B.C.A, B.Sc (Computer Science/IT) ,MCA
& Other Related Diploma & Degree Courses

ASP.NET

Nikisha B. Jariwala
Asst. Professor

Bharat C. Patel
Director-In-Charge

Smt. Tanuben & Dr. Manubhai Trivedi
College of Information Science-Surat

Benison Education

B/10,Suranagar,Nr.Citizen Society, Ellora Park, Vadodara-390 023
Gujarat, India. Mobile:+91-9825377117

E-MAIL : benisonedu@gmail.com URL : www.benisoneducation.com

Benison Education

ASP.NET

ISBN 13: 978-81-88894-35-2

Published by : Rahul Desai
"Benison Education"
B/10, Suryanagar, Nr. Citizen Society,
Ellora Park, Vadodara- 390 023.
Gujarat, INDIA
Mobile: +91-98253 77117
Email: info@benisoneducation.com/ benisonedu@gmail.com
URL: www.benisoneducation.com

Printed at : Ajay Offset, Ahmedabad

Typeset at : Sai Computer, Vadodara

Preface

This is a book for people who are interested in learning web based technology ASP.NET. It represents an extended and thoroughly revised version of a collection of topics on ASP.NET.

This book takes the reader quite far into the world of ASP.NET with VB language, but before that, it presents some elementary topics related to .NET. This book does not assume any previous knowledge about .NET. So this book is intended for beginners in the field of web based programming using ASP.NET.

We have, though, assumed that a person who reads this book has a basic knowledge of Programming and other subjects equivalent to that.

If a person learning with this book wants to do practical example, then he / she should have computer with Windows Operating System and Visual Studio 2008 installed on it.

The text, figures, tables, and examples are presented in a unique and most readable manner in this book. The screen shots within examples are explained with text balloons.

This book should be somewhat informative in its own way because the examples in the book have been carefully written and selected.

We have written this book because we believe that the way of presentation within this book makes learning easier for beginners. We also believe that the examples written in this book are easy to understand. Therefore, we think that this book is an excellent way to study ASP.NET.

we welcome your valuable suggestions and feedback to improve content, you may send the reviews on benisonedu@gmail.com

Nikisha B. Jariwala
Bharat C. Patel

Acknowledgements

We would wish to Thank our family members for their encouragement and support while we were busy preparing this book. We also thank our student, colleagues and our college management for their support and encouragement. Our sincere Thanks to Benison Education for their Perseverance, Enthusiasm and Dedication, which made this book as a Reality.

Nikisha B. Jariwala
Bharat C. Patel

Index

Dedication

This book is dedicated to my Guruji-"Vasudada". Thank you for giving me inspiration.

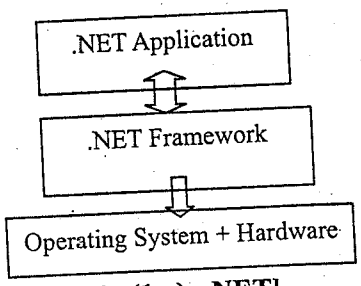
- Nikisha B. Jariwala

This book is dedicated to my parents -
Late Chimanbhai J. Patel and Smt. Parvatiben C. Patel.

-Bharat C. Patel

1 Overview of Microsoft .NET Framework

First let's see what is .NET? **.NET** is Microsoft's strategy of software that provides services to people any time, any place, on any device. **.NET** is the Framework or we can say "Software Platform". It comes with many rich set of tools like Visual Studio to fully develop and build those applications.

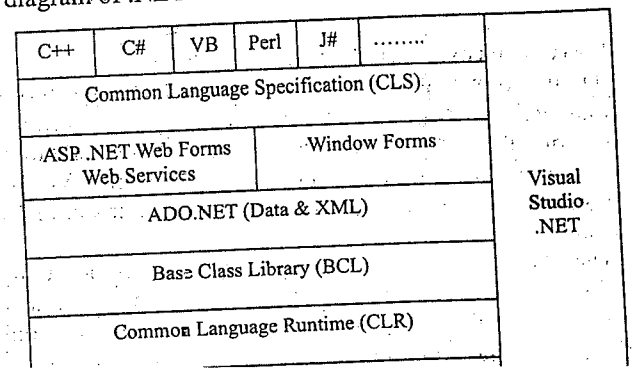


[Fig (1.a): .NET]

1.1 The .NET Framework

In general **framework** is a set of common prefabricated software building blocks that programmer use, extend or customize for specific computing solutions. The **.NET framework** is managed; type – safe environment for building, developing, deploying and executing Windows, Web and Web – Service applications. It manages all aspects of program execution, like, allocation of memory for the storage of data and instructions, granting and denying permissions to the application, managing execution of the application and reallocation of memory for resources that are not needed.

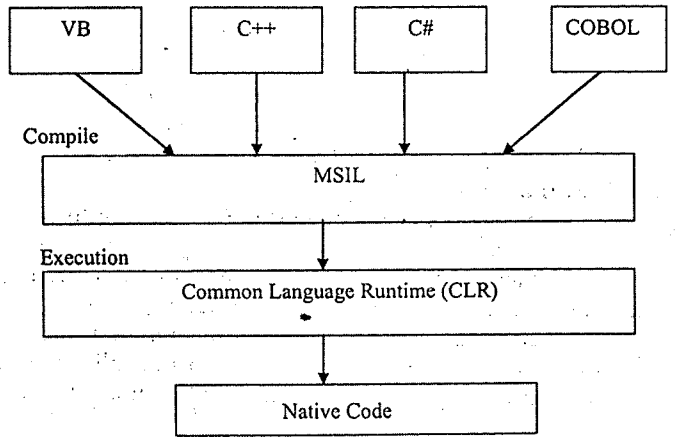
The block diagram of .NET framework is as follows



Now let's see each blocks of the framework one by one. In the above diagram, if we see from bottom to top the first block is of an operating system. The operating system is the system software which is installed on every computer to operate it and all other software work on it.

(i) CLR (Common Language Runtime)

Second block is Common Language Runtime i.e. CLR. It is the heart of .NET framework. It is the engine that compile and run the application. It uses MSIL (Microsoft Intermediate Language) format code which is language independent for execution. The MSIL code is translated by JIT compiler. CLR manages code execution at runtime. It also manages the memory and thread at runtime. (We will see CLR in detail later on in the section 1.2 of same chapter). Following figure shows the Role of CLR at runtime.



[Fig (1.c): CLR Role at runtime]

- **MSIL (Microsoft Intermediate Language)**

  It is also known as common Intermediate Language or Assembly Intermediate Language. A .NET programming language (C#, VB.NET, J# etc.) does not compile into executable code; instead it compiles into an intermediate code called Microsoft Intermediate Language (MSIL). As a programmer one need not worry about the syntax of MSIL - since our source code is automatically converted to MSIL.

  The MSIL code is then send to the CLR (Common Language Runtime) that converts the code to machine language which is then run on the host machine. MSIL is similar to Java Byte code. A Java program is compiled into Java Byte code (the .class file) by a Java compiler, the class file is then sent to JVM which

(ii) Base Class Library (BCL)

It is the second major entity of the .NET framework which is designed to integrate with Common Language Runtime (CLR). It is also known as Framework Class Library (FCL). It is the object oriented collection of reusable types. It is a library available to all languages using .NET. It provide classes which encapsulate a number of common function, including file reading & writing, graphic rendering, database interaction and XML document manipulation. (We will see BCL in detail later on in the section 1.3 of same chapter).

(iii) ADO.NET and XML

It is also known as Data Access Layer. With the help of this layer we can access relational databases. It work with XML and provide the Disconnected Data Model (we will see ADO.NET in detail later on). It is used to access data and data services. It is a part of BCL. It consists of two parts:

1) Data Provider – Connection, Parameter, Data Adapter, Data Reader.
2) Data Set

(iv) Window Forms

It is also known as Win Forms. It is used to create the GUI (Graphical User Interface) for windows desktop application. The idea of Win Form has been taken from Windows Foundation Classes (WFC) which was used for Visual J++. It also provides integrated and unified way of developing GUI. It has a rich variety of windows controls and user interface support. E.g. Text Box, Button, Check Box, Containers etc.

With Win Forms, we can make a single user interface and use it in VC++, VB, and C#. Using Visual Studio.NET, we can simply design the GUI by dragging the controls on a form. And this is all made possible because Visual Studio.NET as it uses the "System.Winforms" namespace to draw the GUI.

(v) Web Forms & Web Services

Web Forms

It provides a tool for web application. We can say that it is a part of ASP.NET. It is the forms engine that provides browser – based user interface. It consists of two parts:

1) Template – It contains HTML based layout information for all GUI elements.
2) Component – It contains the logic behind the controls.

It provides a neat presentation layer and application logic layer separation.

With ASP.NET, Microsoft has provided presentation-business layer separation - by introducing the concept of Web Forms: ASP.NET Web Forms provide an easy and

Web Services

Web Services are the applications that run on a Web Server and communicate with other applications. It uses a series of XML (standard used for storing, carrying and exchanging data over the internet) based communicating protocols that respond to different requests. They are the small unit of code which is design to handle limited set of task. They are operating system and programming languages independent. It connects people, system and devices. One example could be flight schedules and ticket reservation systems.
The protocols on which Web Services are built summarized below:

- **UDDI**: Stands for Universal Discovery and Description Integration. It allows Businesses to search for other Businesses, allowing them to search for the services it needs, know about the services and contact them.
- **WSDL**: Stands for Web Services Description Language, often called as whiz-dull. WSDL is an XML document that describes a set of SOAP messages and how those messages are exchanged.
- **SOAP**: Stands for Simple Object Access Protocol. It's the communication protocol for Web Services.
- **XML, HTTP** and **SMTP**: Stands for Extensible Markup Language, Hyper Text Transfer Protocol and Simple Message Transfer Protocol respectively. UDDI, WSDL and SOAP rely on these protocols for communication.

(vi) CLS (Common Language Specification)

It is a set of rules and constraints that all languages must follow which want to be compatible with .NET framework. It is used to support the theme of .NET i.e. unification and interoperability. That means, if we want the code which we write in a language to be used by programs in other languages then it should adhere to the Common Language Specification (CLS). Thus the CLS describes a set of features that are commn to different languages. The CLS defines the minimum standards that .NET language compilers must conform to, and ensures that any source code compiled by a .NET compiler can interoperate with the .NET Framework. Microsoft has three levels of CLS:

1) Complaint Provider
2) Consumer
3) Extender

1) **Complaint Provider:** Component developed by this type of language can be used by any other languages.
2) **Consumer:** The language in this category can use the class developed by any other languages. In simple words, the language can instantiate classes developed in other languages.
3) **Extender:** Languages in this category cannot just use the classes as in consumer category; but can also extend classes using inheritance.

And on this block there are different languages in which we can develop the application by using .NET framework.

Now let us see some concepts which are important in .NET:

- **Metadata**: Metadata means data about the data. In .NET, metadata is the declarative information which is produced at compile time. It includes all .NET files and assembly. It allows to load and locate code, enforce code security, generate native code, and provide reflection at runtime
- **Assembly**: It is the smallest executable code of the application. It contains the data for the resources. We need specific permission to access the assembly. It is develop in the form of classes. These classes are public so that other assembly can also use it. It also contains the list of external types and references which are used in the application such as references to the other assembly. It also contain the version information (major, minor, revision and build).

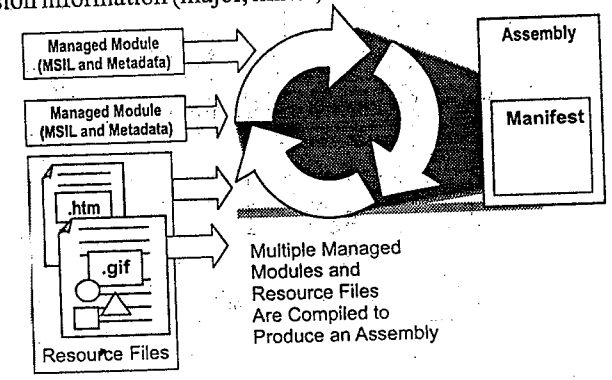


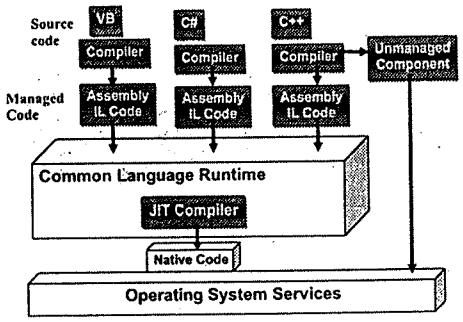
Fig (1.d): Assemblies Formation]

- **Manifest**: It is within the assembly which is formed by collecting the core information for the assembly. It contains version information, security information, list of files that are the part of assembly, type reference information, list of other assemblies, and other custom information such as user – friendly assembly title, description, company name, copyright information and product

1.2 The Common Language Runtime (CLR)

As we have discussed, it is the core component of the .NET framework. It is used to compile the MSIL code to the native code. Figure (1.e) shows the execution model of CLR.

The common language runtime is one of the most essential components of the .NET framework. The CLR provides functionality such as execption handling, security, debugging and versioning support to any languages that targets it. This means that at runtime,it can host a variety of languages and offer a common set of tools across these languages, ensuring interoperability between the codes. The existing runtime environments do not provide this capability. For example, the VB runtime provided in the earlier versions can execute only VB programs. In constrast to this, the CLR can execute programs written in any language. Presently, the languages for the CLR are VB, C# and C++ with managed extensions. The compiler of any of these languages compiles the code to provide the services offered by the CLR. The code that is developed with a language compiler that targets the CLR is called **managed code**. On the other hand, the code that is developed without considering the conventions and requirements of the common language runtime is called **unmanaged code**. Unmanaged code executes in the common language runtime environment with minimal services. For example, unmanaged code may run with limited debugging and without the garbage collection process.The CLR is made up of different components.



[Fig (1.e): CLR – Execution Model]

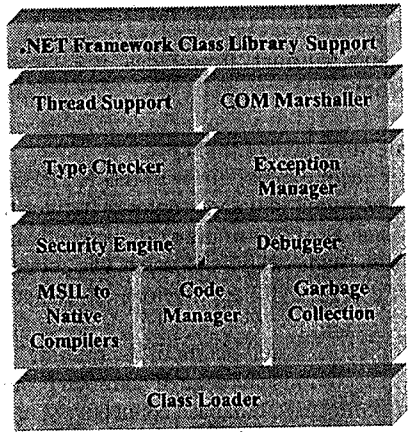
When the .NET program is compiled, the output of the compiler is not an executable file but a file that contains a special type of code called the Microsoft Intermediate Language (MSIL), which is a low-level set of instructions understood by the common language run time. This MSIL defines a set of portable instructions that are independent of any specific CPU. It's the job of the CLR to translate this Intermediate code into executable code when the program is executed, making the program to run in any environment for which the CLR is implemented. And that's how the .NET

Framework achieves Portability. This MSIL is turned into executable code using a JIT (Just In Time) complier. The process goes like this, when .NET programs are executed, the CLR activates the JIT complier. The JIT complier converts MSIL into native code on a demand basis (i.e. each part of the program is needed.)Thus the program executes as a native code even though it is compiled into MSIL making the program to run fast.So, it achieves the portability benefits of MSIL.

- **Purpose / Role of CLR**

Let us see different roles of CLR:

- ✓ **Class Loader**: As and when needed, it loads the classes into the system memory.
- ✓ **Code Manager**: CLR takes care of code management upon program execution and provides various services such as memory management, thread management, security management and other system services. The managed code targets CLR, so it can take benefits of useful features such as cross-language integration, cross-language exception handling, versioning, enhanced security, deployment support, and debugging. CLR written mostly in Microsoft's new language, C.



[Fig (1.f): CLR Role / Components]

- ✓ **Garbage Collection**: The CLR provides the garbage collection feature for managing the lifetime of an object. This process relieves a programmer from the task of manual memory management by deallocating the blocks of memory associated with objects that are no longer being used. The objects whose lifetime is managed by the garbage collection process are called managed data.
- ✓ **MSIL to Native Compilers**: When you compile a program developed in a language that targets the CLR, the compiler translates the code into an

intermediate language. This language is CPU-independent. This means that the code can be executed on any platform that supports the .NET CLR as it converts the MSIL code to Native code.

- ✓ **Debugger**: It performs the debugging at runtime.
- ✓ **Security Engine**: In .NET platform, security is achieved through the Code Access Security (CAS) model. In this model, the CLR enforces retrictions on managed code through the use of objects called 'permissions'. The CLR allows the code to perform only those tasks for which it has permission. In other words, the CAS model specifies what the code can access instead of specifying who can access resources.
- ✓ **Exception Manager**: It handles all the runtime exceptions occurring during the execution of an application.
- ✓ **Type Checker**: This feature ensures that objects are always accessed in compatible ways. Therefore, the CLR will prohibit a code from assigning a 10-bytes value to an object that occupies 8 bytes.
- ✓ **Thread Support**: Thread is a light weight process. Process is nothing but the program which is in execution. Thread provides the multi-threading support to an application.
- ✓ **COM Marshaller**: It allows the communication between the application and COM objects.
- ✓ **Class library support**: It provides BCL classes when application needs at execution time.

1.3 The .NET Framework class Library

Class library is the second major entity of the .NET Framework which is designed to integrate with the common language runtime. It gives the program access to runtime environment. It consists of lots of prewritten code that all the applications created in VB .NET and Visual Studio .NET will use it.The code for all the elements like forms, controls and the rest in VB .NET applications actually comes from the class library.

It is built on the object-oriented nature of the runtime. It provides classes that can be used in the code to accomplish a range of common programming tasks, such as string management, data collection, database connectivity, and file access. One of the most important features of the .NET framework class library is that it can be used in consistent manner across multiple languages. This means that you can use the same set of classes for performing a specific task in VB as well as in VC++. This not only makes the .NET framework types easy to use but also simplifies the learning curve associated with using a new piece of code.

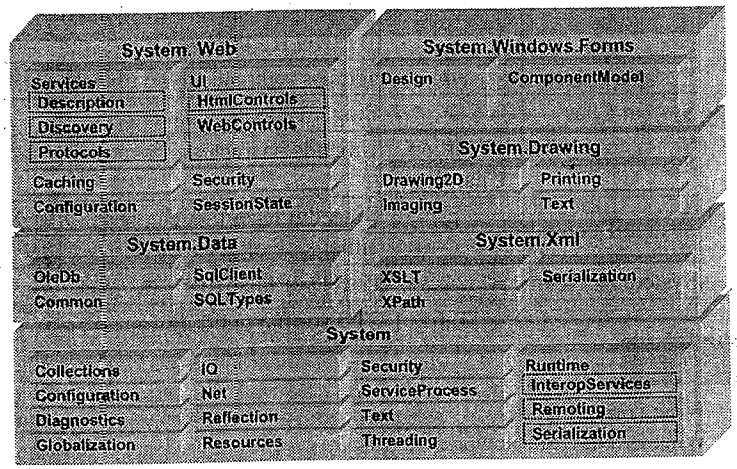
The .NET framework class library comprises namespaces, which are contained within assemblies.

- **Namespaces:**
  Namespaces help you to create logical groups of related classes and interfaces that can be used by any language targeting the .NET framework. Namespaces allow you to organize your classes so that they can be easily accessed in other applications. Namespace can also be used to avoid any naming conflicts between classes that have the same names. For example, you can use two classes with the same name in an application provided they belong to different namespaces.

  You can access the classes belonging to a namespace by simply importing the namespace into an application. The .NET framework uses a dot (.) as a delimiter between classes and namespaces. E.g. System.Console represents the Console class of the System namespace.

  The following are the main areas covered by Class library.

  1) Data Structures
  2) IO management
  3) Windows and Web Controls
  4) Database access
  5) Multithreading
  6) Remoting
  7) Reflections



[Fig (1.g) Class Library]

- **Common Type System (CTS)**
  It describes how types are declared, used and managed. It facilitates cross-language integration, type safety, and high performance code execution. It is a specification that defines the rules to support language integration. This is done in such a way, that programs written in any language (.NET compliant) can interoperate with one another. This also can take full advantage of inheritance, polymorphism, exceptions, and other features. It makes sure that all .NET applications use the same data types.

Data Type	Size in bytes	Range / Values
System.Byte	1	unsigned integer between 0-255
System.Int16	2	signed integer in 32,678 to 32,767
System.Int32	4	signed integer in -2,147,483,648 to 2,147,483,647
System.Int64	8	signed integer in 9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
System.Single	4	Floating point Values. **for negative values:** -3.402823E38 to - 1.401298E-45 **for positive values:** 1.401298E-45 to 30402823E38
System.Double	8	Floating point Values **for negative values:** -1.79769313486231E308 to - 4.964065645841247E-324 **for positive values:** 4.964065645841247E-324 to 1.79769313486232E308
System.Object	4	address reference to an object
System.Char	2	Single Unicode Character.
System.String	2 billion Characters	Unicode characters
System.Decimal	12	Signed integer that can have 28 digits on either side of decimal
System.Boolean	4	true(1) or false (0)

[Table:1 CTS supported data types]

Exercise - 1

1. What is framework?
2. What is .NET Framework?
3. Explain .NET framework in detail.
4. What is MSIL?
5. What is Web Service? Explain protocols supported by it.
6. What are Metadata, Assembly, and Manifest?
7. Explain execution of code with respect to CLR.
8. Explain different roles / components of CLR.
9. What are BCL and Namespace?
10. Explain CTS in detail.
11. Explain various data types supported by .NET.

2 Introduction to Web Based Technologies

2.1 HTTP, HTML, XML, ASP

HTTP

It is the protocol to transfer hypertext requests and information between server and browser. There are various methods available with HTTP such as Get, Post, Put, Delete, Trace, Head, Options, Connect and so on.

- **HTTP structure**
  Initial line
  Zero or more header lines
  A blank line
  Optional message body

- **Initial request line**
  Method / request URL / HTTP / ver
  E.g. GET / virtual dir / Index.html / HTTP / 1.0

- **Response Line**
  HTTP / ver code Description
  E.g. HTTP / 1.0 200 OK
  E.g. HTTP / 1.0 404 Not found message

Here first digit in status code identifies the general category of response. Message body is not available in get method. Data is the part of URL.

- **Status code**
  - ✓ 1xx indicates an information message only
  - ✓ 2xx indicates success of some kind
  - ✓ 3xx indicates redirection of the client to another URL
  - ✓ 4xx indicates an error on the clients port
  - ✓ 5xx indicates an error occur on server port

HTML

It is presentation oriented markup language. It can be use to develop the layout of the page. It contains pre–defined tags for building the layout.

XML

It is content oriented markup language. It is used to describe semantic of data. We can have user–defined tags to store the data. It is also use to transfer the data on the network.

ASP

It is Server side scripting language. The HTML and scripting code (VBScript, Jscript) are written in a single file. It is access via HTTP requests. Scripting code is interpreted on server side.

- **What can we do?**
  - ✓ It is quick and easy to create simple web application.
  - ✓ We are able to generate dynamic web content.
  - ✓ We can use client side scripting for validation.
  - ✓ It supports access to COM components such as database to extend functionality.

- **Example**

```
<html>
<head></head>
<body>
<% if (request.form("b1")<>"")then
  response.write "<p> Hello,the time is" & Now()
  end if %>
</body>
</html>
```

Here Now() returns date and time of server. We can use java script for client side script. It is simple procedural programming language. It can access to COM components such as ActiveX Data Object (ADO), File system object, and Custom components. It is script based (i.e. there is no need for compilation); we just need to edit, save, and run.

- **Short comings of ASP**
  - ✓ Mix layout (HTML) and logic (scripting code)
  - ✓ Interpreting ASP code leads to performance loss
  - ✓ Uses scripting languages that are not strongly typed
    - o MS Jscript
    - o MS VB scripting edition
  - ✓ Browser compatibility
  - ✓ No real state management
    - o No state sharing access to web forms
    - o State is lost when IIS fails
  - ✓ Updates files only when server is down

2.2 ASP.NET

ASP.NET is the Web Based technology developed by Microsoft and incorporated in Visual Studio.NET. With the help of ASP.NET, we can develop web application as well as the web service. It support full fledge Object–Oriented programming languages. Before the execution of the application, it is compiled and then after whenever the page is requested it is served from this compiled code. So, there is performance enhancement with ASP.NET.

It has the facility that it can detect client browser and according to it generate HTML. State Management and cookies data can be stored on Third party server, so if our server crashes then also we can retrieve our critical data.

- **ASP.NET core concepts**
  - ✓ It supports separation of presentation layout and business logic.
  - ✓ It uses the services provided by .NET framework.
  - ✓ Code is compiled only first time when a page is requested.
  - ✓ It supports State management.
  - ✓ To develop the application in ASP.NET. we can make use of programming language i.e. Cross language integration.
  - ✓ We can update files, while server is running
  - ✓ HTML pages are targeted to the capabilities of requesting browser
  - ✓ In ASP.NET, for a new request it creates object of page class, so compilation is done only for first request of application

- **Features**
  - ✓ Web forms
  - ✓ Web services
  - ✓ Built on .NET framework
  - ✓ Simple programming model
  - ✓ Maintain page state
  - ✓ Multi browser support
  - ✓ Xcopy deployment
  - ✓ Xml configuration
  - ✓ Complete object model
  - ✓ Session management
  - ✓ Caching
  - ✓ Debugging
  - ✓ Extensibility
  - ✓ Separation of code and UI
  - ✓ Security
  - ✓ ASPX and ASP side by side
  - ✓ Simplified form validation
  - ✓ Cookieless sessions

2.3 Compiled Code

In order to serve the request given by the user, ASP.NET must first compile the code. We can write the ASP.NET code in any of the language supported by .NET framework. When we compile the code, it is translated into MSIL which is language – independent. Then this MSIL is converted into Machine – Specific instructions.

- **Benefits of Compiled Code**
  - ✓ **Stability**: When the code is compiled, it is checked for syntax error, type safety, etc so that many errors can be eliminated from the later stage.
  - ✓ **Performance**: The execution of the compiled code is faster then the scripts such as VBScript which is much similar to the machine code.
  - ✓ **Security**: Compiled code is difficult to read. So no one can convert compiled code into source code.
  - ✓ **Interoperability**: We are able to use assemblies written in different languages as MSIL supports any .NET language.

2.3.1 Inline code (In–Page Code)

Traditional ASP developers were using this approach. In this approach scripting code working for generating desired output was intermixed with HTML code to create user friendly pages as well as specific functionality was added to the web pages. But often, the combined intermixed code become ambiguous for maintaining the readability of large web pages. For ASP .Net web pages, you can also use inline code but it becomes difficult to maintain the server side events of all the server controls on a single page. The code is written in <Script> tag in the .aspx page that contains the HTML and controls.

2.3.2 Code Behind

ASP.NET framework provides a facility to maintain the code for large pages by designing separate files for HTML code and the code file in which we can write the code in VB or C# language. So this approach of developing the web pages is called Code Behind. It separates the design layout and the coding part.

The HTML and controls are in the .aspx file and the code is in .aspx.vb or .aspx.cs file. At the time of compilation both the files are combined and one class is generated which is then used to serve the user request. Hence, we can say that the web pages are in the form of class. So when the page is requested the instance of the page is served.

2.4 Event Driven Programming

There are two programming approaches: Linear and Event Driven.

Any identifiable occurrence that has significance for system hardware or software is called the Event. So we can say that, Event is the action or the occurrence that is detected by the program. User-generated events include keystrokes and mouse clicks, among a wide variety of other possibilities.

The programming approach, which response to the event or handle the event is called Event Driven Programming. Windows operating system is the Event Driven Program (EDP).

- **ASP.NET Event Model**

  Traditional ASP uses the Linear Model. It means that the code on the page is executed from top to bottom, whenever the page is loaded.

But ASP.NET uses Event Driven Model. User can add controls on the page and according to the need we can decide which event will respond. Here each event handler is the separate method.

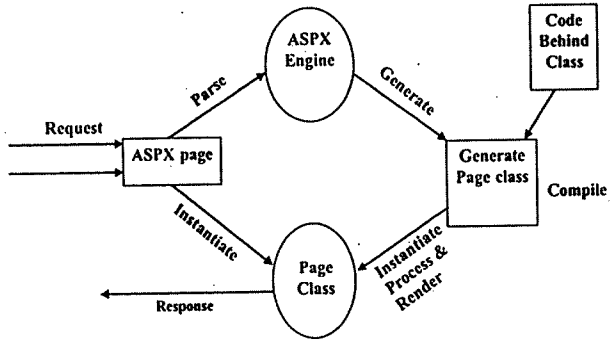
ASP.NET works as follows:

1. When the page is executed for the first time, it creates the page and control objects. Along with, the initialization code is executed and the page is rendered.
2. After the page is served to the user, any event can be triggered due to the action performed by the user.
3. If the Button Click () event generates the postback of the form. ASP.NET responds the page by re-creating it.
4. Then it raises the appropriate event which has triggered the postback.
5. At last the modified page is rendered to HTML and served to the client.

2.5 Page Life Cycle

When the user request the page, ASP.NET checks that the class for the page is available or not. If it is present, then it creates the instance of the page class and gives response. But if the class is not present then it goes under the page life cycle.

When the request comes the ASPX page is given for parsing to the ASPX engine. Then from that parsed page and the Code behind class, a combined page class is generated and it is compiled. At last the instance of the class is created and rendered for the user. In short, Web Forms are in the form of Page Class.



[Fig: (2.a) Page Life Cycle]

2.5.1 Phases of Page Life Cycle

- **Page Request**

  As we have seen in above diagram, the page request occurs before the page life cycle starts. It checks that the page needs to be parsed and compiled or not.

- **Start**

  In this phase, several properties such as Response, Request, IsPostBack and Culture are set. Along with, it checks that the request is the postback (the contents of the form are Posted Back to itself) request or not.

- **Page Initialization**
  During this phase, the controls are available on the page and its ID property is set then themes are applied.If the request is the postback request, the postback data are not loaded and the values of the controls are not restored from the ViewState.
- **Load**
  Now in this phase, the request is the postback request, the postback data is loaded and the values of the controls are not restored from the ViewState.
- **Validation**
  In this phase, if the validation controls are present on the page then its Validate() method is called, which in turn set the IsValid property of the controls and page.
- **Postback event handling**
  If the event handlers of the PostBack are need to be called, then they are called during this phase.
- **Rendering**
  Prior to this phase, the View State for page and controls are saved. During rendering phase Render() method of the controls and page is called which gives the output to the OutputStream of the Response property.
- **Unload**
  The page comes in this phase, when there is a need to dispose the page as it might not be further needed. Request and Response properties of the page is also unloaded.

2.5.2 Page Events

- **Page_PreInit()**
  This event is used for the following purpose:
  - ✓ Check the IsPostBack property to determine whether the page is being processed first time.
  - ✓ Create or re-create dynamic controls.
  - ✓ Set a master page dynamically.
  - ✓ Set the Theme property dynamically.
  - ✓ Read or set profile property values.
- **Page_Init()**
  It is executed after the PreInit() event. It is use to retrieve or initialize control properties.
- **Page_Preload()**
  Prior to this event, the ViewState data and PostBack data are restored to the controls. This event is used if we want to do some task before load event is executed.
- **Page_Load()**
  It calls the Load event of the page as well as each control on the page. This event is use to set the properties of the controls and it can also be used to establish database connection.
- **Control_events**
  They are useful to handle the events of the controls. E.g. click event of the Button.

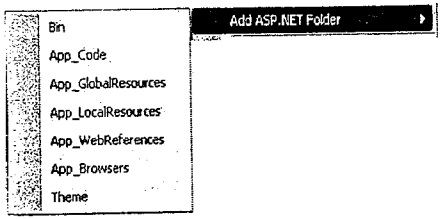
- **Page_PreRender()**
  It is executed for each control on the page. It is used to make final changes to the page and its controls.
- **Page_Unload()**
  It is executed for each control and then for the page. So we can say that it is useful for the final cleanup such as closing the connection, closing files, completing log etc.

2.6 Types of Files and Directory Structure in ASP.NET - Types of Files

File Type	Description
.aspx	ASP.NET web page that contain controls and business logic.
.ascx	Web user control file that define custom functionality that can be added to the web page.
.asax	Global.asax (Global Application Class) file that contains application level events.
.asmx	Web service file that contains methods which can be invoke by other application.
.ashx	Handler file that is invoke in response to web request to generate dynamic web content.
.browser	Browser definition file that contain the features of an individual browser.
.config	Configuration file that contains XML elements for application setting.
.cs, .vb	Source code file that contain the business logic behind the page.
.csproj, .vbproj	Project file for Visual Studio web application.
.disco	An XML Discovery file use to locate web service.
.dll	A Compiled Library or assembly file.
.master	A master page which can be used to define the layout of other web pages.
.mdb, .mdf	Database files.
.resources, .resx	A resource file that contains information about the resources such as images, text etc for the application.
.sitemap	XML sitemap file that defines the logical structure of the web application.
.skin	A theme file that contains the settings for the controls to define consistent formatting.
.sln	A solution file.

[Table:1 Types of files]

Directory Structure



[Fig (2.b) Directory Structure of web application in VS2008]

Directory	Description
App_Browsers	It contains .Browser (browser definition) file that ASP.NET use to determine the browsers and its capabilities.
App_Code	It contains source code files, which can be compiled for application.
App_Data	It contains data files such as .mdb, .mdf, .XML etc.
App_GlobalResources	It contains .recources, .resx (resource files) that are compiled into assemblies and used with global scope.
App_LocalResources	It contains resource files that are associated with specific page, master page or user control.
App_Themes	It contains .skin files.
App_WebReferences	It contains the files which is useful when working with web service.E.g. .wsdl, .disco etc.
Bin	It contains compiled assemblies (.dll).

[Table: 2 Types of Directory]

2.7 Object Oriented Concepts

Object-oriented programming (OOP) is a programming approach that uses "objects" - data structures consisting of data fields and methods - and their interactions to design applications and computer programs.

2.7.1 Class

The class is a template through which we can create an object. In .NET, it contains fields, properties, methods i.e. procedure and functions, and events.

Syntax

[<attrlist>] [Access Modifier]
[Shadows] [MustInherit | NotInheritable] Class name
[Implements interfacename]
  [statements]
End Class

Here access modifiers are Public, Private, Protected, Friend, and Protected Friend.

2.7.1.1 Fields

They are the variables in the class and they can be read or set directly.

```
Public Class Class1
  Public Field1 As Integer
  End Class
```

2.7.1.2 Properties

Properties are retrieved and set like fields but are implemented using Property Get and Property Set procedures which provide more control on how values are set or returned.

```
Private PropertyValue As String
  Public Property Prop1() As String
    Get
      Return PropertyValue
    End Get
    Set(ByVal Value As String)
      PropertyValue = Value
    End Set
  End Property
```


2.7.1.3 Methods

Methods represent the object's built-in procedures. For example, a Class named Employee may have methods named CalculateSalary and CalculateLeave. We can define methods by adding procedures, Sub routines or functions to the class.

Example:

```
Public Class Employee
  Public Sub CalculateSalary()
      Write("--------")
  End Sub
  Public Sub CalculateLeave()
      Write("---------")
  End Sub
End Class
```

2.7.1.4 Events

Events allow objects to perform actions whenever a specific occurrence takes place. For example when we click a button, a click event occurs and we can handle that event using an event handler.

2.7.2 Object

They are the basic runtime entities. They are the variables of the class. It is said to be the instance of the class.

Syntax

[<attrlist>] [{Access Modifier | Static }] [Shared] [Shadows] [ReadOnly] Dim
[WithEvents] name [(boundlist)] [As [New] type] [= initexpr]

2.7.3 Constructor

They are the special procedures that are invoked automatically for object creation.

```
Public Sub New(ByVal newValue As Integer)
    value = newValue
End Sub
```

2.7.4 Destructor

We know how to create constructor, but what about destructor; it is executed automatically when an object is destroyed. We can place code to clean up, such saving state information

and closing files, in a destructor. In VB.NET, you can use the Finalize method for this purpose. The Finalize method is called automatically when the .NET runtime determines that the object is no longer needed.

Example: In this case, we are creating an object of a class named Class1, and adding a Finalize method that beeps when the object is destroyed.

```
Public Class Form1
   Inherits System.Windows.Forms.Form
   Dim Object1 As New Class1()
End Class

Public Class Class1
   Protected Overrides Sub Finalize()
      Beep()
   End Sub
End Class
```

2.7.5 Encapsulation

The process of uniting data members and methods in a single unit is called Encapsulation. In other words, to combine data and function into a single logical unit is known as encapsulation.

When using Data Encapsulation, data is not accessed directly; it is only accessible through the functions present inside the class. Data Encapsulation enables the important concept of data hiding.

Protecting data from access by unauthorized function (external function) is called data hiding.

2.7.6 Inheritance

It is the process by which object of one class acquire the properties, i.e. data / attributes of object, of another class. That means inheritance is the process of forming a new class from an existing class or base class.

The base class is also known as parent class or super class, the new class creatad from base class is called derived class. Derived class is also known as a child class or sub class. Inheritance helps in reducing the overall code size of the program, which is an important concept in object-oriented programming.

This is important because it supports hierarchical classification. In object oriented programming the concepts of inheritance provide the idea of reusability. This means that we can add additional features to an existing class without modifying it.

This is possible by deriving a new class from the existing one. The new class has the combine features of both the classes.

To inherit the class Inherits keyword is used with derived class.

Example:

```
Public Class Person
   'Members of Person class
End Class
Public Class Employee
   Inherits Person
   'Members of Employee class
End Class
```

2.7.7 Abstraction

It is a mechanism of representing essential features without including background details. Data Abstraction increases the power of programming language by creating user defined data types. Data Abstraction also represents the needed information in the program without presenting the details.

When the classes use the concept of data abstraction, they are also known as Abstract DataTypes (ADT).

2.7.7.1 Abstract Class

MustInherit keyword is used to declare a class that cannot be instantiated and can be used only as a base class, also called an Abstract Base Class(ABC).

2.7.8 Interface

Interfaces can act as specifications for class members; when you implement an interface, you also must implement all the specified members. If class wants to use the interface, it needs to use Implements keyword.

Example:

```
Public Interface person
   Sub SetName(ByVal PersonName As String)
   Function GetName() As String
End Interface

Public Class employee
   Implements person
   Dim Name As String

   Sub SetName(ByVal PersonName As String) Implements person.SetName
      Name = PersonName
   End Sub

   Function GetName() As String Implements person.GetName
      Return Name
   End Function
End Class
```

- **Multiple Inheritance**
  VB.Net does not support Multiple Inheritance, where we inherit from multiple base classes at the same time. But it allows us to implement multiple interfaces.

2.7.9 Polymorphism

It is the Greek word, poly means many and morphism means forms i.e. one name many forms.

The ability of functions and operators to act in different ways on different data type is called polymorphism.

To use one function name for many different purposes is known as function overloading. So, Overloading is one type of Polymorphism. It allows an object to have different meanings, depending on its context.

Polymorphism allows routines to use variables of different types whenever required. To implement function overloading, Overloads keyword is used.

Example:

```
Overloads Function Add(ByVal a1 As Integer, ByVal a2 As Integer) As Integer
        res = a1 + a2
        Return res
End Function
Overloads Function Add(ByVal a1 As Integer, ByVal a2 As Integer, ByVal a3 As
Integer) As Integer
        res = a1 + a2 + a3
        Return res
End Function
```

There are two ways to implement polymorphism:

1. Inheritance Based Polymorphism
2. Interface Based Polymorphism

2.7.10 Method Overriding

The process in which subclass provide a specific implementation of a method, that is already provided by one of its super classes is called Method Overriding. The implementation in the subclass overrides (replaces) the implementation in the superclass.For method overriding keywords are as follows:

- ✓ **Overridable**: Allows a property or method in a class to be overridden.
- ✓ **Overrides**: Overrides an Overridable property or method.
- ✓ **NotOverridable**: Prevents a property or method from being overridden. Note that public methods are NotOverridable by default.
- ✓ **MustOverride**: Requires that a derived class override the property or method. MustOverride methods must be declared in MustInherit classes.

2.7.11 Partial Classes

Partial class also known as Page SubClassing.It is a new type of functionality which helps you to split a single class into multiple partial classes. These partial classes can be in different individual files. Partial keyword is used to implement it.

For example, there is the interface named IPartialClass which contains Overloaded Sub Routines. MyPartialClass is the partial class which is splitted into 3 files and it implements IPartialClass interface. Now to use this class, on Button click events object of it is created and subroutines are invoked. The code is as follows:

File1.vb

```
Imports System
Interface IPartialClass
    Sub overloads PrintEmployeeName(ByVal sName As String)
    Sub overloads PrintEmployeeName()
End Interface
Partial Class MyPartialClass
    Private sName As String = "Nikisha Jariwala"
End Class
```

File2.vb

```
Imports System
Partial Public Class MyPartialClass
    Implements IPartialClass
            Public Sub PrintEmployeeName(string s)
            Response.Write(s)
    End Sub
End Class
```

File3.vb

```
Imports System
Partial Public Class MyPartialClass
    Public Sub PrintEmployeeName()
            Response.Write(sName)
    End Sub
End Class
```

Write the following code in the click event of a button control

Default.aspx.vb page

```
Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim PClass as New MyPartialClass()
    PClass.PrintEmployeeName("String from a Partial Class.")
    PClass.PrintEmployeeName()
End Sub
```

- **Advantages**
  - ✓ This type of splitting the class file is particularly useful, if a single class contains thousands lines of code with different functionalities across different methods.
  - ✓ Productivity of the project team is increased since a single class file is split across the team members and implemented as partial classes.
  - ✓ It helps in the isolation of the business logic of an application from its user interface.
  - ✓ It facilitates easier debugging.
  - ✓ They allow programmers on your team to work on different parts of a class without needing to share the same physical file.

Exercise - 2

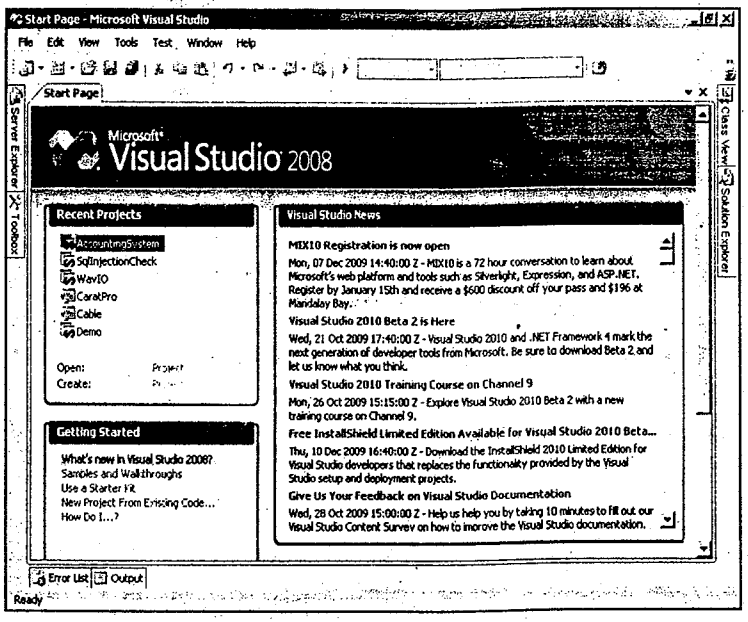
1. Explain structure of HTTP and Status code.
2. What is ASP? Explain its limitations.
3. Explain ASP.NET with its features.
4. What is Compiled code?
5. Differentiate between In – line code and Code Behind.
6. What is Event Driven Programming? Explain it with respect to ASP.NET.
7. Write a short note on Page Life Cycle. Also explain its different phases and Events performed during the life cycle.
8. List out types of files and directories supported by ASP.NET.
9. What is OOP? Explain its Basic Concepts.
10. What is Page Subclassing? Explain it with an example.

3 Visual Studio.NET for ASP.NET

3.1 Visual Studio.NET IDE

Before going to the .NET programming, let us learn about the Integrated Development Environment (IDE) which helps in programming. We will study about the IDE with respect to Visual Studio 2008.

When we start VS 2008, we get startup screen as shown in fig.(3.a).



[Fig (3.a): Startup Screen]

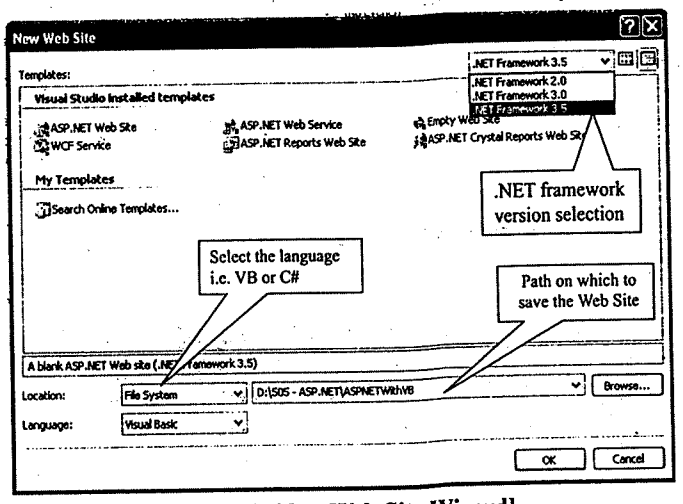
In the above screen we are able to see that it is divided into different panels:

- **Recent Projects**: It displays the list of the projects which we have created recently. In this panel, it also provides link to open existing project which is not in the list or we can create the new project.
- **Getting started**: It displays the link to the topic which provides the help for the Visual Studio 2008.
- **Visual Studio News**: It displays the links for the latest news for the Visual Studio from the internet.

Visual Studio 2008 IDE also contains some tabs such as Server Explorer, Toolbox, Solution Explorer, Class View, Error List, and Output.

3.1.1 Creating ASP.NET Application in Visual Studio.NET

You can create a new Web Site in visual studio.NET with the help of File Menu. **File→New→WebSite (Shift + Alt + N)** option. When you perform one of these actions, the **New Web Site** dialog box is displayed as shown in the fig.(3.b)



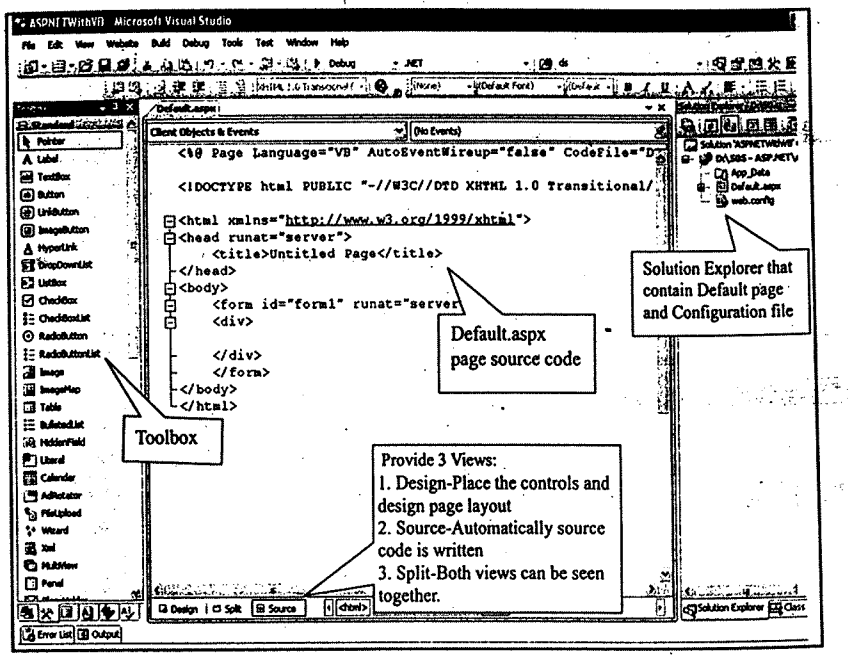
[Fig (3.b): New Web Site Wizard]

In the **New Web Site** dialog box, different types of templates are present. Select any one as per the need.

Templates	Description
ASP.NET Web Site	Creates a new ASP.NET web application that includes a configuration file and one Default.aspx page.
ASP.NET Web Service	Creates a Web Service application that allows you to develop different services which contain different methods that can be accessed by other applications.
Empty Web Site	Creates a new ASP.NET web application that includes a Configuration file but no other files.
WCF Service	Provides a basic class structure for service development. Specifically, these templates provide the basic definitions for service contract, data contract, service implementation and configuration.
ASP.NET Reports Web site	Creates an ASP.NET application that allows user to create the reports for the application.
ASP.NET Crystal Report Web Site	Creates an ASP.NET application that allows user to create the crystal reports and it also contain one sample crystal report.

