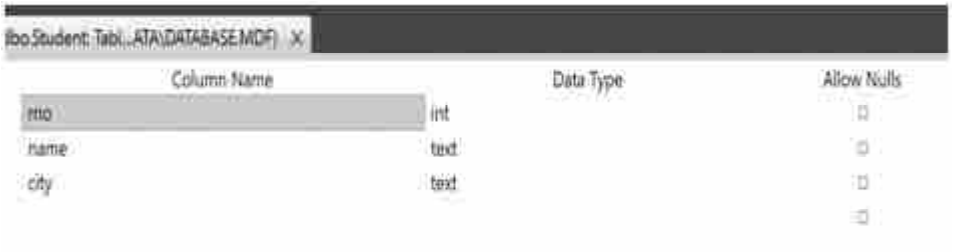
Create a programme that will run on a website by using code that was created in the ADO.NET language. Users of this application should be able to select anything from a database table and then insert, update, or delete records based on their choices using the functionality that this application provides. It is necessary for the users of this application to be able to select many entries at the same time. Users of the application should be able to select several table items at once from the available options in the program's table. It is hoped that users, as a direct result of utilising the application in question, will be able to achieve the aforementioned capability. In addition to that, users of the website should have the ability to delete their own records whenever and however they see fit, and they should be able to do so at any time. This option should be made available to them via the website.

In order to begin utilizing this web application, you will first need to establish a database and give the newly constructed table the name "Students." After that, you can begin using the programme. After that, you will be able to start utilizing the programme. When you have completed this procedure, you will then be able to make use of the application. As can be seen in this screen capture, the screen capture that was shown earlier used to illustrate this particular topic.

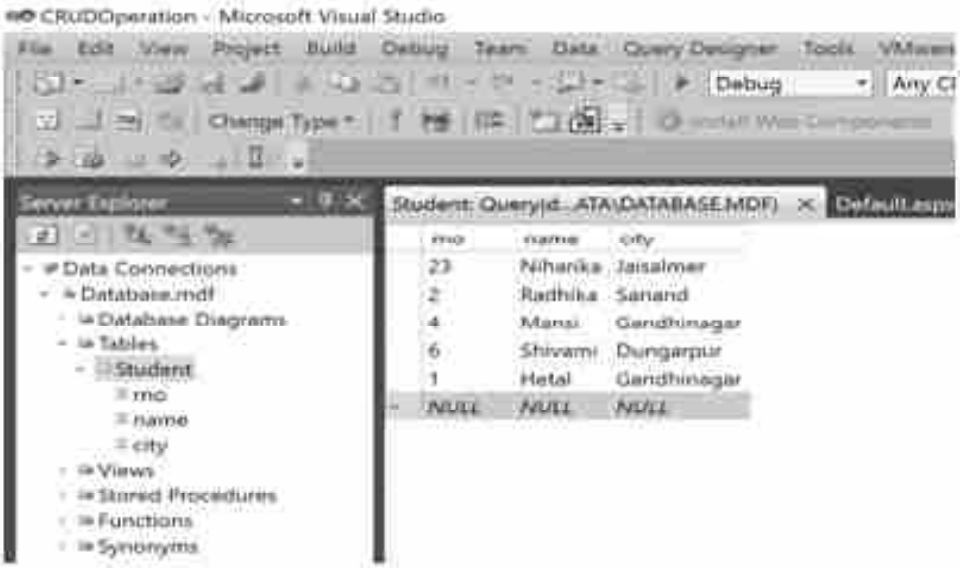
Database : Database.mdf

Table : Student

Fields : rno, name and city



Data in the Student Table :



Design of the WebForm namedDefault.aspx. In this we have used following controls :

- **GridView Control :** For binding data from the database, displaying the data in the Grid and do the Create, Read, Update and Delete operations on it.
- **Labels :** Labels are used to represent the names for inserting the values in the textboxes respectively.
- **Textboxes :** To insert the new data in it. The inserted data will be inserted in the "Student Table" in the database using ExecuteNonQuery. To modify, delete or read the data too these textboxes are used.
- **Buttons :** By clicking on these buttons all operations will be done accordingly.

Once the data will bind with the GridView Control using it's properties and methods. The data will we shown like this on the browser.

If we want to update any record from the database, we need to enter the Roll No. in the txtrno textbox. The entered Roll No. should be from rno column which is shown in the GridView. Once you will enter it modify the name as shown in the below figure. I have written Hetal Bhavsar at the place of Hetal and clicked on Update Button.

After clicking it, data particular record will be updated in the Student Table.

After updating the name it will look like this. We have to refresh the screen.

Same way we can enter the roll no and click on Delete Button to remove the particular record from the Student Table. We can insert new records too from our Web Application by entering new data in all textboxes and need to click on Insert Button. It will insert the data in the Student table.