[Table:1 Templates provided in .NET framework 3.5]

After selecting the template, when we click on the OK button then the default user interface will appear. It will contain one Web Page by default fig.(3.c)

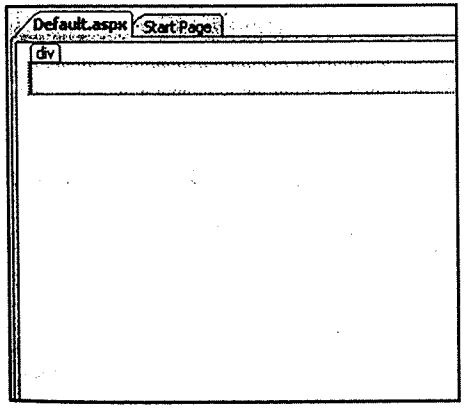


[Fig (3.c) First Look of the application on creation]

3.1.2 User Interface elements of Visual studio.NET IDE

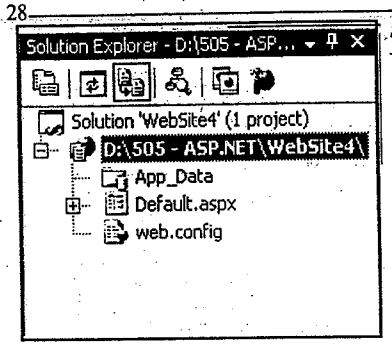
Default Page

When you create a new Web application in ASP.NET, a Page is automatically added to the application fig. (3.d) A page is the basic unit in any web-based application and is used to accept user input and perform some action based on the input.



The web page designer allows you to design the user interface for an application. It allows you to add controls to a page, arrange them as per your requirements, and add code to perform some action.

[Fig (3.d): Default Page that is included in application]

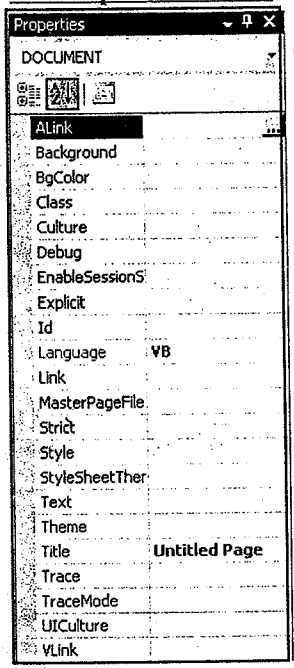


The Solution Explorer Window

The solution explorer window lists the solution name, application name, and all the pages that are used in the application fig.(3.e). You can open a particular file existing in an application by double-clicking the page in the solution explorer window.

[Fig (3.e): Solution Explorer Window]

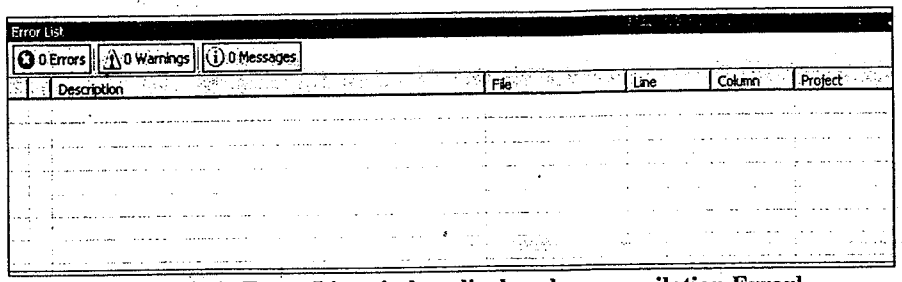
The Properties Window



The properties window displays the properties that are associated with an object fig(3.f). For example, on selecting Page from the Designer window, the properties window lists all the properties associated with Page, such as size, text, bgcolor, etc.

[Fig (3.f): Properties Window]

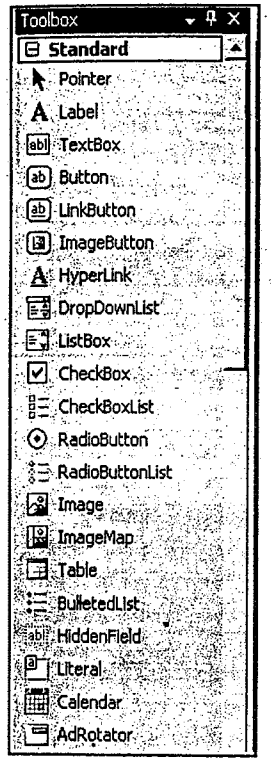
Error List



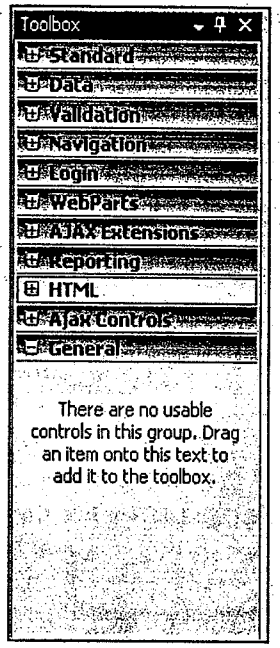
[Fig (3.g): Error List window displayed on compilation Error]

It contains three tabs fig.(3.g): Errors, Warnings, and Messages. It displays description, file, line, column and project in which error/warning/ message appear.

Toolbox



[Fig (3.h): Standard controls]



The toolbox display number of tabs such as Standard, Data, Validation, Navigation, Login, HTML, Webparts, and AJAX Extensions. Each tab contains several items, as shown in figure(3.h) and fig.(3.i). At a time, the items of only a single tab are visible. The Standard tab contains items, such as pointer, label, Textbox, and Button etc fig.(3.h).

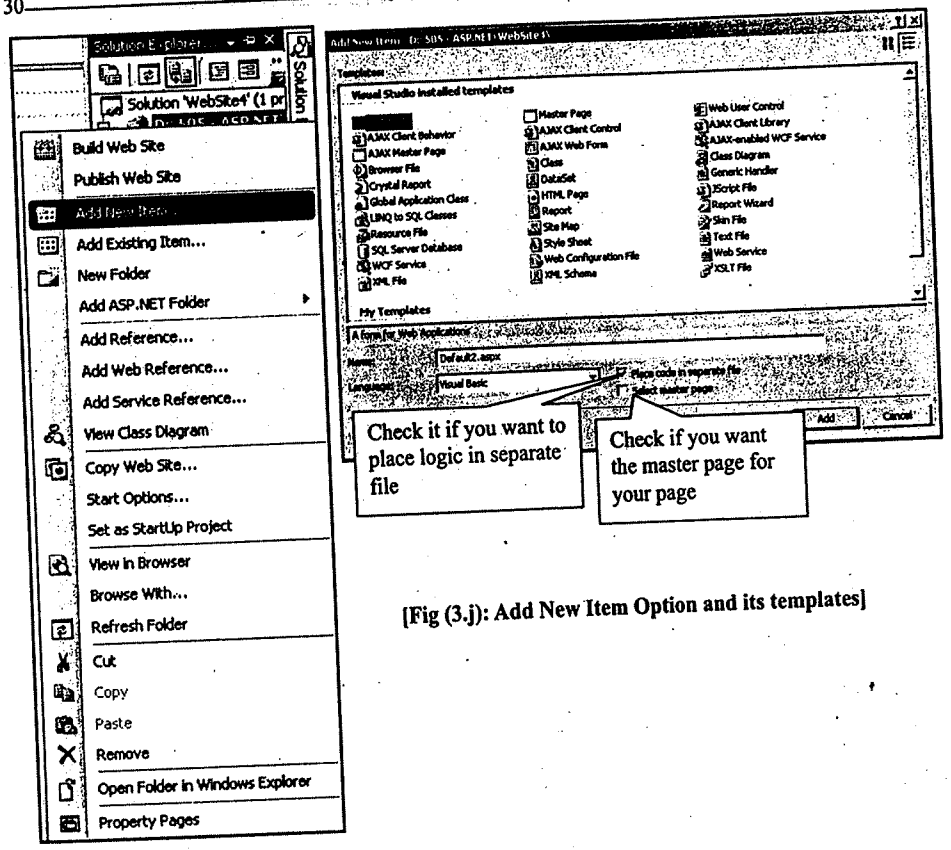
[Fig (3.i): ToolBox and Its tabs]

3.1.3 Adding new item to the Web application Solution

To add new item to the application, right click on the project in solution explorer,it display menu then select Add New Item Option. Select the option which suites for your task. The following fig (3.j) shows this step:

By selecting Add New Item, we will get a dialog box from which we can select the Item, which is to be added into the application. It contains different Templates such as Web Form, Class, Master Page, Crystal Report, SQL Server Database, SiteMap file, Global Application Class and so on fig.(3.j). Select the Item and specify the name for it and click on the Add button. By doing this, the item is added to the application.

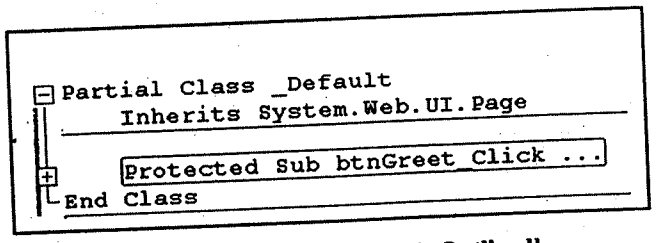
The Visual Studio IDE also contains other Menus which helps us in building, debugging, tracing of the project such as Website, Build, Debug, Format, Tools, and Test etc.



[Fig (3.j): Add New Item Option and its templates]

3.1.4 Outlining feature

By default, all text is displayed in the code editor, but you can choose to hide some code from view. The code editor allows you to select the region of code and make it collapsible, so that it appears under a plus sign (+) fig.(3.k).



[Fig (3.k): btnGreet_Click is Outlined]

You can expand or hide the region by clicking the plus sign (+) next to the symbol. Outlined code is not deleted, it is hidden from view.

When you have completed your work, you can use the stop outlining command to remove the outline information without disturbing your underlying code.

- **Commands for outlining**
  - ✓ Toggle all outlining (Ctrl + M, Ctrl + L)
  - ✓ Hide selection (Ctrl + H)
  - ✓ Stop outlining (Ctrl + P)
  - ✓ Stop hiding current (Ctrl + U)
  - ✓ Collapse to definition (Ctrl + O)
- **To create collapsible section of code**
  1. Select the desired section of code.
  2. Right click on selection, select outlining
  3. Choose hide selection.
- **To turn off automatic outlining**
  1. Right click anywhere in code editor, select outlining.
  2. Choose stop outlining.
- **To restore automatic outlining**
  1. Right click anywhere in code editor, select outlining.
  2. Choose start automatic outlining.

3.1.5 Some short cuts or button available on the Standard Toolbar

- ✓ To start debugging the solution, we can use F5 function key or press button.
- ✓ To save current file we can use Ctrl + S or we can press button.
- ✓ To save all the files in the website i.e. to save the website, we can use Ctrl + Shift + S or press button.
- ✓ To comment the lines, we can use Ctrl + K or Ctrl + C or press button.
- ✓ To uncomment the lines, we can use Ctrl + K or Ctrl + U or press button.
- ✓ To start running the application without debugging, we can use Ctrl + F5.
- ✓ To Add New Item we can use Ctrl + Shift +A.
- ✓ To Add Existing Item we can use Shift + Alt + A
- ✓ To view the application in Browser, use Ctrl + Shift + w.

3.2 Directives

Directives specify settings that are used by the page and user-control compiler when the compiler process ASP.NET Web Forms pages (.aspx files) and user control (.ascx) files. It controls how an ASP.NET page is compiled. The beginning of a directive is marked with the characters <%@ and the end of a directive is marked with the characters %>. A directive can appear anywhere within a page.

Directives	Description
@Assembly	It links the assembly to the current page or user control.
@Control	It defines control-specific attributes used by the ASP.NET page parser and compiler. It can be included only in .ascx files (user controls).
@Implements	It Indicates that a page or user control implements a specified .NET Framework interface.
@Import	It provides facility to imports a namespace into a page or user control explicitly.
@Master	It identifies a page as a master page and defines attributes used by the ASP.NET page parser and compiler. It can be included only in .master files.
@MasterType	It defines the class or virtual path used to type the Master property of a page.
@OutputCache	It maintains the output caching policies of a page or user control.
@Page	It defines page-specific attributes used by the ASP.NET page parser and compiler and can be included only in .aspx files. Page directive configure the runtime environment that will execute the page.
@PreviousPageType	It creates a strongly typed reference to the source page from the target of a cross-page posting
@Reference	It links a page, user control, or COM control to the current page or user control.
@Register	It associates aliases with namespaces and classes, which allow user controls and custom server controls to be rendered when included in a requested page or user control.

[Table.2: List and description of the Directives]

Each directive contains their attributes which describes its properties. Let's take example of @Page directive.

Example:

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Sample.aspx.cs" Inherits="Sample" Title="Sample Page Title" %>.

Here Language, AutoEventWireup, CodeFile, Inherits, Title are the attributes of the @Page directives.

3.3 Page Class

In life cycle of the page, we have seen that whenever the page is requested, the instance of the class is created which reference the page. It contains the code that you have written for the page along with the code that is generated by ASP.NET. This page contains controls and event – handling code.

All web forms are actually instances of the ASP.NET page class, which is defined in the System.Web.UI namespace. The Page class inherits from the TemplateControl class, which in turn inherits from the Control class. As a result, the Page class provides useful properties and methods that we can use in the code.

Properties

Properties	Description
Title	Gets/Sets the title of the page
Background	Gets/Sets the background image of the page
Bgcolor	Gets/Sets the background color of the page
Link	Gets/Sets the color of unvisited links on the page
ALink	Gets/Sets the color of all active links on the page
VLink	Gets/Sets the color of visited links on the page
Text	Gets/Sets the ofeground color of the page
ID	Gets/Sets the Id of body element.
Src	Gets/Sets the code behind class to compile
EnableSessionState	Allows to enable or disable session state
EnableViewState	Allows to maintain view state across pages
EnableViewStateMac	Indicates whether view state should be MAC checked
ErrorPage	Gets/Sets an error page of unhandled requests
Culture	Gets/Sets the culture setting of the page
CodePage	Gets/Sets local code page of the file
Language	Gets/Sets the language used of compiling inline code blocks
IsPostBack	It indicates whether the page is posted back or not.
IsValid	If the validation on the page are successfully validated then it sets true, false otherwise.
MasterPageFile	Gets/Sets the Master Page attached with the page.

[Table.3: List and description of the Properties of Page class]

Methods

Methods	Description
DataBind	Binds a data source to the invoked server control and all its child controls.
FindControl	Searches the current naming container of a server control with the specified id parameter.
Focus	Sets input focus to a control.
GetDataItem	Gets the data item at the top of the data-binding context stack.
GetType	Gets the System.Type of the current instance.
HasControl	Determines if the server control contains any child controls.
LoadControl	Loads a System.Web.UI.Control object from a file based on a specified virtual path.

MapPath	Retrieves the physical path of a virtual path, either absolute or relative, or an application-relative path maps too.
SetFocus	Sets the browser focus to the control with the specified identifier.
Validate	Instructs any validation control included on the page to validate their assigned information.

[Table.4 List and description of Methods of the Page class]

3.4 Structure of ASP.NET page

The following are the list of important elements of an ASP.NET page:

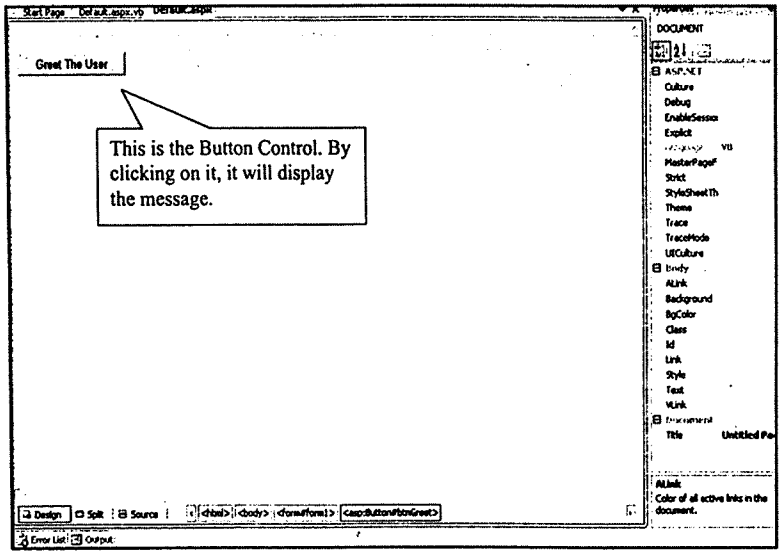
- ✓ **Directives**
  Directives specify settings that are used by the page and user-control compiler when the compiler process ASP.NET Web Forms pages.
- ✓ **Code declaration blocks**
  A code declaration block contains all the application logic for your ASP.NET page and all the global variable declarations, subroutines, and functions. It must appear within a <Script Runat="Server"> tag.

```
Sub Button_Click( s As Object, e As EventArgs )
        {lblMessage.Text = "Hello World!"}
End Sub
```

- ✓ **ASP.NET controls**
  ASP.NET controls can be freely mixed together with the text and HTML content of a page. The only requirement is that the controls should appear within a <form Runat="Server"> tag. One significant limitation of ASP.NET pages is that they can contain only one <form Runat="Server"> tag. This means that you cannot group ASP.NET into multiple forms on a page. If you try, you get an error.
- ✓ **Code render blocks**
  If you need to execute code within the HTML or text content of ASP.NET page, you can do so within code render blocks. There are two types of code render blocks are inline code and inline expressions. Inline code executes a statement or series of statements. This type of code begins with the characters <% and ends with the characters %>.
  Inline expressions, on the other hand, display the value of a variable or method (this type of code is shorthand for Response.Write). Inline expressions begin with the characters <%= and end with the characters %>.
- ✓ **Server-side comments**
  You can add comments to your ASP.NET pages by using server-side comment blocks. The beginning of a server-side comment is marked with the characters <%-- and end of the comment is marked with the characters --%>.
  It can be added to a page for the purposes of documentation. Note that you cannot see the contents of server-side comment tags, unlike normal HTML comment tags, by using the View Source command of Web browser.
  It can also be useful when you are debugging an ASP.NET page. You can temporarily remove both ASP.NET controls and code render blocks from a page by placing these elements within server-side comments.
- ✓ **Server-side include directives**
  You can include a file in an ASP.NET page by using one of the two forms of the server-side include directive. If you want to include a file that is located in the same directory or in a subdirectory of the page including the file, you would use the following directive:
  <!-- #INCLUDE file="includefile.aspx" -->
  Alternatively, you can include a file by supplying the full virtual path.
  E.g.: if you have a subdirectory named myDirectory under the root directory, you can include a file from that directory like this:
  <!-- #INCLUDE virtual="/myDirectory/includefile.aspx" -->
- ✓ **Literal text and HTML tags**
  The final type of element that you can include in an ASP.NET page is HTML content. The static portion of your page is built with plain old HTML tags and text.
  The HTML content in your ASP.NET page is compiled along with the rest of the elements. HTML content in a page is represented with the LiteralControl class. You can use the Text property of the LiteralControl class to manipulate the pure HTML portion of an ASP.NET page.

3.5 First ASP.NET Application (setting startup page)

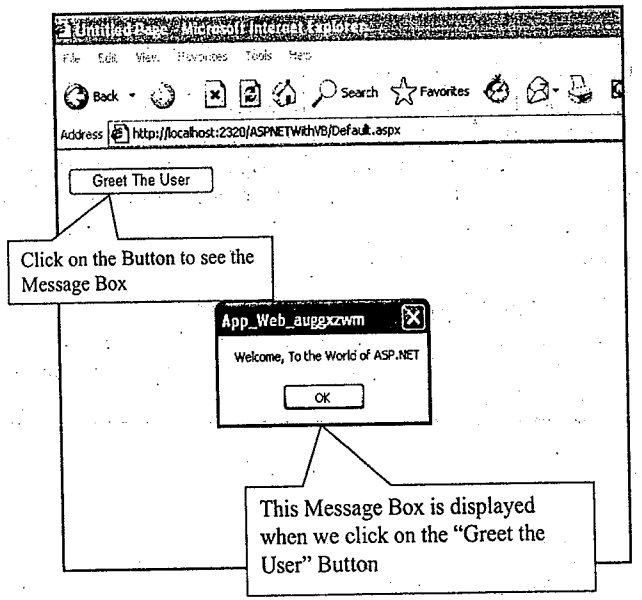
Create the Web Application as specified in Topic 3.1. So you will get one Default.aspx page fig.(3.l). Place Button Control on the page and specify the code in Button Click event fig.(3.m).



[Fig (3.l) Creating first Web Application]

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub btngreet_Click(ByVal sender As Object, ByVa
        MsgBox("Welcome, to the world of ASP.NET")
    End Sub
End Class
```

[Fig (3.m) Creating first Web Application]



[Fig (3.n) View of Web Page in browser]

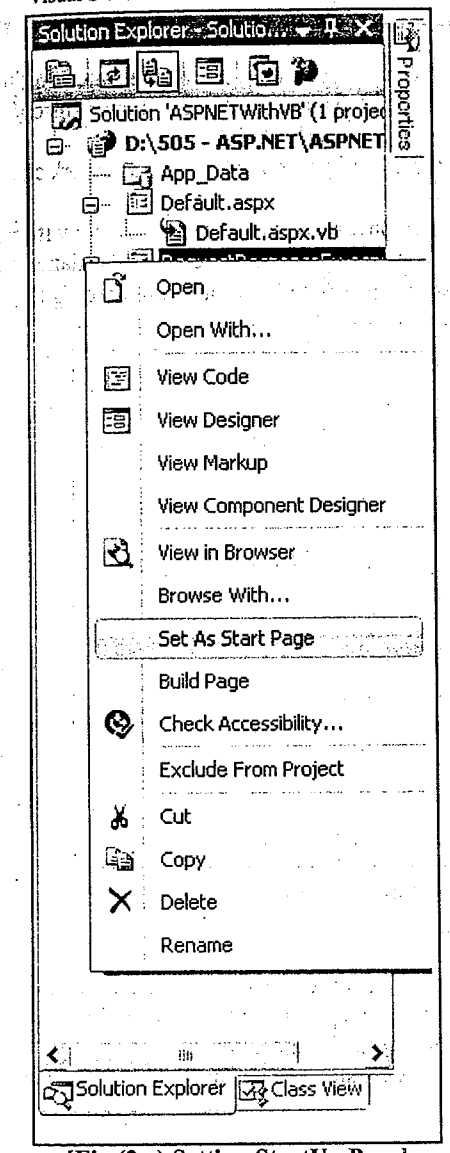
- **Setting the StartUp Page**

  Consider a Web Application containing more than one page. Setting a Startup page means to set one of the pages in the application to execute, when the application is set to run. Following are the steps to set start page:

  ✓ Right Click on the Page in solution explorer, you want to set as startup.

  ✓ Select the Set As Start Page Option fig.(3.o).

  So now whenever you run the website the page selected as startup will run.



[Fig (3.o) Setting StartUp Page]

3.6 Post Back

A Postback is an action taken by an interactive webpage, when the entire page and its contents are sent to the server for processing some information, then the server posts the same page back to the browser.

To determine whether the page is posted back or it is the first request for the page, you can use IsPostBack property of the Page. Once the page is posted back, IsPostBack property becomes true.

Example.

in Page_Load() event: if we write:

ListBox1.Items.Add("ABC")
ListBox1.Items.Add("PQR")
ListBox1.Items.Add("XYZ")

ABC
PQR
XYZ

Now, if we refresh the page or submit the page, the page is send to server and comes back, so this ListBox will again get filled which is shown in following box.

ABC
PQR
XYZ
ABC
PQR
XYZ

So, to avoid this problem use IsPostBack property in the code as follows:

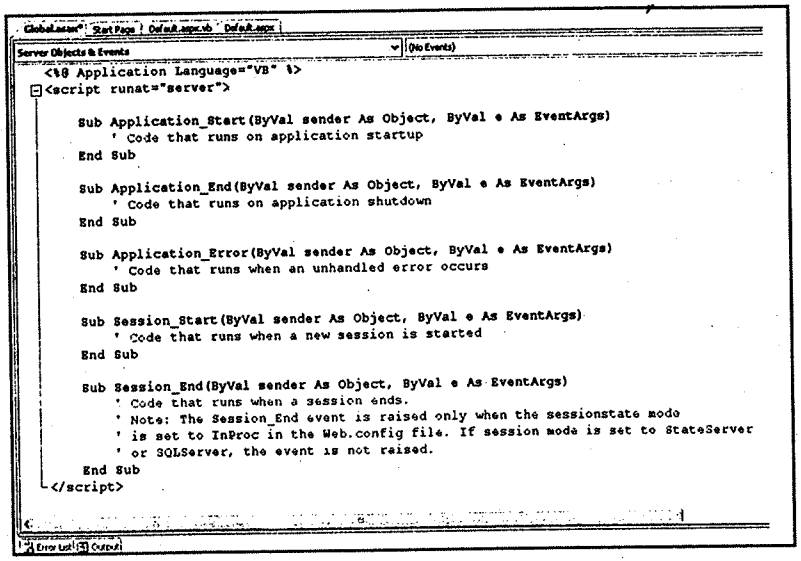
```
If Not IsPostBack then
        ListBox1.Items.Add("ABC")
        ListBox1.Items.Add("PQR")
        ListBox1.Items.Add("XYZ")
End If
```

3.7 Global Application Class

It is also known as ASP.NET application file. It is optional file that is located in the application's root directory and is counterpart of Global.asa file in ASP. It contains code for responding application and session level events raised by ASP.NET.

At runtime, it is parsed and compiled into a dynamically generated .NET framework class derived from HttpApplication base class.

To add this class into the application, Right click on the web site →Select Add New Items →Select Global Application Class from the template. By default, its name is Global.asax.Then click on Add Button. So it will add the file which looks like fig.(3.p).



```
<%@ Application Language="VB" %>
<script runat="server">

    Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
        ' Code that runs on application startup
    End Sub

    Sub Application_End(ByVal sender As Object, ByVal e As EventArgs)
        ' Code that runs on application shutdown
    End Sub

    Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
        ' Code that runs when an unhandled error occurs
    End Sub

    Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
        ' Code that runs when a new session is started
    End Sub

    Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
        ' Code that runs when a session ends.
        ' Note: The Session_End event is raised only when the sessionstate mode
        ' is set to InProc in the Web.config file. If session mode is set to StateServer
        ' or SQLServer, the event is not raised.
    End Sub
</script>
```

[Fig (3.p) Look of Global Application Class]

It contains following events:

- **Application_Start()**: When an application starts, this event is fired. It is executed only one time in an application life cycle. It executes again when we restart the application.
- **Application_End()**: It is fired when application ends its execution.
- **Session_Start():** It is executed whenever new session starts. This is the best place to count user sessions for application.
- **Session_End():** It is executed when any user session ends.
- **Application_Error():** It is executed when an unhandled error or exception occurs in an application.

Exercise - 3

1. Explain different types of templates available in Visual Studio ASP.NET
2. Write a short note on elements of IDE in Visual Studio ASP.NET.
3. What is Outlining?
4. What is Directive? List out its types. Explain any two of them.
5. Explain Page class in detail.
6. What is the significance of IsValid and IsPostBack Properties of Page Class?
7. What is the importance of Validate() Method of Page class?
8. List out the items that can be added in the web application.
9. List out various commands of Outline.
10. Explain @Page directive with example.
11. Explain Global.asax file.
12. Explain the formal structure of ASP.NET page.

4 Client Server Communication

4.1 Web Server - Web Browser Communication

Following steps are performed for web server-web browser communication.

- ✓ The user specifies what domain and port to connect to:
  User type URL in browser. Browser will use default HTTP protocol and port 80. If you need to use another port then you can specify it.
- ✓ The browser must now know which IP to connect to:
  The DNS name is translated to IP. IP is used to connect to the server. The browser will ask the local system for IP. The system will then find IP in one or another way. It will first see in cached of local machine. If not then asks to DNS and then top level DNS server.
- ✓ The client connects to server:
  This is done using IP and port only. The DNS name is not in any way used to make this connection. There is only connection; the server still does not know what to do.
- ✓ The client now sends a request message using HTTP protocol.
  E.g.: GET / index.html HTTP / 1.1. This request does not specify any host header. If web server is configured so that it required host header, the web server does not know what to do with this request. So the server will reply with "400 bad request".
- ✓ The server examines the request message, and takes an action.
  The web server will use the host header to see if there is something matching host header. If there is, it will serve index.html from home folder. If host header not found, it will use the default setting of IIS default website.
- ✓ The server responds to the request by sending some header information and content of web page.

4.2 Predefined Objects of ASP.NET

There are several built – in objects available in ASP.NET which can be directly used to perform some task.

4.2.1 Response

A Response is exactly opposite to the Request. A 'Response' is the message sent from the web server to the client, when client makes a 'Request'. For each request from a client, the server gives a response, unless there is an error. When you type a URL in a web browser or when you click on a hyper link in any web page, your browser makes a 'Request' to the server for the specific URL. Then server processes the request and sends a response back. The response includes several information about the page requested such as the cookies to be saved on the client machine, the actual content to be displayed to the user etc. The browser accepts the response and processes it. Browser does several things including saving the cookies, checking the security etc and then displays the page content to the user. In short, displaying the page content to the user in the browser is only a small portion of the actual work.

- **ASP.NET Response Object**

  The ASP.NET provides a class called HttpResponse which is defined in the namespace System.Web. This class provides various methods and properties which help you to use various information related to a web response.An instance of this class is created by default in all the pages, so that you can use this object without creating again each time in all the pages. The name of this object is Response.

  In ASP, the Response object was the only way to write data to the page. But, ASP.NET provides the event handling mechanism and web controls so that you can write content to the page in an easier way.

  The ASP.NET Response object is used to send output to the user from the server. Its collections, properties, and methods are as follows:

Collections

Collection	Description
Cookies	Sets a cookie value. If the cookie does not exist, it will be created, and take the value that is specified

[Table.1: Collection of Response Object]

Properties

Property	Description
Buffer	Specifies whether to buffer the page output or not
CacheControl	Sets whether a proxy server can cache the output or not
Charset	Appends the name of a character-set to the content-type header in the Response object
ContentType	Sets the HTTP content type for the Response object
Expires	Sets how long (in minutes) a page will be cached on a browser before it expires
ExpiresAbsolute	Sets a date and time when a page cached on a browser will expire
IsClientConnected	Indicates if the client has disconnected from the server
Status	Specifies the value of the status line returned by the server

[Table.2: Properties of Response Object]

Methods

Method	Description
AddHeader	Adds a new HTTP header and a value to the HTTP response
AppendToLog	Adds a string to the end of the server log entry
BinaryWrite	Writes data directly to the output without any character conversion
Clear	Clears any buffered HTML output
End	Stops processing a script, and returns the current result
Flush	Sends buffered HTML output immediately
Redirect	Redirects the user to a different URL
Write	Writes a specified string to the output

[Table.3: Methods of Response Object]

4.2.2 Request

When a browser asks for a page from a server, it is called a request. The Request object is used to get information from a client. Its collections, properties, and methods are as follows:

Collections

Collection	Description
ClientCertificate	Contains all the field values stored in the client certificate.
Cookies	Contains all the cookie values sent in a HTTP request.
Form	Contains all the (inputted) values from a form that uses the post method.
QueryString	Contains all the variable values in a HTTP query string
ServerVariables	Contains all the server variable values

[Table.4: Collection of Request Object]

Properties

Property	Description
TotalBytes	Returns the total number of bytes the client sent to the body of the request
UserHostAddress	It request user's host address and writes using the response object's write method.
ApplicationPath	It requests the application path.
CurrentExecutionFilePath	It requests the current execution path
FilePath	It requests the path to the file that you are currently working with.
HttpMethod	It gets the Http Method being used.
Browser	It gets the browser that is being used.
UserAgent	It gets the user information about his computer.

[Table.5: Properties of Request Object]

Methods

Method	Description
BinaryRead	Retrieves the data sent to the server from the client as a part of post request and stores it in a safe array
MapPath	Retrieves the physical path of a virtual path, either absolute or relative, or an application-relative path maps too.

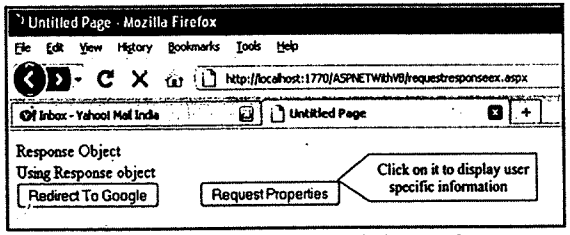
[Table.6: Methods of Request Object]

Example:
Following code show the use of various methods and properties of Request and Response Object.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
```



```
    Dim Ro As String = "Response Object"
    Response.Write(Ro)
    Response.Write("Using Response object")
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System. EventArgs)
Handles Button1.Click
    Response.Redirect("http://www.google.com")
End Sub
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

        Dim s As String = Request.UserHostAddress
        Response.Write(s)

        Dim a As String = Request.ApplicationPath
        Response.Write(a)

        Dim b As String = Request.CurrentExecutionFilePath
        Response.Write(b)

        Dim c As String = Request.FilePath
        Response.Write(c)

        Dim d As String = Request.HttpMethod
        Response.Write(d)

        If Request.Browser.Browser = "IE" Then
            Response.Write("You are using IE")
        Else
            Response.Write("You are using some other browser")
        End If

        Response.Write("Your computer is/has the following" & Request.Useragent)
End Sub
```



[Fig (4.a) View of the page in browser]

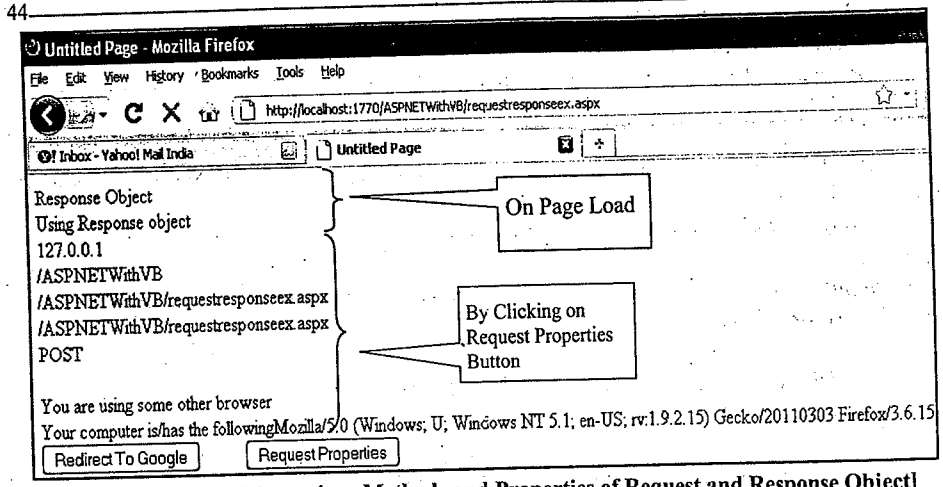


Fig (4.b) Output of the various Methods and Properties of Request and Response Object]

4.2.3 Server

This object is the instance of the System.Web.HttpServerUtility class. It provides MachineName property which provides the name of the computer on which the page is running.

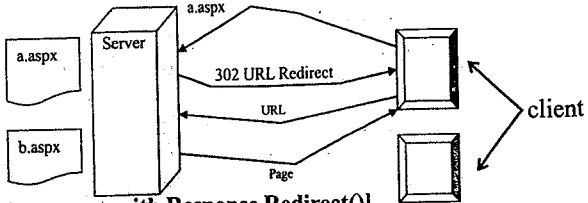
Methods

Method	Description
GetLastError	It retrieves the exception object for the most recently encountered error. This error must have occurred while current request is being processed.
HtmlEncode and HtmlDecode	It changes an ordinary string into a string with legal HTML characters (Vice versa)
UrlEncode and UrlDecode	It changes an ordinary string into a string with legal URL characters (Vice versa)
MapPath	It returns the physical file path that corresponds to a specified virtual file path on the web server.
Transfer	It transfers execution to another web page in the current application.

[Table.7: Methods of Server Object]

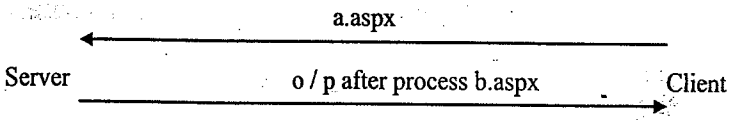
- **Redirect vs. Transfer**

Response.Redirect()



[Fig (4.c) Request sequence with Response.Redirect()]

Server.Tansfer()



[Fig (4.d) Request sequence with Server.Transfer()]

Server.Transfer()	Response.Redirect()
It transfers the execution control within the server i.e. within the same domain.	It can transfer to another domain.
We get the old URL (a.aspx).	We get the new URL (b.aspx).
Transfer is the responsibility of server.	Redirection is supported by browser i.e. rely on browser
There is no additional round trip.	One additional round trip is required.
It executies faster.	It executies Slower.

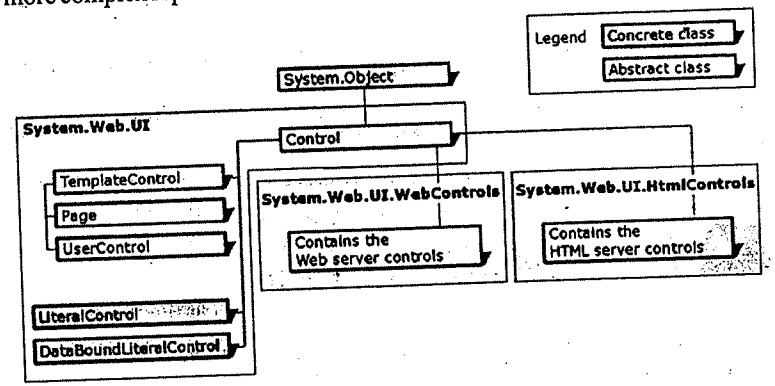
[Table.8: Difference between Server.Transfer and Response.Redirect]

Exercise - 4

1. Write a short note on Communication with Web Browser.
2. List out Built – In object of ASP.NET.
3. Which are the In – Built object of ASP.NET? Explain them with their properties and methods.
4. Which classes are used to create Response, Request and Server object?
5. Differentiate between Response.Redirect() and Server.Transfer().
6. What is the significance of MapPath()?
7. Which method can be used to increase the performance of the application (Response.Redirect() or Server.Transfer())?
8. What is the importance of UserHostAddress()?
9. Differentiate between HtmlEncode() and UrlEncode()
10. Which object contains MachineName property?

5 ASP.NET Controls

ASP.NET server controls are a fundamental part of the ASP.NET architecture. They are classes in the .NET Framework that represent visual elements on a web form. Some of these classes are relatively straightforward and map closely to a specific HTML tag. Other controls render a more complex representation from multiple HTML elements.



[Fig (5.a) ASP.NET Server Controls Hierarchy]

Control Class

All server controls are derived from the base Control class in the System.Web.UI namespace. It also applies to the Page class from which all web forms derived. Because all controls derive from the base Control class, so you can use it to manipulate any control on the page, even if you don't know the specific control type.

Properties

Properties	Description
ID	It is the name through which you can access the control from the server – side scripts or the Code Behind Class.
Page	It returns the reference of the page object that contained the control.
Parent	It returns the reference to the parent of control. It can be either page or another control.
Visible	It gets / sets the Boolean value indicating whether the control should be rendered or not.
ClientID	It returns the identifier of the control. It is the unique name created by the ASP.NET at the time of page instantiated.
Controls	It returns the collection of the child controls.
EnableViewState	It gets / sets the Boolean value indicating whether the control should maintain its states across the postbacks of its parent page or not.

[Table: 1 Properties of Control Class]

Methods

Methods	Description
DataBind()	It is used to bind the control and its child controls to the specified data source or expression.
FindControl()	It searches for the child control with a specific name in the current control and all contained control. If the child control is found the method returns the reference of general type to control. You can cast this general type to the specific control type.
HasControls()	It returns the Boolean value indicating whether the control has child controls or not. The control must be container tag to have child controls.
Render()	It writes the html output for the control according to its current state. ASP.NET implicitly calls it.

[Table: 2 Methods of Control Class]

Events

Events	Description
DataBinding	It occurs when the server control binds to a data source.
Disposed	It occurs when a server control is released from memory, which is the last stage of the server control lifecycle when an ASP.NET page is requested.
Init	It occurs when the server control is initialized, which is the first step in its lifecycle.
Load	It occurs when the server control is loaded into the Page object.
PreRender	It occurs after the Control object is loaded but prior to rendering.
UnLoad	It occurs when the server control is unloaded from memory.

[Table: 3 Events of Control Class]

As all the controls are derived from the Control Class, the properties and methods of the Control Classes are also inherited in those classes. So, all the controls will have properties (Table: 1), methods (Table: 2) and Events (Table: 3) in common.

Commonly used controls belong to System.Web.UI:

- ✓ **Page** – It is important because every ASP.NET page is compiled to a Page control by the ASP.NET page framework.
- ✓ **UserControl** – They are developed using the same programming model as ASP.NET pages and are saved as .ascx text files.
- ✓ **LiteralControl** - It allows text to be encapsulated as a control.

The ASP.NET server controls provide a user interface that is organized into two namespaces:

System.Web.UI.HtmlControls

System.Web.UI.WebControls

Types of Server Controls

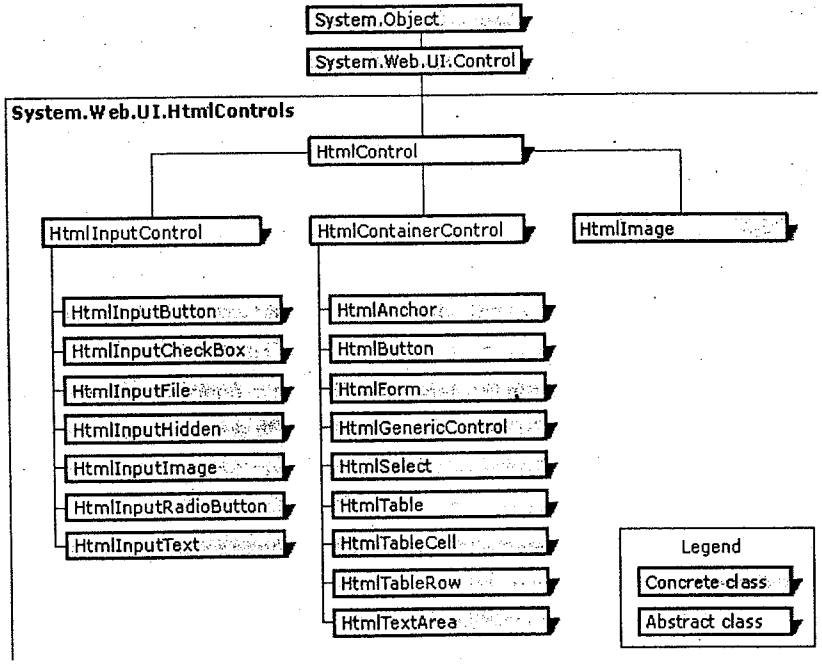
- HTML Server Controls
- Web Controls
  - ✓ Standard Controls
  - ✓ Rich Controls
  - ✓ Validation Controls
  - ✓ Navigation Controls
  - ✓ Login Controls

5.1 HTML Server Controls

They are classes that wrap the standard HTML elements. The declaration also remains the same.

Example

HtmlAnchor (for the <a> tag)
HtmlSelect (for the <select> tag)



[Fig (5.b) HTML Server Controls Hierarchy]

HTML server controls derive directly or indirectly from the base class System.Web.UI.HtmlControls.HtmlControl and map directly to HTML elements. HTML server controls are useful for migrating ASP applications to ASP.NET applications.

You can also turn any HTML tag into a server control, and if there is no direct class, ASP.NET will use the HtmlGenericControl class. To change an ordinary HTML element into a server control, simply add the runat="server" attribute to the element tag.

- **HtmlControl Class**

All the HTML server controls are derived from the base class HtmlControl. Additional properties of HtmlControl class are shown in Table:4:

Properties

Properties	Description
Attributes	It allows you to access or add attributes to the control tag.
Disabled	It gets / sets the Boolean value that indicates whether the control is disabled or not.
Style	It returns the collection of the CSS attributes that are applied to the control.
TagName	It returns the control's tag name. (For HtmlImage control it returns img).

[Table: 4 Properties of HtmlControl Class]

- **The HtmlContainerControl Class**

Any HTML tag has both an opening and a closing tag that can contain other HTML content or controls.

Example

Anchor tag <a>...., which usually wraps text or an image with the tags.

Many other HTML tags also work as containers, including everything from the tag which allows you to format a block of content to the <b> tag which applies bold formatting. These tags don't map to specific HTML server control classes, but you can still use them with the runat="server" attribute.

Properties

Properties	Description
InnerHtml	It gets or sets the HTML text inside the opening and closing tags. When you use this property, all characters within these tags are as it is.
InnerText	It gets / sets the text inside the opening and closing tags. When you use this property, any characters that would be interpreted as special HTML syntax such as < or > (the angle brackets) are automatically replaced with the HTML tags.

[Table: 5 Properties of HtmlContainerControl Class]

- **HtmlInputControl Class**

The HTML input controls allow for user interaction. These include check box, text box, button, and list box. All of these controls are generated with the <input> tag. The type attribute indicates the type of input control, such as <input type="text"> for text box, <input type="submit"> for submit button, and <input type="file"> controls for uploading a file.

Properties

Properties	Description
Type	It gets the type of an HtmlInputControl. **Example** If this property is set to text, the HtmlInputControl is a text box for data entry.
Value	It gets / sets the value associated with an input control. The value associated with a control depends on the type of control. **Example** In a text box, this property contains the text entered in the control. For buttons, this defines the text on the button.

[Table: 6 Properties of HtmlInputControl Class]

- **Html Server Control Classes**

Classes

Tag	Class	Properties	Event
<a runat="server">	HtmlAnchor	HRef, Target, Title, Name	ServerClick
<button runat="server">	HtmlButton	CausesValidation, ValidationGroup	
<form runat="server">	HtmlForm	Enctype, Method, Target, DefaultButton, DefaultFocus	-
<img runat="server">	HtmlImage	Align, Alt, Border, Height, Src, Width	-
<input type="button" runat="server">	HtmlInputButton	Type, Value, CausesValidation, ValidationGroup	ServerClick
<input type="reset" runat="server">	HtmlInputReset	Type, Value	-
<input type="submit" runat="server">	HtmlInputSubmit	Type, Value, CausesValidation, ValidationGroup	ServerClick
<input type="checkbox" runat="server">	HtmlInputCheckBox	Checked, Type, Value	

<input type="file" runat="server">	HtmlInputFile	Accept, MaxLength, PostedFile, Size, Type, Value	
<input type="hidden" runat="server">	HtmlInputHidden	Type, Value	ServerChange
<input type="image" runat="server">	HtmlInputImage	Align, Alt, Border, Src, Type, Value, CausesValidation, ValidationGroup	ServerClick
<input type="radio" runat="server">	HtmlInputRadioButton	Checked, Type, Value	ServerChange
<input type="text" runat="server">	HtmlInputText	MaxLength, Type, Value	
<input type="password" runat="server">	HtmlInputPassword		
<select runat="server">	HtmlSelect	Multiple, Size, Value, SelectedIndex, DataSource, Items (Collection) DataTextField, DataValueField	
<table runat="server">, <td runat="server">	HtmlTable	Align, BgColor, Border, BorderColor, CellPadding, Width, CellSpacing, Height, NoWrap, Rows (collection)	-
<th runat="server">	HtmlTableCell	Align, BgColor, Border, BorderColor, ColSpan, Height, NoWrap, Width, RowSpan, VAlign	-
<tr runat="server">	HtmlTableRow	Align, BgColor, Border, BorderColor, Height, VAlign, Cells (collection)	-
<textarea runat="server">	HtmlTextArea	Cols, Rows, Value	ServerChange

3439

[Table: 7 Properties of HtmlInputControl Class]

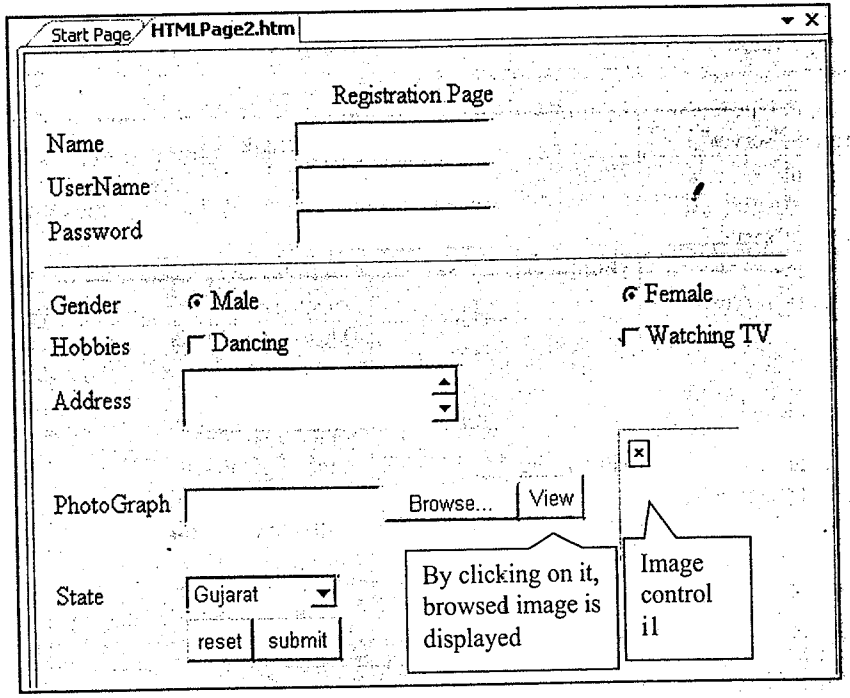
The ServerClick event is simply a click that is processed on the server side. It is provided by most button controls, and it allows to take action immediately.

The ServerChange event responds when a change has been made to a text or selection control. This event does not occur until the page is posted back. The ServerChange event occurs for all changed controls, followed by the appropriate ServerClick.

Example

Create a registration page using HTML Server controls.

- Design the page (Fig (5.c))



[Fig (5.c) Design View of the Html page]

- Add Click event of View button in the source code within JavaScript of <head> tag.

```
<script language="javascript" type="text/javascript">
        //<!CDATA[
        function btnView_onclick() {
document.getElementById('i1').src=document.getElementById('File1').value;
}
//]]>
```



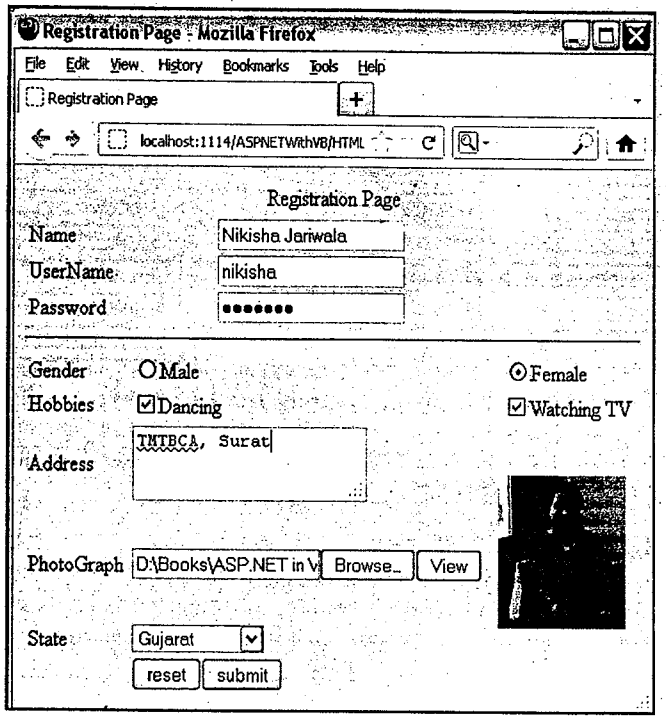
```
</script>
```

Here "i1" is the name of the image control (Fig (5.c)).

- Make changes in the <input> tag of the View Button.

```
<input id="btnView" type="button" value="View" onclick="return
btnView_onclick()"/>
```

- Save and execute the application (Fig (5.d)).

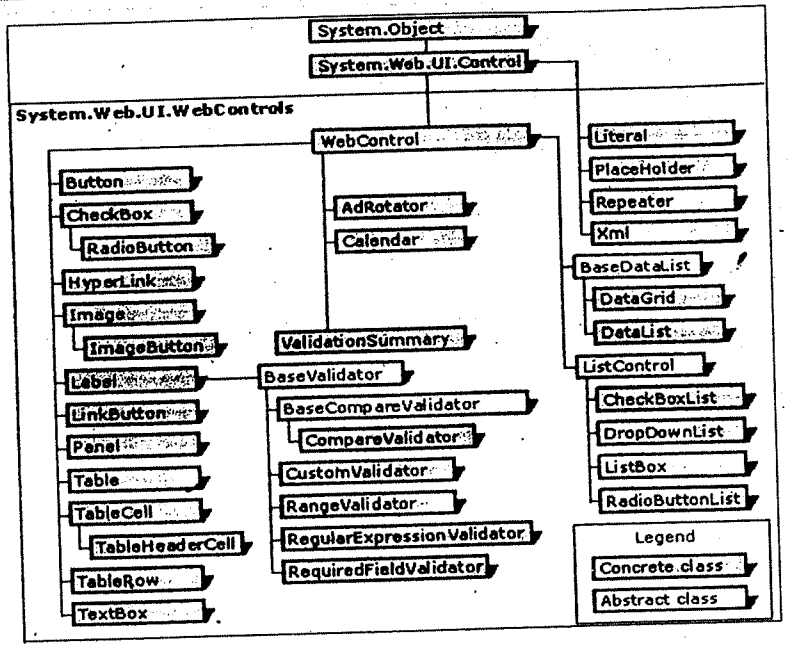


[Fig (5.d) Page after filling information]

5.2 Web Server Controls

Their functionalities are same as HTML elements but have a more consistent and meaningful set of properties and methods that make it easier to declare and access them. Some examples are the HyperLink, ListBox, and Button controls. In Visual Studio, you will find the basic web forms controls in the Standard tab of the Toolbox.

Most Web server controls are derived directly or indirectly from the base class System.Web.UI.Control.WebControl. (Fig (5.e))



[Fig (5.e) Web Server Control Class Hierarchy]

Note: All Web Server Controls has one common property runat; which specifies that the control is a server control and must be set to "server".

WebControl base Class

All the web controls are inherited from the WebControl base class. The WebControl class is derived from Control class. So, many of its properties and methods—such as Controls, Visible, and FindControl() are similar to those of the HTML server controls.

Properties	Description
AccessKey	It gets or sets the keyboard shortcut key that allows the user to quickly navigate to the control. **Example** If this property is set to A, the user can move the focus to this control by pressing Alt+A.
BackColor	It gets or sets the background color.
BorderColor	It gets or sets the border color.
BorderStyle	It gets or sets one of the values from the BorderStyle enumeration, including Dashed, Dotted, Double, Groove, Ridge, Inset, Outset, Solid, and None.
BorderWidth	It gets or sets the border width.
CssClass	It gets or sets the CSS style to associate with the control.

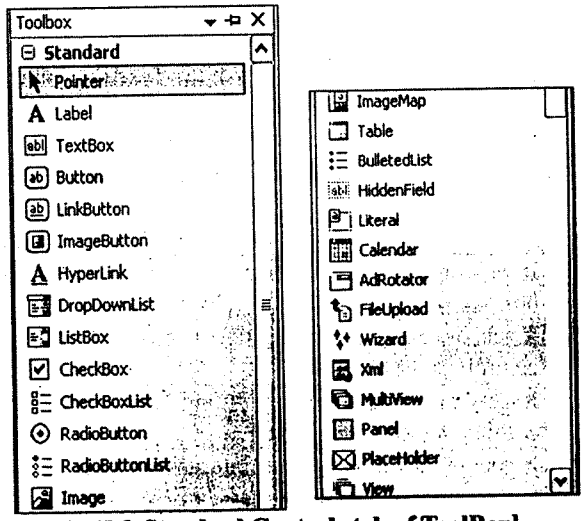
Enabled	It gets or sets the control's enabled state. If false, the control is usually rendered grayed out and is not usable.
Font	It returns an object with all the style information of the font used for the control's text. This property includes sub properties that can also be set.
ForeColor	It gets or sets the foreground color (text of the control).
Height	It gets or sets the control's height.
ID	It gets or sets the programmatic identifier assigned to the server control.
RunAt	It specifies that the code is to be run on server and not on client.
SkinId	It allows you to assign a skin to the control for its appearance.
Style	It gets the HTML style of the control as a collection of text attributes.
TabIndex	It indicates a number that allows you to control the tab order. When the page loads, the control with a TabIndex 0 has the focus. Pressing Tab moves the focus to the enabled control of next TabIndex.
Tooltip	It displays a text message when the user hover the mouse above the control.
Width	It gets or sets the control's width.

[Table: 8 Properties of WebControl Class]

It inherits the Methods and Events of the Control base class.

5.2.1 Standard Controls

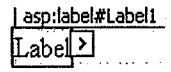
All Standard controls are defined in the System.Web.UI.WebControl namespace, which in turn is derived from the Control base class, which provides a consistent model than the HTML server controls. Web controls have the features, such as automatic postback. Fig (5.f) shows the Standard Controls of the ASP.NET. Basic Web Controls have same functionality as HTML server controls.



[Fig (5.f) Standard Controls tab of ToolBox]

Note: Standard Controls inherits all the properties, methods and events of WebControl class and in additions; these controls also have their own properties and events.

- **Label Control**



[Fig (5.g) Label Control

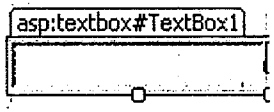
The label control is used to display information to the user (fig. (5.g)).
Its ASP.NET tag is <asp:Label> and HTML is ....

Properties

Properties	Description
Text	It gets or sets the Text to be shown for the label.
AssociatedControlID	It gets or sets the ID of the control associated with the label.

[Table: 9 Label Control Properties]

- **TextBox Control**



[Fig (5.h) TextBox Control]

The TextBox control is an input control that allows user to enter a text (fig. (5.h)). By default, it displays a single-line text box, as TextMode property of the control is set to TextBoxMode.SingleLine. TextBox control can also be use to display a multiline text box or a text box that masks user input by changing the value of the TextMode property to TextBoxMode.MultiLine or TextBoxMode.Password, respectively.
Its ASP.NET tag is <asp:TextBox> and HTML tags are <input type="text"/> or <textarea>. . .</textarea>.

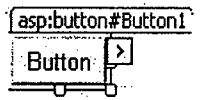
Properties

Properties	Description
Text	It gets or sets the text content of the TextBox control.
TextMode	It gets or sets the behavior mode (single-line, multiline, or password).
MaxLength	It gets or sets the maximum number of characters allowed in the text box.
Rows	It gets or sets the number of rows displayed in a multiline text box.
Columns	It gets or sets the number of columns displayed in a multiline text box.
ReadOnly	It gets or sets a value indicating whether the contents of the TextBox control can be changed.
Wrap	It gets or sets a value indicating whether the text content wraps within a multiline text box.
AutoPostBack	It gets or sets a value that indicates whether an automatic postback to the server occurs when the TextBox control loses focus.

[Table: 10 TextBox Control Properties]

✓ **Default Event:** TextChanged()

- **Button Control**



[Fig (5.i) Button Control]

The Button control is used to create a push button on the Web page (fig. (5.i)). You can create either a Submit button or a Command button. By default, a Button control is a Submit button. A Submit button does not have a command name specified by the CommandName property associated with the button and simply posts the Web page back to the server. User can provide an event handler for the Click event of control which can programmatically control the actions, when the Submit button is clicked.
Its ASP.NET tag is <asp:Button> and HTML tags are <input type="submit"/> or <input type="button"/>.

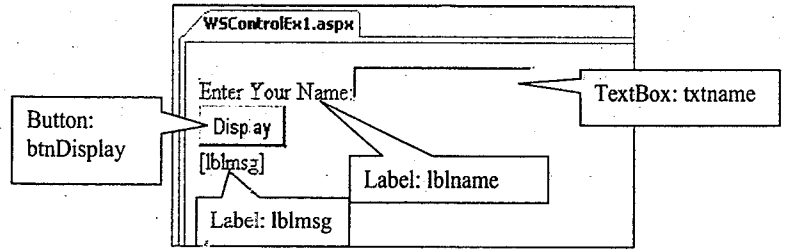
Properties

Properties	Description
Text	It gets or sets the caption to be displayed on the Button control.
CausesValidation	By default, page validation is performed when a Button control is clicked. Page validation determines whether the input controls associated with a validation control on the page all pass the validation rules specified by the validation control. To prevent page validation from occurring, set the CausesValidation property to false.
PostBackUrl	It gets or sets the URL of the page to be posted when the Button is clicked.
ValidationGroup	It gets or sets the group of controls for which the Button control causes validation when it posts back to the server.
UseSubmitBehavior	It gets or sets a value indicating whether the Button control use the submit mechanism of client browser or the postback mechanism of ASP.NET.
TextAlign	It gets or sets the alignment of the caption on the control.

[Table: 11 Button Control Properties]

✓ **Default Event:** Click()

Example
Design the page as shown in fig (5.j).

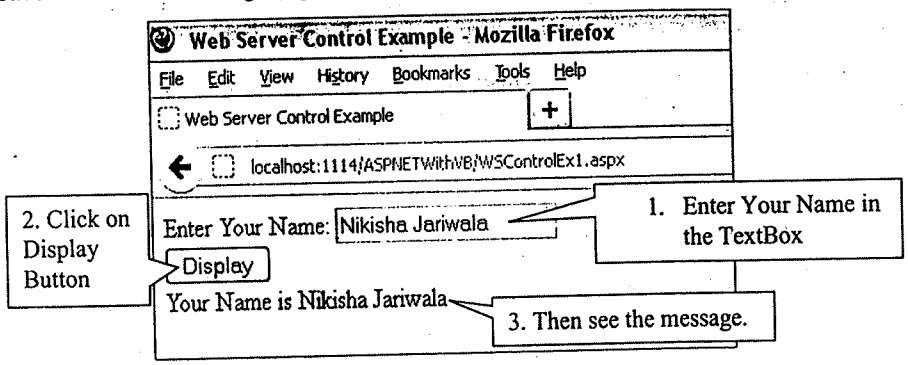


[Fig (5.j) Label, Button, TextBox Control Example]

Write the code on the Click() Event of the btnDisplay Button:

```
Protected Sub btnDisplay_Click(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles btnDisplay.Click
    lblmsg.Text = "Your Name is " + txtname.Text
End Sub
```

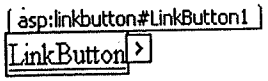
Save and execute the Page (fig (5.k).



[Fig (5.k) Example page at runtime]

- **LinkButton Control**

The LinkButton control is used to create a hyperlink-style button on the Web page (fig. (5.l)). The LinkButton control has the same appearance as a HyperLink control, but has the same functionality as a Button control.



[Fig (5.l) LinkButton Control]

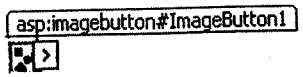
Its ASP.NET tag is <asp:LinkButton> and HTML Tag is <a> <img/>.

Its properties, methods and default event are same as Button Control, except it does not contain UseSubmitBehavior Property of Button Control.

- **ImageButton Control**

The ImageButton control is used to display an image that responds to mouse clicks (fig. (5.m)). Both the Click and Command events are raised when the ImageButton control is clicked. By using the OnClick event handler, you can programmatically determine the coordinates where the image is clicked. User can then code a response, based on the values of the coordinates. Note that the origin (0, 0) is located at the upper left corner of the image.

Its ASP.NET tag is <asp:ImageButton> and HTML tag is <input type="image"/>.



[Fig (5.m) ImageButton Control]

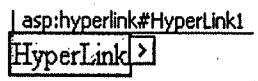
Properties

Properties	Description
CausesValidation	By default, page validation is performed when ImageButton control is clicked. Page validation determines whether the input controls associated with a validation control on the page, pass the validation rules specified by the validation control. To prevent page validation from occurring, set the CausesValidation property to false.
PostBackUrl	It gets or sets the URL of the page to be posted when the Button is clicked.
ValidationGroup	It gets or sets the group of controls for which the Button control causes validation when it posts back to the server.
ImageAlign	It gets or sets the alignment of the caption on the control.
AlternateText	It gets or sets the alternate text displayed in the control when the image is unavailable.
ImageUrl	It gets or sets the URL that provides the path to an image to display in the control.

[Table: 12 ImageButton Control Properties]

- **HyperLink Control**

The HyperLink control is used to create a link to another Web page (fig. (5.n)). The HyperLink control is displayed as text specified by the Text property. It can also be displayed as an image specified by the ImageUrl property. If both the Text and ImageUrl properties are set, the ImageUrl property takes precedence. If the image is unavailable, the text in the Text property is displayed.



[Fig (5.n) HyperLink Control]

Its ASP.NET tag is <asp:HyperLink> and HTML tag is <a>....

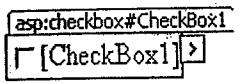
Properties

Properties	Description
ImageUrl	It gets or sets the path to an image to be display for the HyperLink control.
Text	It gets or sets the Text to be displayed on the control.
NavigateUrl	It gets or sets the URL to link to when the HyperLink control is clicked.
Target	It gets or sets the target window or frame in which you want to display the Web page content linked to when the HyperLink control is clicked.

[Table: 13 HyperLink Control Properties]

- **CheckBox Control**

The CheckBox control is used to allow user to select one or more choice from various values (fig. (5.o)). The check box, when clicked, displays a check mark (✓) meaning user has selected the check box option and check mark goes away when the user clicks the check box once again.



[Fig (5.o) CheckBox Control]

Its ASP.NET tag is <asp:CheckBox> and HTML tag is ☐.

Properties

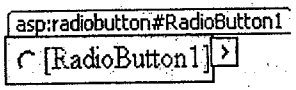
Properties	Description
Checked	It gets or sets the value indicating that the CheckBox control is checked or not.
Text	It gets or sets the Text to be displayed on the control.
AutoPostBack	It gets or sets a value that indicates whether an automatic postback to the server occurs when the CheckBox control loses focus.
TextAlign	It gets or sets the alignment of the caption on the control.
CausesValidation	It gets or sets a value indicating whether validation is performed when the CheckBox control is selected.

[Table: 14 CheckBox Control Properties]

✓ **Default Event:** CheckedChanged()

- **RadioButton Control**

The RadioButton control is used to create mutually-exclusive option (fig. (5.p)). When we click one of the RadioButton in a group, any other button previously selected will automatically turn off. The buttons are grouped logically if they all share the same GroupName property.



[Fig (5.p) RadioButton Control]

Its ASP.NET tag is <asp:RadioButton> and HTML tag is ☐.

Properties

Properties	Description
Checked	It gets or sets the value indicating that the RadioButton control is checked or not.
Text	It gets or sets the Text to be displayed on the control.
AutoPostBack	It gets or sets a value that indicates whether an automatic postback to the server occurs when the RadioButton control loses focus.
TextAlign	It gets or sets the alignment of the caption on the control.
CausesValidation	It gets or sets a value indicating whether validation is performed when the RadioButton control is selected.
GroupName	It gets or sets the name of the group that the radio button belongs to.

[Table: 15 RadioButton Control Properties]

✓ **Default Event:** CheckedChanged()

- **List Controls**

The list controls generate list boxes, drop-down lists, and other repeating controls that can be either bound to a data source such as a database or programmatically filled with items. Most of list controls allow the user to select one or more items.

Properties

Properties	Description
AutoPostBack	It gets or sets a value indicating whether a postback to the server automatically occurs when the user changes the list selection or not.
Items	It gets or sets the collection of <asp:ListItem>.
SelectedIndex	It gets or sets the index of the selected items in the list.
SelectedItem	It gets the selected item in the list control.
SelectedValue	It gets the value of the selected item in the list control, or selects the item in the list control that contains the specified value.
DataSource	It gets or sets the table from which the control retrieves its list of data items.
DataMember	It gets or sets the name of the list of data that the control binds to when data source contains more than one table.
DataTextField	It gets or sets the field of the data source that is to be displayed as the text content of the list items.
DataValueField	It gets or sets the field of the data source is to be used as value of each list item.
DataTextFormatString	It gets or sets the formatting string used to control how data is to be displayed.

[Table: 16 Common properties to all List Control (ListControl Class)]

✓ **Default Event:** SelectedIndexChanged()

The Items in the List control is stored in the form of ListItem, which is the Class. But in ASP.NET it is considered as a control.

- **ListItem Class**

A ListItem control contains an individual data item within a list control, such as a ListBox or a CheckBoxList control.

Its ASP.NET tag is <asp:ListItem> and HTML tag is <Option/>.

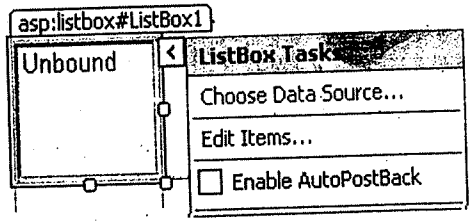
Properties

Properties	Description
Enabled	It gets or sets a value indicating whether the list item is enabled or not.
Selected	It gets or sets a value indicating whether the item is selected or not.
Text	It gets or sets the text displayed in a list control for the ListItem.
Value	It gets or sets the value associated with the ListItem.

[Table: 17 Properties of ListItem Control]

- **ListBox Control**

The ListBox control is used to create a list control that allows single or multiple item selection (fig. (5.q)).
Its ASP.NET tag is <asp:ListBox> and HTML tag is <Select/>.



[Fig (5.q) ListBox Control]

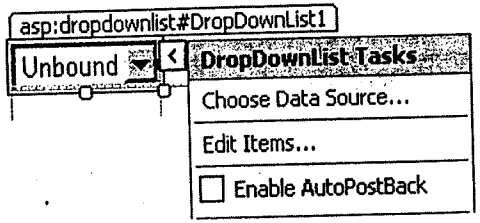
Properties

Properties	Description
Rows	It gets or sets the number of rows displayed in the ListBox control that means you can specify the height of the control.
SelectionMode	It gets or sets the selection mode of the ListBox control that means you want to select Single or Multiple items.

[Table: 18 Properties of ListBox Control]

- **DropDownList Control**

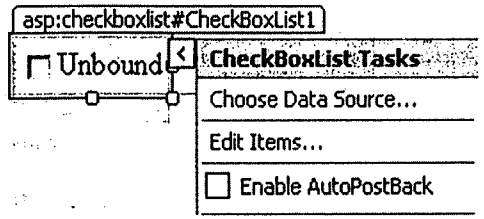
The DropDownList control is used to create a single-selection drop-down list control (fig. (5.r)).
You can set the appearance of the DropDownList control by setting the BorderColor, BorderStyle, and BorderWidth properties.
Its ASP.NET tag is <asp:DropDownList>.



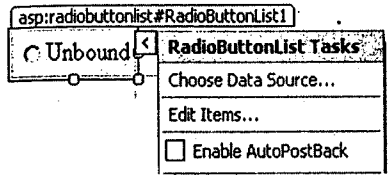
[Fig (5.r) DropDownList Control]

- **CheckBoxList Control**

The CheckBoxList control is used to provide a multi selection check box group that can also be generated dynamically with data binding (fig. (5.s)). To check which item is checked, you can use Selected property.
Its ASP.NET tag is <asp:CheckBoxList>.



[Fig (5.s) CheckBoxList Control]



[Fig (5.t) RadioButtonList Control]

- **RadioButtonList Control**

The RadioButtonList control is used to provide a single-selection radio button group that can also be generated dynamically with data binding (fig. (5.t)). To check which item is selected, you can use SelectedItem property.

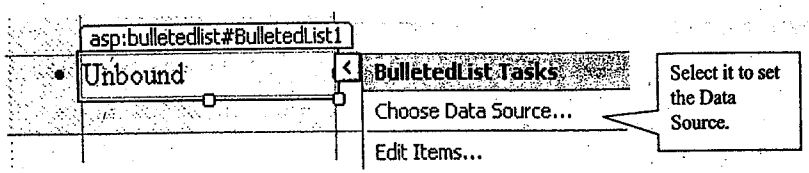
Properties

Properties	Description
RepeatLayout	It gets or sets a value that specifies whether the list will be rendered by using a <table> element, a element, a element, or a element.
RepeatDirection	It gets or sets a value that indicates whether the control displays vertically or horizontally.
RepeatColumns	It gets or sets the number of columns to display in the control, if RepeatLayout Property is set to table.
CellPadding, CellSpacing, TextAlign	If RepeatLayout is set to Table, then these properties set the spacing and alignment of the cells of the table.

[Table: 19 Common properties to RadioButtonList and CheckBoxList Control]

- **BulletedList Control**

The BulletedList control is used to display the list of values as bulleted list or numbered list (fig. (5.u)). As with all list controls, you can set the collection of items that is to be displayed in the list, but the BulletedList displays a static list.
Its ASP.NET tag is <asp:BulletedList> and HTML tag is <ul> (unordered list) or <ol> (ordered list).



[Fig (5.u) BulletedList Control]

Note: it supports all the common properties of List controls and it also adds its own properties (Table: 20).

Properties

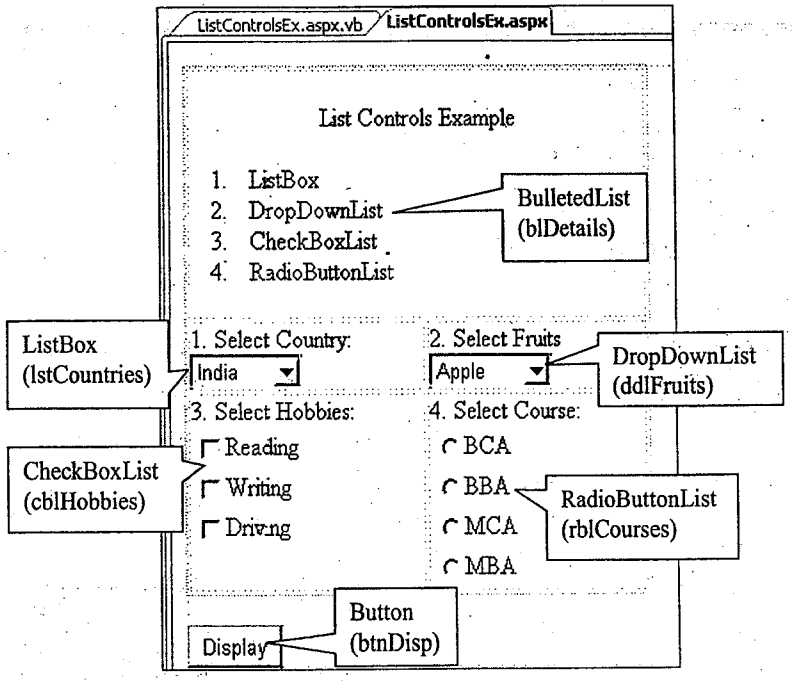
Properties	Description
BulletStyle	It determines the type of list. **Example** Numbered (1, 2, 3...), LowerAlpha (a, b, c...) and UpperAlpha (A, B, C...), LowerRoman (i, ii, iii...) and UpperRoman (I, II, III...), and the bullet symbols Disc, Circle, Square, or CustomImage.
BulletImageUrl	If the BulletStyle property is set to CustomImage, this property contains the Image path that is to be placed to the left of each item as a bullet.
FirstBulletNumber	If the BulletStyle is set to Numbered, LowerAlpha, UpperAlpha, LowerRoman, or UpperRoman styles, then this property sets the first value for the order in the list. **Example** If you set FirstBulletNumber to 4, the list will start from 4, 5, 6 (for Numbered) or D, E, F (for UpperAlpha).
DisplayMode	It determines whether the text of each item is rendered as text or a hyperlink (LinkButton or HyperLink).

[Table: 20 BulletedList Control Properties]

✓ **Default Event:** Click()

To set the Data Source for any of the List control, you can use DataSource property or select "Choose Data Source" option from the smart tag. User can use the Data Source as explain in Chapter 7.

Example
Design the page as shown in fig (5.v).



[Fig (5.v) List Controls Example (Design View)]

Write the following code on the Click() Event of the btnDisp button of the page:

```
Protected Sub btnDisp_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnDisp.Click
    Dim ct, fr, hb, cs As String
    ct = lstCountries.SelectedItem.Text
    fr = ddlNames.SelectedItem.Text
    For Each lstitm As ListItem In cblHobbies.Items
      If lstitm.Selected = True Then
        hb = hb + lstitm.Text
        hb = hb + " "
      End If
    Next
    cs = rblCourses.SelectedItem.Text
      MsgBox("Your Selected Country:" + ct + vbCrLf + "Your Selected Fruit:" + fr +
    vbCrLf + "Your Hobbies:" + hb + vbCrLf + "Your Course:" + cs,
    MsgBoxStyle.Information, "Info")
  End Sub
```

Save & execute the page.

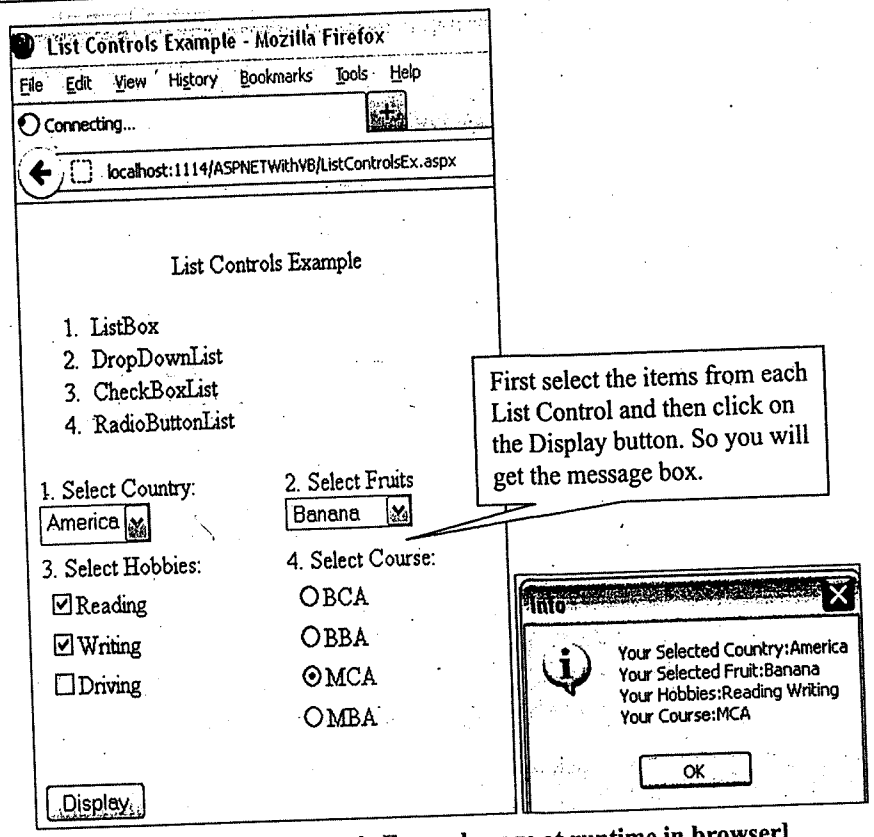
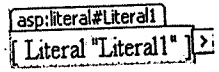


Fig (5.w) List Controls Example page at runtime in browser]

- **Literal Control**

The Literal control is used to reserve a location on the Web page to display text (fig. (5.x)). The Literal control is similar to the Label control, except the Literal control does not allow you to apply a style to the displayed text.
Its ASP.NET tag is <asp:Literal/>.



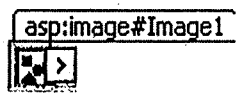
[Fig (5.x) Literal Control]

Properties

Properties	Description
Text	It gets or sets the Text displayed in the Literal control.

[Table: 21 Literal Control Properties]

- **Image Control**



[Fig (5.y) Image Control]

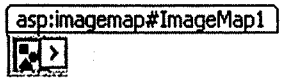
The Image control is used to display any valid image supported by the requesting browser on the Web page (fig. (5.y)).
Its ASP.NET tag is <asp:Image/> and HTML tag is <img/>.

Properties

Properties	Description
AlternateText	It gets or sets the Text displayed in the Image control when the image is not available.
ImageAlign	It gets or sets alignment of the image in relation to other elements on the Web page.
ImageUrl	It gets or sets the path to the image.
DescriptionUrl	It gets or sets the location of detailed description of the image.

[Table: 22 Image Control Properties]

- **ImageMap Control**



[Fig (5.z) ImageMap Control]

The ImageMap control is used to create an image that contains hot spot regions (fig. (5.z)). When a user clicks on hot spot region, the control can either generate a postback to the server or navigate to a specified URL.
Its ASP.NET tag is <asp:ImageMap/> and HTML tag is <map>.
This control is used to display a map of any country. When a user clicks on specific state of the map, the control navigates to a URL that provides additional data about the selected state. You can also use this control to generate a postback to the server and run specific code based on the hot spot region that was clicked.
You can also specify one hot spot region to navigate to a URL and another hot spot region to post back to the server.

Properties

ImageMap Control supports all the properties of Image control (Table: 22), and it has its own properties (Table: 23)

Properties	Description
HotSpots	It gets a collection of HotSpot objects that represents the defined hot spot regions in the control. **Example** CircleHotSpot (Properties: Radius, X, Y) RectangleHotSpot (Properties: Top, Right, Bottom, Left) PolygonHotSpot (Property: Coordinates) You can use these 3 classes to define hot spot regions.
HotSpotMode	It gets or sets the default behavior for the HotSpot objects of the control when the HotSpot objects are clicked. Its value can be Navigate or PostBack or Inactive.

[Table: 23 ImageMap Control Properties]

- **FileUpload Control**

The FileUpload Control is used to displays a text box control and a browse button that enable users to select a file on the client and upload it to the Web server (fig. (5.aa)). You can specify the file to upload by entering the full path of the file on the local computer (example, D:\Test.txt) in the text box of the control, or you can also select the file by clicking the Browse button, and then locating it in the Choose File dialog box.

[Fig (5.aa) FileUpload Control]

Its ASP.NET tag is <asp:FileUpload/> and HTML is <imput type="file">.

Properties

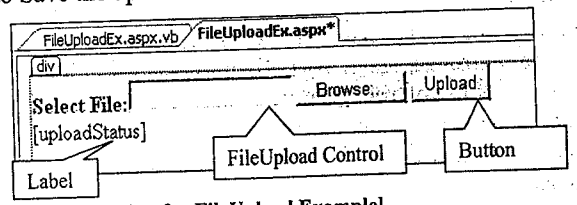
Properties	Description
FileBytes	It gets all the bytes in the form of array of the file.
FileName	It gets the name of a file on a client to upload.
FileContent	It gets a Stream object that point to a file to upload.
HasFile	It gets a value indicating whether the FileUpload control contains a file or not.
PostedFile	It gets the underlying HttpPostedFile object for a file that is uploaded by using the FileUpload control.

[Table: 24 FileUpload Control Properties]

You can use SaveAs() method to Save the uploaded file on the server.

Example

Design the page as Fig (5.x)

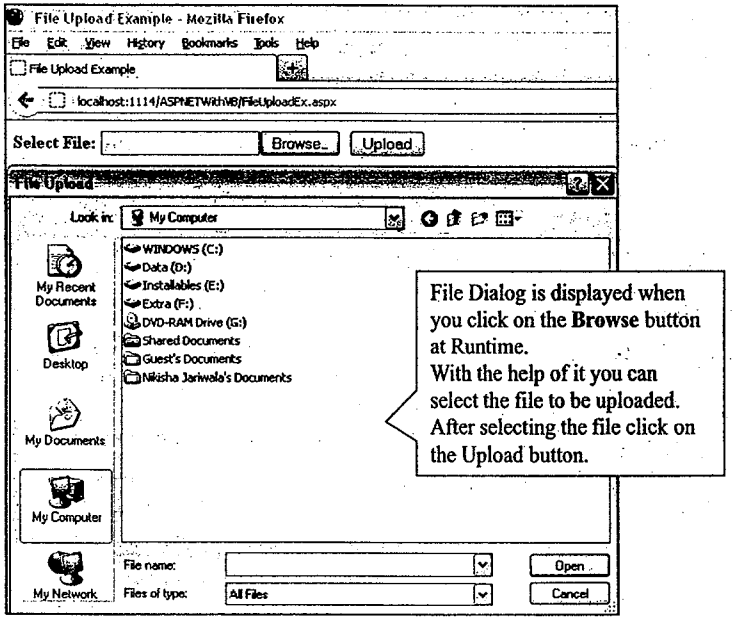


[Fig (5.ab) Page design for FileUpload Example]

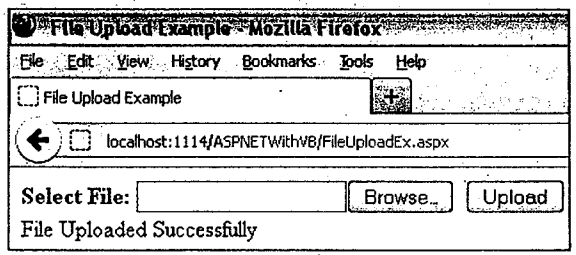
Write the following code on the BtnUpload() Click Event:

```
Protected Sub btnUpload_Click
    Dim saveDir As String = "\UploadedFiles\"
    Dim appPath As String = Request.PhysicalApplicationPath
    If FileUpload1.HasFile Then
      Dim savePath As String = appPath + saveDir + _
        Server.HtmlEncode(FileUpload1.FileName)
      FileUpload1.SaveAs(savePath)
      uploadStatus.Text = "File Uploaded Successfully"
    Else
      uploadStatus.Text = "Not able to Upload File"
    End If
  End Sub
```

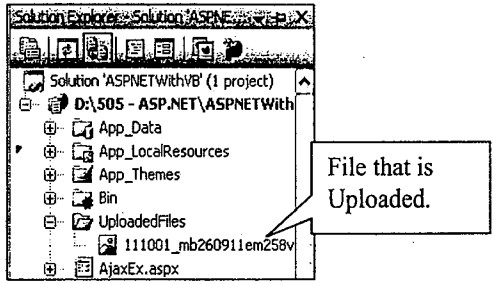
Save and execute the page (fig (5.y), (5.z) & (5.aa).



[Fig (5.ac) Runtime Page View of FileUploadEx page]



[Fig (5.ad) Successful upload message]

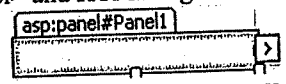


[Fig (5.ae) File in UploadedFiles Folder after the successful Upload]

Panel Control

The Panel control is used as a container for other controls (fig. (5.af)). It is useful when you want to generate controls programmatically, or hide/show a group of controls.

Its ASP.NET tag is <asp:Panel> and HTML tag is ... .



[Fig (5.af) Panel Control]

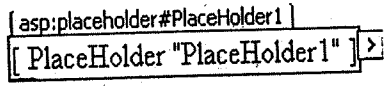
Properties

Properties	Description
BackImageUrl	It gets or sets the URL of the background image.
Direction	It gets or sets the direction in which to display controls that include text in a Panel control.
GroupingText	It gets or sets the caption for the group of controls.
HorizontalAlign	It gets or sets the horizontal alignment of the contents.
Scrollbars	It gets or sets the visibility and position of scroll bars.
Wrap	It gets or sets a value indicating whether the content wraps or not.

[Table: 25 Panel Control Properties]

- **PlaceHolder Control**

The PlaceHolder control is used as a container to store server controls that are dynamically added to the Web page (fig. (5.ag)). It just holds the place on the page. It does not give any visible output. You can use the Control.Controls collection to add, insert, or remove a control in the PlaceHolder control. It does not have a specific property.

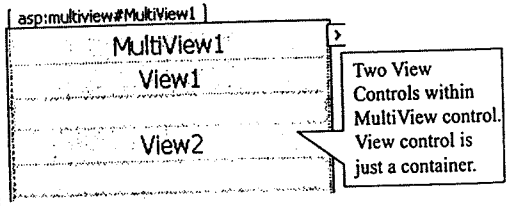


[Fig (5.ag) PlaceHolder Control]

Its ASP.NET tag is <asp:PlaceHolder/>.

- **MultiView Control**

The MultiView control is used as a container for a group of View controls (fig. (5.ah)). It allows you to define a group of View controls, where each View control contains child controls. You can also use the MultiView control to create wizards. Only one View control at a time can be defined as an active view within the MultiView control. When a View control is defined as the active view, the child controls that it contains are rendered to the client. You can also use SetActiveView() method to set the active View at runtime.



[Fig (5.ah) MultiView Control

Its ASP.NET tag is <asp:MultiView/> and <asp:View/>.

Properties

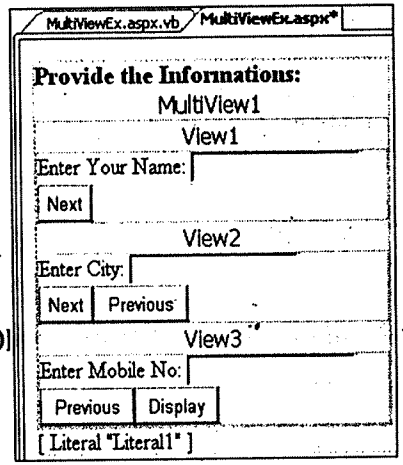
Properties	Description
ActiveViewIndex	It gets or sets the index of the active View control within a MultiView control.
Views	It gets the collection of View controls in the MultiView control.

[Table: 26 MultiView Control Properties]

✓ **Default Event:** ActiveViewChanged()

Example

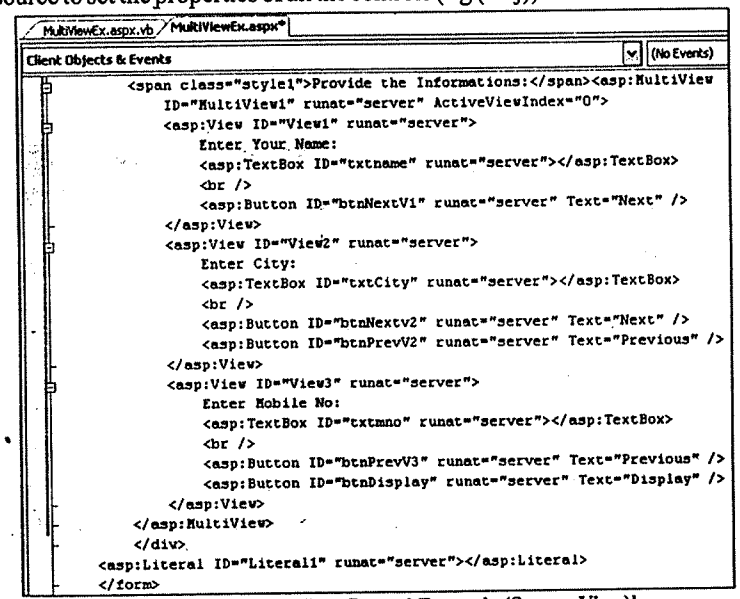
Design the page. Place MultiView control and 3 View controls within it. Then place other controls (fig (5.ai)). Set ActiveViewIndex property to 0 of MultiView control, so when the page is rendered in the browser, by default View1 is displayed.



[Fig (5.ai) MultiView Control Example (Design View)]

Source View

View the source to set the properties of all the controls (fig (5.aj))

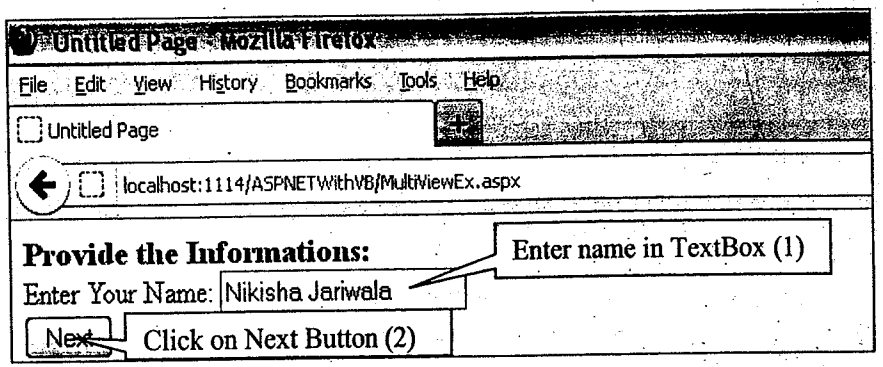


[Fig (5.aj) MultiView Control Example (Source View)]

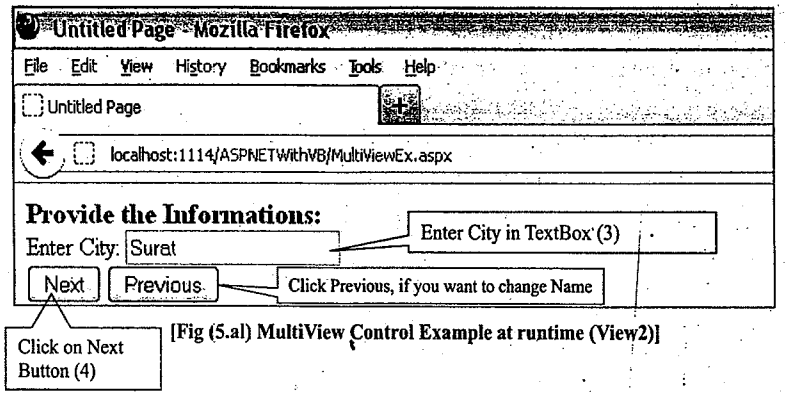
Write the following code on the Click() Event of the buttons on the page:

```
Protected Sub btnNextV1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnNextV1.Click
    MultiView1.ActiveViewIndex += 1
  End Sub
Protected Sub btnNextv2_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnNextv2.Click
    MultiView1.ActiveViewIndex += 1
  End Sub

  Protected Sub btnPrevV2_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnPrevV2.Click
    MultiView1.ActiveViewIndex -= 1
  End Sub

  Protected Sub btnPrevV3_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnPrevV3.Click
    MultiView1.ActiveViewIndex -= 1
  End Sub

  Protected Sub btnDisplay_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnDisplay.Click
    Literal1.Text = txtname.Text + " " + txtCity.Text + " " + txtmno.Text
  End Sub
```

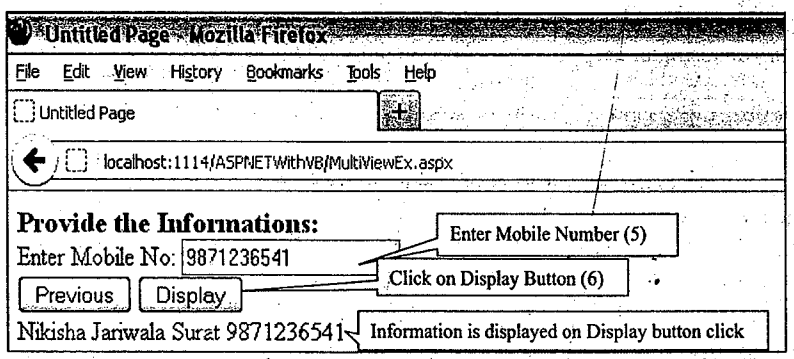
Save and execute the page (fig. (5.ak)).



[Fig (5.ak) MultiView Control Example at runtime (View1)]



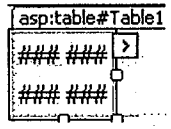
[Fig (5.al) MultiView Control Example at runtime (View2)]



[Fig (5.am) MultiView Control Example at runtime (View3)]

- **Table Control**

The Table control is used to build an HTML table (fig. (5.an)). A table can be built at design time, but you can also build the table programmatically with dynamic contents.
Its ASP.NET tag is <asp:Table> and HTML tag is <table/>.



[Fig (5.an) Table Control]

Source View

```
<asp:Table ID="Table1" runat="server">
    <asp:TableRow runat="server">
      <asp:TableCell runat="server"></asp:TableCell>
      <asp:TableCell runat="server"></asp:TableCell>
    </asp:TableRow>
    <asp:TableRow runat="server">
      <asp:TableCell runat="server"></asp:TableCell>
      <asp:TableCell runat="server"></asp:TableCell>
    </asp:TableRow>
  </asp:Table>
```

Properties

Properties	Description
HorizontalAlignment	It gets or sets the horizontal alignment of the Table control on the page.
Rows	It gets the collection of rows in the Table control.
GridLines	It gets or sets the grid line style to display in the Table control.

[Table: 27 Table Control Properties]

TableRow Class

The TableRow class is used to represents a row in a Table control. This class allows you to control how the contents of the row are displayed.
Its ASP.NET tag is <asp:TableRow/> and HTML tag is<tr/>.

Properties

Properties	Description
HorizontalAlignment	It gets or sets the horizontal alignment of the Table control on the page.
VerticalAlignment	It gets or sets the vertical alignment of the contents in the row.

[Table: 28 TableRow Class Properties]

TableCell Class

The TableCell class is used to represents a cell in a Table control. The TableCell class allows you to control how the contents of the cell are displayed.
Its ASP.NET tag is <asp:TableCell/> and HTML tag is <td/>.

Properties

Properties	Description
HorizontalAlignment	It gets or sets the horizontal alignment of the Table control on the page.
VerticalAlignment	It gets or sets the vertical alignment of the contents in the row.
Text	It gets or sets the text contents of the cell.
Wrap	It gets or sets a value that indicating whether the contents of the cell can be wrapped or not.
RowSpan	It gets or sets the number of rows in the Table control that the cell spans.
ColumnSpan	It gets or sets the number of columns in the Table control that the cell spans.