When we click on the Button named First, it will show the first record of the Student Table. When we click on the Button named Last, it will show the last record which we entered in our Student table.

We can view all records on the same screen in the GridView. But if we want to see one by one record then we can see in the textboxes using Buttons which we have used in this application as Navigation Buttons like, First, Last, Next and Previous. The Design Code and Source code for the same is given below.

Now write this HTML Source Code for your WebPage named "Default.aspx"

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile=
"Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
<table border="2px"><tr><td align="center" colspan="2">CRUD
Operations</td> </tr> </table>
    <asp:GridView ID="GridView1" runat="server" CellPadding="4"
    ForeColor="#333333"
        GridLines="None">
        <AlternatingRowStyle BackColor="White" />
        <EmptyDataRowStyle BackColor="#2461BF" />
        <FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor=
        "White" />
        <HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor=
        "White" />
        <PagerStyle BackColor="#2461BF" ForeColor="White" Horizontal
        Align="Center" />
        <RowStyle BackColor="#EFF3FB" />
    <SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True"
    ForeColor="#333333" />
        <SortedAscendingCellStyle BackColor="#F5F7FB" />
        <SortedAscendingHeaderStyle BackColor="#6D95E1" />
        <SortedDescendingCellStyle BackColor="#E9EBEF" />
        <SortedDescendingHeaderStyle BackColor="#4870BE" />
    </asp:GridView>
    <br />
    <asp:Label ID="Label1" runat="server" Text="Roll No:"></asp:Label>
    <asp:TextBox ID="txtrno" runat="server"></asp:TextBox>
    <br />
    <asp:Label ID="Label2" runat="server" Text="Name:"></asp:Label>
    <asp:TextBox ID="txtname" runat="server"></asp:TextBox>
    <br />
    <asp:Label ID="Label3" runat="server" Text="City:"></asp:Label>
```

```
<asp:TextBox ID="txtcity" runat="server"></asp:TextBox>
<br />
<asp:Button ID="btninsert" runat="server" Text="Insert"
   onclick="btninsert_Click" />
<asp:Button ID="btnupdate" runat="server" Text="Update"
   onclick="btnupdate_Click" />
<asp:Button ID="btndelete" runat="server" Text="Delete"
   onclick="btndelete_Click" />
<asp:Button ID="btnselect" runat="server" Text="Select"
   onclick="btnselect_Click" />
<br />
<asp:Button ID="btnfirst" runat="server" Text="First"
   onclick="btnfirst_Click" />
<asp:Button ID="btnnext" runat="server" Text="Next" onclick=
"btnnext_Click" />
<asp:Button ID="btnprevious" runat="server" Text="Previous"
   onclick="btnprevious_Click" />
<asp:Button ID="btnlast" runat="server" Text="Last" onclick=
"btnlast_Click" />
</div>
</form>
</body>
</html>
```

Now write this C# Source Code (CodeBehind) for your WebPage named "Default.aspx.cs"

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.IO;
public partial class _Default : System.Web.UI.Page
{
SqlConnection cn = new SqlConnection("Data Source=.\\SQLEXPRESS;
AttachDbFilename=C:\\Users\\BAPS\\Documents\\Visual Studio 2010\\WebSites
\\CRUDOperation\\App_Data\\Database.mdf;Integrated Security=True;User
Instance=True");
```

```
static int recptr = 0;
static DataTable dt = new DataTable();
SqlDataAdapter da = new SqlDataAdapter();
DataSet ds = new DataSet();
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        recptr = 0;
        fillTextbox();
        fillGrid();
        }
    }
    public void fillGrid()
    {
        cn.Open();
        SqlCommand cmd = new SqlCommand("select * from student", cn);
        da.SelectCommand = cmd;
        da.Fill(ds);
        GridView1.DataSource = ds.Tables[0];
        GridView1.DataBind();
        cn.Close();
    }
    public void clearTextbox()
    {
        txtrno.Text = "";
        txtname.Text = "";
        txtcity.Text = "";
    }
    protected void btninsert_Click(object sender, EventArgs e)
    {
        String str;
        cn.Open();
        str="insert into student values('"+txtrno.Text+"','"+txtname.Text+"',
        '"+txtcity.Text+"')";
        SqlCommand cmd = new SqlCommand(str, cn);
        cmd.ExecuteNonQuery();
        cn.Close();
        fillGrid();
    }
```

```
protected void btnupdate_Click(object sender, EventArgs e)
{
    String str;
    cn.Open();
    str = "update student set name='" + txtname.Text + "',city='" +
    txtcity.Text + "' where rno=" +txtrno.Text;
    SqlCommand cmd = new SqlCommand(str, cn);
    cmd.ExecuteNonQuery();
    cn.Close();
    fillGrid();
    clearTextbox();
}
protected void btndelete_Click(object sender, EventArgs e)
{
    String str;
    cn.Open();
    str = "delete from student where rno=" + txtrno.Text;
    SqlCommand cmd = new SqlCommand(str, cn);
    cmd.ExecuteNonQuery();
    cn.Close();
    fillGrid();
    clearTextbox();
}
protected void btnselect_Click(object sender, EventArgs e)
{
    String str;
    cn.Open();
    str = "select * from student where rno=" + txtrno.Text;
    SqlCommand cmd = new SqlCommand(str, cn);
    SqlDataReader dr;
    dr = cmd.ExecuteReader();
    if (dr.Read())
    {
        txtrno.Text = dr[0].ToString();
        txtname.Text = dr[1].ToString();
        txtcity.Text = dr[2].ToString();
    }
    cn.Close();
}
```

```
    public void fillTextbox()
    {
        dt.Clear();
        da = new SqlDataAdapter("select * from student", cn);
        da.Fill(dt);
        txtrno.Text = dt.Rows[recptr][0].ToString();
        txtname.Text = dt.Rows[recptr][1].ToString();
        txtcity.Text = dt.Rows[recptr][2].ToString();
    }
    protected void btnfirst_Click(object sender, EventArgs e)
    {
        recptr = 0;
        fillTextbox();
    }
    protected void btnnext_Click(object sender, EventArgs e)
    {
        if (recptr < dt.Rows.Count - 1)
        {
            recptr++;
            fillTextbox();
        }
    }
    protected void btnprevious_Click(object sender, EventArgs e)
    {
        if (recptr > 0)
        {
            recptr--;
            fillTextbox();
        }
    }
    protected void btnlast_Click(object sender, EventArgs e)
    {
        recptr = dt.Rows.Count - 1;
        fillTextbox();
    }
}
```