[Table: 29 TableCell Class Properties]

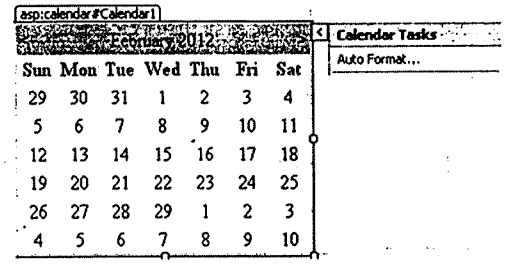
5.2.2 Rich Controls

Rich controls are web controls that have complex user interface elements. It renders itself with a complex sequence of HTML elements and may even use client-side JavaScript. There are two Rich Controls:

1. Calender
2. AdRotator

(1) Calender

The Calender control is used to create a rich functionally and good-looking calendar that shows one month at a time ((fig. (5.ao)). The user can move from month to month, select a date, and even select a range of days.
Its ASP.NET tag is <asp:Calender/>.



[Fig (5.ao) Calender Control]

Properties

Properties	Description
DayHeaderStyle	It gets the style properties for the section that displays the day of the week.
DayNameFormat	It gets or sets the name format for days of the week.
DayStyle	It gets the style properties for the days in the displayed month.
FirstDayOfWeek	It gets or sets the day of the week to be display in the first day column.
NextMonthText	It gets or sets the text displayed for the next month navigation control.
NextPrevFormat	It gets or sets the format of the next and previous month navigation elements in the title.
NextPrevStyle	It gets the style properties for the next and previous month navigation elements.
PrevMonthText	It gets or sets the text displayed for the previous month navigation control.
SelectedDate	It gets or sets the selected date.
SelectedDates	It gets a collection of System.DateTime objects that represent the selected date.
SelectionMode	It gets or sets the date selection mode that specifies whether the user can select a single day, a week, or an entire month.
SelectMonthText	It gets or sets the text displayed for the month selection element in the selector column.

SelectWeekText	It gets or sets the text displayed for the week selection element in the selector column.
ShowDayHeader	It gets or sets a value indicating whether the heading for the days of the week is displayed or not.
ShowGridLines	It gets or sets a value indicating whether the days are separated with gridlines or not.
ShowNextPrevMonth	It gets or sets a value indicating whether the next and previous month navigation elements are to be displayed or not.
ShowTitle	It gets or sets a value indicating whether the title is displayed or not.
TodaysDate	It gets or sets the value for today's date.

[Table: 30 Calender Control Properties]

✓ **Default Event:** SelectionChanged()

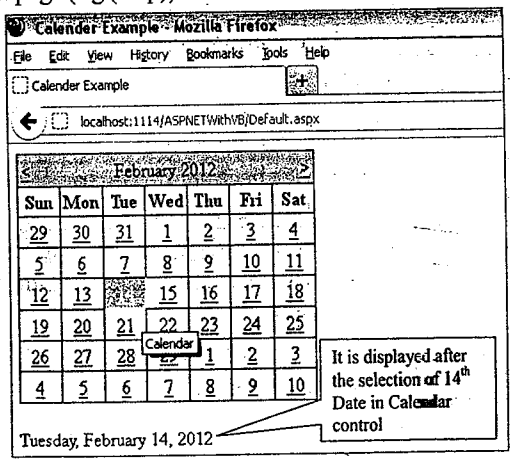
Example

Place the Calender control on the page (fig. (5.ap)). Set its properties as you want.

Write the following code on the SelectionChanged() Event of the calendar control.

```
Protected Sub Calendar1_SelectionChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Calendar1.SelectionChanged
    lblSelectedDate.Text = Calendar1.SelectedDate.ToLongDateString
End Sub
```

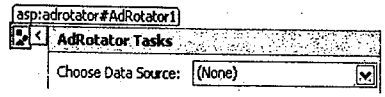
Save and execute the page (fig (5.ap)).



[Fig (5.ap) Calender Control Example page at runtime]

(2) AdRotator Control

The AdRotator control is used to randomly select the banner from the list that is specified in the external XML file (fig. (5.aq)). Before using AdRotator control you need to create the XML file. Its ASP.NET tag is <asp:AdRotator/>.



[Fig (5.aq) AdRotator Control]

Properties

Properties	Description
AdvertisementFile	It gets or sets the path to an XML file that contains advertisement information.
AlternateTextField	It gets or sets a custom data field to use in place of the AlternateText attribute for an advertisement.
ImageUrlField	It gets or sets a custom data field to use in place of the ImageUrl attribute for an advertisement.
NavigateUrlField	It gets or sets a custom data field to use in place of the NavigateUrl attribute for an advertisement.
KeyWordFilter	It gets or sets a category keyword to filter for specific types of advertisements in the XML advertisement file.

[Table: 31 AdRotator Control properties]

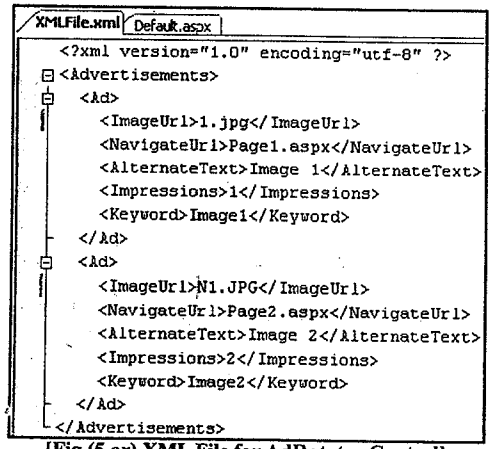
✓ **Default Event:** AdCreated()

Example

To add XML file,

- Right Click on the Website in the Solution Explorer
- Select Add New Item from the Popup menu
- Select XML File Template from the wizard
- Give appropriate name and Click on Add button.

Write following code in XML file (fig (5.ar)):



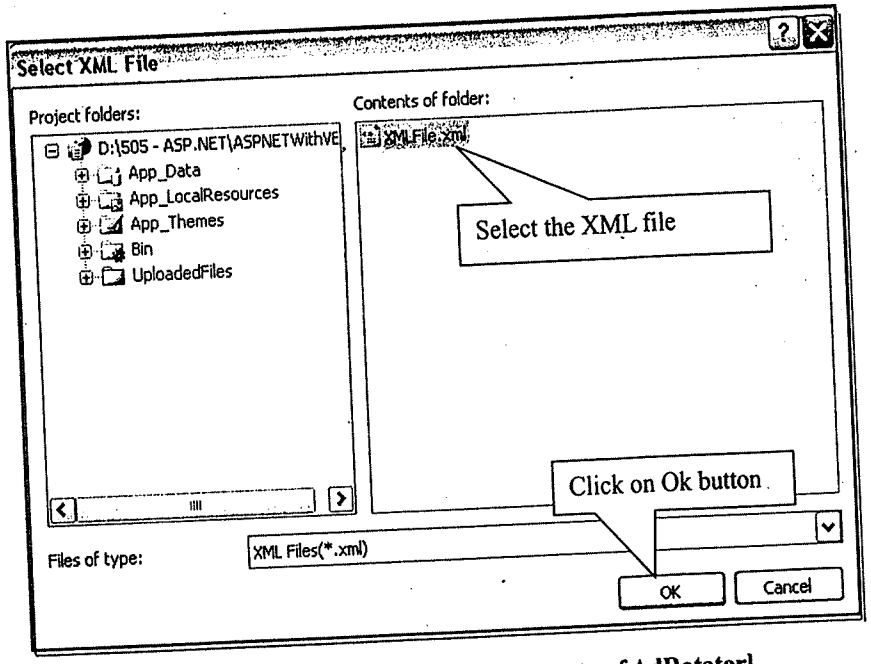
[Fig (5.ar) XML File for AdRotator Control]

Following are the elements of XML File used for storing information of Advertisements

Elements	Description
ImageUrl	The absolute or relative path of the image.
NavigateUrl	The URL of a page to link to if the user clicks the ad.
AlternateText	The text display in place of the image when the image specified by the ImageUrl property is not available
Keyword	A category for the advertisement, so you can filter.
Impressions	A number that indicates the importance of the ad. The larger the number, the more often the ad is displayed.

[Table: 32 Elements of XML file for Advertisement]

Set the AdvertisementFile Property of the AdRotator control to the XML file that you have added in the Website (fig (5.as).

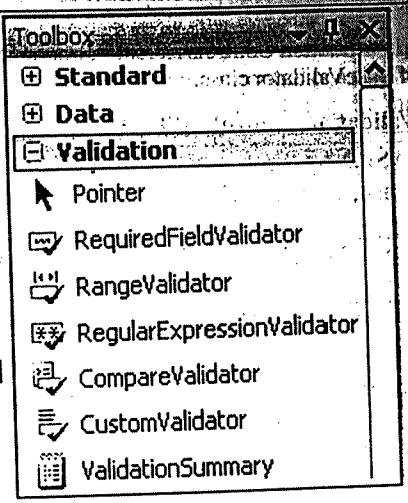


[Fig (5.as) Setting AdvertisementFile property of AdRotator]

Save and execute the page.

5.2.3 Validation Controls

These controls allow you to validate an input control against several standard or user- defined rules. If validation fails, you can prevent page processing or allow these controls to show error messages in the page. In Visual Studio, these controls are found in the Validation tab (fig. (5.aq)) of the Toolbox.



[Fig (5.at) Validation Controls in ToolBox]

BaseValidator Class

The BaseValidator class provides the core implementation for all validation controls. Validation controls are used to validate user input in an associated input control. When the user enters a value that does not match the validation criteria, the validation control displays an error message. Because a validation control is separated from the input control, you can position the error message anywhere on the page relative to the input control. ASP.NET provides several validation controls that perform specific types of validation. The following table describes these controls.

Properties

Properties	Description
ControlToValidate	It gets or sets the input control to validate.
ErrorMessage	It gets or sets the text for the error message displayed in a ValidationSummary control when validation fails.
Text	It gets or sets the text displayed in the validation control when validation fails.
Display	It gets or sets the display behavior of the error.
IsValid	It gets or sets a value that indicates whether the associated input control passes validation or not.
SetFocusOnError	It gets or sets a value that indicates whether focus is set to the control specified by the ControlToValidate property when validation fails.
PropertiesValid	It gets a value that indicates whether the control specified by the ControlToValidate property is a valid control or not.
ValidationGroup	It gets or sets the name of the validation group to which this validation control belongs.
ValidationEmptyText	It gets or sets the value that indicates whether the validator validates the control when the text of the control is empty.

[Table: 33 BaseValidator Class Properties]

All Validation Controls except ValidationSummary control inherits all the common properties of BaseValidator class.

Validation Controls ASP.NET tag and Design View

ASP.NET Tag	Control
<asp:RequiredFieldValidator/>	asp:requiredfield...#RequiredField... RequiredFieldValidator
<asp:RangeValidator/>	asp:rangevalidator#RangeValidator1 RangeValidator
<asp:CompageValidator/>	asp:comparevalidator#CompareValida... CompareValidator
<asp:RegularExpressionValidator/>	asp:regularexpres...#RegularExpres... RegularExpressionValidator
<asp:validationSummary/>	asp:validationsum...#ValidationSum... • Error message 1. • Error message 2.
<asp:CustomValidator/>	asp:customvalidator#CustomValidator1 CustomValidator

[Table: 34 ASP.NET tag and Design View of Validation Controls]

- **RequiredFieldValidator**

The RequiredFieldValidator control is used to make an input control a required field (Table: 34).

It checks whether the control is empty or not when the form is submitted. Multiple validators can also be associated with the same input control.

Properties

Properties	Description
InitialValue	It gets or sets the initial value of the field.

[Table: 35 RequiredFieldValidator Control Properties]

- **RangeValidator**

The RangeValidator control is used to tests whether the value of an input control is within a specified range or not (Table: 34).

Properties

Properties	Description
MinimumValue	It gets or sets the minimum value of the validation range.
MaximumValue	It gets or sets the maximum value of the validation range.
Type	It gets or sets the data type that the values being compared are converted to before the comparison is made. Data type supported are String, Integer, Double, Date, Currency.

[Table: 36 RangeValidator Control Properties]

- **CompareValidator**

The CompareValidator control is used to compare the value entered by the user in an input control, such as a TextBox control, with the value entered in another input control or a constant value (Table: 34). The CompareValidator control passes validation if the value of the input control matches the criteria.

Properties

Properties	Description
ControlToCompare	It gets or sets the input control to compare with the input control being validated.
ValueToCompare	It gets or sets a constant value to compare with the value entered by the user in the input control being validated.
Operator	It gets or sets the comparison operation to perform.
Type	It gets or sets the data type that the values being compared are converted to before the comparison is made. Data type supported are String, Integer, Double, Date, Currency.

[Table: 37 CompareValidator Control Properties]

- **RegularExpressionValidator**

The RegularExpressionValidator control is used to check whether the value of an input control matches a pattern defined by a regular expression (Table: 34). This type of validation allows you to check e-mail addresses, telephone numbers, and postal codes.

Properties

Properties	Description
ValidationExpression	It gets or sets the regular expression that determines the pattern used to validate a field. There are various predefine expression available, so you can also use it.

[Table: 38 RegularExpressionValidator Control Properties]

- **ValidationSummary**

The ValidationSummary Control is used to summarize the error messages from all validators on a Web page in a single location (Table: 34). You can also summarize the error messages from particular group of validators using ValidationGroup property

Properties

Properties	Description
DisplayMode	It gets or sets the display mode of the validation summary (List, BulletList, SingleParagraph).
ShowSummary	It gets or sets a value indicating whether the validation summary is displayed or not.
ShowMessageBox	It gets or sets a value indicating whether the validation summary is displayed in a message box or not.

[Table: 39 ValidationSummary Control Properties]

- **CustomValidator**

The CustomValidator control is used to provide a user-defined validation function for an input control (Table: 34). Validation controls always perform validation on the server. They also have complete client-side implementation that allows script-enabled browsers to perform validation on the client.

To create a server-side validation function, you need to provide a handler for the ServerValidate event that performs the validation. The value of the input control that is to be validated can be accessed by using the Value property of the ServerValidateEventArgs object passed into the event handler as a parameter. The result of the validation is then stored in the IsValid property of the ServerValidateEventArgs object.

To create a client-side validation function, first add the server-side validation function. Next, add the client-side validation script function to the ASP.NET (.aspx) page.

Properties

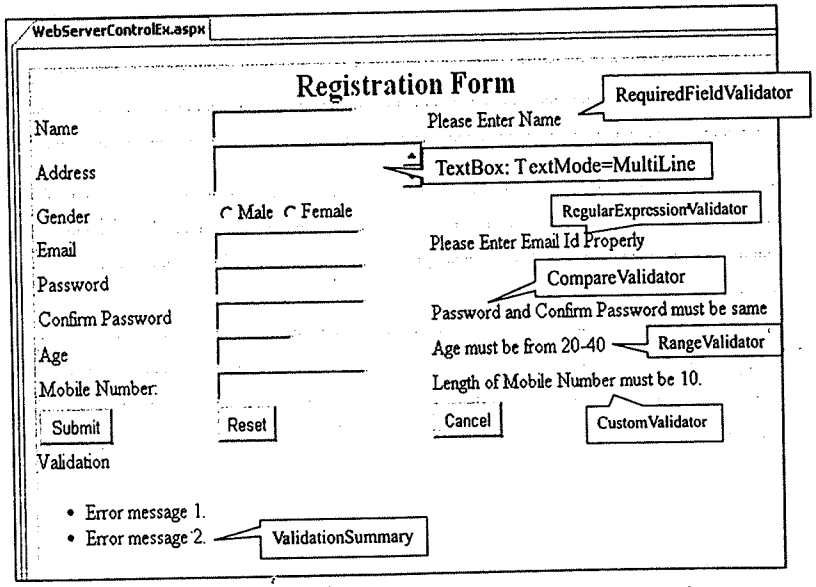
Properties	Description
ClientValidationFunction	It gets or sets the name to the client script validation function.

[Table: 40 CustomValidator Control Properties]

✓ **Default Event:** ServerValidate()

Example

Design the page (fig (5.au))



[Fig (5.au) Standard & Validation Controls Example (Design Page)]

By viewing the Source View of the page, user will be able to set the properties of each control on the page.

Source View

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Registration Form</title></head>
<body>
  <form id="form1" runat="server">
  <table>
    <tr>
          <td align="center" colspan="4"> <asp:Label ID="Label1" runat="server"
          BorderColor= "#009933" Font-Bold="True" Font-Size="Larger" ForeColor="Red"
          style="text-align: center" Text= "Registration Form"></asp:Label> </td>
    </tr>
    <tr>
          <td><asp:Label ID="Label2" runat="server" Text="Name"> </asp:Label></td>
          <td><asp:TextBox ID="txtfname" runat="server" Width= "122px">
          </asp:TextBox> </td>
          <td colspan="2"> <asp:RequiredFieldValidator ID= "RequiredFieldValidator1"
          runat="server"
          ControlToValidate="txtfname" ErrorMessage="Please Enter Name"
          SetFocusOnError="True">Please Enter Name </asp:RequiredFieldValidator>
          </td>
    </tr>
    <tr>
      <td>Address</td>
          <td> <asp:TextBox ID="txtAddress" runat="server" TextMode= "MultiLine">
          </asp:TextBox> </td>
      <td colspan="2"> </td>
    </tr>
    <tr>
      <td>Gender</td>
          <td><asp:RadioButton ID="rdbMale" runat="server" GroupName="g"
          Text="Male" /> <asp:RadioButton ID="rdbfemale" runat="server"
          Text="Female" /></td>
      <td colspan="2"> </td>
    </tr>
    <tr>
          <td><asp:Label ID="Label4" runat="server" Text="Email" ></asp:Label></td>
          <td><asp:TextBox ID="txtemail" runat="server" Width="125px"
          SkinID="TextBoxEmail" ></asp:TextBox></td>
          <td> <asp:RegularExpressionValidator ID= "RegularExpressionValidator1"
          runat="server" ControlToValidate= "txtemail" ErrorMessage="Please Enter Email
          Id Properly" SetFocusOnError="True"
          ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+ ([-.]\w+)*">Please
          Enter Em1ail Id Properly </asp: RegularExpressionValidator> </td>
```

```
</tr>
<tr>
        <td><asp:Label ID="Label5" runat="server" Text=
        "Password"></asp:Label></td>
        <td> <asp:TextBox ID="txtpwd" runat="server" TextMode="Password">
        </asp:TextBox></td>
  <td colspan="2"> </td>
</tr>
<tr>
        <td><asp:Label ID="Label6" runat="server" Text="Confirm
        Password"></asp:Label></td>
        <td> <asp:TextBox ID="txtcpwd" runat="server" TextMode="Password">
        </asp:TextBox></td>
        <td colspan="2"> <asp:CompareValidator ID= "CompareValidator1"
        runat="server"
        ControlToCompare="txtpwd"ControlToValidate="txtcpwd"
        ErrorMessage="Password and Confirm Password must be same"
        SetFocusOnError="True" Type="Integer">Password and Confirm Password must
        be same </asp:CompareValidator> </td>
</tr>
<tr>
        <td><asp:Label ID="Label7" runat="server" Text= "Age"></asp:Label></td>
        <td><asp:TextBox ID="txtAge" runat="server"
        Width="65px"></asp:TextBox></td>
        <td colspan="2"> <asp:RangeValidator ID="RangeValidator1" runat="server"
        ControlToValidate="txtAge" ErrorMessage="Age must be from 20-40"
        MaximumValue="40" MinimumValue="20" SetFocusOnError="True"
        Type="Integer">Age must be from 20-40 </asp:RangeValidator> </td>
</tr>
<tr>
  <td>Mobile Number:</td>
        <td><asp:TextBox ID="txtmno" runat= "server"> </asp:TextBox></td>
        <td colspan="2"> <asp:CustomValidator ID= "CustomValidator1" runat="server"
        ControlToValidate="txtmno" ErrorMessage="Length of Mobile Number must be
        10."></asp:CustomValidator> </td>
</tr>
<tr>
        <td><asp:Button ID="btnsubmit" runat="server" Text= "Submit" /></td>
        <td><asp:Button ID="btnreset"runat="server"Text="Reset" /></td>
        <td colspan="2"><asp:Button ID="cancel" runat="server" Text="Cancel" /></td>
</tr>
</table>
    <div> <asp:ValidationSummary ID="ValidationSummary1" runat="server"
    HeaderText="Validation" ShowMessageBox="True" /> </div>
</form></body></html>
```

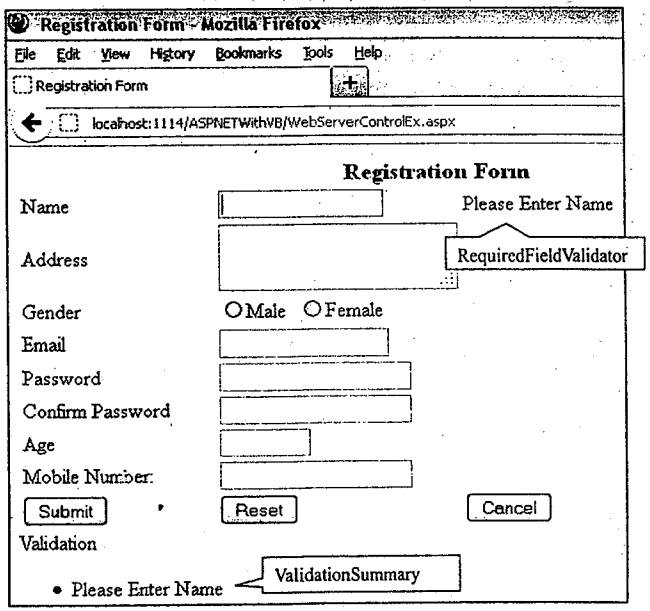
Write the following code for the CustomerValidator control in the Code View of the page:


```
Sub LengthCheck(ByVal sender As System.Object, ByVal args As ServerValidateEventArgs)
Handles CustomValidator1.ServerValidate
    Dim len As Integer = args.Value.Length
    If len = 10 Then
      args.IsValid = True
    Else
      args.IsValid = False
    End If
End Sub
```

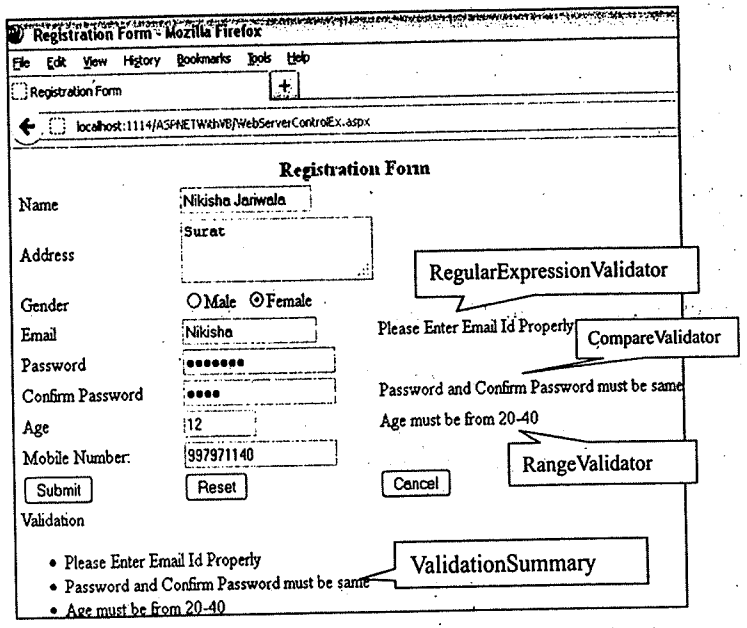
Write the following code in the Click() Event of the btnSubmit Button of the Page:

```
Protected Sub btnsubmit_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnsubmit.Click
    If Page.IsValid Then
      Response.Redirect("Welcome.aspx")
    End If
End Sub
```

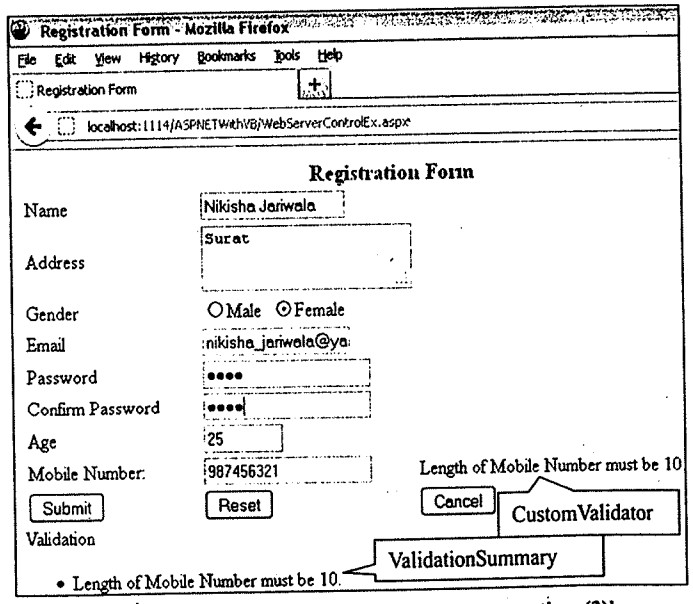
Save & execute the page (fig (5.av), (5.aw) and (5.ax)).



[Fig (5.av) Validation Controls Example page at runtime (1)]



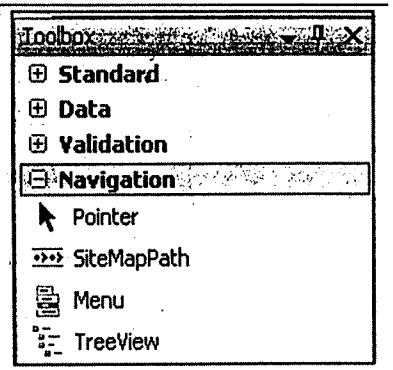
[Fig (5.aw) Validation Controls Example page at runtime (2)]



[Fig (5.ax) Validation Controls Example page at runtime (3)]

5.2.4 Navigation Controls

Navigation is the fundamental component of any website. Navigation controls (fig (5.ay)) allow you to design and display site maps and also allow the user to navigate from one page to another.

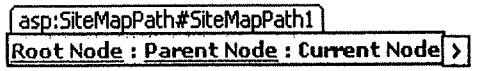


[Fig (5.ay) Navigation Controls in ToolBox]

- **SiteMapPath Control**

The SiteMapPath Control is used for defining the navigation structure of the website (fig. (5.az)). You can bind this control to TreeView and Menu controls. It is made of Nodes (root, parent, current).

Its ASP.NET tag is <asp:SiteMapPath/>.



[Fig (5.az) SiteMapPath Control]

Properties

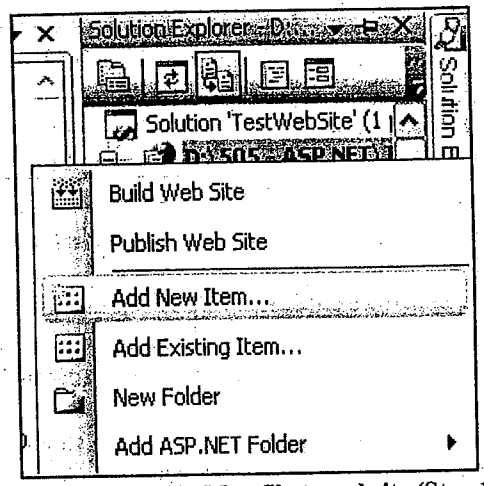
Properties	Description
PathDirection	It gets or sets the order that the navigation path nodes are rendered in.
PathSeperator	It gets or sets the string that delimits SiteMapPath nodes in the rendered navigation path.
SiteMapProvider	It gets or sets the name of the SiteMapProvider used to render the site navigation control.
RootNodeStyle	It gets the style for the root node display text.
CurrentNodeStyle	It gets the style for the current node display text.
NodeStyle	It gets the style used for the display text for all nodes in the site navigation path.

[Table: 41 SiteMapPath Control Properties]

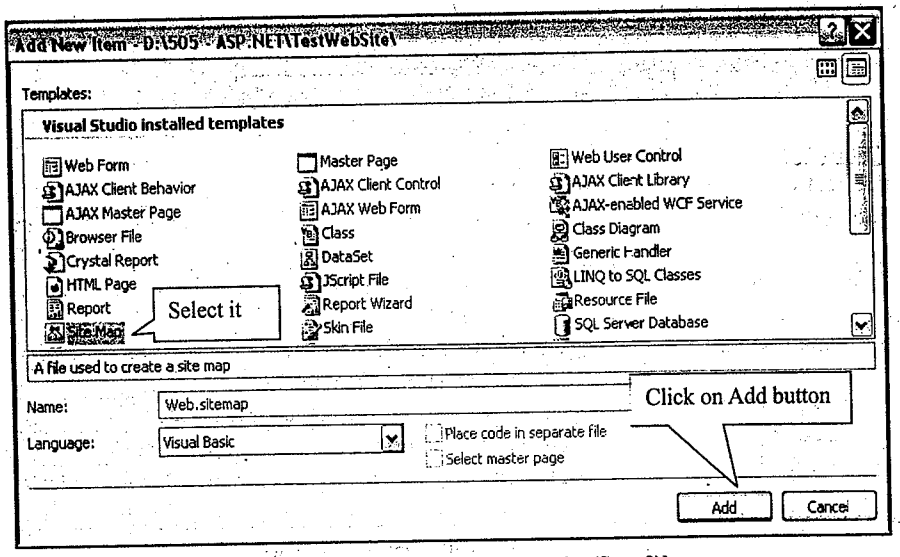
You need web.sitemap file to create Site Map for the website. When you add the web.sitemap file to your website, it automatically gets attach with the SiteMapPath control.

To add web.sitemap file in your website:

- Right Click on the Web site in solution explorer (fig (5.ba)).
- Select Add New Item (fig (5.ba))
- Select Site Map template from the wizard (fig (5.bb)).
- Give appropriate name
- Click on Add button.



[Fig (5.ba) Adding SiteMap file to website (Step 1 & 2)]



[Fig (5.bb) Adding SiteMap file to website (Step 3)]

The web.sitemap file (fig (5.bc)) contains <siteMap> and <siteMapNode> tags in it. You can define the site map with <siteMapNode>.

The SiteMap file must contain <sitemap> node. The <siteMap> tag can only have one <siteMapNode> child node. Each <siteMapNode> can have several child nodes. Each <siteMapNode> has attributes defining page title and URL.

Web2.sitemap | Web.sitemap | NavigationEx.aspx.vb | NavigationEx.aspx* | Start Page

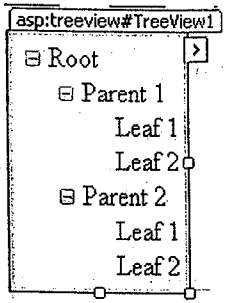
```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="" title="" description="">
        <siteMapNode url="" title="" description="" />
        <siteMapNode url="" title="" description="" />
    </siteMapNode>
</siteMap>
```

[Fig (5.bc) web.sitemap file]

- **TreeView Control**

The TreeView control is used to display hierarchical data, such as a table of contents or file directory or sitemap links, in a tree structure (fig. (5.bd)). It supports data binding that allows the nodes of the control to be bound to XML, tabular, or relational data. It also provides Site navigation through integration with the SiteMapDataSource control. The Node text of the TreeView Control can be displayed as either plain text or hyperlinks and can also have check box next to each node.

Its ASP.NET tag is <asp:TreeView/>.



[Fig (5.bd) TreeView Control]

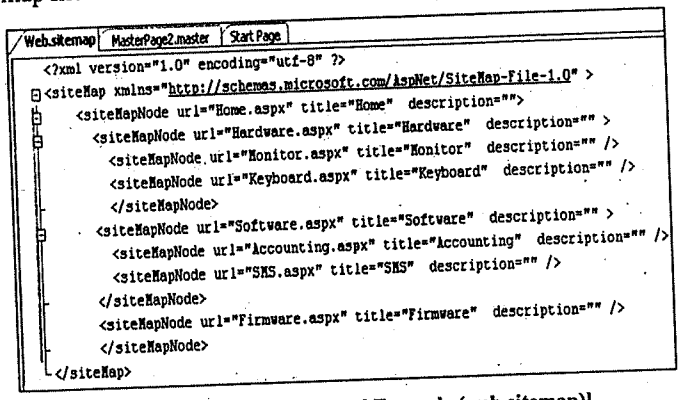
Properties

Properties	Description
CheckedNodes	It gets a collection of TreeNode objects that represent the nodes in the TreeView control that display a selected check box.
CollapseImageUrl	It gets or sets the URL to a custom image for the collapsible node indicator.
ExpandImageUrl	It gets or sets the URL to a custom image for the expandable node indicator.
DataSource	It gets or sets the object from which the data-bound control retrieves its list of data items.
DataSourceID	It gets or sets the ID of the control from which the data-bound control retrieves its list of data items.
HoverNodeStyle	It gets the appearance of a node when the mouse pointer is positioned over it.
LeafNodeStyle	It gets the appearance of leaf nodes.
LevelStyles	It gets a collection of Style objects that represent the node styles at the individual levels of the tree.
NodeIndent	It gets or sets the indentation amount (in pixels) for the child nodes of the TreeView control.

Add following Content Pages (Chapter 6: Topic 6.1 Adding Content Page to Master Page) for the above created Master Page.

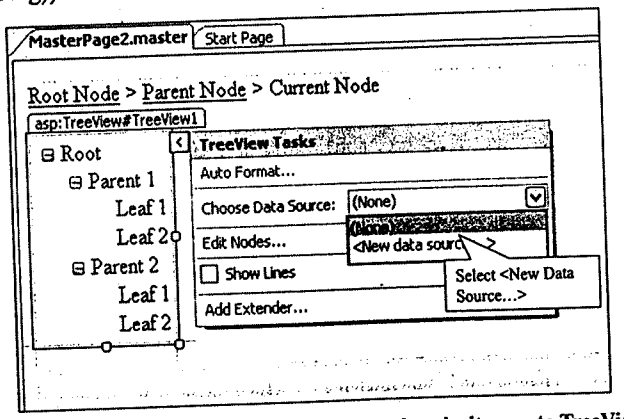
Home.aspx
Hardware.aspx
Monitor.aspx
Keyboard.aspx
Software.aspx
Accounting.aspx
SMS.aspx

Add web.sitemap file to the website (Fig (5.ba) and (5.bb)) and add the <siteMapNode/> in it (fig (5.bf))



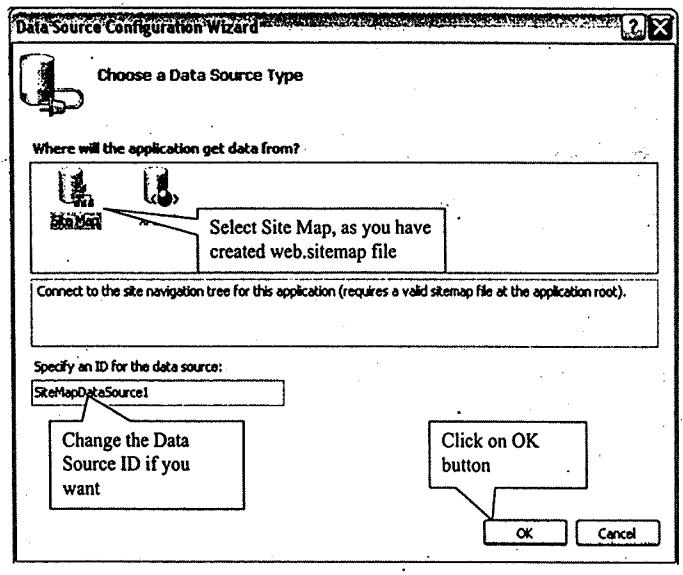
[Fig (5.bf) Navigation Control Example (web.sitemap)]

As you create web.sitemap file, it will automatically get attach with SiteMapPath control. But, you need to explicitly attach the same web.sitemap file with the TreeView Control on the Master Page (fig (5.bg)).



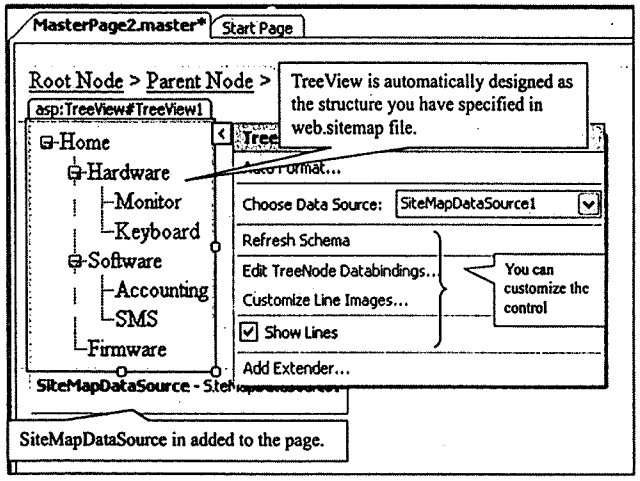
[Fig (5.bg) Navigation Control Example (attach web.sitemap to TreeView)]

The <New Data Source..> Option will open the wizard (fig (5.bh)), from which you can set the site map data source.



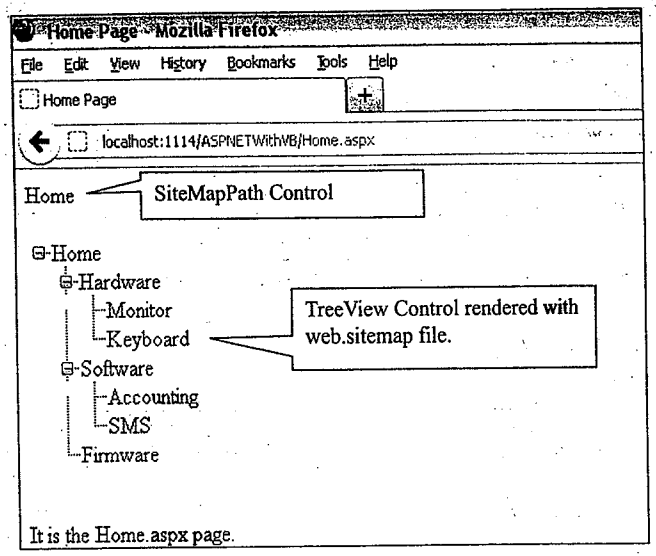
[Fig (5.bh) Navigation Control Example (Attaching Data Source)]

As you have set Site Map data source, it will automatically take web.sitemap file as data source. User will get Master Page as shown in fig (5.bi)).

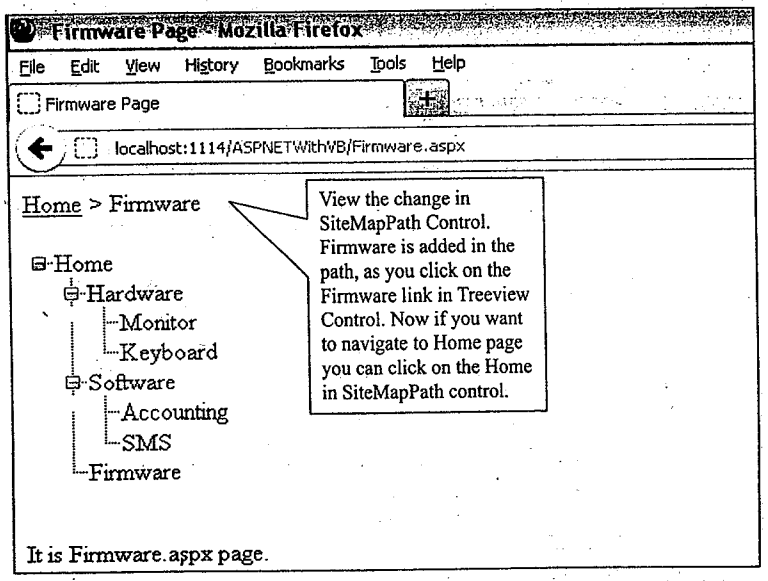


[Fig (5.bi) Navigation Control Example (TreeView after attaching data source)]

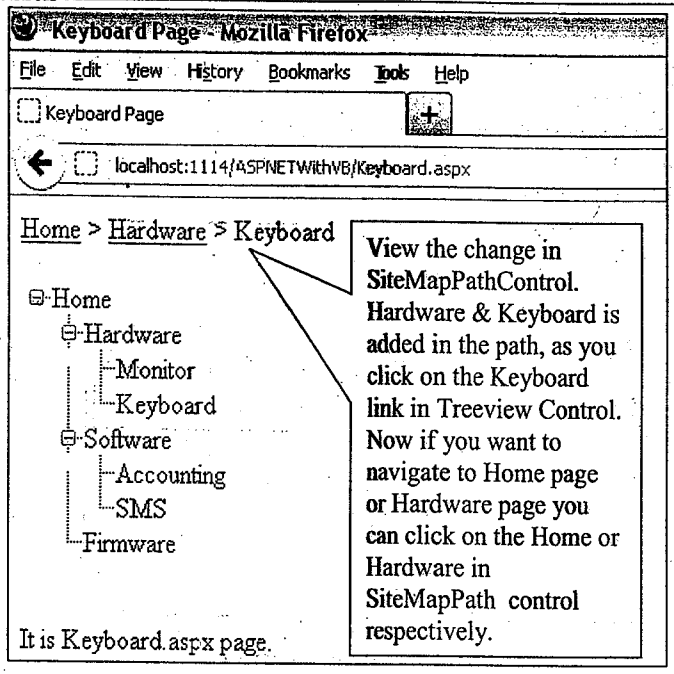
Now set Home.aspx as Startup page, save and execute the web site (fig (5.bj), fig (5.bk), fig (5.bl) and fig (5.bm)).



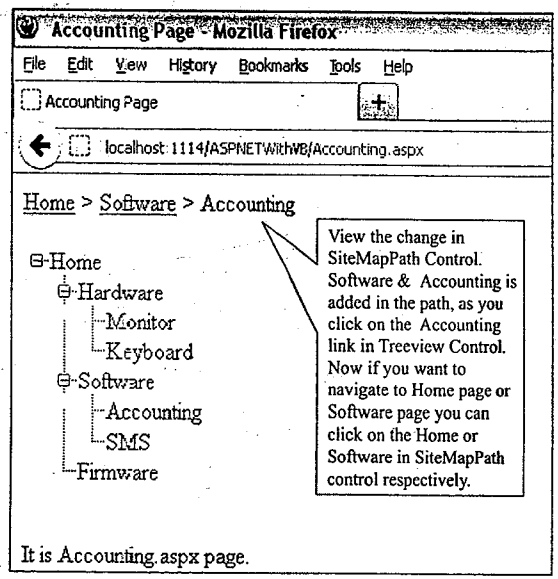
[Fig (5.bj) Navigation Control Example at runtime (Home.aspx)]



[Fig (5.bk) Navigation Control Example at runtime (Firmware.aspx)]



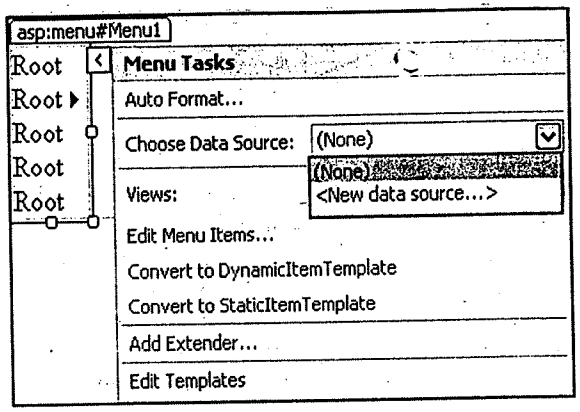
[Fig (5.bl) Navigation Control Example at runtime (Keyboard.aspx)]



[Fig (5.bm) Navigation Control Example at runtime (Accounting.aspx)]

- **Menu Control**

The Menu control is used to display a menu in a Web Forms page (fig.(5.bn)). It can also be used with a SiteMapDataSource control for navigating a Web site. It supports dynamically creation of the menus, customizable appearance, and data binding. When the user clicks a menu item, it can either navigate to a linked Web page or simply post back to the server.
Its ASP.NET tag is <asp:Menu/>.



[Fig (5.bn) Menu Control]

As specified in TreeView Control, you can also set data source in the Menu control by selecting <New Data Source..> Option.

Properties

Properties	Description
DataSource	It gets or sets the object from which the data-bound control retrieves list of data items.
DataSourceID	It gets or sets the ID of the control from which the data-bound control retrieves list of data items.
Items	It gets a MenuItemCollection object that contains all menu items.
ItemWrap	It gets or sets a value indicating whether the text for menu items should wrap or not.
Orientation	It gets or sets the direction in which the Menu control is render.
PathSeparator	It gets or sets the character that is used as a separator for the item.
SelectedItem	It gets the selected menu item.
SelectedValue	It gets the value of the selected menu item.
Style	It gets a collection of text attributes that will be rendered as a style attribute of the Web server control.
Target	It gets or sets the target window or frame in which the Web page content is display.

[Table: 44 Menu Control Properties]

✓ **Default Event:** MenuItemClick()

MenuItem Object

The Menu control is made up of items. It is represented by MenuItem object.

Its ASP.NET tag is <asp:MenuItems/>.

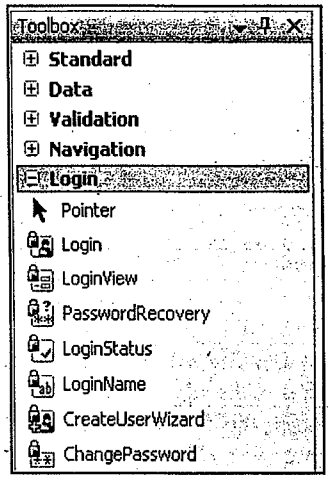
Properties

Properties	Description
Selected	It gets or sets a value that indicates whether the current menu item is selected or not.
ChildItems	It gets a MenuItemCollection collection that contains sub-menu items of the current menu item.
ImageUrl	It gets or sets the URL to an image that is displayed next to the menu item.
NavigateUrl	It gets or sets the URL to navigate to when the item is clicked.
Parent	It gets the parent item of the current item.
Text	It gets or sets the text displayed for the item in the menu control.
Value	It gets or sets value used to store any additional data about the menu item.
ValuePath	It gets the path from the root menu item to the current item.
Target	It gets or sets the target window or frame in which the Web page content is display.
SeparatorImageUrl	It gets or sets the URL to an image that is displayed at the bottom of a menu item to separate it from other menu items.
Selectable	It gets or sets a value that indicates whether the MenuItem object can be selected or not.
Depth	It gets the level at which a menu item is displayed.
DataItem	It gets the data item that is bound to the menu item.
DataBound	It gets a value indicating whether the menu item was created through data binding or not.
DataPath	It gets the path to the data that is bound to the menu item.

[Table: 45 MenuItem Properties]

5.3.5 Login Controls

These controls (fig.(5.bo)) support an ASP.NET model for authenticating users against a database and tracking their status. The model is called forms authentication. They are available in Login Controls tab in the ToolBox. When you use built – in Login Controls, they use ASPNETDB.mdf database which is automatically created with the default tables.

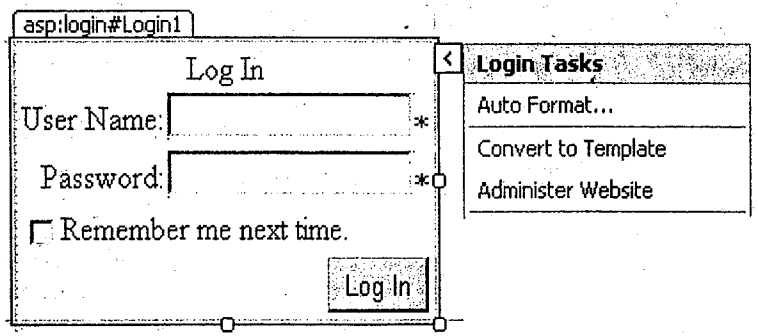


[Fig (5.bo) Login Controls in ToolBox]

- **Login Control**

The Login control is used to display a user interface for user authentication (fig.(5.bp)). It contains text boxes for the user name and password and a check box. A check box allows users to indicate whether user want the server to store their identity using ASP.NET membership and will be automatically authenticated next time when the user visit the same site. If you use the Login control with ASP.NET membership, you do not need to write code to perform authentication. But, if you want to create your own authentication logic, you can handle the Login control's Authenticate event and add custom authentication code.

Its ASP.NET tag is <asp:Login/>.



[Fig (5.bp) Login Control]

Properties

Properties	Description
CreateUserIconUrl	It gets the location of an image to display next to the link to a registration page for new users.
CreateUserText	It gets or sets the text of a link to a registration page for new users.
CreateUserUrl	It gets or sets the URL of the new-user registration page.
DestinationPageUrl	It gets or sets the URL of the page displayed to the user when a login attempt is successful.
FailureAction	It gets or sets the action that occurs when a login attempt fails.
FailureText	It gets or sets the text displayed when a login attempt fails.
FailureTextStyle	It gets a reference to a collection of properties that define the appearance of error text in the Login control.
HelpPageIconUrl	It gets the location of an image to display next to the link to the login Help page.
HelpPageText	It gets or sets the text of a link to the login Help page.
HelpPageUrl	It gets or sets the URL of the login Help page.
InstructionText	It gets or sets login instruction text for the user.
LoginButtonImageUrl	It gets or sets the URL of an image to use for the login button.
LoginButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the login button.
LoginButtonText	It gets or sets the text for the Login button.
LoginButtonType	It gets or sets the type of button to use for login button.
MemberShipProvider	It gets or sets the name of the membership data provider used by the control.
Orientation	It gets or sets a value that specifies the position of the elements of the Login control on the page.
Password	It gets the password entered by the user.
PasswordLabelText	It gets or sets the text of the label for the Password text box.
PasswordRecoveryIconUrl	It gets the location of an image to display next to the link to the password recovery page.
PasswordRecoveryText	It gets or sets the text of a link to the password recovery page.

PasswordRecoveryUrl	It gets or sets the URL of the password recovery page.
PasswordRequiredErrorMessage	It gets or sets the error message to display in a ValidationSummary control when the password field is left blank.
RememberMeSet	It gets or sets a value indicating whether to send a persistent authentication cookie to the user's browser or not.
RememberMeText	It gets or sets the text of the label for the Remember Me check box.
TitleText	It gets or sets the title of the Login control.
UserName	It gets the user name entered by the user.
UserNameLabelText	It gets or sets the text of the label for the UserName text box.
UerNameRequiredErrorMessage	It gets or sets the error message to display in a ValidationSummary control when the user name field is left blank.
VisibleWenLoggedIn	It gets or sets a value indicating whether to show the Login control after the user is authenticated.

[Table: 46 Login Control Properties]

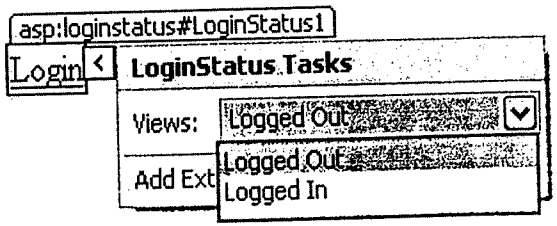
✓ **Default Event:** Authenticate()

When the user logged in to the web site and has been authenticate then LoggedIn() event is fired and when the login error is detected LoginError() event is fired.

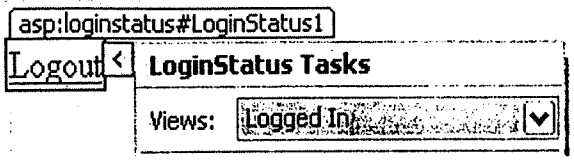
- **LoginStatus Control**

The LoginStatus control (fig.(5.bq) and fig.(5.br)) is used to display a login link for users who are not authenticated and a logout link for users who are authenticated. The login link takes the user to a login page and logout link resets the current user's identity to an anonymous user.

Its ASP.NET tag is <asp:LoginStatus/>.



[Fig (5.bq) LoginStatus Control (Logged Out View)]



[Fig (5.br) LoginStatus Control (Logged In View)]

Properties

Properties	Description
LoginImageUrl	It gets or sets the URL of the image used for the login link.
LoginText	It gets or sets the text used for the login link.
LogoutAction	It gets or sets a value that determines the action taken when a user logs out of a Web site.
LogoutImageUrl	It gets or sets the URL of the image used for the logout button.
LogoutPageUrl	It gets or sets the URL of the logout page.
LogoutText	It gets or sets the text used for the logout link.

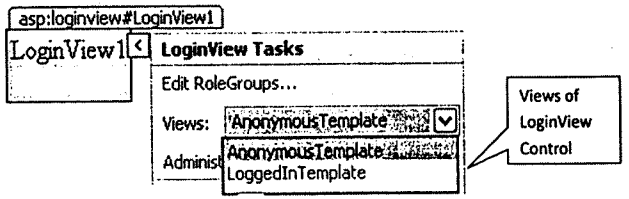
[Table: 47 LoginStatus Control Properties]

✓ **Default Event:** LoggingOut ()

- **LoginView Control**

The LoginView control is used to display different information to anonymous and logged-in users (fig.(5.bs)). The control displays one of two templates: the AnonymousTemplate or the LoggedInTemplate. You can design these templates by adding markup and controls that display information appropriate for anonymous and authenticated users.

Its ASP.NET tag is <asp:LoginView/>.



[Fig (5.bs) LoginView Control]

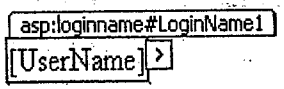
Properties

Properties	Description
AnonymousTemplate	It gets or sets the template that is displayed to users who are not logged in to the Web site.
LoggedInTemplate	It gets or sets the template that is displayed to Web site users who are logged in to the Web site.

[Table: 48 LoginView Control Properties]

✓ **Default Event:** ViewChanged()

- **LoginName**



[Fig (5.bt) LoginName Control]

The LoginName control is used to display a user's login name if the user has logged in using ASP.NET membership (fig.(5.bt)). Alternatively, if your site uses integrated Windows authentication, the control displays the user's Windows account name. It means that, it displays the name contained in the User property of the page.

Its ASP.NET tag is <asp:LoginName/>.

Properties

Properties	Description
FormatString	It provides a string containing the format item for displaying user name.

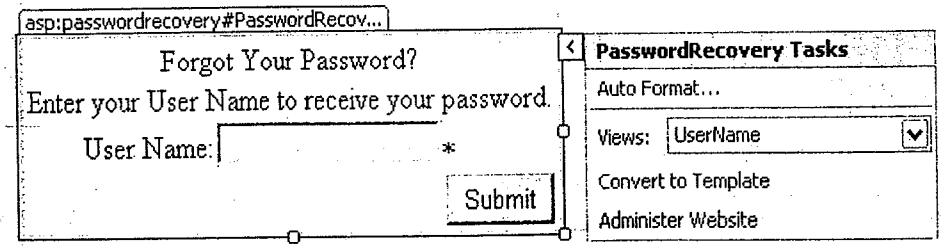
[Table: 49 LoginName Control Properties]

- **PasswordRecovery Control**

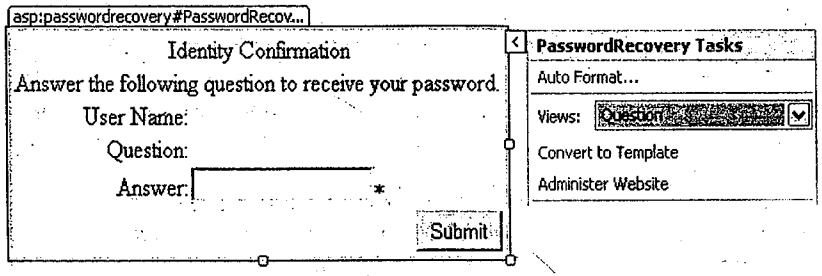
The PasswordRecovery control (fig.(5.bu), fig.(5.bv) and fig.(5.bw)) is used to retrieve the user password based on the e-mail address that was used when the account was created. It sends an e-mail message containing a password to the user. You can configure ASP.NET membership to store passwords using non-reversible encryption. In that case, the PasswordRecovery control generates a new password instead of sending the original password to the user.

You can also configure membership to include a security question that the user must answer to recover a password. If you do, the PasswordRecovery control asks the question and checks the answer before recovering the password. The PasswordRecovery control requires that your application can forward e-mail message to a Simple Mail Transfer Protocol (SMTP) server. In short, it assists the users who have forgotten their passwords.

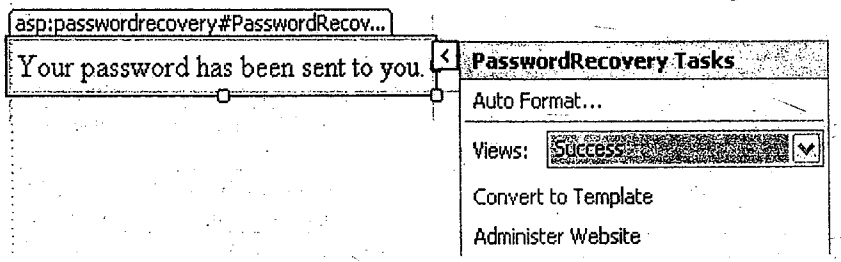
Its ASP.NET tag is <asp:LoginName/>.



[Fig (5.bu) PasswordRecovery Control (UserName View)]



[Fig (5.bv) PasswordRecovery Control (Question View)]



[Fig (5.bw) PasswordRecovery Control (Success View)]

Properties

Properties	Description
Answer	It gets the answer to the password recovery confirmation question entered by the user.
AnswerLabelText	It gets or sets the label text for the password confirmation answer text box.
AnswerRequiredErrorMessage	It gets or sets the error message displayed to the user when the Answer text box is blank.
GeneralFailureText	It gets or sets the error message to display when there is a problem with the membership provider.
HelpPageIconUrl	It gets the location of an image to display next to the link to the login Help page.
HelpPageText	It gets or sets the text of a link to the login Help page.
HelpPageUrl	It gets or sets the URL of the login Help page.
MailDefinition	It gets a reference to a collection of properties (From, Subject, To) that define the characteristics of e-mail messages used to send new or recovered passwords to users.
MembershipProvider	It gets or sets the membership provider used to look up user information.

Question	It gets the password recovery confirmation question established by the user on the Web site.
QuestionFailureText	It gets or sets the text to display when the user's answer to the password recovery confirmation question does not match the answer stored in the Web site data store.
QuestionInstructionText	It gets or sets the text to display in the Question view to instruct the user to answer the password recovery confirmation question.
QuestionLabelText	It gets or sets the text of the label for the Question text box.
QuestionTemplate	It gets or sets the template used to display the Question view.
QuestionTitleText	It gets or sets the title for the Question view.
SubmitButtonImageUrl	It gets or sets the URL of an image to use for the submit button.
SubmitButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the submit button.
SubmitButtonText	It gets or sets the text for the submit button.
SubmitButtonType	It gets or sets the type of button to use for submit button.
SuccessPageUrl	It gets or sets the URL of the page to display after sending a password successfully.
SuccessTemplate	It gets or sets the template used to display the Success view.
SuccessText	It gets or sets the text to display after sending a password successfully.
UserName	It gets or sets the text that appears in the User Name text box.
UserNameLabelText	It gets or sets a text of the label for User Name text box.
UserNameRequiredErrorMessage	It gets or sets the error message displayed to the user when the User Name text box is blank.
UserNameFailureText	It gets or sets the text displayed when the user name entered by the user is not a valid user name for the Web site.
UserNameInstructionText	It gets or sets the text to display in the UserName view to instruct the user to enter a user name.
UserNameTemplate	It gets or sets the template used to display the UserName view.
UserNameTitleText	It gets or sets the title for the UserName view.

[Table: 50 PasswordRecovery Control Properties]

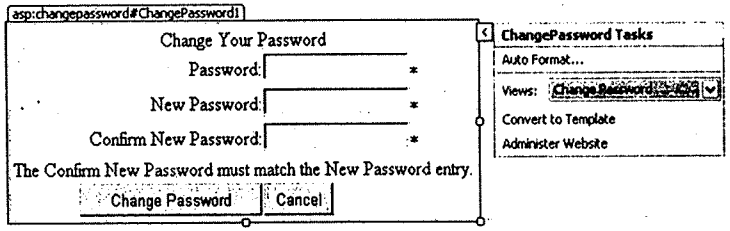
✓ **Default Event:** SendingMail()

- **ChangePassword Control**

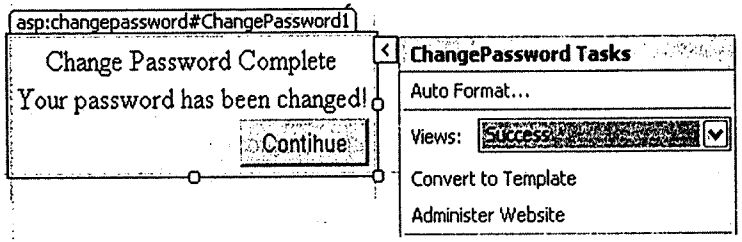
The ChangePassword control (fig.(5.bx) and fig.(5.by)) is used to allow user to change their password. It simply queries the user name as well as the old password from the user. Then it requires the user to enter the new password and confirm the new password. If the user is already logged on, the control automatically hides the text field for the user name and uses the name of the authenticated user. The control also includes support for sending an e-mail message about the new password.

It is made up of two template views that are displayed to the user. The first is the ChangePasswordTemplate, which displays the user interface used to gather the data required to change the user password. The second template is the SuccessTemplate, which defines the user interface that is displayed after a user password has been successfully changed.

Its ASP.NET tag is <asp:ChangePassword/>.



[Fig (5.bx) ChangePassword Control (Change Password View)]



[Fig (5.by) ChangePassword Control (Success View)]

Properties

Properties	Description
CancelButtonImageUrl	It gets or sets the URL of an image to use for the cancel button.
CancelButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the cancel button.
CancelButtonText	It gets or sets the text for the cancel button.

CancelButtonType	It gets or sets the type of button to use for cancel button.
CancelDestinationPageUrl	It gets or sets the URL of the page that the user is shown after clicking the Cancel button.
ChangePasswordButtonImageUrl	It gets or sets the URL of an image displayed next to the Change Password.
ChangePasswordButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the change password button.
ChangePasswordButtonText	It gets or sets the text for the change password button.
ChangePasswordButtonType	It gets or sets the type of button to use for change password button.
ChangePasswordFailureText	It gets or sets the text displayed when the user's password is not changed.
ChangePasswordTemplate	It gets or sets the template used to display the change password view.
ChangePasswordTitleText	It gets or sets the title for the change password view.
ConfirmNewPassword	It gets the duplicate password entered by the user.
ConfirmNewPasswordLabelText	It gets or sets the label text for the ConfirmNewPassword text box.
ConfirmPasswordCompareErrorMessage	It gets or sets the message that is displayed when the new password and the duplicate password entered by the user are not identical.
ConfirmPasswordRequiredErrorMessage	It gets or sets the error message that is displayed when the Confirm New Password text box is left empty.
ContinueButtonImageUrl	It gets or sets the URL of an image to use for the continue button.
ContinueButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the continue button.
ContinueButtonText	It gets or sets the text for the continue button.
ContinueButtonType	It gets or sets the type of button to use for continue button.

ContinueDestinationPageUrl	It gets or sets the URL of the page that the user is shown after clicking the continue button.
CreateUserIconUrl	It gets or sets the URL of an image to display next to the link to the Web page that contains a CreateUserWizard control for the Web site.
CreateUserText	It gets or sets the text of the link of the Web page that contains a CreateUserWizard control for the Web site.
CreateUserUrl	It gets or sets the URL of the Web page that contains a CreateUserWizard control for the Web site.
CurrentPassword	It gets the current password for the user.
DisplayUserName	It gets or sets a value indicating whether the ChangePassword control should display the UserName control and label or not.
EditProfileIconUrl	It gets or sets the URL of an image to display next to the link to the user profile editing for the Web site.
EditProfileText	It gets or sets the text of the link to user profile editing page for the Web site.
EditProfileUrl	It gets or sets the URL of the user profile editing page for the Web site.
HelpPageIconUrl	It gets the location of an image to display next to the link to the change password Help page.
HelpPageText	It gets or sets the text of a link to the change password Help page.
HelpPageUrl	It gets or sets the URL of the change password Help page.
MailDefinition	It gets a reference to a collection of properties (From, Subject, To) that define the characteristics of e-mail messages that is send to the user after their password is changed.
InstructionText	It gets or sets informational text that appears between the input boxes and the ChangePasswordTitleText.
MembershipProvider	It gets or sets the membership provider used to look up user information.

NewPassword	It gets the new password entered by the user.
NewPasswordLabelText	It gets or sets the text of the label for the New password text box.
NewPasswordRequiredErrorMessage	It gets or sets the error message to display in a ValidationSummary control when the new password field is left blank.
NewPasswordRegularExpression	It gets or sets the regular expression that is used to validate the password provided by the user.
NewPasswordRequiredErrorMessage	It gets or sets the error message that is shown when the password entered does not pass the regular expression criteria defined in the NewPasswordRegularExpression property.
PasswordHintText	It gets or sets informational text about the requirements for creating a password for the Web site.
PasswordLabelText	It gets or sets the text of the label for the current password text box.
PasswordRecoveryIconUrl	It gets the location of an image to display next to the link to the password recovery page.
PasswordRecoveryText	It gets or sets the text of a link to the password recovery page.
PasswordRecoveryUrl	It gets or sets the URL of the password recovery page.
PasswordRequiredErrorMessage	It gets or sets the error message that is displayed when the user leaves the Current Password text box empty.
SuccessPageUrl	It gets or sets the URL of the page to display after a password is changed successfully.
SuccessTemplate	It gets or sets the template used to display the Success view.
SuccessText	It gets or sets the text that is displayed on the Success view between the SuccessTitleText and the Continue button.

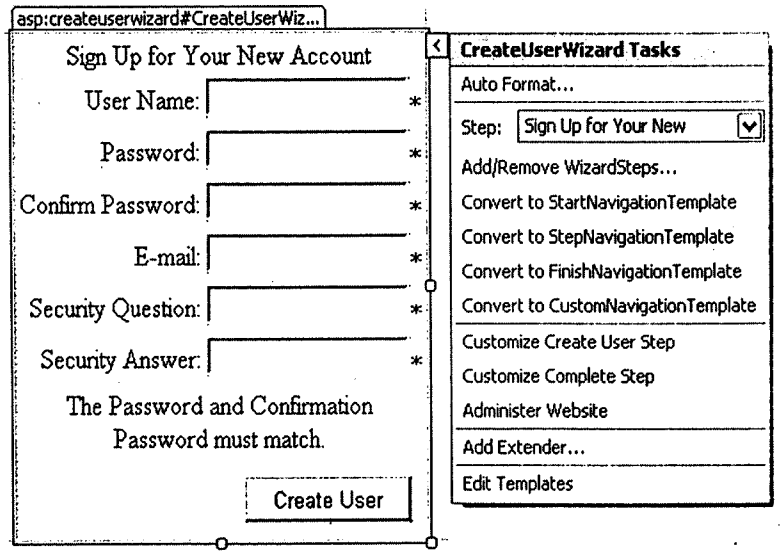
SuccessTitleText	It gets or sets the title of the Success view.
UserName	It gets or sets the web site user name for which the password is to be changed.
UserNameLabelText	It gets or sets a text of the label for User Name text box.
UserNameRequiredErrorMessage	It gets or sets the error message displayed to the user when the User Name text box is blank.

[Table: 51 ChangePassword Control Properties]

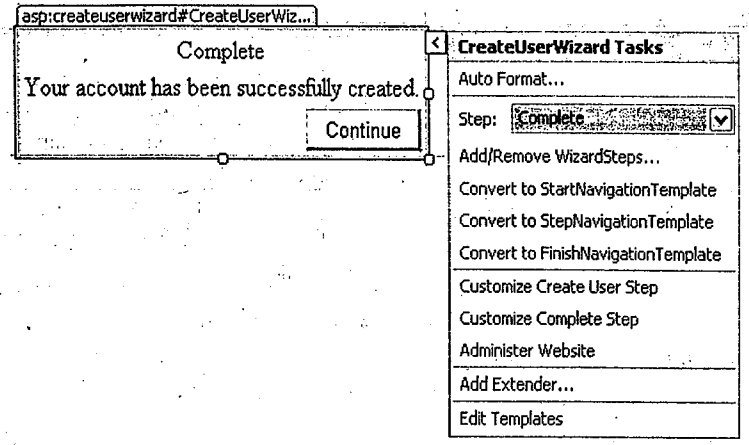
✓ **Default Event:** ChangedPassword()

- **CreateUserWizard Control**

The CreateUserWizard control (fig.(5.bz) and fig.(5.ca)) is used to collects information from the users. By default, the CreateUserWizard control adds the new user to the ASP.NET membership system. The CreateUserWizard control gathers user information such as user name, password, confirm password, E-mail address, Security question, and security answer. This information is used to authenticate users and recover user password.
Its ASP.NET tag is <asp:CreateUserWizard/>.



[Fig (5.bz) CreateUserWizard Control (Sign Up for Your New Step)]



[Fig (5.ca) CreateUserWizard Control (Complete Step)]

Properties

Properties	Description
Answer	It gets or sets the user answer to the password recovery confirmation question.
AnswerLabelText	It gets or sets the label text for the password confirmation answer text box.
AnswerRequiredErrorMessage	It gets or sets the error message displayed to the user when the Answer text box is blank.
ActiveStep	It gets the step in the WizardSteps collection that is currently displayed to the user.
ActiveStepIndex	It gets or sets the step that is currently displayed to the user.
AutoGeneratePassword	It gets or sets a value indicating whether or not to automatically generate a password for the new user account.
CancelButtonImageUrl	It gets or sets the URL of an image to use for the cancel button.
CancelButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the cancel button.
CancelButtonText	It gets or sets the text for the cancel button.

CancelButtonType	It gets or sets the type of button to use for cancel button.
CancelDestinationPageUrl	It gets or sets the URL of the page that the user is shown after clicking the Cancel button.
CompleteStep	It gets a reference to the final user account creation step.
CompleteSuccessText	It gets or sets the text displayed when a Web site user account is created successfully.
CompleteSuccessTextStyle	It gets a reference to a collection of properties that define the appearance of the text displayed when a Web site user account is created successfully.
ConfirmPassword	It gets the second password entered by the user.
ConfirmPasswordLabelText	It gets or sets the label text for second Password text box.
ConfirmPasswordCompareErrorMessage	It gets or sets the message that is displayed when the user input two different passwords in password and the confirm password fields.
ConfirmPasswordRequiredErrorMessage	Gets or sets the error message displayed when the user leaves the confirm password text box empty.
ContinueButtonImageUrl	It gets or sets the URL of an image to use for the continue button.
ContinueButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the continue button.
ContinueButtonText	It gets or sets the text for the continue button.
ContinueButtonType	It gets or sets the type of button to use for continue button.
ContinueDestinationPageUrl	It gets or sets the URL of the page that the user is navigated after clicking the continue button.
CreateUserButtonImageUrl	It gets or sets the URL of an image displayed for the Create User button.

CreateUserButtonText	It gets or sets the text caption displayed on the Create User button.
CreateUserButtonType	It gets or sets the URL of the Web page that contains a CreateUserWizard control for the Web site.
CreateUserStep	It gets a reference to the template for the user account creation step.
DisableCreateUser	It gets or sets a value indicating whether the new user should be allowed to log on to the Web site.
DisplayCancelButton	It gets or sets a Boolean value indicating whether to display a Cancel button.
DuplicateEmailErrorMessage	It gets or sets the error message displayed when the user enters an e-mail address that is already in use in the membership provider.
DuplicateUserNameErrorMessage	It gets or sets the error message displayed when the user enters a user name that is already in use within the membership provider.
EditProfileIconUrl	It gets or sets the URL of an image to display next to the link to the user profile editing for the Web site.
EditProfileText	It gets or sets the text of the link to user profile editing page for the Web site.
EditProfileUrl	It gets or sets the URL of the user profile editing page for the Web site.
Email	It gets the Email entered by the user.
EmailLabelText	It gets or sets the text of the label for the Email text box.
EmailRequiredErrorMessage	It gets or sets the error message to display in a ValidationSummary control when the Email field is left blank.
EmailRegularExpression	It gets or sets the regular expression that is used to validate the Email provided by the user.
EmailRequiredErrorMessage	It gets or sets the error message that is shown when the Email entered does not pass the regular expression criteria defined in the EmailRegularExpression property.

FinishCompleteButtonImageUrl	It gets or sets the URL of an image to use for the finish button.
FinishCompleteButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the finish button.
FinishCompleteButtonText	It gets or sets the text for the finish button.
FinishCompleteButtonType	It gets or sets the type of button to use for finish button.
FinishDestinationPageUrl	It gets or sets the URL of the page that the user is shown after clicking the finish button.
FinishNavigationTemplate	It gets or sets the template that is used to display the navigation area on the Finish step.
FinishPreviousButtonImageUrl	It gets or sets the URL of an image to use for the previous button of finish step.
FinishPreviousButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the previous button of finish step.
FinishPreviousButtonText	It gets or sets the text for the previous button of finish step.
FinishPreviousButtonType	It gets or sets the type of button to use for previous button of finish step.
HeaderTemplate	It gets or sets the template that is used to display the header area on the control.
HeaderText	It gets or sets the text caption that is displayed for the header area on the control.
HelpPageIconUrl	It gets the location of an image to display next to the link to the change password Help page.
HelpPageText	It gets or sets the text of a link to the change password Help page.
HelpPageUrl	It gets or sets the URL of the change password Help page.
MailDefinition	It gets a reference to a collection of properties (From, Subject, To, CC) that define the characteristics of e-mail messages that is send to the user after their password is changed.

InstructionText	It gets or sets informational text that appears between the input boxes and the ChangePasswordTitleText.
MembershipProvider	It gets or sets the membership provider used to look up user information.
InvalidAnswerErrorMessge	It gets or sets the message displayed when the password retrieval answer is not valid.
InvalidEmailErrorMessage	It gets or sets the message displayed when entered e-mail address is not valid.
InvalidPasswordErrorMessage	It gets or sets the message displayed when the password entered is not valid.
InvalidQuestionErrorMessage	It gets or sets the message displayed when the password retrieval question entered is not valid.
LoginCreatedUser	It gets or sets a value indicating whether to logged-in as the new user after creating the user account.
Password	It gets the password entered by the user.
PasswordHintText	It gets or sets informational text about the requirements for creating a password for the Web site.
PasswordLabelText	It gets or sets the text of the label for the current password text box.
PasswordRequiredExpression	It gets or sets a regular expression used to validate the provided password.
PasswordRegularExpressionErrorMessage	It gets or sets the error message shown when the password entered does not confirm to the site's password requirements.
PasswordRequiredErrorMessage	It gets or sets the error message that is displayed when the user leaves the Current Password text box empty.
Question	It gets or sets the password recovery confirmation question entered by the user.
QuestionLabelText	It gets or sets the text of the label for the question text box.
QuestionRequiredErrorMessage	It gets or sets the error message that is displayed when the user does not enter a password confirmation question.

RequiredEmail	It gets or sets a value indicating whether an e-mail address is required for the Web site user.
SideBarTemplate	It gets or sets the template that is used to display the sidebar area on the control.
StartNavigationTemplate	It gets or sets the template that is used to display the navigation area on the Start step.
StartNextButtonImageUrl	It gets or sets the URL of an image to use for the Start Next button.
StartNextButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the Start Next button.
StartNextButtonText	It gets or sets the text for the start next button.
StartNextButtonType	It gets or sets the type of button to use for start next button.
StepNavigationTemplate	It gets or sets the template that is used to display the navigation area on any Wizard Step Base object.
StepNextButtonImageUrl	It gets or sets the URL of an image to use for the Step Next button.
StepNextButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the Step Next button.
StepNextButtonText	It gets or sets the text for the step next button.
StepNextButtonType	It gets or sets the type of button to use for step next button.
StepPreviousButtonImageUrl	It gets or sets the URL of an image to use for the Step Previous button.
StepPreviousButtonStyle	It gets a reference to the Style object that allows you to set the appearance of the Step Previous button.
StepPreviousButtonText	It gets or sets the text for the step Previous button.
StepPreviousButtonType	It gets or sets the type of button to use for step previous button.
UnknownErrorMessage	It gets or sets the error message displayed when an error returned by the membership provider is not defined.

UserName	It gets or sets the web site user name for which the password is to be changed.
UserNameLabelText	It gets or sets a text of the label for User Name text box.
UserNameRequiredErrorMessage	It gets or sets the error message displayed to the user when the User Name text box is blank.

[Table: 52 CreateUserWizard Control Properties]

✓ **Default Event:** CreatedUser()

By default, all Login Controls works with Windows Authentication. But, if you want that Login controls works with Forms Authentication, then you have to make changes in the <authentication/> tag of web.config file (fig.(5.cb) and fig.(5.cc)).

```
<!--
        The <authentication> section enables configuration
        of the security authentication mode used by
        ASP.NET to identify an incoming user.
    -->
<authentication mode="Windows"/>
```

By default mode is Windows, you have to change it to Forms

[Fig (5.cb) authentication tag in web.config file (by default)]

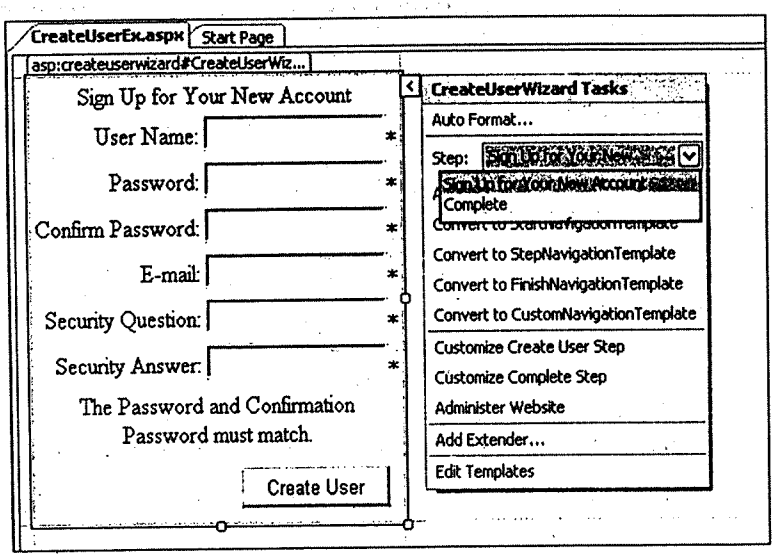
```
<!--
        The <authentication> section enables configuration
        of the security authentication mode used by
        ASP.NET to identify an incoming user.
    -->
<authentication mode="Forms"/>
```

Mode changed to Forms

[Fig (5.cc) authentication tag in web.config file (changed to Forms)]

Example

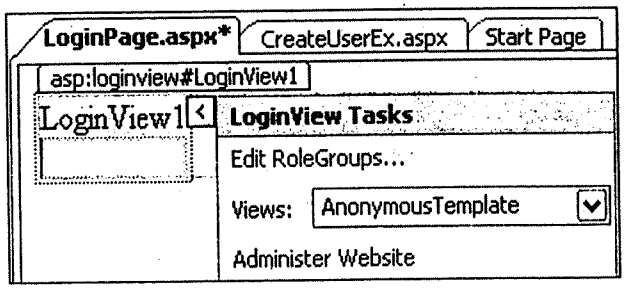
Create CreateUserEx.aspx page (fig (5.cd)), LoginPage.aspx (fig (5.ce), (5.cf)and(5.cg)), PasswordRecoveryPage.aspx (fig (5.ch)), ChangePasswordPage.aspx (fig (5.ci)).



[Fig (5.cd) CreateUserEx.aspx page]

Set ContinueDestinationPageUrl property to "~/LoginPage.aspx" of CreateUserWizard Control on CreateUserEx.aspx page.

Place LoginView control on LoginPage.aspx (fig (5.ce)).



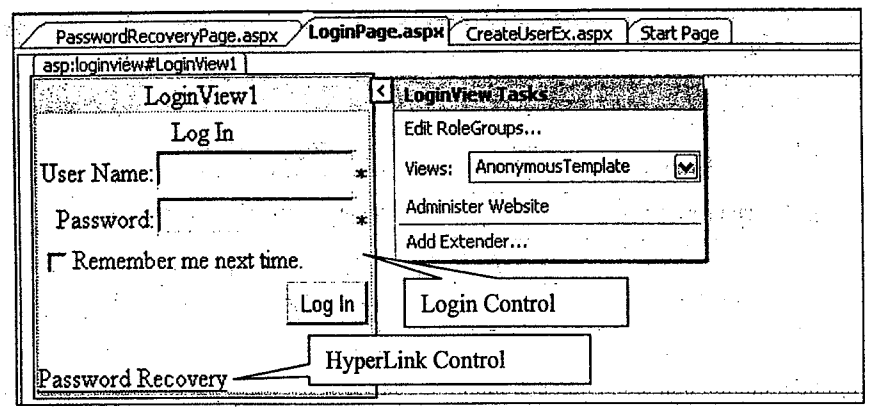
[Fig (5.ce) LoginPage.aspx page (LoginView Control)]

Place Login Control and HyperLink Control in AnonymousTemplate View of LoginView Control. (fig (5.cf)).

HyperLink Control:

Text property – Password Recovery

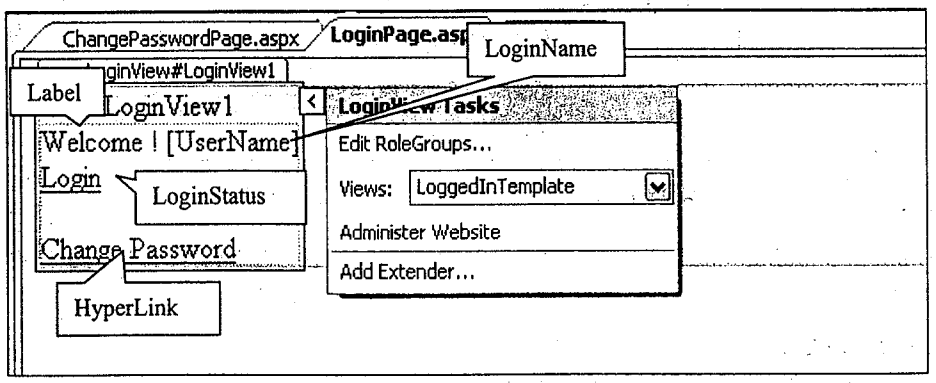
NavigateUrl property - ~/PasswordRecoveryPage.aspx



[Fig (5.cf) LoginPage.aspx page (Login Control in AnonymousTemplate View)]

Now change the View in LoginView control from AnonymousTemplate to LoggedInTemplate. (fig (5.cg)).

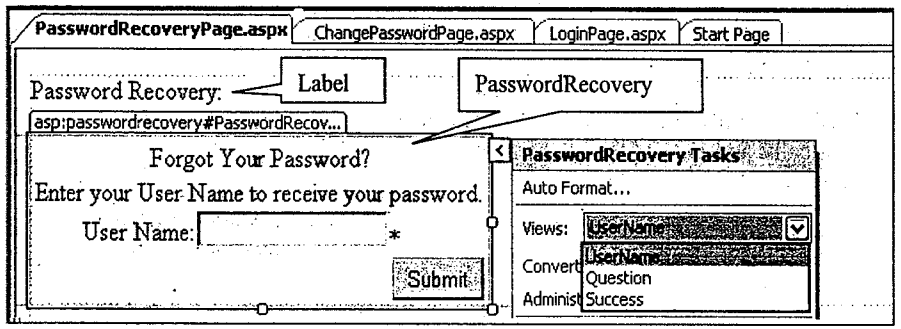
Place following controls in LoggedInTemplate View:
LoginName Control
LoginStatus Control
Label Control:
Text property – Welcome !
HyperLink Control:
Text property – Change Password
NavigateUrl - ~/ChangePasswordPage.aspx



[Fig (5.cg) LoginPage.aspx page (LoggedInTemplate View)]

Now on PasswordRecoveryPage.aspx, place following Controls:
PasswordRecovery Control:
If you want your password in the mail then you need to set MailDefinition property of this control and Setup the SMTP server.

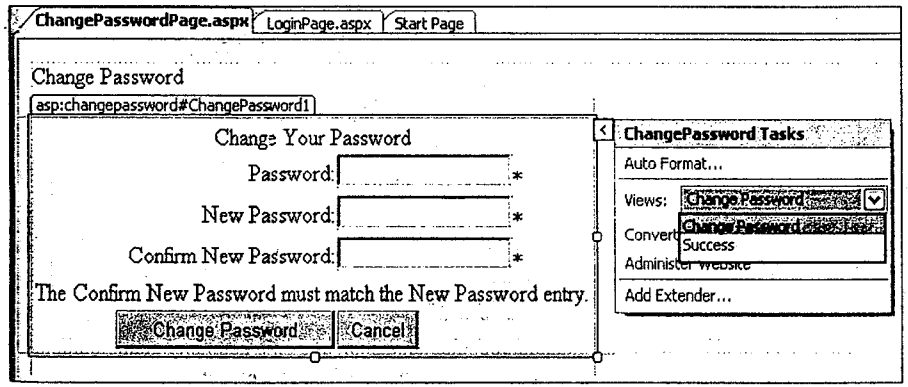
Label Control:
Text property – Password Recovery



[Fig (5.ch) PasswordRecoveryPage.aspx page (UserName View)]

Now on ChangePasswordPage.aspx, place following Controls:
ChangePassword Control:
ContinueDestinationPageUrl property - "~/LoginPage.aspx".

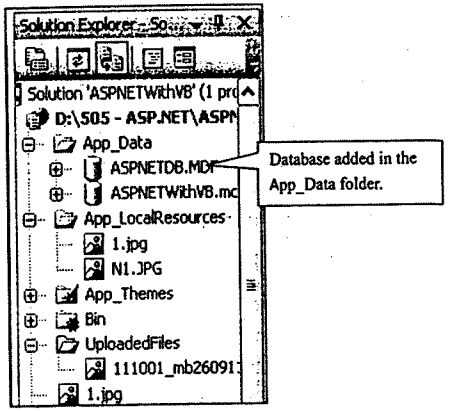
Label Control:
Text property – Change Password



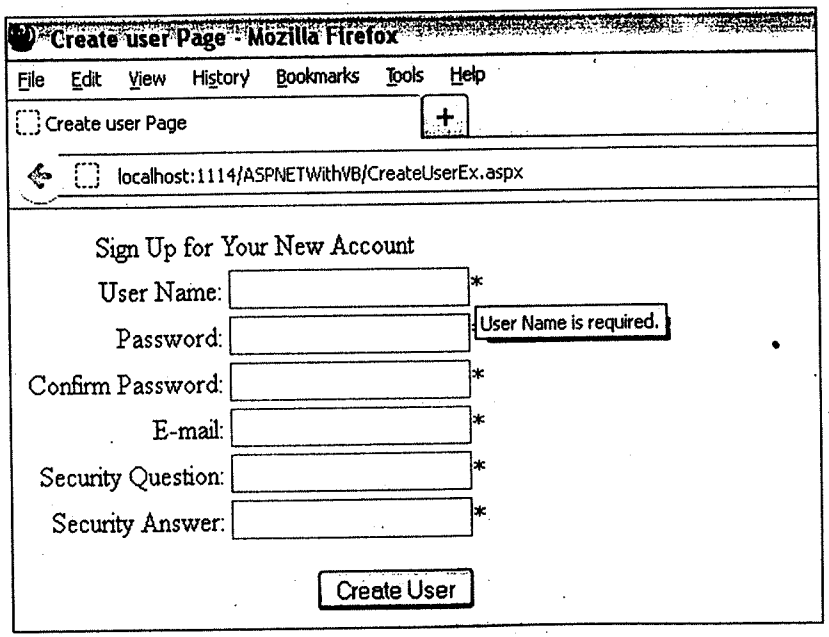
[Fig (5.ci) ChangePasswordPage.aspx page (ChangePassword View)]

By default, each and every Login Control has the Validation implemented in them. So if the user do any mistake in data input then Validation Mark (*) is shown. And if you take your cursor on the Validation Mark then it will also show you Validation message in Tool Tip (fig (5.ck)).

In web.config file, change <authentication/> tag (fig (5.cc)). Set CreateUserEx.aspx as Startup page. Save and execute.
When you execute the page for the first time, ASPNETDB.mdf data base is created automatically and it is added in the App_Data folder of the website (fig (5.cj)).

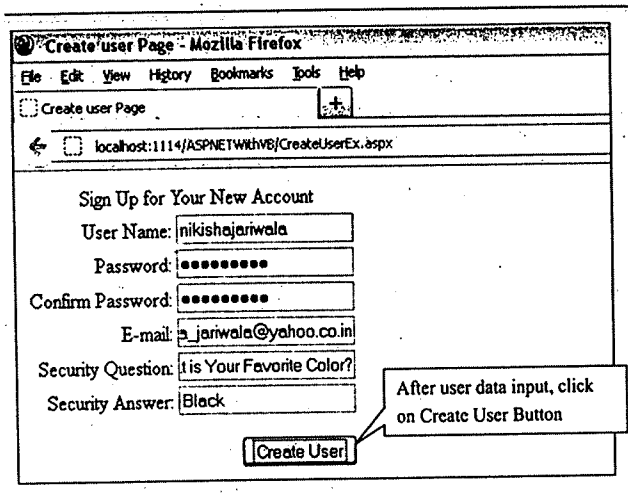


[Fig (5.cj) ASPNETDB database in Solution Explorer]

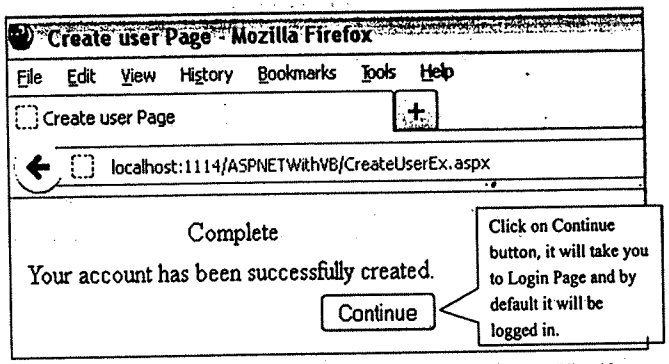


[Fig (5.ck) Validation in the CreateUserWizard Control]

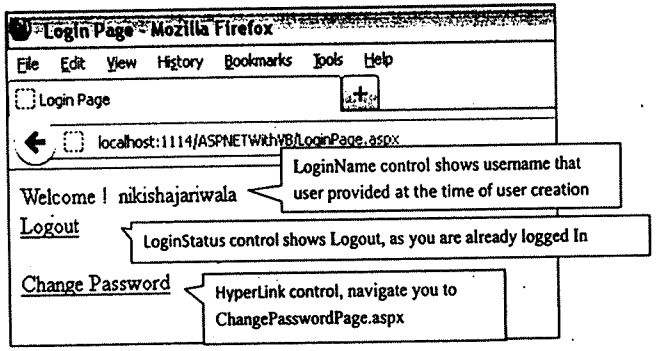
Enter valid user data on the CreateUserEx.aspx page (fig (5.cl)).



[Fig (5.cl) CreateUserEx.aspx page at runtime]

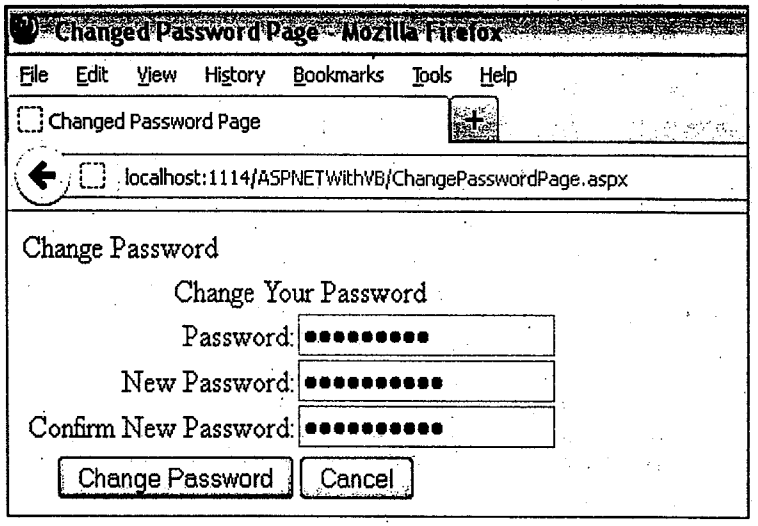


[Fig (5.cm) CreateUserEx.aspx page at runtime (Success View)]

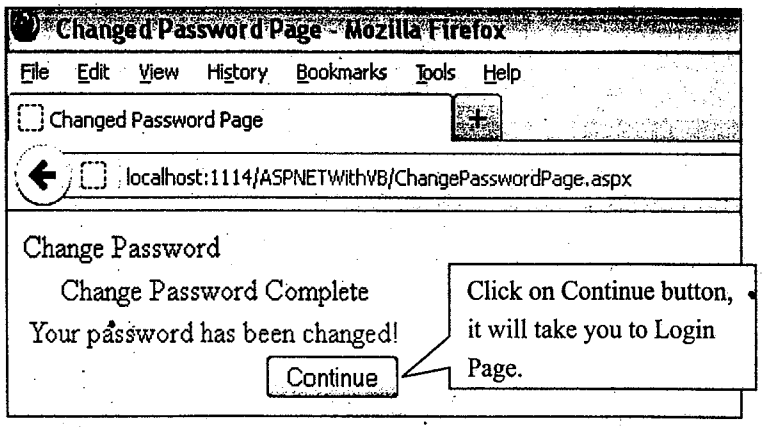


[Fig (5.cn) LoginPage.aspx page at runtime (LoggedIn View)]

Click on Change Password Link, it will take you to ChangePasswordPage.aspx (fig (5.co)). Provide old and new password on the page and click on Change Password button.

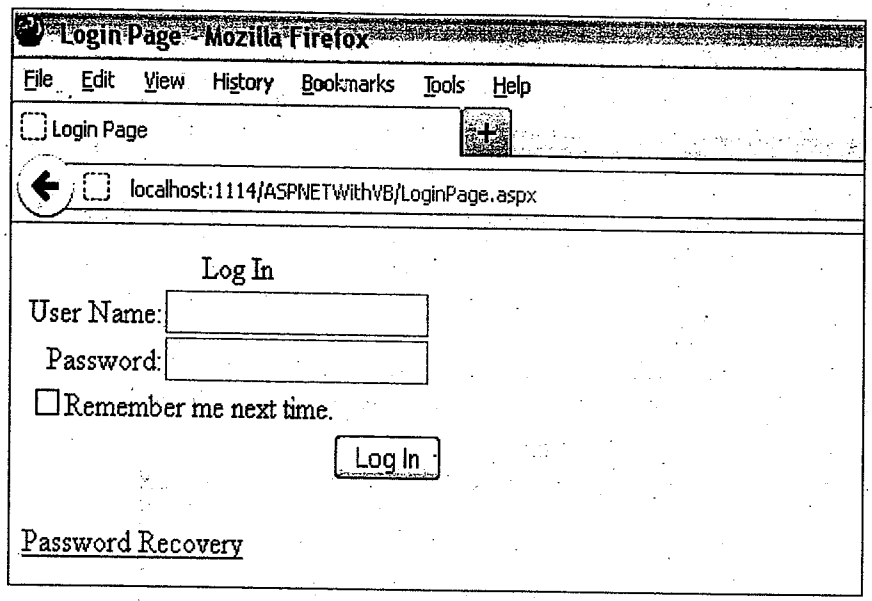


[Fig (5.co) ChangePasswordPage.aspx page at runtime]



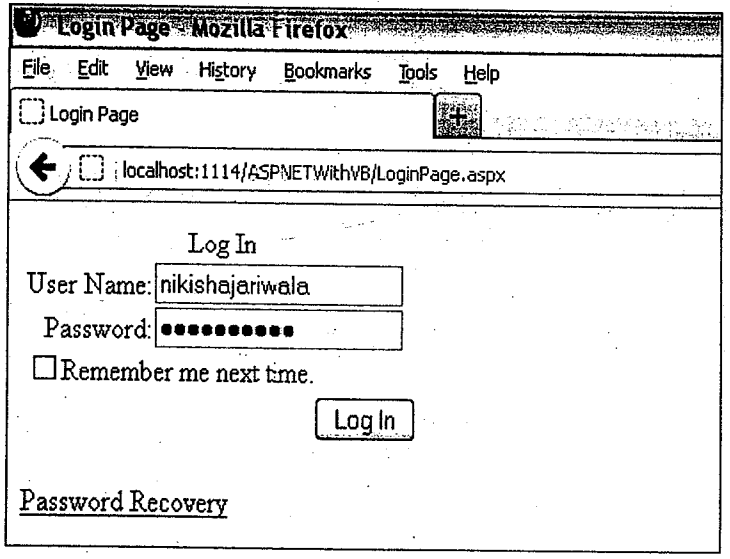
[Fig (5.cp) ChangePasswordPage.aspx page at runtime (Success View)]

Click Logout on LoginPage.aspx. So it will take you to AnonymousTemplate View of the LoginPage.apsx (fig (5.cq)).

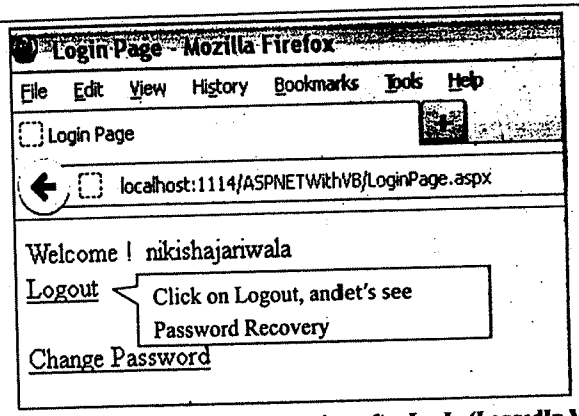


[Fig (5.cq) LoginPage.aspx page at runtime (AnonymousTemplate View)]

Provide the User Name and new changed Password on LoginPage.aspx and Click on Login Button (fig (5.cr)).

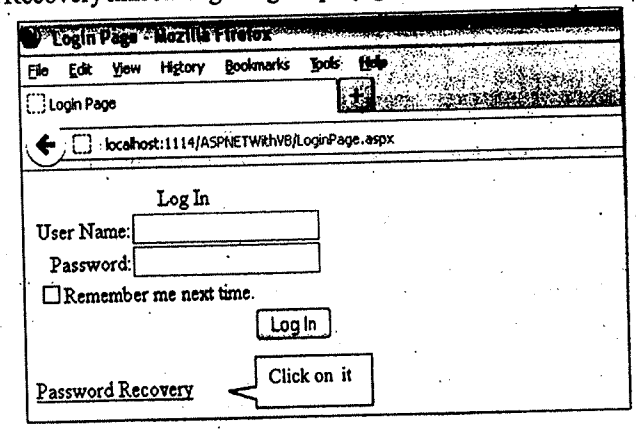


[Fig (5.cr) LoginPage.aspx page at runtime (Logged In with new password)]

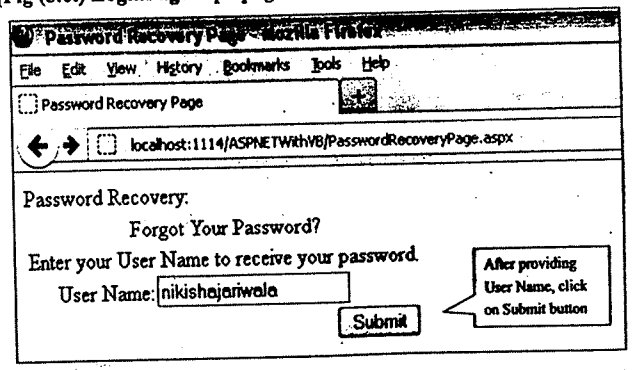


[Fig (5.cs) LoginPage.aspx page at runtime after LogIn (LoggedIn View)]

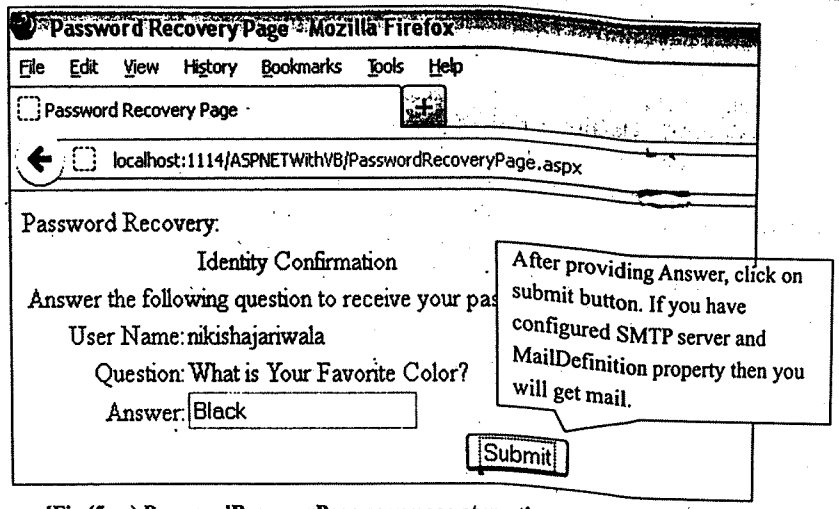
Click Password Recovery link on LoginPage.aspx (fig (5.ct)).



[Fig (5.ct) LoginPage.aspx page at runtime (Password Recovery link)]



[Fig (5.cu) PasswordRecoveryPage.aspx page at runtime (UserName View)]



[Fig (5.cv) PasswordRecoveryPage.aspx page at runtime (Question View)]

Exercise-5

1. Which is the base class of all the Control classes?
2. List out properties, methods and events of Control class.
3. List out various types of Server Controls.
4. Draw the class hierarchy for the Html server controls.
5. Which is the base class of all the Html Server Control classes?
6. List out various Html server controls with its properties and default events.
7. What is the difference between InnerHtml and InnerText property?
8. Which is the base class of all the Web Server Control classes?
9. List out various Web server controls with its properties and default events.
10. Explain common properties of Web Server Control.
11. Explain different types of Buttons.
12. Explain various List Controls.
13. Differentiate between various List Controls.
14. Which is the default event of CheckBox and RadioButton Control?
15. Differentiate between Label and Literal control.
16. Explain ImageMap control in detail.
17. Explain HyperLink Control.
18. Explain FileUpload Control with example.
19. Which method is used to save uploaded file with FileUpload Control?
20. What is the significance of HasFile property?
21. Which is the HTML tag similar to Panel control of ASP.NET?
22. What is the significance of PlaceHolder control?
23. Explain MultiView Control in detail.

24. Explain Table control with TableCell and TableRow classes.
25. List out Rich Controls? Explain them in detail.
26. Explain Calender control in detail.
27. Explain XML file along with its tags used for Advertisement in AdRotator control.
28. List out various Validation controls.
29. Which is the base class of all the Validation Control Classes? List out its properties.
30. Explain RequiredFieldValidator & RangeValidator Controls.
31. Which Validation control can be used to check that two values are similar or not?
32. Explain RegularExpressionValidator & ValidationSummary controls in detail.
33. What is the purpose of Menu control?
34. Explain CustomValidator control with Example.
35. What is the significance of Navigation controls?
36. What is the use of SiteMapPath control?
37. Explain web.sitemap file with example.
38. Explain TreeView control in detail.
39. List out properties of TreeNode object.
40. List out various Login Controls available in ASP.NET.
41. Which is the default event of Login control?
42. Explain Login Control in detail.
43. What is the significance of LoginStatus control?
44. Which are the Views available with LoginView Control?
45. What is the significance of LoginName control?
46. Explain MailDefinition property?
47. Explain PasswordRecovery control in detail.
48. What is the significance of <authentication/> tag of web.config file?
49. Which database is created automatically if we use built-in Login controls?
50. Explain ChangePassword & CreateUserWizard controls.
51. Design an application that demonstrates all Login Controls.

6 Structuring and Formatting ASP.NET Page

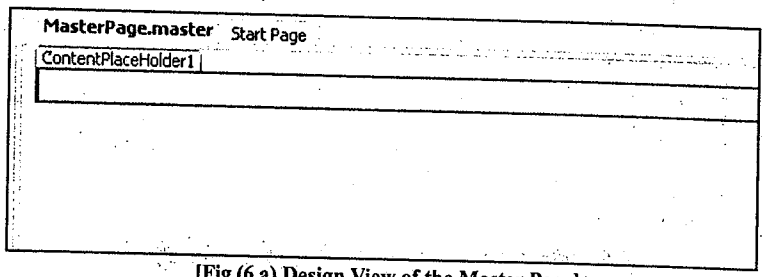
6.1. Master Page

ASP.NET Master Page allows you to create a consistent layout for the pages in your application. A single master page defines the look and standard behavior such as header, footer, links and so on of all the pages or a group of pages. The design view and the source view of the Master Page are shown in fig. (6.a) and (6.b) respectively.

You can create individual content page that contain the content you want to display from the Master Page. When users request the content pages, ASP.NET merges with the master page to produce output that combines the layout of the master page with the content of the content page.

Master pages are not executed individually; they need to be combined with other web forms. Website may contain more than one master page.

Master page actually consist of two parts: the master page itself and one or more content pages.



[Fig (6.a) Design View of the Master Page]