14.4 Let Us Sum Up :

After completing this programme, you should have a much deeper understanding of how to construct online applications by utilising ASP.NET and C#. This should be the case because the programme is designed to teach you these skills. Because you will have learned a significant amount of new information, this ought to be the case. This should be the case because you have completed all of the tasks that were required during the session, so it is reasonable to expect that it has already come to an end. At this point, each one of us has an in-depth grasp of the processes that need to be carried out in order to make a connection between our application and the database. This is because we have been working on this problem for quite some time. Utilizing the SQLDataSource Control was what ultimately allowed us to successfully establish a connection to the database on our end. There is a chance that you won't be able to complete this task. We were able to successfully retrieve data from the database as a direct consequence of this. We made use of a component known as the GridView Control in order to present the data records in a format that was readable on the screen. Because of this, we were able to carry out the task in an organised fashion. This was finished with the intention of making our experience more pleasant. Our team has designed a wide range of distinct components, including, amongst other things, text boxes and buttons. Our group fabricated each of these individual parts. The users of our website have access to buttons that enable them to make changes to previously saved records, delete entries that are already in the database, and add new records to the database. These functions are all available to them through our website. These tasks can be accomplished by going to the appropriate part of the database and then selecting the corresponding button on the screen. When users visit our website, they are granted access to each of these functions in their entirety. This provides some proof that the CRUD operations have been incorporated into our programme in some manner, shape, or form. [Create, Read, Update, and Delete]

BLOCK SUMMARY :

This block covered user account creation and management. Learned how to give users Membership and Roles. How to set user access rules. We also learned how to create website user accounts. We also learned password recovery, password generation, and required email attributes.

This block taught us about Website Login Controls, Logout, and Logout Properties. We learned how to use Login view and Login Name Web Controls. Change password Control was learned. We studied about MasterPages and their importance in our online apps. This taught us about Master Pages and themes for web pages.

This block has covered configuring Master Pages and Themes. We studied modifying master page and theme material. We also studied Master Pages, Events, and topics. This block covered filestreams. We learned about System.IO. We explored ASP.NET Drives, Directories, and Files.

We learned how ACL secures files and folders. We learned Directory, DriveInfo, and DirectoryInfo classes. We learned how to construct and copy directories, subdirectories, and how to get files from a directory. We also learned how to create new files, copy their contents, and read them in ASP.NET using C#. Appending multiple files was learned. ASP.NET allows field renaming. ASP.NET's properties and methods can compress files.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. What is ACL ?
2. Explain the working with Login view and Login Name Web Controls.
3. Write a note on DirectoryInfo Class ?
4. Write short note on System.IO namespace ?

❖ **Long Questions :**

1. Explain how to create Accounts Via Websites in detail.
2. Explain various Properties and Methods of Master Pages and Themes.
3. Explain how to copy and retrieve Files from a Directory in detail.

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	11	12	13	14
No. of Hrs.				

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....

DR.BABASAHEB AMBEDKAR
OPEN UNIVERSITY

'Jyotirmay' Parisar,
Sarkhej-Gandhinagar Highway, Chharodi, Ahmedabad-382 481.
Website : www.baou.edu.in