```
<%@ Master Language="VB" CodeFile="MasterPage.master.vb"
Inherits="MasterPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitiona

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Master Page Example</title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1"
        runat="server">

        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>
```

[Fig (6.b) Source code of the Master Page]

Components added to the master page are also appeared on all the pages that inherit it. Master page must contain atleast one ContentPlaceHolder control. It is used to specify the region that can be customized.

The extension of the Master Page is .master. It is identified by a special @Master directive. Along with the @Master directive, Master Page also contain html tags such as <Html>, <Head> and <Form>. You can use any HTML and ASP.NET elements as a part of Master Page.

ContentPlaceHolder

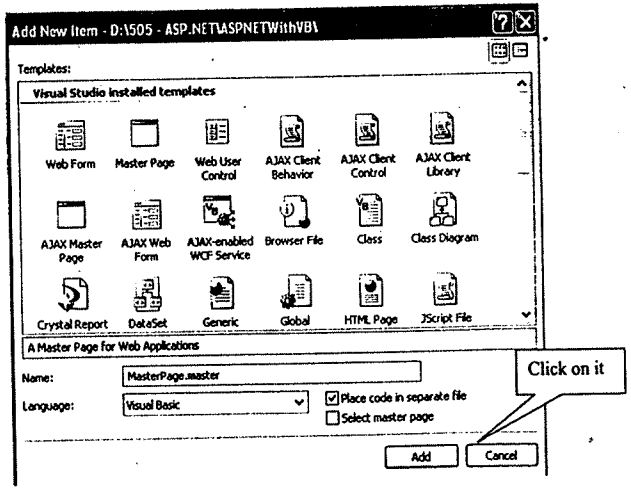
When the Master Page is created, by default you will see one ContentPageHolder on it (As shown in Fig (6.a)). You can include one or more PlaceHolders on the Master Page. These place holders define the region where the content of the Content Pages will be seen.

Content Pages

You can bind the Master Page to the ASP.NET Web Forms (.aspx) which defines the content of the place holders of the Master Pages. These Web Forms are known as Content Pages. The binding is done with the help of MasterPageFile attribute of the @Page directive or the MasterPageFile property of the Page.

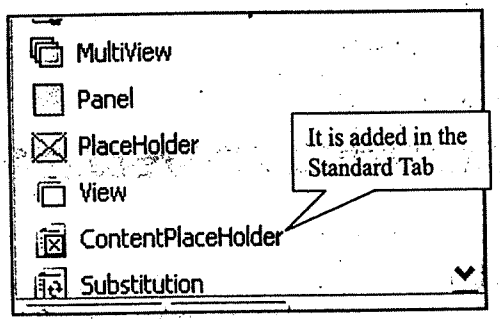
E.g. <% @Page Language="VB" MasterPageFile="~/MasterPage.master" Title="Master Page Demo" %>

- **Steps to create Master Page**
  - ✓ Right Click on the Website in Solution Explorer
  - ✓ Select Add New Item from popup menu
  - ✓ The Dialog Box will appear as shown in Fig (6.c). Select Master Page from it.
  - ✓ By default its name is MasterPage.master
  - ✓ Click on Add Button



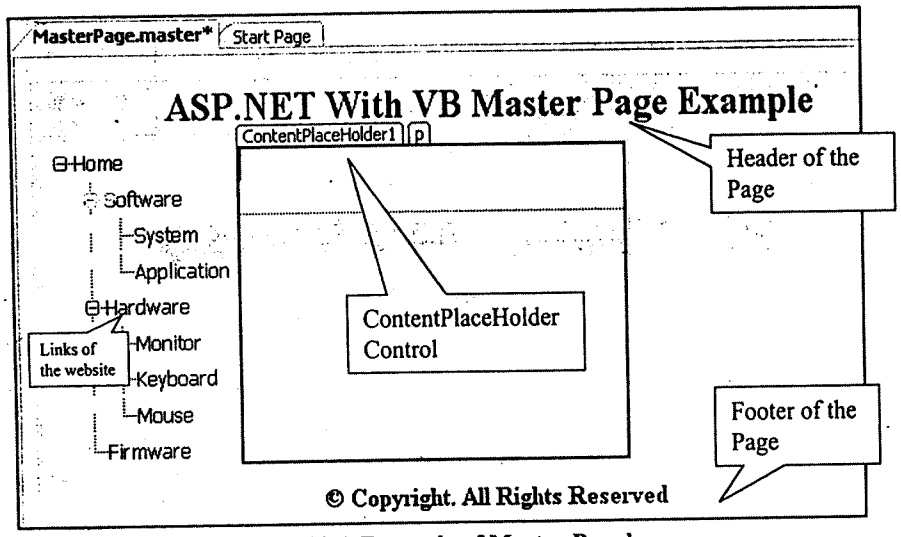
[Fig (6.c) Select Master Page]

- ✓ You will get the Master Page as shown in the Fig (6.a) which contains by default one ContentPlaceHolder.
- ✓ As soon as you add Master Page to your website ContentPlaceHolder control is added to the Standard tab of the ToolBox as shown in Fig (6.d).



[Fig (6.d) ContentPlaceHolder Control]

- ✓ Design the Master Page as shown in Fig (6.e) as per the requirement and place ContentPlaceHolder at the area on the Master Page where you need the Content Page content.



[Fig (6.e) Example of Master Page]

- **Adding the Content Page for the Master Page**

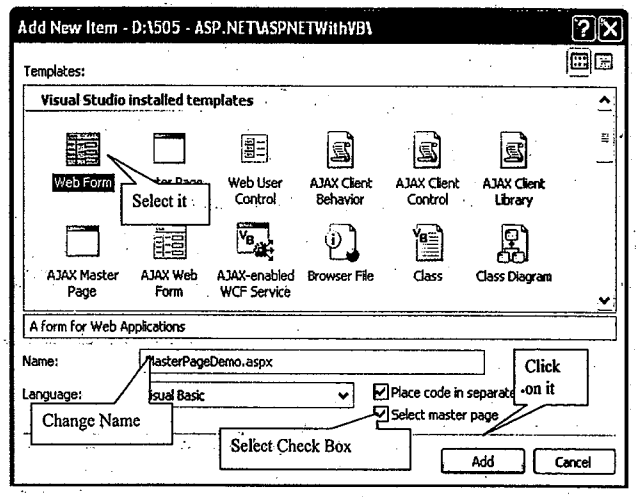
There are different ways to add Content Page:

- ✓ Right click on website in Solution Explorer
- ✓ Select Add New Item from popup menu

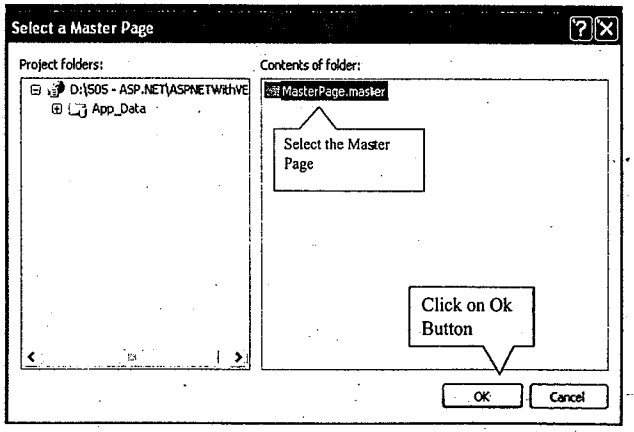
- ✓ Select Web Form (as shown in fig. (6.f)) and change the name accordingly
- ✓ Select the Check Box "Select Master Page" and click on Add Button
- ✓ Select appropriate Master Page from the list (as shown in fig. (6.g))
- ✓ Click on Ok Button

Another Way:

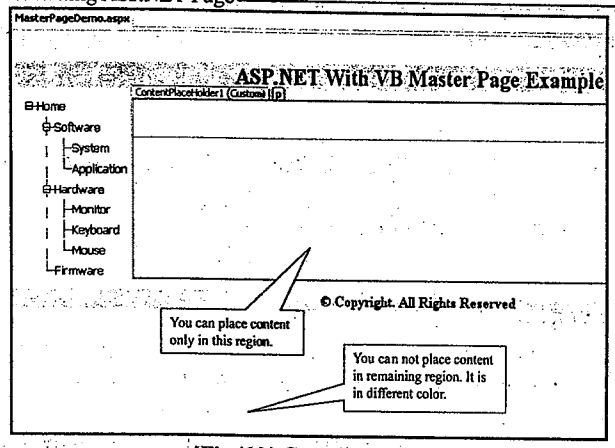
- ✓ Right Click on the Master Page
- ✓ Select Add Content Page from popup menu
- ✓ Change the name accordingly



[Fig (6.f) Adding Content Page]

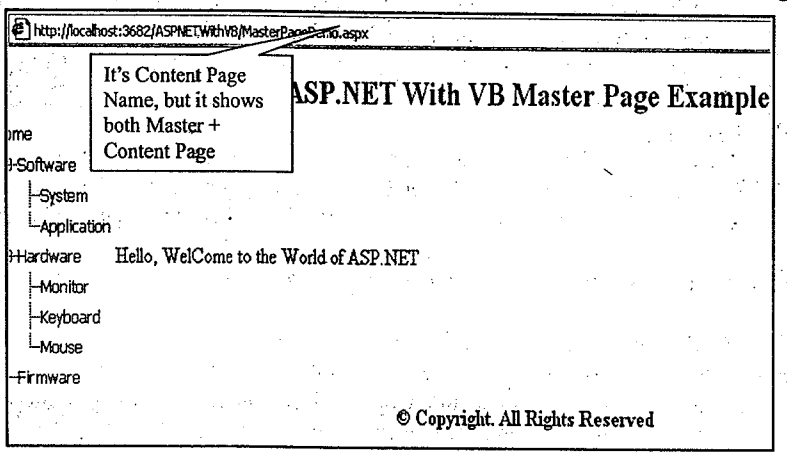


[Fig (6.g) Selecting Master Page]



[Fig (6.h) Content Page]

We can only execute the Content Page and not the Master Page as shown in fig. (6.i)):



[Fig (6.i) Executing the Page]

- **The Master Page is handled at Runtime in following Sequence:**
  - ✓ First user will request the page using the Content Page URL.
  - ✓ The page is retrieved and the @Page directive is examine. If, it contains the Master Page name then the Master Page is also retrieved. If the request is for the first time, then both the pages are compiled.
  - ✓ The Master Page is merged to the content page.
  - ✓ The content of the ContentPlaceHolder controls is merged to the Master Page.
  - ✓ At last the merged page is rendered and given to the browser.

- **Attaching Master Page at runtime**

We can attach Master Page to the Content Page at runtime using the PreInit event of the Content Page.

Example

```
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.PreInit
    Me.MasterPageFile="MasterPage.master"
End Sub
```

- **Advantages of Master Page**

Traditionally, if the developer wants the uniform look and behavior of the pages, then they need to copy – paste the content to all the pages, but in ASP.NET this is done with the help of Master Page.

- ✓ It allow you to centralize the common functionality of your pages, so that if any changes comes at later stage then it can be done from the single place.
- ✓ It is easy to create a common set of controls and code, and then apply it to all the pages.
- ✓ It is easy to control the rendering of the PlaceHolder control.
- ✓ You can customize the Master Page individually without affecting the Content Page.

Nested Master Page

You have already seen that how you can create a single Master Page and create content pages through it. You can also create one Master Page from another Master Page. So, when one Master Page references another as its Master Page, then it is said to be a Nested Master Page.

Example.

MasterPage.master→ChildMaster.master→ContentPage.aspx

MasterPage.master is the Master Page for entire website were you can design Header, Footer and keep ContentPlaceHolder to add content of ChildMaster.master

ChildMaster.master contains Links and ContentPlaceHolder for the ContentPage. aspx.

ContentPage.aspx contains the content of the particular page.

In short, Master Pages can be nested, as one Master Page references another as its master. You can combine several Master Pages together and create one Master Page. The Child Master Pages also have the extension .master.

6.1.Themes

They are skin templates that allow you to define the look of pages and controls, which can then be applied to all the pages in your application to provide consistency for entire application.

A skin is a set of properties and templates that can be used to standardize the size, font, and other characteristics of controls of a page. You can combine multiple skins and Stylesheets to specify the overall look of the website.

There are two types of the Skins:

1. Default Skin 2.Named Skin

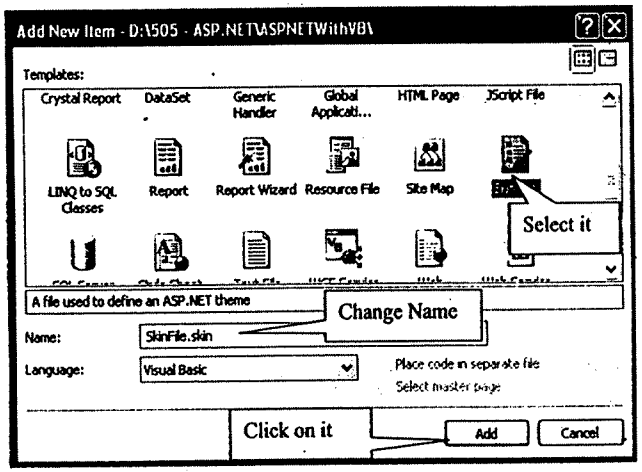
The skin that does not have SkinID is called Default Skin. The Skin that has SkinID is called Named Skin.

- **Adding Theme with Skin File to the Website**

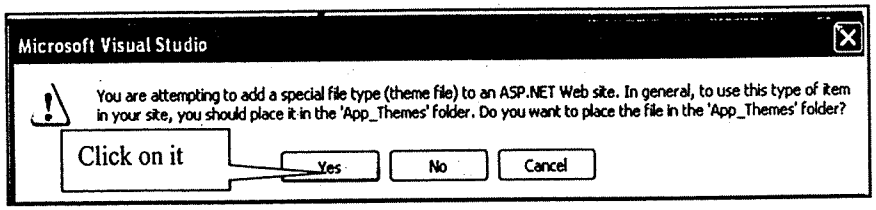
Steps to add Skin File are as follows:

- ✓ Right Click on the Website in Solution Explorer
- ✓ Select Add New Item from popup menu
- ✓ Select Skin File from the wizard (as shown in fig. (6.j))
- ✓ Change the Name if needed
- ✓ Click on Add button
- ✓ It will ask you to add this skin file in App_Theme folder (see fig. (6.k)), and then click on Yes Button.
- ✓ The skin file is added to the website

By default, the name of the file is SkinFile.skin where .skin is an extension. The Skin File contains the comments that specify different types of themes (as shown in fig. (6.l)).



[Fig (6.j) Adding Skin File]



[Fig (6.k) Adding Skin File to App_Theme Folder]

App_Themes/Sk.../SkinFile.skin

```
<%--
Default skin template. The following skins are provided as examples only.

1. Named control skin. The SkinId should be uniquely defined because
   duplicate SkinId's per control type are not allowed in the same theme.

<asp:GridView runat="server" SkinId="gridviewSkin" BackColor="White" >
   <AlternatingRowStyle BackColor="Blue" />
</asp:GridView>

2. Default skin. The SkinId is not defined. Only one default
   control skin per control type is allowed in the same theme.

<asp:Image runat="server" ImageUrl="~/images/image1.jpg" />
--%>
```

[Fig (6.l) Skin File]

Now within the Skin File, add the Default Skin to provide different effect to the control (as shown in fig. (6.m)).

App_Themes/Sk.../SkinFile.skin

```
<asp:TextBox runat="server" BackColor="#FFFF99"
BorderColor="Maroon" BorderStyle="Dashed"
ForeColor="Maroon">
</asp:TextBox>
```

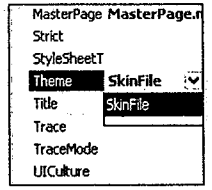
[Fig (6.m) Default Skin for the TextBox in SkinFile.Skin]

- **Different ways to apply Skin File are as follow:**
  - ✓ @Page Directive (fig. (6.n)) or property of the Page (fig. (6.o)).

You can apply the theme to the page by setting the Theme attribute of the @Page directive or by setting the Theme property of the page to the SkinFile from the Property Window

```
<%@ Page Language="VB"  Theme="SkinFile"
```

[Fig (6.n) @Page Directive Theme Attribute]



[Fig (6.o) Theme property of the Page]

- ✓ Applying theme at runtime.

We can attach Skin File to the Page at runtime using the PreInit event of the Page.

Example.

```
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.PreInit
        Page.Theme="SkinFile.skin"
End Sub
```

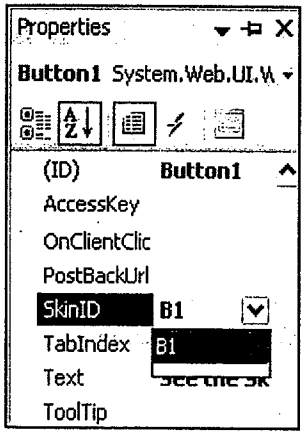
- ✓ Applying theme globally within web.config file

```
<configuration>
        <system.web>
                <pages theme="SkinFile"></pages>
        </system.web>
</configuration>
```

You can also create Named Skin in the same way as you create Default Skin. The only difference is that you need to add SkinID attribute in the skin (as shown in fig. (6.p)).

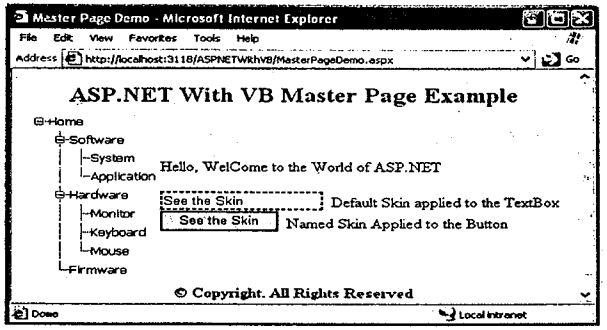
```
<asp:Button SkinID="B1" runat="server" BackColor="#99CCFF"
BorderColor="#000099" BorderStyle="Groove" ForeColor="#000099"/>
```

[Fig (6.p) Named Skin for the Button in SkinFile.Skin]



Named Skin is to be applied to the particular control on the Page. To apply the Named skin, set SkinID property of the control (as shown in fig. (6.q)).

←[Fig (6.q) SkinID property of the Button]



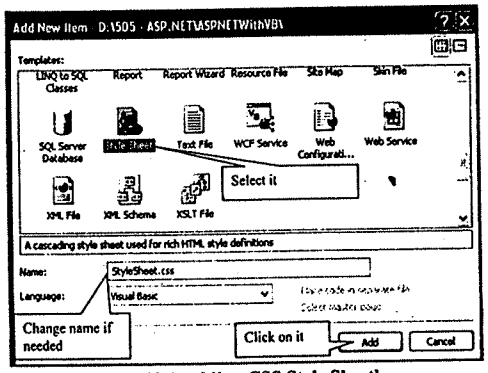
[Fig (6.r) Control with Skin on the Page]

After applying the theme to the page, the Page looks as shown in Fig (6.r)

6.3. CSS

CSS is use for the standardization of website formatting and provides consistent appearance throughout the application. It is supported by all modern browsers.

- **Creating Style Sheet**
  - ✓ Right Click on the Website in Solution Explorer.
  - ✓ Select Add New Item from popup menu
  - ✓ Select Style Sheet from the wizard (fig. (6.s)).
  - ✓ Change the name if needed
  - ✓ Click on Add Button



[Fig (6.s) Adding CSS Style Sheet]

By default, the name of Style Sheet is StyleSheet.css where .css is an extension.

Style sheets consist of rules which define the formatting information for the particular elements on the web page. Each rule-name has two parts. The portion before the period indicates the tag to which the rule applies. The portion after the period is a unique name (called the CSS class name). Rules can be specified in following manner:

1. If standard tag name (body, h1, td etc) is specified without any period then the rules are applied to all such tags under the web page.

```
H1
{
font – weight : bold;
}
```

Constraint will be set to all H1 tag.

2. If only css class name is mentioned (portion after the period) then the rules are applicable to all the tags with specified class name.

```
.BoldText
{
font – weight : bold;
}
```

Constraint will be set to all tags where css class name is specified as "BoldText".

3. If complete rule name is mentioned (portion at the both side of period) then the rules are applied to the specified tag with the specified class name.

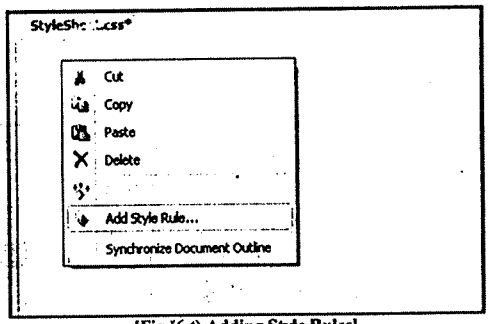

```
H1.BoldText
{
font – weight : bold;
}
```

Constraint will be set to all H1 tags where css class name is specified as "BoldText".

- **Creating CSS in ASP.NET**

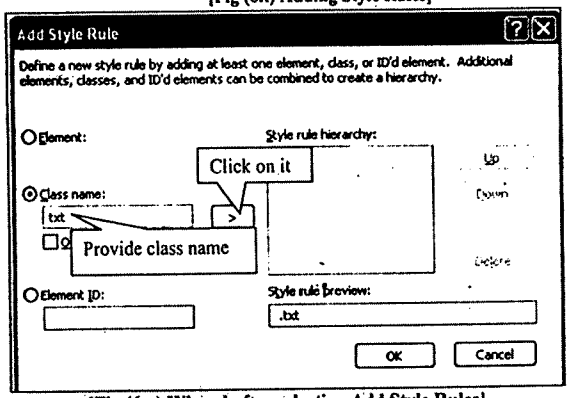
On the StyleSheet.css file follow the steps:

- ✓ Right click on the StyleSheet.css file
- ✓ Select Add Style Rules from popup menu (fig. (6.t))

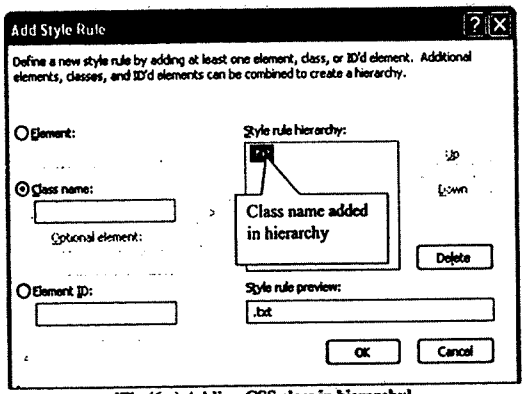


[Fig (6.t) Adding Style Rules]

- ✓ Select appropriate option from the Wizard (fig. (6.u))



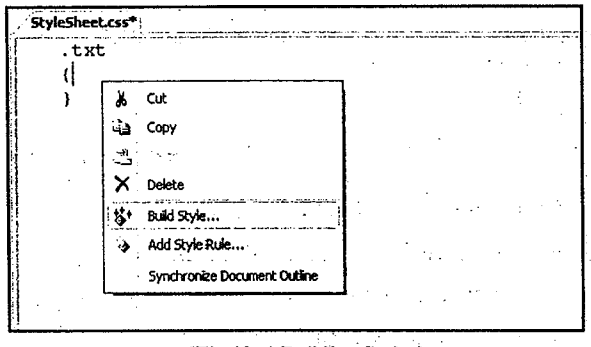
[Fig (6.u) Wizard after selecting Add Style Rules]



[Fig (6.v) Adding CSS class in hierarchy]

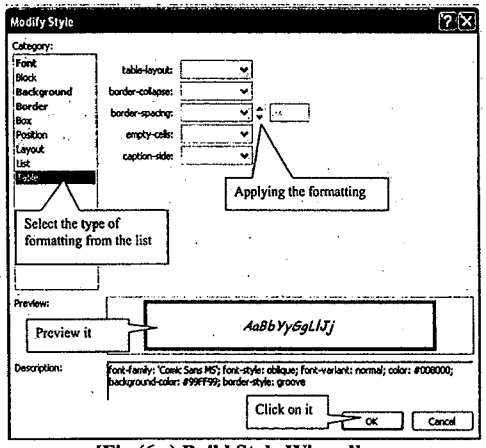
- ✓ If Class name is selected, then specify its name and click on > Button. So that the Class name appear in the hierarchy box (fig. (6.v)).

✓ The StyleSheet.css file will look as shown in Fig (6.w). Now right click and select Build Style Option from popup menu (fig. (6.w)).



[Fig (6.w) Building Styles]

✓ From the wizard apply the appropriate style (fig. (6.x))



[Fig (6.x) Build Style Wizard]

✓ After the Style is applied the StyleSheet.css will be as shown in Fig (6.y)

StyleSheet.css*

```
.txt
{
    font-family: 'Comic Sans MS';
    font-style: oblique;
    font-variant: normal;
    color: #008000;
    background-color: #99FF99;
    border-style: groove;
}
```

[Fig (6.y) Style Code in .css file]

Now you can apply the Style to the Text Box on the Page.

- **Applying Style Sheet to the web page**

There are different ways to attach the StyleSheet.css file to the web page.

a) Link tag

Add the <link > tag under the <Head> section and give reference of the specific stylesheet.

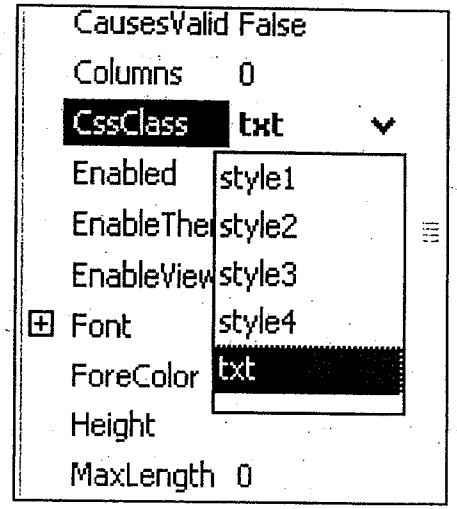
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />

Where href attribute is set to the location of the required stylesheet class (.css).

b) Through the StyleSheetTheme property of the page.

It is used only, if you have created the StyleSheet.css file within the SkinFile folder within the App_Themes Folder of the Website.

Now the rules, under the stylesheet, can be applied to the web controls by setting CssClass property Fig (6.z):



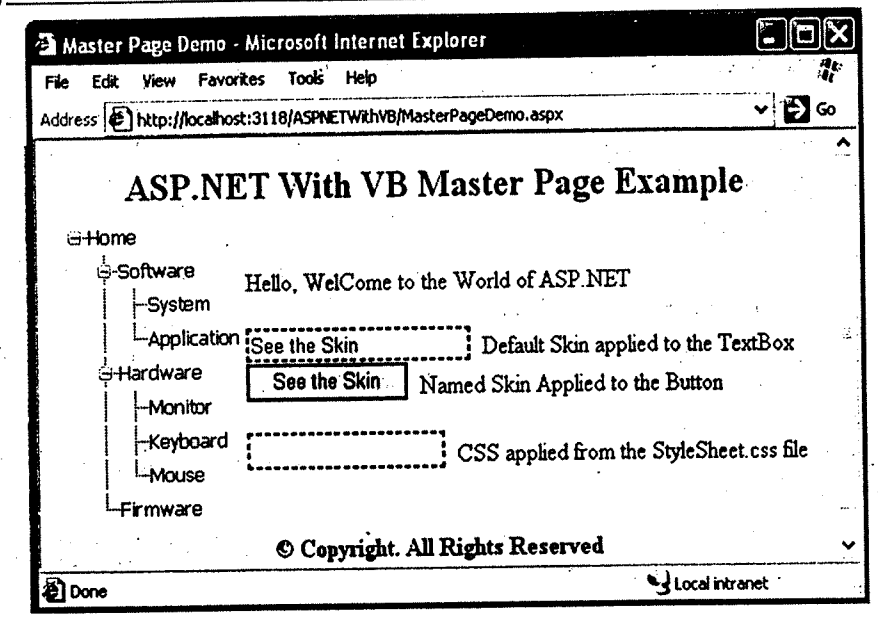
[Fig (6.z) Setting CssClass Property of Text Box]

<asp:TextBox ID="TextBox2" runat="server" CssClass="txt">

</asp:TextBox>

.txt is the class name of the CSS which we have created in the StyleSheet.css file.

Now, save & execute the page.fig(6.aa)



[Fig (6.aa) Text Box after applying CSS]

Exercise - 6

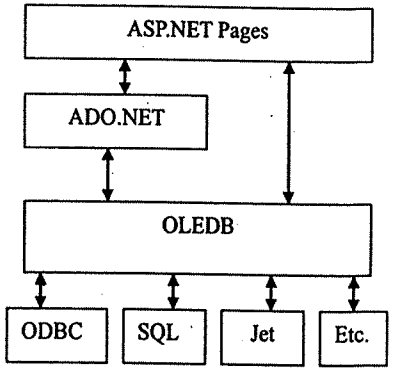
1. Explain Master Page along with different ways to create Content Page.
2. What are the advantages of Master Page?
3. What is Nested Master Page?
4. How to apply Theme and Master Page at runtime?
5. Explain different types of Skin.
6. How to apply Named Skin to the controls of the page?
7. What is CSS? Explain it in detail
8. How CSS is implemented in ASP.NET?
9. How to apply the CSS styles to the controls of the page?

7 Database Interaction

7.1 Introduction to ADO.NET

ADO.NET is the major revision of ADO (ActiveX Data Objects) that enables ASP.NET pages to present data in much more efficient and different ways. For example, it fully makes use of XML and is easily able to communicate with any XML – complaint application. ADO.NET offers a lot of new features.

In terms of ASP.NET development, ADO.NET provides the framework for accessing any type of data that can be used with ASP.NET pages. This allows users to view or change information stored in any kind of database, text Files, and XML Data dynamic application development. It shares Common Type System, design pattern and naming conventions.



[Fig (7.a): ASP.NET & ADO.NET]

- **Evolution of ADO.NET**

The first data access model created was DAO (Data Access Object) for local databases. It consists of built-in Jet engine which had performance and functionality issues. Then RDO (Remote Data Object) and ADO (Active Data Object) came. They were designed for Client Server architectures but soon ADO took over RDO. ADO was a good architecture, but with it, all the data are stored in a recordset object which had problems when implemented on the network. ADO was a connected data access, which means that when a connection to the database is established the connection remains open until the application is closed. So, if the connection is open for the lifetime of application, it raises issues like database security, network traffic and productivity. For example, an application with connected data access may work well when connected to two clients, the same may work poorly when connected to 10 and might be stopped when connected to 100 or more clients. When database connection is kept open for long period of time then it uses system resources to a maximum extent making the system performance less effective.

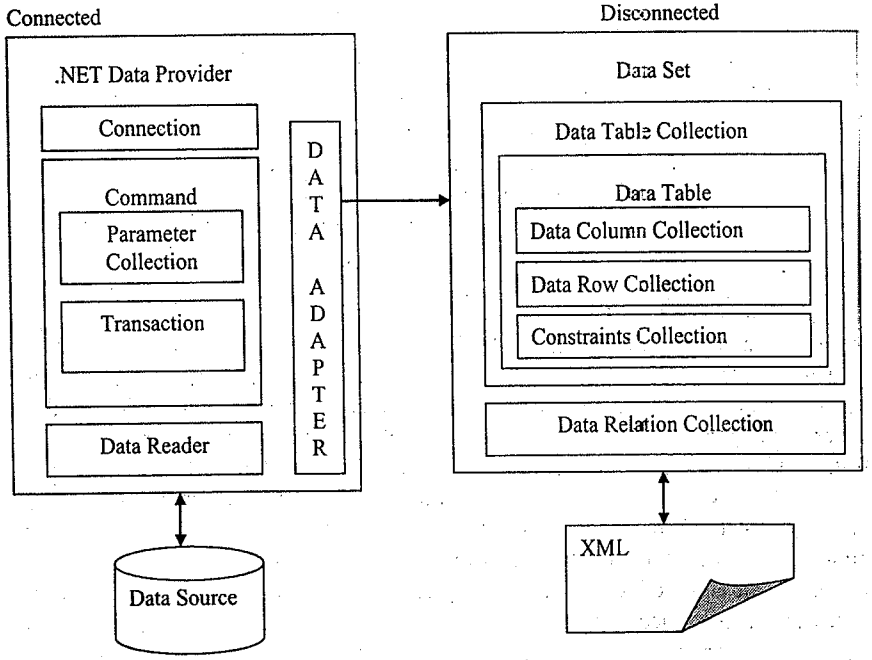
- **Why ADO.NET?**

To handle the problems mentioned above, ADO.NET came into existence. Above mention problems are removed in ADO.NET as it maintain a disconnected data model which means, when an application interacts with the database, the connection is opened to serve the request of the application and is closed as soon as the request is completed. Likewise, if a database is updated, the connection is opened until the completion of the update operation. By keeping connections open for only a minimum period of time, ADO.NET use less system resources and provides maximum security for databases and also has less impact on system performance. Another significance of ADO.NET is that, when interacting with the database, it uses XML and converts all the data into XML format for database related operations which makes it more efficient.

7.2 ADO.NET Architecture

ADO.NET is an object-oriented set of libraries that allows you to interact with data sources. Commonly, the data source is a data base, but it could also be a text file, an Excel spread sheet, or an XML file. There are many different types of databases available such as Microsoft SQL Server, Microsoft Access, Oracle, Borland Interbase, IBM DB2, etc.

- **Connected & Disconnected Data (Architecture)**



[Fig (7.b): Connected Disconnected Architecture of ADO.NET]

The data access with ADO.NET consists of two parts:

(1) Data Provider
(2) DataSet

ADO.NET allows us to interact with different types of data sources and different types of data bases. There are many set of classes that allow you to access different types of databases. Since different data sources are interpreted with different protocols, we need a way to communicate with the right data source using the right protocol. Some older data sources use the ODBC (Open DataBase Connectivity) protocol, many new data sources use the OleDb (Object Linking and Embedding DataBase) protocol, and there are many other data sources that allow you to communicate with them directly through .NET ADO.NET class libraries.

(1) Data Provider

The Data Provider is used for providing and maintaining the connection to the database. It is a set of related components that work together to provide data in an efficient manner

Provider Name	API prefix	Data Source Description
ODBC Data Provider	Odbc	Data Sources with an ODBC interface. Normally older data bases.
OleDb Data Provider	OleDb	Data Sources that expose an OleDb interface, i.e. Access or Excel.
Oracle Data Provider	Oracle	For Oracle Data Bases.
SQL Data Provider	Sql	For interacting with Microsoft SQL Server.
Borland Data Provider	Bdp	Generic access to many data bases such as Interbase, SQL Server, IBM DB2, and Oracle.

[Table:1 List of Various Data Providers]

ADO.NET provides a common way to interact with data sources, but with different sets of libraries. These libraries are called Data Providers and are usually named for the protocol or data source type that allow you to interact with the database. Table:1 lists some well known data providers, the API prefix they use, and the type of data source they allow you to interact with.

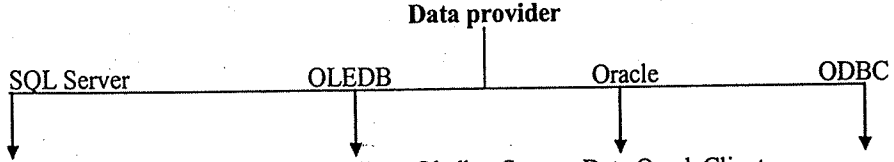
- **Example**

First Object of ADO.NET that you will learn is the connection object, which allows you to establish a connection to a data source. If we were using the OleDb Data Provider to connect to a data source that uses an OleDb interface, then we would use a connection object named OleDbConnection. Similarly, the connection object name would be prefixed with Odbc or Sql for an OdbcConnection object on an Odbc data source or a SqlConnection object on a SQL Server data base, respectively. Since we are using MSDE, all the API objects will have the Sql prefix.

i.e. SqlConnection.

Each DataProvider consists of the following classes:

The Connection object which provides a connection to the database. The Command object which is used to execute a command. The DataReader object which provides a forward-only, read only, connected recordset. The DataAdapter object which populates a disconnected DataSet with data and performs update



SqlConnection	OledbConnection	OracleConnection	OdbcConnection
SqlCommand	OledbCommand	OracleCommand	OdbcCommand
SqlDataAdapter	OledbDataAdapter	OracleDataAdapter	OdbcDataAdapter
SqlDataReader	OledbDataReader	OracleDataReader	OdbcDataReader

[Fig (7.c): Data Providers & Namespaces available in ADO.NET]

In Fig (7.c), System.Data.SqlClient, System.Data.Oledb, System.Data.OracleClient and System.Data.ODBC are the namespaces.

- **ADO.NET Objects**

ADO.NET consists of many objects that we use to work with data.

a) Connection Object
b) Command Object
c) Parameter Object
d) DataReader Object
e) DataAdapter Object

a) Connection Object

To establish connection with a database, you must have an object. The connection helps to identify the database server, the database name, user name, password, and other parameters that are required for connecting to the database. A connection object is used by command objects so they will know which database to execute the command on.

Connection String – A string that specifies information about a data source and the means of connecting to it is called Connection String. It is passed in code to an underlying driver or provider in order to initiate the connection.

Parameters of Connection String:

- ✓ AttachDBFileName / Initial FileName
- ✓ ConnectTimeOut / Connection TimeOut

- ✓ DataSource / Server / Address / Addr / Network address
- ✓ Initial catalog / database
- ✓ Trusted connection / integrated security
- ✓ Persist security info.

Example

```
Data Source=myServerAddress;Initial Catalog=myDataBase;User Id=myUsername;
Password=myPassword;
Server=myServerAddress;Database=myDataBase;User ID=myUsername;Password=
myPassword;Trusted_Connection=False;
Data Source=myServerAddress;Initial Catalog=myDataBase;Integrated Security=
SSPI;
```

Note: In this chapter all examples are with respect to Microsoft SQL Server as a Database.

- **SqlConnection Class**

The purpose of creating a SqlConnection object is that you can connect to SQL Server database. Other ADO.NET objects, such as a SqlCommand and a SqlDataAdapter use a connection object as a parameter. The sequence of operations that occurs with SqlConnection is as follows:

1. Instantiate the SqlConnection.
2. Open the connection.
3. Pass the connection to other ADO.NET objects.
4. Perform data base operations with the other ADO.NET objects.
5. Close the connection.

To create SqlConnection Object:

Dim conn As New SqlConnection("Data Source=DatabaseServer;Initial Catalog=DBName;User ID=YourUserID;Password=YourPassword")

Properties

Properties	Description
ConnectionString	It stores the connection string that is passed to the connection object at the time of creating its object.
Database	It stores the name of the database to which you need to connect.
State	It return the state of the connection i.e. IsClose or IsOpen
ConnectionTimeOut	Gets the time to wait while trying to establish a connection before terminating the attempt and generating an error.

[Table:2 Properties of Connection Object]

Methods

Methods	Description
Open	It opens the connection.
Close	It closes the connection.
BeginTransaction	It creates the Transaction Object
CreateCommand	It creates and returns a SqlCommand object associated with the SqlConnection.
ChangeDatabase	It changes the current database for an open SqlConnection.

[Table:3 Methods of Connection Object]

b) Command Object

It is used to retrieve a subset of data. Also invoking SQL insert, Update and Delete statement directly require you to set certain parameters on the command before executing the statement. Another common use of the command object is to execute stored procedure and pass the appropriate parameters to the database for the stored procedure.

Properties

Properties	Description
Connection	To set a connection object.
CommandText	It specifies the SQL string or stored procedure to be executed.
CommandType	It is used to determine how to interpret command text. i.e. CommandText is StoredProcedure or Text or DirectTable.
Transaction	It is used to set the Transaction Object that is to be used for this command.
Paremeters collection	It is used to specify placeholders instead of direct values.
CommandTimeOut	Gets the time to wait while trying to execute the command before terminating the attempt and generating an error.

[Table:4 Properties of Command Object]

Methods

Methods	Description
ExecuteNonQuery	It will execute the SQL statement and returns the number of rows affected by the query.
ExecuteScalar	It will execute the SQL statement which return the singleton value.
ExecuteReader	It will execute the SQL statement and returns the records in the form of DataReader. i.e. it is used to create the Object of DataReader.
CreateParameter	It creates and returns a SqlParameter object associated with the SqlCommand.
Cancel	It is used to cancel the command given for execution.
Prepare	It is used to compile the statement before the execution.
ResetCommandTimeOut	It is used to reset Command time out property to its default value

[Table:5 Methods of Command Object]

Example

```
Dim conn As New SqlConnection("Data Source=.\SQLEXPRESS; AttachDbFilename =
|DataDirectory|\ASPNETWithVB.mdf;Integrated Security=True;User Instance= True")
Dim cmd As New SqlCommand
cmd.Connection=conn
cmd.CommandText= "Select Count(*) from tblEmployee"
Dim EmpCnt As Integer
conn.Open()
EmpCnt=cmd.ExecuteScalar()
```



```
conn.Close()
Response.Write("Total Employee=" + EmpCnt.ToString())
```

c) SqlParameter Object

When we are working with data, you may also need to filter results based on some criteria. Typically, this is done by accepting input from a user and using that input in SQL query. For example, an Admin want to see the data of Employee working in specific department. Another query might be to filter Employees by City. Now you know that the SQL query assigned to a SqlCommand object is a string. So, if you want to filter records, you need to build the string dynamically. But this is not the good programming approach.

Example

```
Dim cmd As New SqlCommand("Select * from tblEmployees where City = ' " +
varCity + "'",conn)
```

Here the input variable, varCity, is retrieved from a TextBox control on the Web Page. Anything placed into that TextBox control will be put into varCity and added to your SQL string. This situation invites a hacker to replace that string with something malicious.
Instead of dynamically building a string, use parameters. Anything placed into a parameter will be treated as field data, not part of the SQL statement, which makes your application much more secure.
There are three steps to use parameterized queries:

1. Construct the SqlCommand command string with parameters.
2. Declare a SqlParameter object, assigning appropriate values.
3. Assign the SqlParameter object to the SqlCommand object's Parameters property.

The following sections take you step-by-step through this process.

1. Preparing a SqlCommand Object for Parameters

In first step you need to build a command string containing parameter placeholders. These placeholders are filled in with actual parameter values when the SqlCommand executes. You have to use an '@' symbol as a prefix for the parameter name as shown below:

```
Dim cmd As New SqlCommand("Select * from tblEmployees where City
=@City",conn)
```

In the SqlCommand constructor, the first argument contains a parameter declaration, @City. This example used one parameter, but you can have as many parameters as needed to customize the query. Each parameter will match a SqlParameter object that must be assigned to this SqlCommand object.

2. Declaring a SqlParameter Object

Each and every parameters used in a SQL command must be defined. You must define a SqlParameter instance for each parameter in a SqlCommand object's SQL command. The following code defines a parameter for the @City parameter from the previous section:

```
Dim param As New SqlParameter
```

```
param.ParameterName = "@City"
param.Value = varCity
```

Notice that the ParameterName property of the SqlParameter instance must be exactly the same as the parameter that is used in the SqlCommand query string. You must also specify a value for the command. When the SqlCommand object executes, the parameter will be replaced with this value.

3. Associate a SqlParameter Object with a SqlCommand Object

You must define SqlParameter for each parameter defined in the SQL command string. You must also let the SqlCommand object know about the SqlParameter by assigning the SqlParameter instance to the Parameters property of the SqlCommand object. The following code shows how to do this:

```
cmd.Parameters.Add(param)
```

Here the SqlParameter created above is added as an argument to the Add method of Parameters collection of SqlCommand Object and this SqlParameter must be unique for each parameter.

Example

```
Imports System
Imports System.Data
Imports System.Data.SqlClient
Dim conn As New SqlConnection("Data Source=.\SQLEXPRESS;
AttachDbFilename = |DataDirectory| \ASPNETWithVB.mdf;Integrated
Security=True;User Instance=True")
Dim cmd As New SqlCommand
cmd.Connection=conn
cmd.CommandText="Select Count(*) from tblEmployee Where City=@City"
Dim param As New SqlParameter
param.ParameterName = "@City"
param.Value = varCity
cmd.Parameters.Add(param)
Dim EmpCnt As Integer
conn.Open()
EmpCnt=cmd.ExecuteScalar()
conn.Close()
Response.Write("Total Employee=" + EmpCnt.ToString())
```

d) DataReader Object

A SqlDataReader is used to read data in the most efficient manner. You cannot use it for writing data. You can read forward – only and in sequential manner from SqlDataReader. Once you have read some data, you must save it because you will not be able to go back and read it again. A single row of data is in the memory at a time. When the SqlDataReader object is created the pointer is placed before the first row.

Properties

Properties	Description
FieldCount	It stores number of fields in a row.
HasRows	It specifies that the rows are selected or not for reading.
IsClosed	It specifies that DataReader is closed or not.
RecordsAffected	It returns -1 as DataReader is created on server.
Item	It gets the value of the specified column name.

[Table:6 Properties of DataReader Object]

Methods

Methods	Description
Read	It read the Next Record for DataReader.
Close	It is used to Close the DataReader Connection with the database.
IsDBNull	It checks that the value of the column is Null or not.
GetSchemaTable	It returns the object of the Data Table for which the DataReader is created.
GetName(index)	It returns the Name of the field whose index is passed as an argument.
GetOrdinal(string name)	It returns the position of the field whose name is passed as an argument.
GetFieldType(index) / GetFieldType(FieldName)	It returns the data type of the field whose index is passed as an argument.
Get<Type> E.g. GetString(index) / GetString(FieldName)	It returns the value of the field in the form of String whose index is passed as argument.
GetValue(index) / GetValue(FieldName)	It returns the value of the field whose index is passed as an argument. It returns the value in the form of object.
GetValues	It returns the array of the values for the row.
NextResult	It is used to navigate from one record set to another when more than one record sets are used in the command.

[Table:7 Methods of DataReader Object]

- **Steps to create DataReader**
  - ✓ Create a connection and open it
  - ✓ Create command object with the appropriate select query
  - ✓ Use the command.ExecuteReader() method
  - ✓ Move through the rows using Read() method
  - ✓ Close DataReader() and Connection()

Example

```
Imports System
Imports System.Data
Imports System.Data.SqlClient
```

```
Dim conn As New SqlConnection("Data Source=.\SQLEXPRESS; AttachDbFilename =
|DataDirectory|\ASPNETWithVB.mdf;Integrated Security=True;User Instance=True")
    Dim cmd As New SqlCommand
    Dim dr As SqlDataReader
    cmd.Connection=conn
    cmd.CommandText="Select EmpNo,EnpName from tblEmployee"
    conn.Open()
    dr=cmd.ExecuteReader
    while dr.Read()
        Response.Write(dr.GetInt32("EmpNo").ToString + " " + dr.GetString
    ("EmpName"))
    Wend
    dr.Close()
    conn.Close()
```

Here note that the object of DataReader is not created directly with New keyword. The Object is created with the ExecuteReader method of the Command Object.

e) DataAdapter

It acts as a bridge between data source and in-memory data objects such as the Dataset.

Properties

Properties	Description
SelectCommand	It is used to hold a Command that retrieves data from the data source.
UpdateCommand	It is used to hold a Command that updates data to the data source.
DeleteCommand	It is used to hold a Command that delete data from the data source.
InsertCommand	It is used to hold a Command that insert data into the data source.
CommandType	It indicates CommandText property contains SQL statement or stored procedure. If CommandText property contains stored procedure than set the value to CommandType.StoredProcedure. Default value is CommandType.Text for SQL statement.

[Table:8 Properties of DataAdapter Object]

Methods

Methods	Description
Fill	It is used to populate a dataset object with the data that the DataAdapter object retrieve from the data store using its SelectCommand. But before that we must initialize a Dataset object.
Update	It is used to update the database according to the changes that are made in the DataSet.

[Table:9 Methods of DataAdapter Object]

Example

```
Imports System
Imports System.Data
Imports System.Data.SqlClient
    Dim conn As New SqlConnection("Data Source=.\SQLEXPRESS;
AttachDbFilename = |DataDirectory|\ASPNETWithVB.mdf;Integrated
Security=True;User Instance= True")
Dim da As SqlDataAdapter
da.SelectCommand.Connection = conn
da.SelectCommand.CommandText = "Select * from tblEmployee" or  "Stored procedure
name"
'If commandText contains stored procedure name than include following statement.
da.SelectCommand.CommandType= CommandType.StoredProcedure
da = new SqlDataAdapter("spEmployeeCount",conn) 'spEmployeeCount is procedure
name
da.SelectCommand.CommandType = CommandType.StoredProcedure
da.SelectCommand.Parameter.Add("EmpNo",number,2).value=n
da.SelectCommand.Parameter.Add("ECount",number,2).Direction=ParameterDirection.
Output
```

CommandBuilder

When we use Update() method to update the database, the changes remains only in the DataSet, it is not reflected in the Database. To implement changes in the database, you need to use CommandBuilder. SqlCommandBuilder class of System.Data.SqlClient namespace automatically generates single – table commands that are used to update the changes made to the DataSet with the associated SQL server database. It uses the SelectCommand property of the DataAdapter to generate Insert, Update, and Delete queries.

You need to add following code:

```
Dim cb As New SqlCommandBuilder(da)
Da.Update(ds,"Emp")
```

If you call Dispose, the SqlCommandBuilder is disassociated from the SqlDataAdapter, and the generated commands are no longer used.

(2) DataSet

The dataset is a disconnected, in-memory representation of data. It can be considered as a local copy of the portions of the database. The DataSet is persisted in memory and it can be manipulated and updated independent of the database. When the use of this DataSet is finished, changes can be made back to the central database for updating. The data in DataSet can be loaded from any valid data source like Microsoft SQL server database, an Oracle database or from a Microsoft Access database. The DataSet class resides in the System.Data namespace. The Dataset object contains a collection of tables, relationships and contraints that are consistent with the data read from the data store.

Example

```
Dim conn As New SqlConnection("Data Source=.\SQLEXPRESS;
AttachDbFilename = |DataDirectory|\ASPNETWithVB.mdf;Integrated
Security=True;User Instance= True")
Dim da As New SqlDataAdapter
da.SelectCommand.Connection = conn
da.SelectCommand.CommandText = "Select * from tblEmployee"
Dim ds As New DataSet
da.Fill(ds,"Emp")
```

- **Objects of DataSet**

Objects of Dataset are DataTable, DataColumn, DataRow, Constraints and DataRelation. DataSet is the collection of DataTable and DataRelation. DataTable is made up of DataRow and DataColumn. DataColumn may have Constraints.

Example

```
Dim tmprow As DataRow
Dim mycol As New DataColumn("EmpNo", type.gettype("System.Int32"))
EmployeeTable.Columns.Add(mycol)

EmployeeTable.columns.add("EmpName", Type.GetType("System.String"))
EmployeeTable.Columns.Add("Salary", Type.GetType("System.Single"))
EmployeeTable.Columns.Add("Department", Type.GetType("System.Int32"))
EmployeeTable.PrimaryKey="EmpNo"

Tmprow = EmployeeTable.NewRow()
Tmprow("EmpNo") = 1
Tmprow("EmpName") = "Nikisha"
Tmprow("DeptNo") = "10"
EmployeeTable.Rows.Add(tmprow)
```

DataRelation

It is the object used to maintain the relationship within the DataSet.

Example

```
Dim datar As New
DataRelation("deptemp",ds.Tables("tblDepartment").Columns("DeptNo"),
dsTables("tblEmployee").Columns("DeptNo"))
ds.DataRelations.Add(datar)
Dim dno, eno As Integer
Dim temprow, emprow As DataRow
For Each temprow In ds.Tables("dept").Rows
    dno=temprow("DeptNo")
    For Each emprow In temprow.GetChildRows()
        eno=emprow("EmpNo")
    Next
Next
```


- **DataView**

It is used for sorting, filtering, searching, editing and navigating the data from a dataSet. It provides custom view of a DataTable – i.e. we can sort or filter the rows. You can create a DataView from existing DataSet as follow:

```
Dim dv As New DataView(ds.Tables("Emp"))
```

Properties

Properties	Description
RowFilter	It is used to filter the rows of data that it will contain in DataView. Only rows that match the criteria will remain in the view. It is similar to specifying where clause in SQL queries. E.g. dv.RowFilter = "EmpName="Nikisha" dv.RowFilter = "EmpNo>=6 And EmpNo<=10"
Sort	It sorts the data stored in DataView. Sorting operation on a DataView are always performed in an ascending order by default. If we wanted to perform the sort in descending order, we would need to specify the DESC keyword. E.g. dv.Sort = "column_name" dv.Sort = "sname, rno" dv.Sort = "sname Desc"

[Table:10 Properties of DataView Object]

Methods

Methods	Description
Find	To search specific row of data in DataView. Here DataView need to be sorted first, searching process done as per sort key column of DataView. It returns –1 if no match found otherwise returns position of the record in DataView. E.g. Dim pos As Integer dv.Sort = "EmpName" Pos = dv.Find ("Nikisha")

[Table:9 Methods of DataAdapter Object]

Example

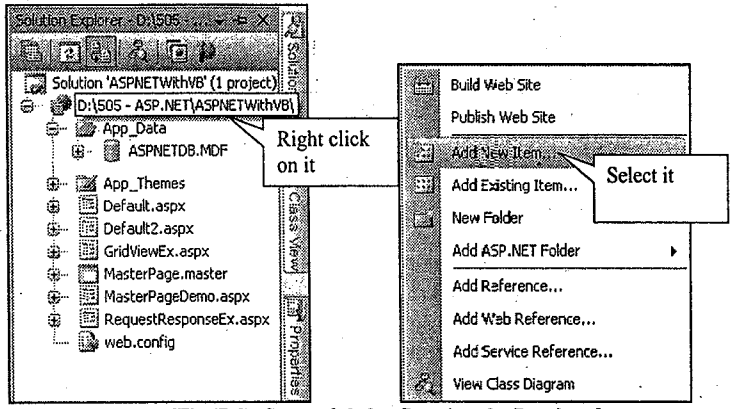
If we want to sort on more than one column, we need to supply an array of values to the Find method instead of just a single value.

```
Dim pos As Integer
Dim vals(1) as object
dv.Sort = "fname, lname"
Vals(0)="Nikisha"
Vals(1)="Jariwala"
```

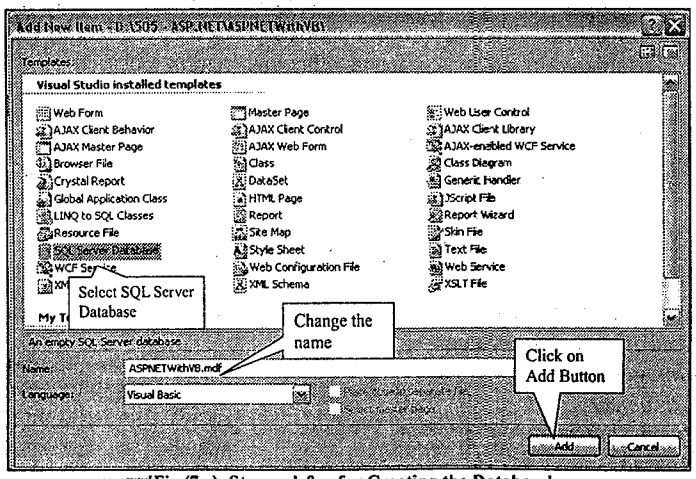
Pos = dv.Find (vals)

- **Creating a Database in Visual Studio.NET**
  a) Right click on the WebSite in Solution Explorer (Fig (7.d))
  b) Select Add New Item from the List (Fig (7.d))
  c) Select SQL Server Database from the Templates (Fig (7.e))
  d) Change the Name of the Database as you need (Fig (7.e))
  e) Click on Add Button.
  f) It will ask to add the Database in App_Data Folder Click on Yes Button. (Fig (7.f))

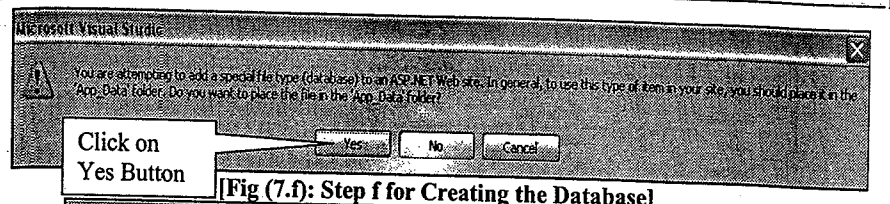
The Database is created and you are able to see it in Solution Explorer within App_Data folder of WebSite and when you double – click on the database, it is viewed in Server Explorer where you can manage the objects of it.



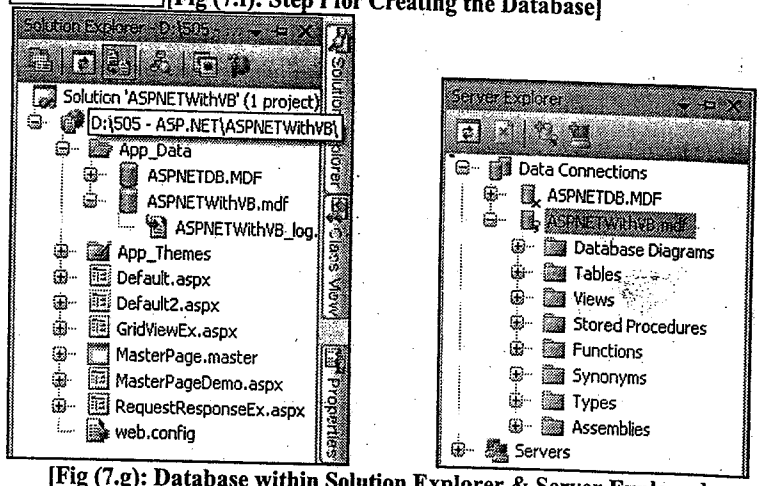
[Fig (7.d): Step a & b for Creating the Database]



[Fig (7.e): Step c, d & e for Creating the Database]

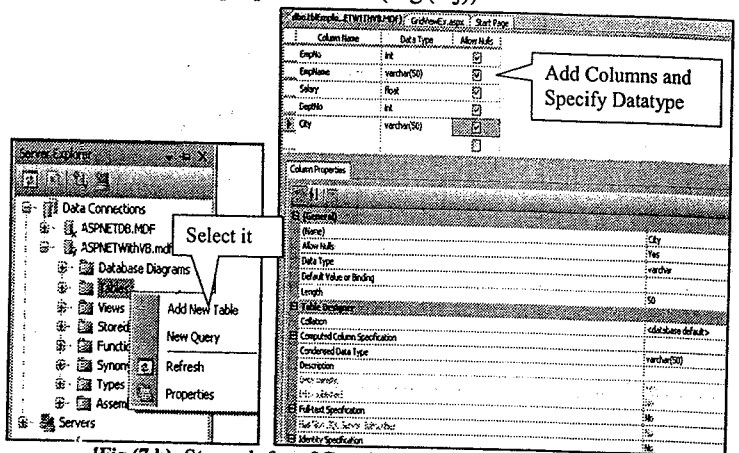


[Fig (7.f): Step f for Creating the Database]

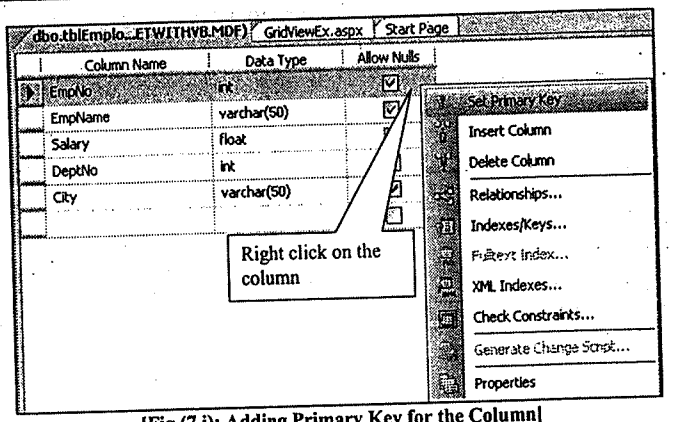


[Fig (7.g): Database within Solution Explorer & Server Explorer]

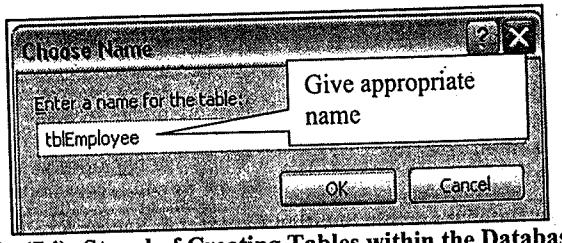
- **Creating Tables within the Database**
  a) Right click on the Tables (Fig (7.h))
  b) Select Add New Table, it will open the Table Creation Wizard.
  c) Add Columns and Set the constraints for it. (Fig (7.i))
  d) Save the Table with the appropriate name. (Fig (7.j))



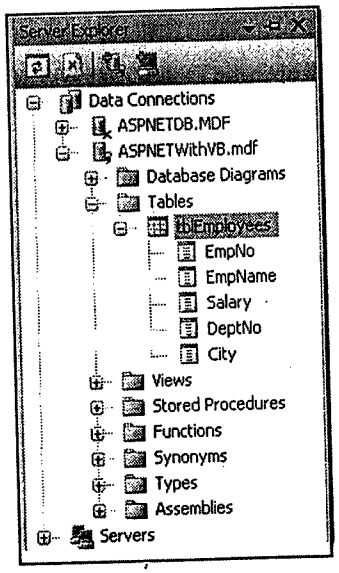
[Fig (7.h): Step a, b & c of Creating Tables within the Database]



[Fig (7.i): Adding Primary Key for the Column]



[Fig (7.j): Step d of Creating Tables within the Database]



[Fig (7.k): View of Tables and Columns in Database within Server Explorer]

7.3 Data Binding and Data Controls

Data Binding

It is used to bind the data from the database or any other data objects to the Data Controls. There are two types of Data Binding:

(1) Single Value Data Binding
(2) Repeated Value Data Binding

(1) Single Value Data Binding

In Single Value Data Binding only one value is bind to the Web Controls. It is implemented in two ways: <%#Expression %> and <%$Expression %>.

- **<%#Expression %>**

  Here expression can be Public, Private or Protected. It can be value of a property, member variable, return value of the function, and static value.

  Example

  Default.aspx (Source)

```
<asp:Label ID="Label2" runat="server" Text="<%#stnam %>"></asp:Label>
<img src="<%#ImagePath()%>" ID="image1" />
```

  Default.aspx.vb

```
Public stnam As String
Public Sub Page_Load()
    stnam = txtstname.Text
    Me.DataBind()
End Sub
Public Function ImagePath() As String
    Return "/images/logo.gif"
End Sub
```

  In #Expression, to evaluate the expression, it is mandatory to use DataBind() method of the Page.

- **<%$Expression %>**

  It is used to bind Application Setting and Connection Strings to the Web Controls. To create Application Settings, we can use Website Menu→ASP.NET configuration→. Application Tab →Create application setting link. After creating the Application Setting, it can be used with web controls.

  Example

```
<asp:Literal ID="Literal1" runat="server" Text="<% $ AppSettings: Test %>">
</asp:Literal>
```

  - Here the name of Application Setting is Test. Application Settings are name–value pair. So, wherever you use name of the Setting you will get its value. In this example the value which is stored in Test AppSetting is assign to the Text property of the Literal control.

  With $Expression we don't need to use DataBind() method, as it is evaluated by the

expression builder itself on each request.

Example

*<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString=" <%$ ConnectionStrings:DatabaseConnectionString %>" SelectCommand="SELECT * FROM [tblEmployee]"></asp:SqlDataSource>*

Here in this example the connection string names DatabaseConnectionString is bind with the SqlDataSource control.

(2) Repeated Value Data Binding

In Repeated Value Data Binding, list of information is associated with the controls. Controls that can be used with it are ListBox, DropDownList, CheckBoxList, RadioButtonList, BulletedList. The control properties that are used to bind the values are DataSource, DataTextField, DataValueField. The collections that are used to store the data are ArrayList, SorteList, DataReader, DataSet etc.

Example

```
Dim arlst As New ArrayList
arlst.Add("ABC")
arlst.Add("XYZ")
arlst.Add("MNO")
ListBox1.DataSource = arlst
```

Data Controls

There are many Data Controls available in ASP.NET which represents Data in different forms. Such Data Controls are GridView, ListBox, DataList, Repeater, FormView, DetailsView etc. The common events present with all Data Controls are as follows:

- ✓ **ItemCreated** – it gives you a way to customize the item-creation process.
- ✓ **ItemDataBound** – it gives you the ability to customize the items, but after the data is available for inspection.
- ✓ **EditCommand** – it puts the current row into edit mode by setting the EditItemIndex property.
- ✓ **DeleteCommand** – it puts the current row into the delete mode.
- ✓ **UpdateCommand** – it extract the contents of current row and update the data source, cancel the edit mode of the row, and rebind the updated data to the Data controls.
- ✓ **CancelCommand** – it Cancel edit mode of the row, and rebind the original data to the DataGrid control.

These events are not available with ListBox and Repeater controls as they are read only controls.

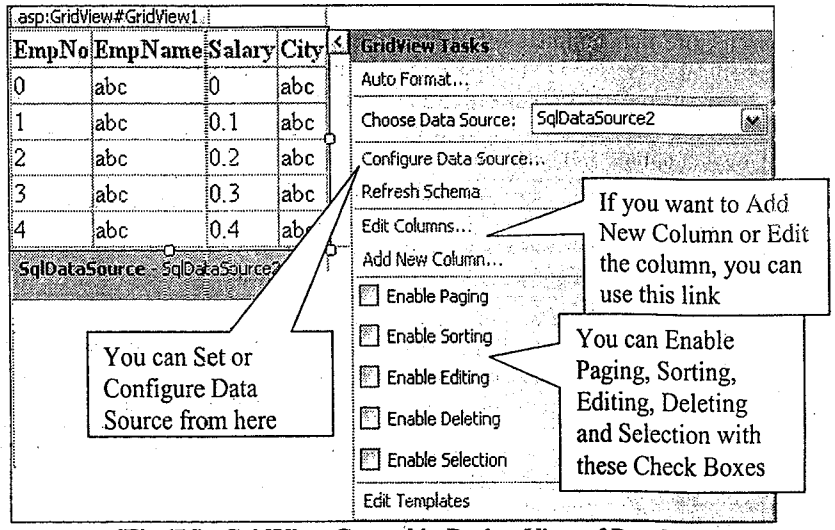
- **GridView**

  It is a fully featured, multicolumn, data-bound grid control. It must be bound to a data source via its DataSource property or it will not be rendered on the page. Typical data sources are the DataSet and DataReaders. You can also customize the layout of individual columns, by setting the column type to "templated" and modify the column's templates. It

can also be rendered without templates, so that it can be used for creating reports. It also supports selection, editing, deletion, paging, and sorting by column and button columns. Different types of columns in GridView Control are as follows:

Type of Column	Description
Bound column	It allows you to specify which data source field to display and also allows you to specify the format of that field.
Hyperlink column	It displays information as hyperlinks so that users can click to navigate to a separate page that provides details about that item.
Button column	It allows you to add a button for each item in the grid and define custom functionality for that button. For example, you might create a button labeled "Add to Shopping Cart" that runs your custom logic when a user clicks it. You can also add predefined buttons for Select, Edit, Update, Cancel, and Delete functions.
Edit, Update, Cancel column	It allows you to create in-place editing representation of the GridView.
Template column	It allows you to create combinations of HTML text and server controls to design a custom layout for a column. The controls within a template column can be data-bound and you can define the layout and functionality of the grid contents, because you have complete control over how the data is displayed and what happens when users interact with rows.

[Table:10 Types of Columns with GridView]



[Fig (7.1): GridView Control in Design View of Page]

- **ListBox**

It is used to create a single- or multi-selection drop-down list. Each item in a ListBox control is defined by a ListItem element. Properties that can be used to bind the data are DataSource, DataTextField, DataValueField.

Example

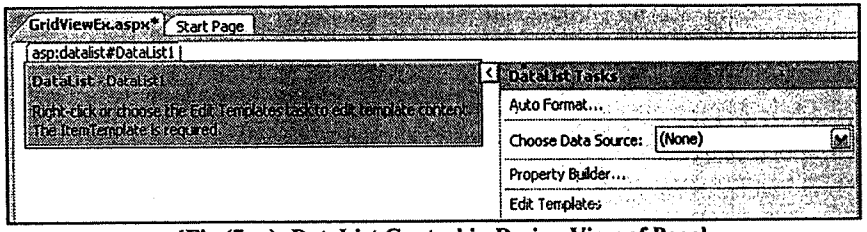
```
Imports System.Data.SqlClient
Imports System.Data

Public Function AllEmployees() As DataSet
    Dim constr As String = "<Constr>"
    Dim con As New SqlConnection (constr)
    Dim cmd As New SqlCommand("Select * From tblEmployees", con)
    Dim da As New SqlDataAdapter(cmd)
    Dim ds as New DataSet
    da.fill(Ds,"Emp")
    Return Ds
End Function
Public Sub Page_Load()
    Dim ds as new Dataset=AllEmployees()
    lstEmp.DataSource=ds.Tables("Emp")
    lstEmp.DataTextField="EmpName"
    lstEmp.DataValueField="EmpNo"
    Page.DataBind()
End Sub
```

- **DataList**

It is a feature-rich, templated, data-bound list. You can modify the templates to customize this control. It is useful to represent the data in repeating structure, such as a table. There are different properties for direction and layout in which the record will be displayed.

Layout option	Description
Flow layout	It renders the items inline, as in a word-processing document.
Table layout	It renders the items as in HTML table. This gives you more options for specifying the look of items.
Vertical and horizontal layout	By default, the items in a DataList control are displayed in a single vertical column. However, you can specify that the control contain more than one column. If so, you can further specify whether the items are ordered vertically or horizontal.
Number of columns	You can specify how many columns the list will have. This permits you to control the rendered width of the Web page, usually to avoid horizontal scrolling.

[Table:11 Layout Options with DataList]



[Fig (7.m): DataList Control in Design View of Page]

- **Repeater**

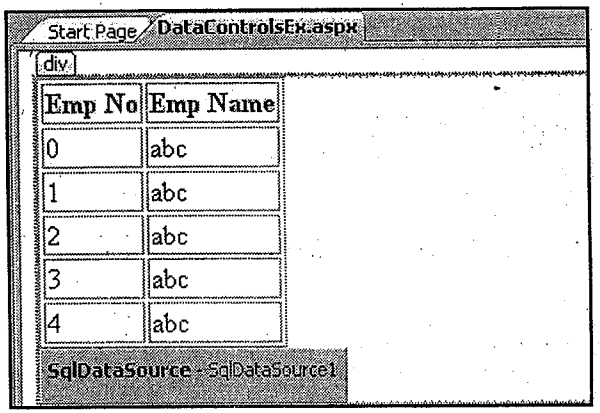
It is a templated, data-bound list. But it is "lookless" that means, it does not have any built-in layout or styles. Therefore, you must explicitly declare all HTML layout, formatting, and style tags in the control's templates.

Template property	Description
Item Template	It contains the HTML elements and controls to render once for each data item in the data source.
Alternating Item Template	It contains the HTML elements and controls to render once for every other data item in the data source. Mostly, it is used to create a different look for the alternating items.
Header Template and Footer Template	It contains the text and controls to render at the beginning and end of the list, respectively.
Separator Template	It contains the elements to render between each item.

[Table:12 Templates with Repeater]

Example

```
<asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1">
<HeaderTemplate>
    <table>
        <tr> <th> DeptNo</th> <th>DeptName</th>  </tr>
</HeaderTemplate>
<ItemTemplate>
    <tr><td><asp:Label runat="server" ID="did" Text='<%# Eval("DeptId") %>' />
    <td><asp:Label runat="server" ID="dnm" Text='<%# Eval("deptName")
    %>'/></td>
    </tr>
</ItemTemplate>
 <FooterTemplate>
    </table>
</FooterTemplate>
</asp:Repeater>
```

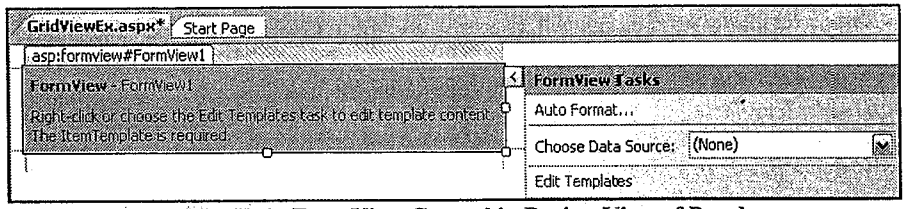


[Fig (7.n): Repeater Control in Design View of Page]

To display data within Repeater control, it is mandatory to use <ItemTemplate> within it. There are no properties that you can set and display the data. You need to write the code in the Source view to use Repeater control.

- **FormView**

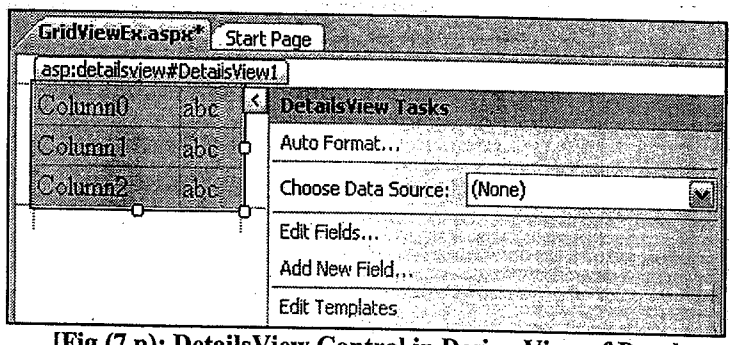
It is a data-bound user interface control that renders a single record at a time from its associated data source. It also provides paging buttons to navigate between records. The templates available are ItemTemplate, HeaderTemplate, FooterTemplate, EmptyDataTemplate, PagerTemplate, EditItemTemplate, InsertItemTemplate.
Like DataList control it does not provide the pre – defined layout for display and / or updating the data. Instead, when the page is rendered the FormView is also renders and the content of the tables are displayed.



[Fig (7.o): FormView Control in Design View of Page]

- **DetailsView**

It is a data – bound control that renders a single row at a time. It provides navigation, insertion, updation and deletion option. It is implemented through <table> tag of HTML.

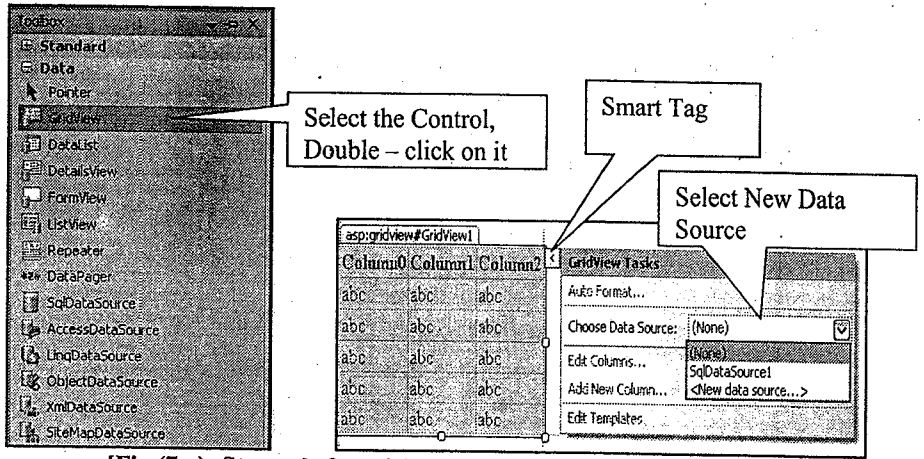


[Fig (7.p): DetailsView Control in Design View of Page]

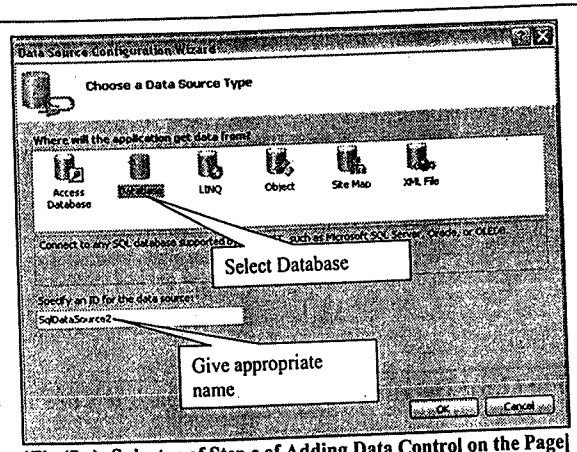
- **Adding Data Control on the page and Use it.**

The Data Controls are available within Data tab of the ToolBox.

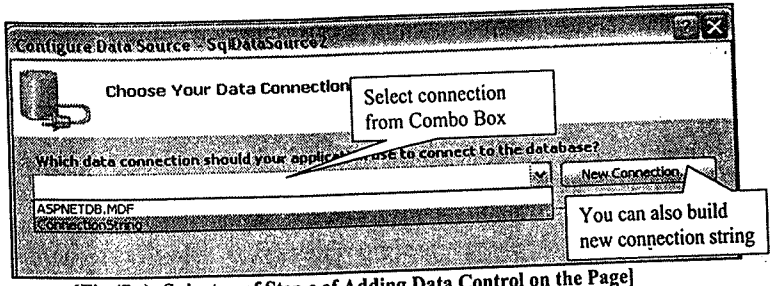
- ✓ Select the Appropriate Control from the Data Tab.
- ✓ Double – click or Drag Drop the Control on the Page.
- ✓ Set the Data Source from the Smart tag.



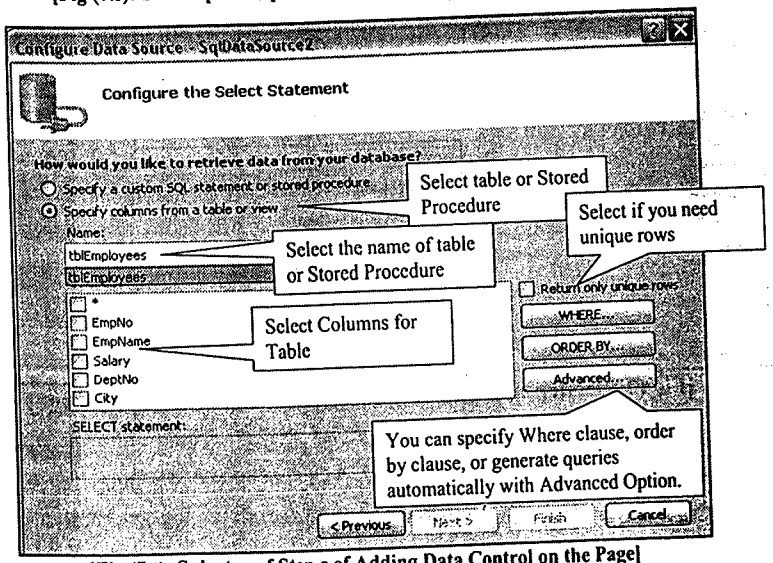
[Fig (7.q): Step a, b & c of Adding Data Control on the Page]



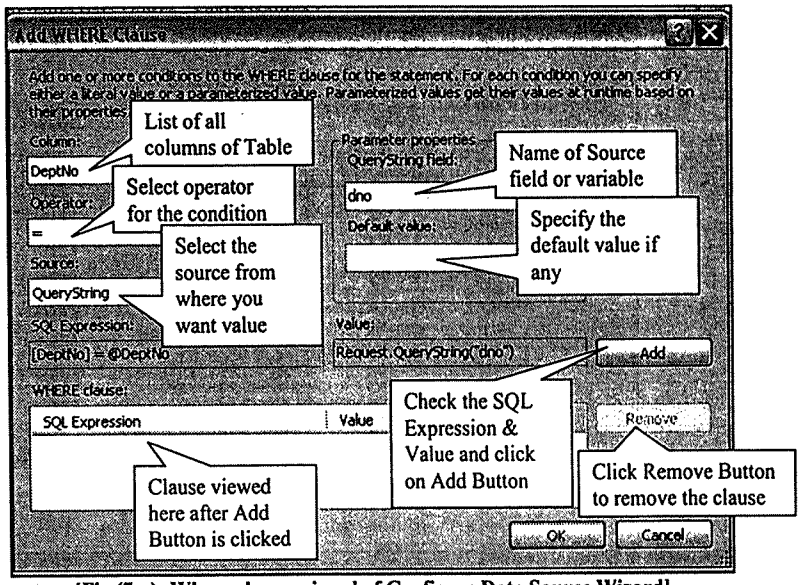
[Fig (7.r): Sub step of Step c of Adding Data Control on the Page]



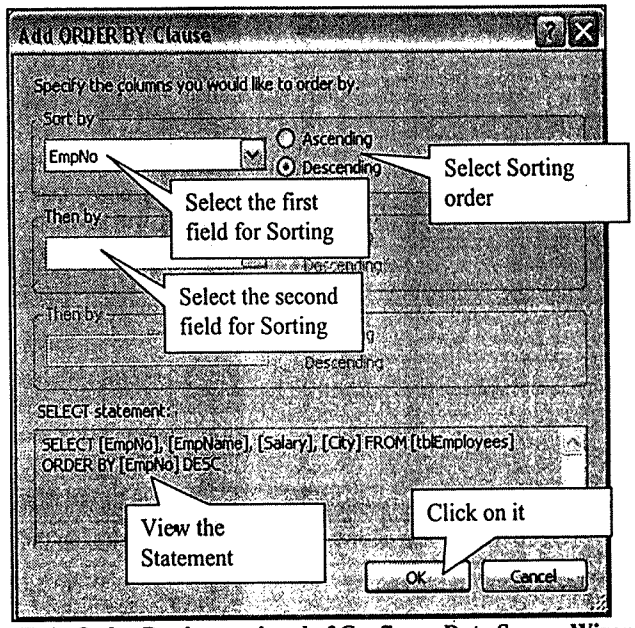
[Fig (7.s): Sub step of Step c of Adding Data Control on the Page]



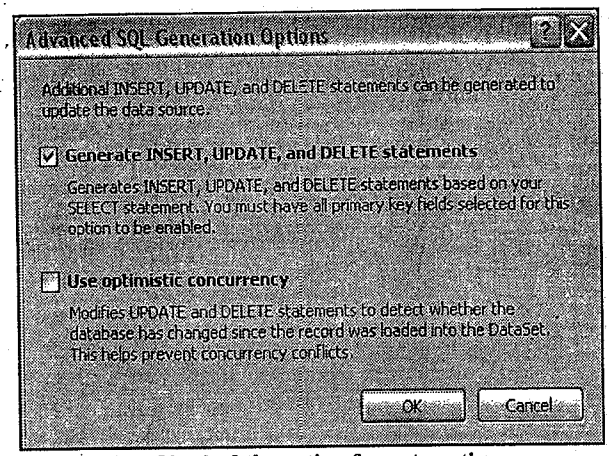
[Fig (7.t): Sub step of Step c of Adding Data Control on the Page]



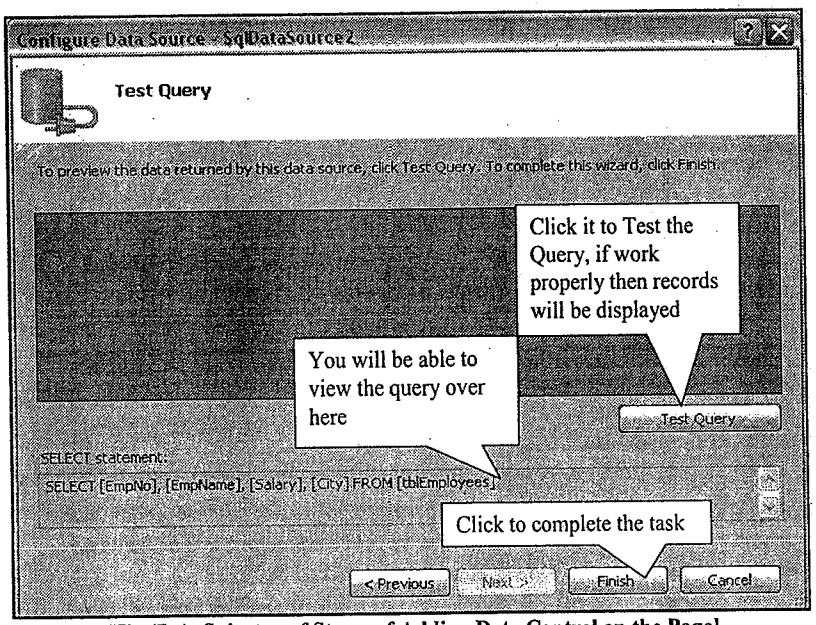
[Fig (7.u): Where clause wizard of Configure Data Source Wizard]



[Fig (7.v): Order By clause wizard of Configure Data Source Wizard]



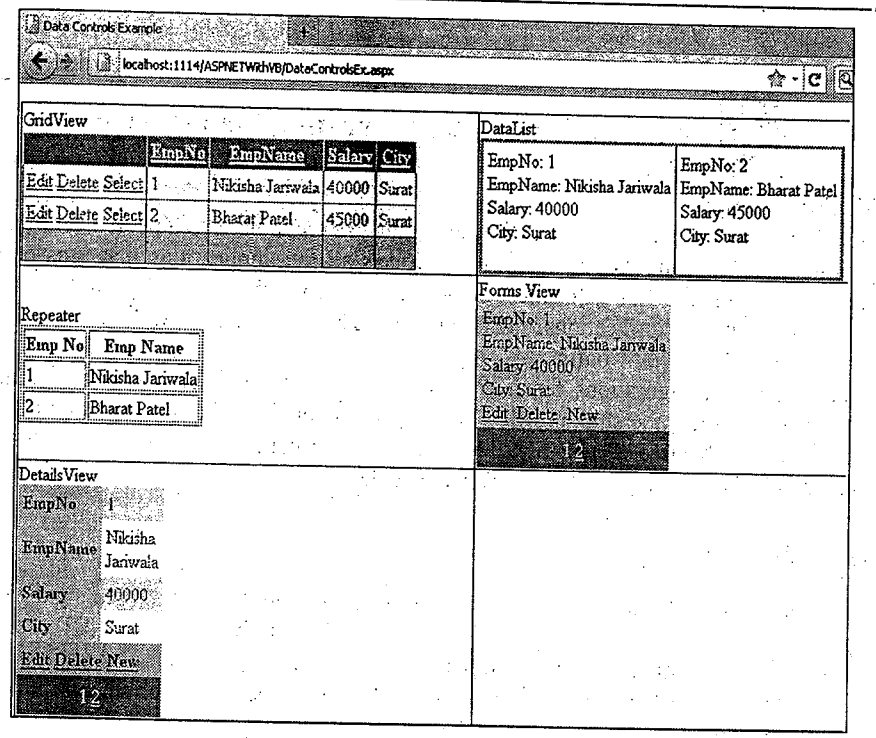
[Fig (7.w): Wizard to Checked the option for automatic query generation of Configure Data Source Wizard]



[Fig (7.x): Sub step of Step c of Adding Data Control on the Page]

Example

As the Steps specified in the "Adding Data Controls on the Page", in this example you will be able to see the various data controls on the page.



[Fig (7.y): Various Data Controls on the Page]

Exercise-7

1. What is ADO.NET?
2. Why ADO.NET came into existence?
3. Write a note on ADO.NET Architecture?
4. What is Data Binding?
5. Write a detailed note on Data Binding.
6. Difference between #Expression and $Expression Data Binding.
7. List out various Data Controls and their common events.
8. Explain Repeater control in detail.
9. Explain the steps that are used to Add Data Controls on the Page and Binding it with Data Source.

8 Web Application Configuration Management

8.1 Machine.Config

It is used to configure all the application according to a particular machine. It means that, configuration done in machine.config file affects all the applications that runs on a particular machine. Usually, this file is not altered and only web.config (Topic 8.2) is used which configures applications. This file resides in c:\Windows\Microsoft.Net\Framework\v2.0.50727\Config.

It defines the configurations for ASP.NET processes and register providers for the features such as membership, profiles, security and so on. Web.Config file inherits the settings from Machine.Config file. To improve performance, machine.config should contain only those settings that differ from their defaults.

Element

<machineKey>

It allows you to set the server- specific key used for encrypting data and creating digital signatures. You can use encryption with several ASP.NET features. ASP.NET uses it automatically to protect the forms authentication cookie and you can also apply it to protected view state data. The key is also used for out-of- process session state providers for authentication.

Example

```
<machineKey validationKey="AutoGenerate" decryptionKey=" AutoGenerate"
validation="SHA1"/>
```

<processModel>

It allows you to configure how the ASP.NET worker process recycles application domains, and the Windows account it executes under, which determines its privileges.

Example

```
<processModel autoConfig="true"/>
```

8.2 Web.Config

It is a configuration file for the ASP.NET web application. An ASP.NET application has one web.config file which keeps the configurations required for the corresponding application. But, if the website contains multiple folders then each folder can have separate Web.config file. This file is written in XML with specific tags having specific meanings.

It acts as a central location for storing the information to be accessed by web pages. This information could be a Connection String stored at a centralized location so that it can be accessed in a data-driven page. If the connection string changes then it is to be changed at one place.

It can also store other information such as Session states, Error handling, Security issues,

and global information for the pages etc. It contains information in the form of Key – value pair.

Example

```
<?xml version="1.0"?>
  <configuration>
      <appSettings/>
      <connectionStrings>
  <add name="condb" providerName="System.Data.SqlClient"
  connectionString="server=localhost;uid=abc;pwd=;database=CompanyDB"/>
      </connectionStrings>
  </configuration>
```

In classic ASP such global information was typically stored as an application variable.

We can have multiple Web.config file, but it must be in separate folder.

Elements

<appSettings>

It allows you to add the custom settings for the application in the web.config file.

Example

```
<?xml version="1.0"?>
  <configuration>
      <appSettings>
          <add name="SiteName" value="ASPNETWithVB"/>
      </appSettings>
  </configuration>
```

It is very easy to access the information stored in <appSettings> element. There is ConfigurationManager class in System.Configuration namespace which provides the properties to access this information.

Example

```
Dim ASettings As String = ConfigurationManager.AppSettings("SiteName")
```

<system.web>

It contains the entire configuration specific to the ASP.NET. It contains different types of elements such as authentication, authorization, compilation, customErrors, identity, pages, sessionState, trace and so on. These elements centralize the configuration of the ASP.NET application.

Elements	Usage
authentication	It determines how you will verify the User's identity when the page is requested.
authorization	It controls which Users have access to the resources of the application.
Compilation	It contains the assemblies that are present in the GAC (Global Assembly Cache) which is used by your application.
customErrors	It allows you to specify the URL to which it is redirected when the error occurs.
pages	It defines the page settings such as theme and so on.
sessionState	It allows you to maintain various information for the user session.
trace	It allows you to display the diagnostic (tracing) information.

[Table.1: Elements within <system.web>]

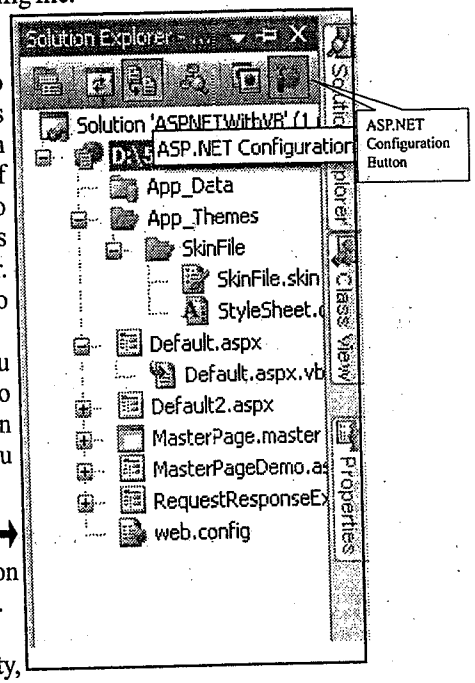
- **Accessing information of web.config**

ASP.NET provides WebConfigurationManager class in the System.Web.Configuration namespace that contains different types of properties and methods through which you can easily access the information of web.config file.

8.3 Web Site Administration Tool (WAT)

WebConfigurationManager is used to configure ASP.NET application. It is designed to allow developers to build custom configuration tools that simplify the work of configuring web applications. ASP.NET also includes a graphical configuration tool that is based on the WebConfigurationManager. This tool is called WAT. It allows you to configure parts of Web.Config file.

To open WAT, Select Website Menu → ASP.NET Configuration or you can also open WAT from Solution Explorer. In browser, it will open WAT through which you can configure Web.Config file.



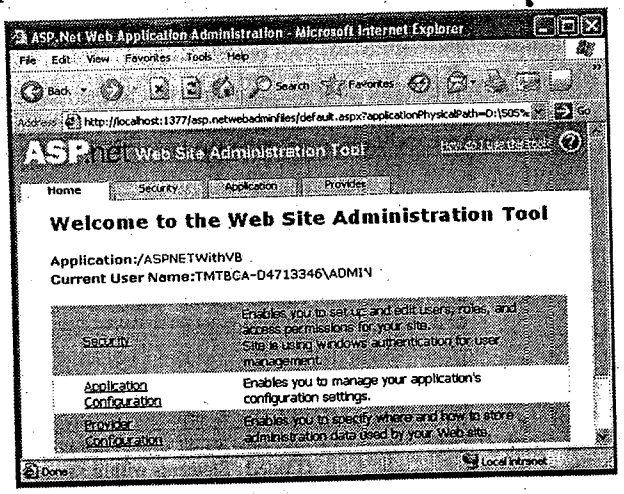
[Fig (8.a) Running WAT from Solution Explorer] →

You can click on the ASP.NET Configuration Tool button of the Solution Explorer (fig. (8.a)).

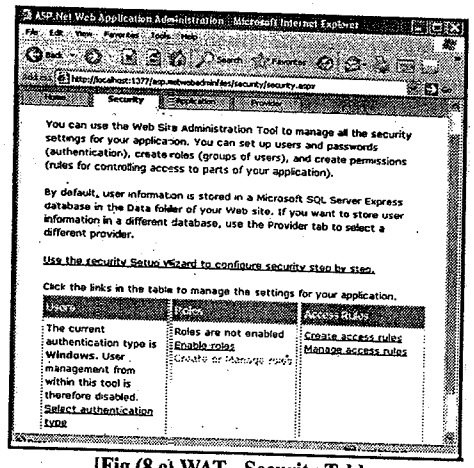
It contains four Tabs (fig. (8.a)): Home, Security, Application, and Provider. The Web.Config file is written in XML, so it is difficult to modify. But each tab of WAT provides the GUI through which you can easily maintain Web.Config file.

Home tab (fig.8.b) allows you to navigate to other tabs through links. It also contains description of each tab.

Security Tab (Fig (8.c)) allows you to create different Users and Roles through which you can provide authentication and authorization for the application.

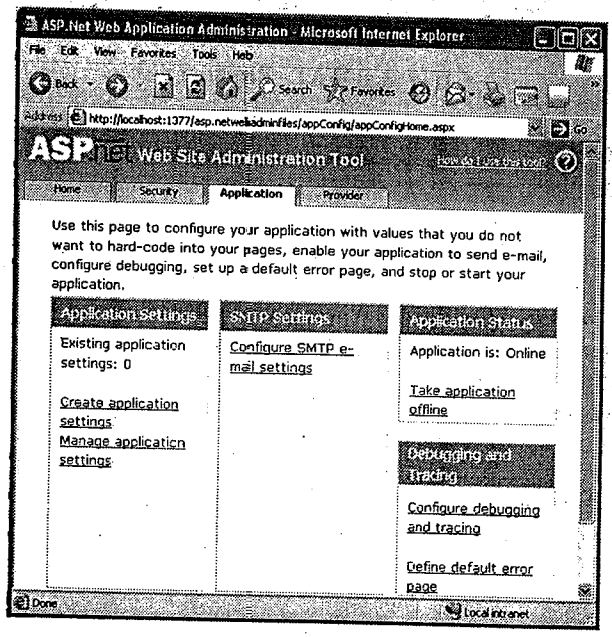


[Fig (8.b) WAT] →



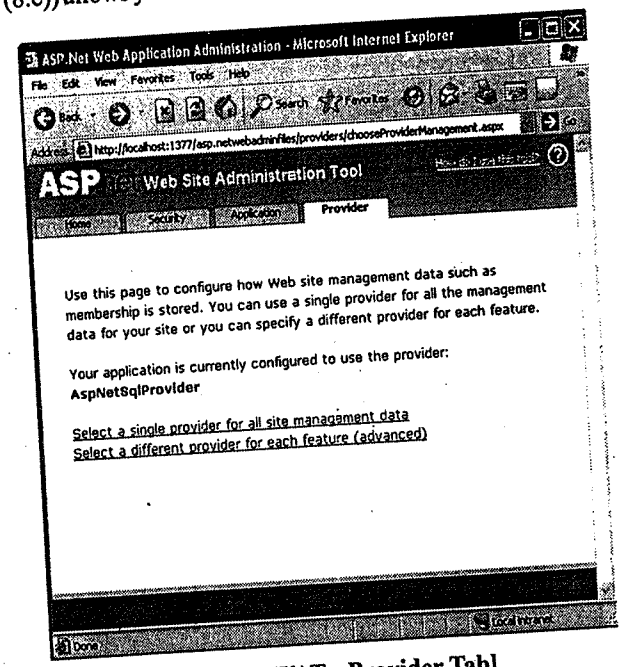
[Fig (8.c) WAT - Security Tab]

Application tab (Fig (8.d)) allows you to Manage Application Settings, SMTP settings and also allow you to configure debugging and tracing the application. Along with, it allows you to set the default error page which will be displayed to the user if any error occurs in the application.



[Fig (8.d) WAT - Application Tab]

Provider tab (Fig (8.e)) allows you to configure the database provider for the application.



[Fig (8.e) WAT - Provider Tab]

Exercise - 8

1. Which language is used to write Machine.Config and Web.Config file?
2. Is it possible to have multiple Web.Config files in the Website?
3. What is Machine.Config file? Where does it reside?
4. Is it possible to have multiple Machine.Config files?
5. What does Web.Config file contain?
6. Which is the GUI tool through which you can configure Web.Config file?
7. How to open WAT for the application?
8. What is the significance of Security tab of WAT?
9. Which tab allows you to configure the debugging and tracing facility for application?

9 State Management Techniques

9.1 Client Side State Management Techniques

In this technique, the user state information is stored on the client side and not on the server, so it's called Client Side State Management Technique.

9.1.1 Cookies

The small text stored on client machine by web application is called Cookies. It is stored as text file or in browser memory. This text is sent to web application with each subsequent request by browser. For example, user can define the back color of their home page. So the page is render with selected color and it is given to the client. If we store data on client machine, we need to pass it with each request. So data size depends on browser capability. So, it is useful to maintain small user information specific to pages.

- **Creating Cookies**

To create the cookie, you need to create the object of HttpCookie class. With the help of this object, you can add the name – value pair within it. After the cookie is created, you have to add the object of the cookie within the Cookies collection of the Response object, so that it can be retrieve.

Example

Write the following code on btnCreateCookie_Click event of CreateCookie.aspx page.

```
Protected Sub btnCreateCookie_Click(ByVal sender As Object, ByVal e As System.
EventArgs) Handles btnCreateCookie.Click
    If Not Request.Cookies("color") Is System.DBNull.Value Then
        Dim c As New HttpCookie("color")
        c.Values.Add("FC", "Blue")
        c.Values.Add("BC", "Yellow")
        Response.Cookies.Add(c)
        Response.Redirect("RetrieveCookie.aspx")
    End If
End Sub
```

- **Retrieving Cookies**

To retrieve the cookie created on CreateCookie.aspx page, you first need to create the reference of the HttpCookie class and then within it you need to assign the Cookie object which you can retrieve with the help of Request object.

Example

Write the following code on btnRetrieveCookie_Click event of RetireveCookie.aspx page.

```
Protected Sub btnRetrieveCookie_Click(ByVal sender As Object, ByVal e As System.
```

```
EventArgs) Handles btnRetrieveCookie.Click
    Dim cookie As HttpCookie = Request.Cookies("color")
    If Not cookie Is System.DBNull.Value Then
        Dim f, b As New Color
        f = Color.FromName (cookie.Values("FC").ToString)
        b = Color.FromName (cookie.Values("BC").ToString)
        lblColor.BackColor = b
        lblColor.ForeColor = f
    End If
End Sub
```

Types of Cookies

There are two types of Cookies:

(1) Persistent Cookie
(2) Non – Persistent Cookie

(1) Persistent Cookie

The Cookie which is stored in the Text file i.e. stored physically on disk is called Persistent Cookie. It is slower than Non – Persistent Cookie. To create Persistent Cookie, you need to set expires property of the Cookie object of HttpCookie class.

Example

```
Cookie.expires=datetime.now.adddays(1) 'Cookie will be persistent for 1 day
        or
Cookie.expires=datetime.maxvalue 'Cookie will remain persistent for life time.

Cookie.expires=datetime.now.adddays(-1) 'it will remove the cookie.
```

(2) Non – Persistent Cookie

The cookie which is stored in the browser memory i.e. stored in primary memory is called Non – Persistent Cookie. If you don't specify expires property then Non – Persistent Cookie is created automatically.

Advantages

- ✓ You are able to configure expiration time.
- ✓ No server resources are required, so there will be no load on server.
- ✓ It is simple to create and use the cookie.
- ✓ By using Persistent cookie the data will remain persistence for the user.

Disadvantages

- ✓ You can store small amount of information.
- ✓ There is security risk for the critical information because the information is stored on client machine.

9.1.2 QueryString

It is the portion of the URL after the question mark. Mostly, it is made up of name – value pair. If you want to store the information in query string you need to place it by yourself. There is no collection to store query string directly. You have to build query string as:

Website path?varname=value&varname=value.....

Example

http://localhost:1114/ASPNETWithVB/QueryStringEx.aspx?q=computer

Here it defines a single variable q, which contains the value computer.

The advantage of query string is that it's light – weight and doesn't apply any kind of burden on the server.

- **To use Query String in ASP.NET you have to write it as:**

```
Dim rec As Integer = 1
Response.Redirect("QueryStringEx.aspx?no=" & rec.ToString())
```

- **You can also send multiple parameters as long as you separate them with an ampersand (&) sign.**

```
Dim rec As Integer = 1
Dim nm As String = "NikishaJariwala"
Response.Redirect("QueryStringEx.aspx?no=" & rec.ToString() & "&name=" & nm)
```

- The QueryString which you will create is stored in the QueryString Collection of the Request object automatically.

- **To access the Query String collection of Request object you have to specify:**

Write the following code on Page_Load event of QueryStringEx.aspx page.

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
    Response.Write(Request.QueryString("no"))
    Response.Write(Request.QueryString("name"))
End Sub
```

Information is always retrieved as a string, which can then be converted to another simple data type. Values in QueryString collection are indexed by the variable name.

URL Encoding

The problem with the Query String is the characters that are not allowed in a URL.

All characters must be alphanumeric or one of small set of special character including $(dollar), -(Hyphen), _(underscore), .(dot), +(plus), !(exclamation), *(asterisk), '(single quote), ( )(parenthesis). Other characters are & (ampersand) which is used to separate multiple query string parameters, +(plus) is alternate to represent a space,(# (hash) number sign is used to point to a specific bookmark in a web page.it can also be represented as character sequence %20)or + sign.

You can use the HttpServerUtility class to encode your data automatically.

Example

```
Dim pnam as string="flying carpet"
Response.Redirect("UrlEncodeEx.aspx?pname="&Server.UrlEncode(pnam))
```

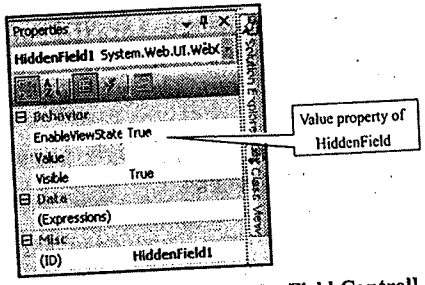
On UrlEncodeEx.aspx

```
Response.Write(Server.UrlDecode (Request.QueryString(pname))
```

You can use HttpServerUtility.UrlDecode() method to return a URL – encoded string to its original value.

9.1.3 HiddenField

It is used to store the data on the page. Hidden Field can store single variable value at a time. It is not rendered by the browser. In ASP.NET, it is in the form of control. So that you can set its properties. The important property on HiddenField is Value (fig. (9.a)).



[Fig (9,a) Properties of HiddenField Control]

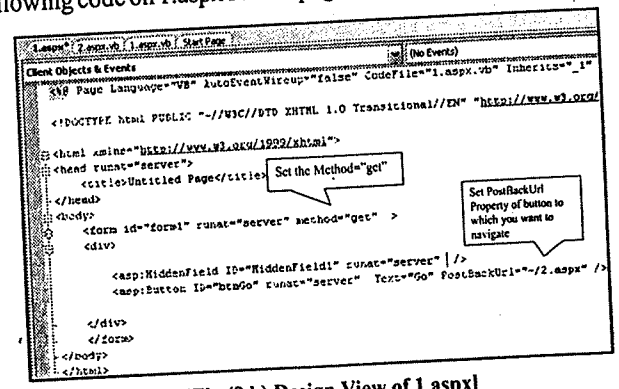
If you want to use the value of HiddenField for the single page then it is written as:

Example

```
HiddenField1.Value = 10          'to store the value in HiddenField

Dim No As New Integer
No=HiddenField1.Value            'to Retrieve the value of HiddenField
```

- **Passing the value from one page to another using HiddenField**

  Write the following code on 1.aspx source page.



[Fig (9.b) Design View of 1.aspx]

As you specify "get" method in the form of 1.aspx, so that all the values of the controls are passed in the URL. So that it can be retrieved through QueryString.

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
    HiddenField1.Value = 10
End Sub
```

Write the following code on Page_Load event of 2.aspx page.

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
    Response.Write(Request.QueryString("HiddenField1"))
End Sub
```

Advantages

- ✓ No server resource required to use HiddenField.
- ✓ It is simple to implement.

Disadvantages

- ✓ It can store small amount of data.
- ✓ There are security issues as the values are stored at client side.

9.1.4 ViewState

It is used to maintain the state for single ASP.NET page. Each item is indexed with unique string

- **To store the value in ViewState:**

```
ViewState("id")=1
```

- **Retrieving the value from ViewState:**

```
Dim Sid As Integer
If Not ViewState("id") is System.DBNull.Value then
        Sid=Convert.toInt32(ViewState("id"))
End if
```

ASP.NET engine create separate hidden fields and store its value in encoded base64 form.

Advantages

- ✓ No server resource required to use ViewState.
- ✓ It is simple to implement.
- ✓ Enhanced security feature is used.

  MAC encoding (tampered proof) technique is embedded with data, if following attribute is set to true.

```
<%@ Page enableViewStateMac="true"%>
```

Disadvantages

- ✓ With each request, encryption and decryption is required.
- ✓ Performance is degraded, as during page loading encryption and decryption takes place.
- ✓ ViewState values are implicitly stored in hidden fields and it might be some devices does not support hidden field. Therefore on such devices ViewState will not work.

9.2 Server Side State Management Techniques

In this technique, the user state information is stored on the Server, so it's called Server Side State Management Technique.

9.2.1 Session

In general, the time period for which the user interacts with the web application is called the user session. In ASP.NET, Session is the instance of HttpSessionState Class. It allows you to save the session information for each user of the web application separately. It stores the information in the form of object. The unique identifier or Session ID is given to each session which is generated by the server.

There are two events related to the Session in Global Application Class (Global.asax): (1)Session_Start (2) Session_End (see Topic 3.7 of this book)

Properties

Property	Description
SessionID	Get the session ID of a user's current session. It is read only.
IsNewSession	Get the value that indicates whether the session is new or not.
IsCookieLess	Get the value that indicates whether the SessionID is embedded in the URL or stored within the Cookie.
IsReadOnly	Get the value that indicates whether the session is read only or not.
TimeOut	Get / set the value in minutes which indicate the time after which the session is terminated if user is idle.
Count	Get the number of items within the Session

[Table.1: Properties of Session]

Methods

Method	Description
Clear()	Removes all Key – Value pairs from Session State
Abandon()	Kill / Cancel the Session.
Remove()	Remove the item from the Session State whose name / key is provided
RemoveAll()	Remove all the items from the Session State.
RemoveAt()	Remove the item from the Session State whose index is provided

[Table.2: Methods of Session]

- **Store values in the Session**

General Syntax:

Session ("Key") =Value

Example

```
Protected Sub btnCreateSession_Click(ByVal sender As Object, ByVal e As System.
EventArgs) Handles btnCreateSession.Click
    Session ("No") = txtNo.Text
    Session ("Name") =txtName.Text
End Sub
```

It works as a collection.

- **Retrieve values from Session**

Example

```
Protected Sub btnRetrieveSession_Click(ByVal sender As Object, ByVal e As System.
EventArgs) Handles btnRetrieveSession.Click
Dim id As Integer = Convert.toInt32 (Session ("No"))
Dim nm As String = Session ("Name").toString()
End Sub
```

- **<SessionState>**

It is the element of Web.Config file (See Topic 8.2 of this book). It allows you to maintain the several information of the user session.

Example

```
<SessionState mode="Inproc" SqlConnectionString= data source=127.0.0.1; user
id=sa" cookieless="use cookies" timeout="20"/>
```

Attributes

Attributes	Description
Mode	It specifies the storage location of the session data. Its values are: ✓ *InProc* – stored in process memory (IIS process memory) ✓ *StateServer* –it specifies that it if application restart, the data is to be persistent in memory. Separate process is created to store session data. ✓ *SqlServer* – Table is used to store the session data as large number of users are requesting simultaneously. ✓ *Off*- if session code is not written then also ASP.NET will create own session. So if you don't want to use session state management technique then it can be set to off. ✓ *Custom* – it stores data in other data store, file system, database. You need to provide provider code.
TimeOut	It specifies the lifetime of the session. By default, if session is idle for 20 minutes then the session is destroyed automatically.

Cookieless	It specifies whether the session data is to be stored in Cookie or in URL. Its values are: ✓ *UseCookies* – it stores the session data into the cookies. ✓ *UseUri* – it embed the session data in URL. It is embedded before virtual directory in the URL. General format of the URL with session id is: Protocol – servername – portno – sessionid – virtual directory – page This type of URL is called modified URL. ✓ *AutoDetect* – if cookie is supported then it will use it, otherwise use URL.
SqlConnectionString	If the mode is set to SqlServer, then you can specify the connection string within this attribute.

[Table.3: Methods of Session]

Session is automatically ended, if user closes the browser or TimeOut property is reached or Abandon () method is called.

Advantages

- ✓ It is simple to implement.
- ✓ It provides the feature that session data can be stored in database, so the session data is persistent.
- ✓ Cookieless implementation is also supported, so you can store crucial data.
- ✓ Global Application Class contains Session specific events, so you can manage various tasks at the creation and termination of the session.

Disadvantages

- ✓ If session data is embedded in the URL then you don't get original URL.
- ✓ Session ID visible in URL.

9.2.2 Application

It is the single data repository which is available to all the users. It is used to store small amount of frequently used data because the application data continuously occupy memory until application terminate.The information that is global to the web application is stored in Application. In ASP.NET, Application is the instance of HttpApplicationState Class. There are two events related to the Session in Global Application Class (Global.asax): (1) Application_Start (2) Application_End (3) Application_Error (see Topic 3.7 of this book)

Properties

Property	Description
Count	Get the number of items within the Application.
Item	Get the value of the specified index

[Table.4: Properties of Application]

Methods

Method	Description
Clear()	Removes all Key–Values pair from Application State
Remove()	Remove the item from the Application State whose name / key is provided
RemoveAll()	Remove all the items from the Application State.
RemoveAt()	Remove the item from the Application State whose index is provided
Lock()	It locks the Application object to synchronize the access i.e. for concurrency.
UnLock()	It unlocks the Application object that was lock previously.

[Table.5: Methods of Application]

- **Store values in the Application**

General Syntax:

Application ("Key") =Value

Example

```
Protected Sub btnCreateApplication_Click(ByVal sender As Object, ByVal e As System.
EventArgs) Handles btnCreateApplication.Click
    Application.Lock()
    Application ("Count") = Application ("Count") + 1
    Application.UnLock()
End Sub
```

It works as a collection.

- **Retrieve values from Application**

Example

```
Protected Sub btnRetrieveApplication_Click(ByVal sender As Object, ByVal e As
System. EventArgs) Handles btnRetrieveApplication.Click
Response.Write (Convert.toInt32 (Application ("Count"))
End Sub
```

9.2.3 User Profile

There is a need to store user-specific data that is accessible site-wide. The Profile feature is used to define user-specific data as well storing and retrieving user data. A user's profile is a collection of properties that define information you want to store for your web site's users. The user's profile is defined using XML syntax in a configuration file (machine.config and/or web.config). On the Page. it is reference with the help of Profile property of the page.

ASP.NET automatically generates a class that is accessible from the Profile property on a page by reading the pre - defined schema in configuration. You can access properties of the Profile, in the same way as you access properties of any other class.

Although, the most common use of the Profile feature is storing data for authenticated users. The Profile feature also supports storing information for anonymous users. Storing profile information for anonymous users depends on the Anonymous Identification feature. In addition to the Profile property, the Profile feature provides support for administration of

profiles (both for authenticated users and anonymous users) with the ProfileManager. ProfileManager allows you to search the information about all profiles, determines the number of profiles that is not modified for a long period of time, and deletes individual profiles based on modification date.

Example

Defining Profile properties in Web.Config file.

```
<system.web>
<anonymousIdentification enabled="true"/>
<profile >
   <properties>
         <add name="Name" allowAnonymous="true" />
         <add name="Age" allowAnonymous="true" type="System.Int16"/>
   </properties>
</profile>
<compilation debug="true"/>
</system.web>
```

Write the following code on UpdProfileButton _Click and Page_Load events of ProfileEx.aspx page.

```
Protected Sub UpdProfileButton _Click(ByVal sender As Object, ByVal e As System.
EventArgs) Handles UpdProfileButton.Click
Profile.Name = txtname.Text                'Storing values within the Profile
   Profile.Age = Int16.Parse(txtage.Text)   'properties.
End Sub
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System. EventArgs)
Handles Me.Load

If Not IsPostBack Then
            txtname.Text = Profile.Name      'Retrieving values from the Profile
            txtage.Text = Profile.Age.ToString()
 End if
End Sub
```

The Unidentified User

By default, Profile properties are only available for authenticated users. But if you want a property to be available for an anonymous user, we need to add allowAnonymous ="true" to the property.

Without this attribute, the runtime will throw an exception if the current user is anonymous. In order for allowAnonymous to work, you need to configure ASP.NET to track anonymous users. You can configure tracking of anonymous user using the anonymousIdentification section of web.config.

By default, Anonymous user tracking is off , but enabled="true" will tell the ASP.NET runtime to load an AnonymousIdentificationModule into the HttpModule request pipeline.

Profile Provider

Profile management feature is implemented with the help of Provider Model. A base ProfileProvider class defines the interface or contract, that all profile providers must implement. SqlProfileProvider class is used as a ProfileProvider that uses SQL Server. ProfileProvider method such as FindProfilesByUserName() is implemented by executing stored procedures in SQL Server.

Grouping

You can create a hierarchy of profile properties using the group element. Grouping allows you to categorize properties, which can be helpful when there are a large number of profile properties. You can specify more than one property group, but you cannot nest a group under another group element, because the compiler generates a class with strongly typed profile properties.

Example

```
<profile>
 <properties>
  <add name="Name" allowAnonymous="true" />
  <add name="Age" allowAnonymous="true" type="System.Int16"/>
   <group name="UI">
     <add name="MasterPage" defaultValue="layout1.master"/>
     <add name="Theme" defaultValue="Red"/>
   </group>
 </properties>
</profile>
```

```
Protected Sub UpdProfileButton _Click(ByVal sender As Object, ByVal e As System.
EventArgs) Handles UpdProfileButton.Click
Profile.Name = txtname.Text                'Storing values within the Profile
   Profile.Age = Int16.Parse(txtage.Text)   'properties.
Profile.UI.Theme = "Reddish"
Profile.UI.MasterPage = "layout1.master"
End Sub
```

Profile Configuration

The machine.config file configures the default Profile provider for all web sites on the computer. The machine.config file is located in the framework installation config directory, typically Windows\Microsoft.NET\Framework\v2.0.xxxx\Config.

Example

Write the following code in web.config file.

```
<profile>
<providers>
  <add name="AspNetSqlProfileProvider"
      connectionStringName="LocalSqlServer"
      applicationName="/"
      type="System.Web.Profile.SqlProfileProvider />
 </providers>
</profile>
<configuration>
<connectionStrings>
 <remove name="LocalSqlServer"/>
 <add name="LocalSqlServer" connectionString="server=global;
database=aspnetdb; integrated security=sspi;" />
 </connectionStrings>
 ...
</configuration>
```

But you can configure it in your Web.Config File.

```
<profile defaultprovider="MyProfileProvider">
<providers>
<add name="MyProfileProvider" connectionStringName="MySqlServer"
applicationName="/"  type="System.Web.Profile.SqlProfileProvider" />
 </providers>
 </profile>
```

9.2.4 Caching

Caching means storing data and pages in the server memory, so that subsequent request may be satisfied from the cache if certain conditions are met. It is similar to Application (Topic 9.2.2 of this book) but cache items can be discarded automatically from memory when server is low, the items in cache are expire or items dependency changes. The scope of Cache data and Application is same.

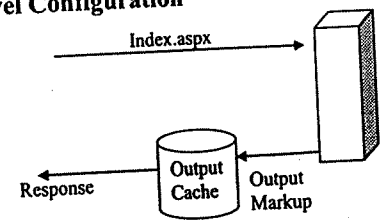
There are two types of Caching: (1)Output Caching (2)Data Caching

(1) Output Caching

To implement output caching the configuration can be done at two levels:

a) Page Level Configuration b) Application Level Configuration

a) Page Level Configuration



[Fig (9.c) Serving Page from Output Cache]

@OutputCache directive is used to implement Output Caching at page level after the @Page directive.

Example

```
<%@OutputCache Duration="15" VaryByParam="none"%>
```

Here the active cache duration is specified which indicates that for specified duration the page will remain cached. It is in seconds.

```
<%@OutpuCache Duration="10" VaryByParam="StudID"%>
```

Here the VaryByParam value is given, so caching is dependent on this parameter. It searches this parameter in the QueryString (Topic 9.1.2 of this book). If its value is different than new page is served otherwise it serves the page from the Output cache.

b) Application Level Configuration

It is written in web.config file.

Example

```
<system.web>
  <caching >
   <outputCacheSettings >
    <outputCacheProfiles>
     <add name="c1" enabled="true" duration ="10"/>
    </outputCacheProfiles>
   </outputCacheSettings>
  </caching>
</system.web>
```

Write the following code after @page directive in source view of the page that you want to cache:

```
<%@OutputCache CacheProfile="c1" VaryByParam="none"%>
```

Write the following code on code view of the page that you want to cache:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System. EventArgs)
Handles Me.Load
      lblTime.Text = "Now="+ System.DateTime.Now
End Sub
```

Then execute the page and refresh it for some time, you see the difference between the cached page and page without caching.

Advantages

- ✓ It reduces the bandwidth usage.
- ✓ It also reduces server load.
- ✓ It provides better response time.

Disadvantages

- ✓ If the load on server increases,the cached page can be discarded automatically before the specified time.
- ✓ The dynamic nature of the page is lost, because the request is served from the cache. So, the page which is saved within the cache is served and not the new one until the active cache duration is reached.

(1) Data Caching

The data which require extensive resources for its creation can be stored in the cache. So, whenever required, it can be served from it. This is called Data Caching.

- **Storing data in cache**

```
Cache("Name") = "Nikisha Jariwala"
    'Or
Cache.Insert("Name", "Nikisha Jariwala")
```

- **Retrieving data from cache**

```
Dim nm As String
nm=Cache("Name").toString()
```

- **Removing data from cache**

```
Cache.Remove("Name")
```

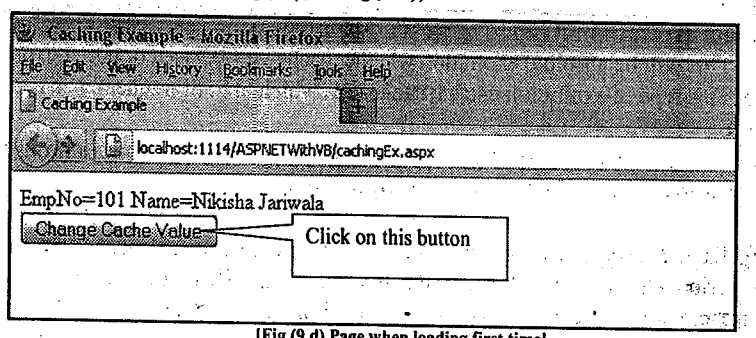
Data Caching is dependent on other cache items or on files. So, depending on it you can recreate the cache items or discard it.

Example

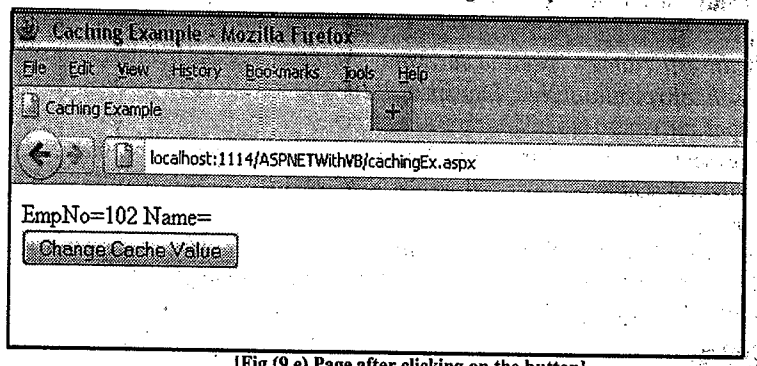
Dependency on other cache item:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load

    If Not IsPostBack Then
        Cache("EmpNo") = "101"
        Dim dependencies() As String = {"EmpNo"}
        Cache.Insert("Name", "Nikisha Jariwala", New System.Web. Caching.
        CacheDependency (Nothing, dependencies))
    End If
 Response.Write("EmpNo=" + Cache("EmpNo").ToString + " Name=" +
Cache("Name"))
End Sub

Protected Sub btnChangeCache_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnChangeCache.Click
    Cache("EmpNo") = "102"
End Sub
```


Now,save & execute the page.(fig(9.d) & fig(9.e))



[Fig (9.d) Page when loading first time]



[Fig (9.e) Page after clicking on the button]

Dependency on files:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
    If Not IsPostBack Then
        Cache("EmpNo") = "101"
        Cache.Insert("Salary", "40000", New System.Web. Caching. CacheDependency
        (Server.MapPath("Data.txt")
        Dim dependencies() As String = {"EmpNo"}
        Dim d1 As New System.Web.Caching.CacheDependency(Nothing, dependencies)
        Dim d2 As New System.Web.Caching. CacheDependency (Server.MapPath
        ("data.txt"))
        Dim acd As New System.Web.Caching.AggregateCacheDependency
        acd.Add(d1)
        acd.Add(d2)
        Cache.Insert("Report", "data", acd)
    End If
End Sub
```

In this example, along with file dependency more than one dependencies is added in the cache.

Expiration

Absolute expiration: It expire cache after some time period set at the time of activating cache. This will be absolute expiration whether cache will be used or not, it expire the cache. This type of expiration used to cache data which are not frequently changing.

Example

Cache.Insert("Name", "Nikisha Jariwala", Nothing, DateTime.Now.AddMinutes(1), System.Web.Caching.Cache.NoSlidingExpiration)

Sliding Expiration: If any request is not made during this time period,it expire cache after time period which is set at the time of activating cache. This type of expiration is useful when there are so many data to cache. So, it will put those items in the cache which are frequently used in the application.

Example

Cache.Insert("Name", "Nikisha Jariwala", Nothing, System.Web.Caching. Cache. NoAbsoluteExpiration, New TimeSpan(0, 10, 0))

Priority

You can set the priority of the cache data..So, if server memory is low than low priority data will be discarded.

Example

Cache.Insert("Name", "Nikisha Jariwala", Nothing, System.Web.Caching.Cache .NoAbsoluteExpiration, System.Web.Caching.Cache.NoSlidingExpiration, CacheItemPriority.High, Nothing)

Exercise - 9

1. List out the types of State Management Technique.
2. What is Cookies? How to use it?
3. Which are the types of Cookies?
4. What are the advantages and disadvantages of Cookies?
5. Write a note of QueryString?
6. What is URL Encoding? Which methods are used to implement it?
7. What is the use of HiddenField?
8. Develop an application in which HiddenField is used to pass the value from one page to another.
9. Write a note on ViewState.
10. Where are the ViewState data stored?
11. What is Session? How to implement it? What are its advantages and disadvantages?
12. Explain <sessionState> with its attributes.
13. Write a note on Application.
14. Which methods are used to provide concurrency for the Application?
15. What is User Profile?
16. Explain the implementation of User Profile.
17. How to enable User Profile for Anonymous users?
18. How to store User Profile in User define database?
19. .Is it possible to Group the User Profile?
20. What is Caching? Explain its types with example.

10 User Control

10.1 Introduction

In addition to Web server controls in ASP.NET Web pages, you can also create your own custom, reusable controls using the same techniques that are used for creating ASP.NET Web pages. These controls are called User Controls.

A user control is a kind of composite control that works much like an ASP.NET Web page. You can add existing Web server controls and markup to a user control then define properties and methods for the control. You can then embed them in ASP.NET Web pages, where they act as a unit.

There are two types of User Controls:

- ✓ **Web User Control**: They are containers into which you can put markup and Web server controls. You can then treat the user control as a unit and define properties and methods for it.
- ✓ **Custom Control**: It is a class that you write and derives from Control or Web Control.

Web User Control

They are easier to create than custom controls, because you can reuse existing controls. You can create complex user interface. An ASP.NET Web User Control (.ascx) is similar to a complete ASP.NET Web page (.aspx file), with both a user interface page and code. You can create the user control in much the same way as you create an ASP.NET page and then add the markup and child controls that you need. A user control can include code to manipulate its contents like a page can, including performing tasks such as data binding.

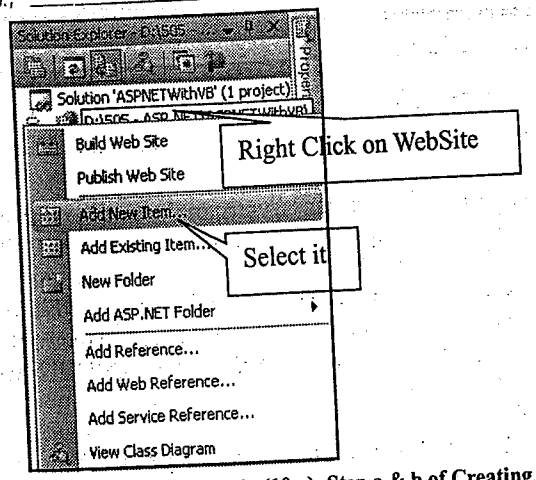
10.2 Creation and Usage

To create the Web User Control you can use the same HTML elements (except the html, body, or form elements) and Web Server Controls on Web User Control template as you do on an ASP.NET Web page.

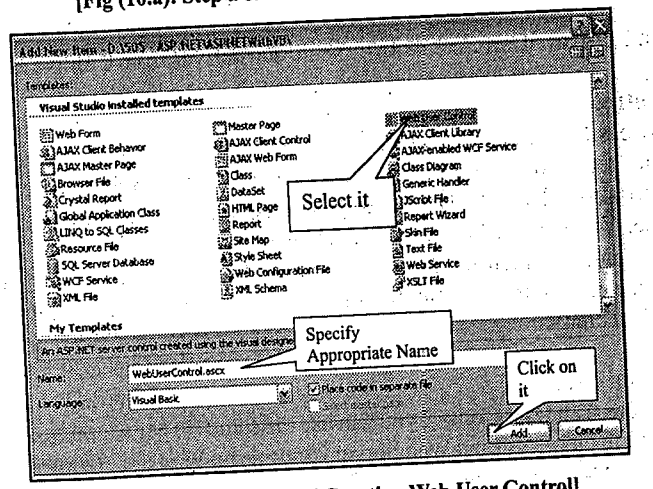
For example, if you are creating a Web User Control to use as a toolbar, you can put Buttons of Web Server Control onto the user control and create event handlers for the buttons.

Creating the User Control that Greet the User whose name is entered in TextBox

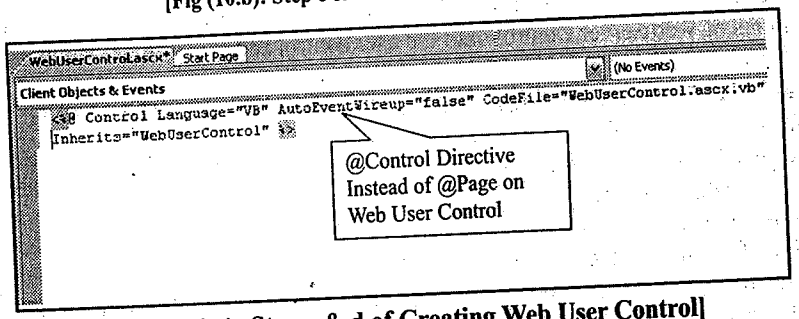
- Create the User Control
  a. Right Click on the Website name in solution explorer.
  b. Select Add→new Item from popup menu (fig. (10.a))
  c. Select Web User Control from wizard (fig. (10.b))
  d. Specify Name and then click on Add
  e. The page for Web User Control will open. Its source code contains @Control Directive instead of @Page as it is Web User Control (fig. (10.c)).



[Fig (10.a): Step a & b of Creating Web User Control]

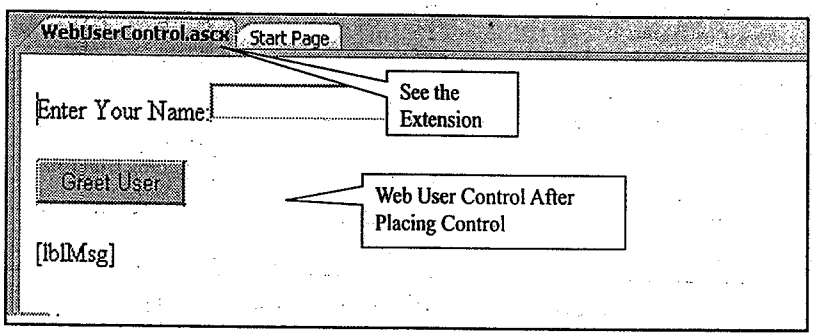


[Fig (10.b): Step c & d of Creating Web User Control]



[Fig (10.c): Step c & d of Creating Web User Control]

- Now place the Web Server Control which you want to use.
  - ✓ Label : Enter Name (lblName)
  - ✓ Textbox: to take input (txtName)
  - ✓ Button: to generate event (btnGreet)
  - ✓ Label: to display welcome message (lblMsg)



[Fig (10.d): Web User Control Design View]

- On button click event write:

```
Protected Sub btnGreet_Click(ByVal sender As Object, ByVal e As System. EventArgs)
Handles btnGreet.Click
    lblMsg.Text = "Hi " & txtName.Text & " Welcome to ASP.NET!"
End Sub
```

- Save the Web User Control

Using Web User Control on the Page

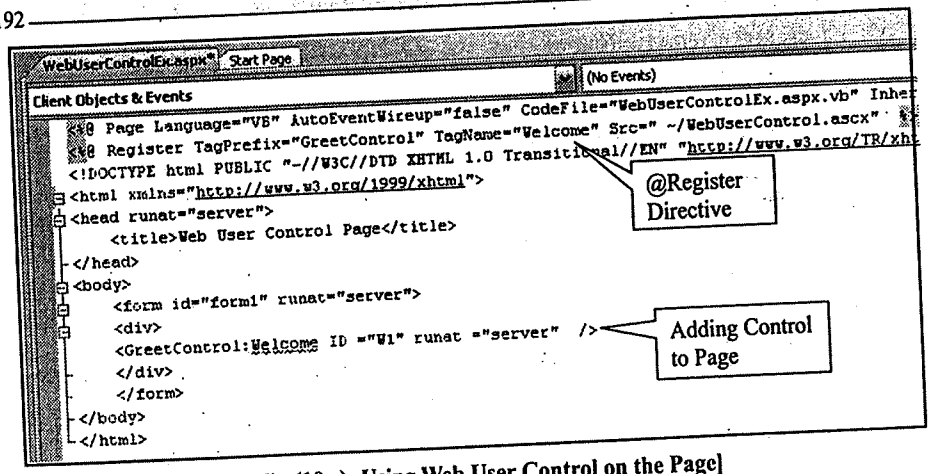
- Open the page on which you want to use the control
- Go to Source View, register the control by placing the Registering line after the @Page Directive.

```
<%@ Register TagPrefix="GreetControl" TagName="Welcome" Src="
~/WebUserControl.ascx" %>
```

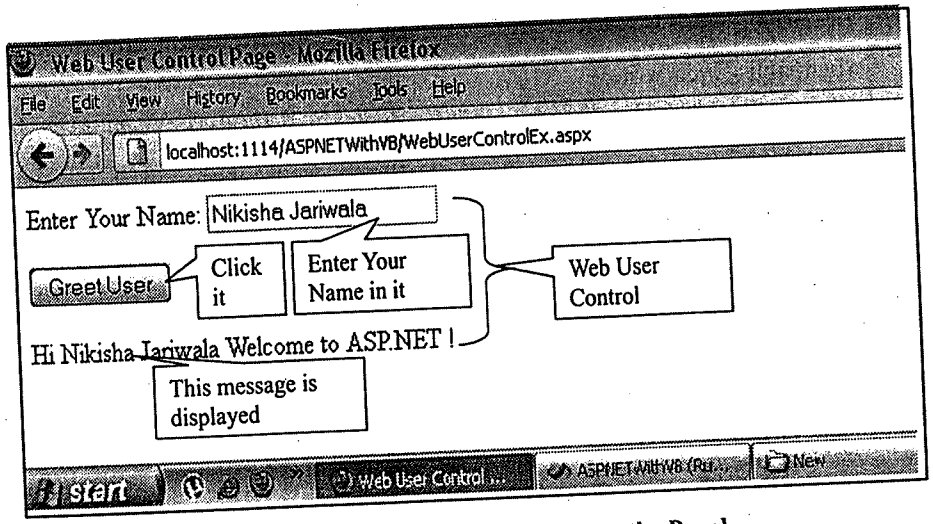
- Then in body tag write following lines:

```
<body>
<form id="form1" runat="server">
<div>
<GreetControl:Welcome ID ="W1" runat ="server" />
</div>
</form>
</body>
```

- Save and execute the Page.



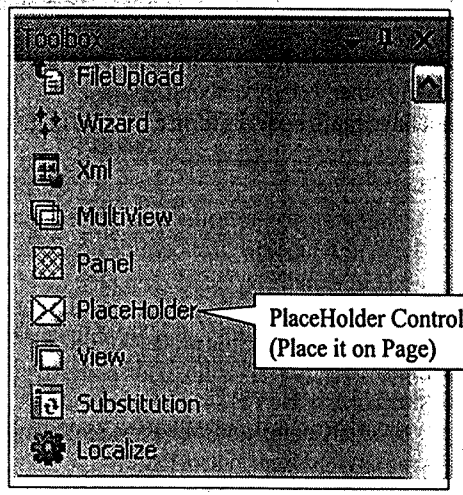
[Fig (10.e): Using Web User Control on the Page]



[Fig (10.f): Using Web User Control on the Page]

10.3 Dynamically Adding User Control

- Create the Control as describe in topic 10.2.
- Place the Button (btnAdd) on the page on which you want to add Web User Control dynamically.
- Place the PlaceHolder Control (it just holds the space on the page) below Button.



[Fig (10.g): PlaceHolder Control in the ToolBox Standard Tab]

Source View:

```
<asp:Button ID="btnAdd" runat="server" Text="Create control" /><br />
<asp:PlaceHolder ID="phUserControl" runat="server"> </asp:PlaceHolder>
```

Code View:

- Place the following code in Code View of the page on which you want to add Web User Control Dynamically on Button Click.

```
Imports ASP.webusercontrol_ascx
Partial Class _Default
    Inherits System.Web.UI.Page
    Dim UCtl As ASP.webusercontrol_ascx
    Protected Sub Page_Load(ByVal sender As Object,  ByVal e As System. EventArgs)
    Handles Me.Load
        UCtl =CType(LoadControl("~/WebUserControl.ascx"),
        ASP.webusercontrol_ascx)
    End Sub
    Protected Sub btnAdd_Click(ByVal sender As  Object, ByVal e As System
    .EventArgs) Handles  btnAdd.Click
        phUserControl.Controls.Add(UCtl)
    End Sub
End Class
```

10.4 Comparison between User Control and Web Page

- ✓ The file name extension for the user control is .ascx.
- ✓ Instead of a @ Page directive, the user control contains a @ Control directive that defines

configuration and other properties.

- ✓ User controls cannot run as stand-alone files. Instead, you must add them to ASP.NET pages, as you would add any other web server control.
- ✓ The user control does not have html, body, or form elements in it. These elements must be in the hosting page.

Exercise - 10

1. What is User Control?
2. What is Web User Control in ASP.NET?
3. Difference between Web User Control and Web Page of ASP.NET.
4. Which Directive must be in Web User Control Page?
5. Write a note of Creation and Usage of Web User Control.
6. Is it possible to add Web User Control dynamically on the page? Justify.

11 Web Services

11.1 Basics of Web Services

Web service is a programmable application component accessible via standard web protocols. It is small code for limited task. IIS and ASP.NET infrastructure support web services. .NET framework contains classes, attributes and protocol for implementing web services. VS.NET provides powerful tools for developing web services that means for implementation, testing, administration of IIS, generation of proxy (WSDL file).
It is middleware for distributed application. It is used for remote procedure calls and data exchange. It is open standard based on XML. It is independent of programming language and OS. It utilizes existing internal protocol and server architecture.

Example

There are various ATM booths of different banks available to withdraw the money. If you are having ATM Card of Bank ABC and you are at the ATM booth of Bank XYZ, then also from ATM booth of XYZ Bank, you are able to withdraw the money. This is because of Web Service. XYZ Bank has the Web Service of ABC bank. So with the help of web service, XYZ bank will allow transaction of ABC bank by calling ABC bank web service.

W3C

- ✓ World Wide Web Consortium.
- ✓ It is software application identified by URI.
- ✓ Its Interface description is in XML and XML encoded messages.
- ✓ It uses internal protocols for Message exchange.

SOAP

- ✓ Simple Object Access Protocol.
- ✓ It is XML standard for message encoding.
- ✓ It is independent of transport protocol and client-server implementation.

It is simple XML based protocol, which allows application to exchange information over HTTP. It is used for accessing web services. It is a communication protocol. It is platform and language independent, simple and extensible. It allows you to get around firewalls. It is important for application development to allow internal communication between programs.

WSDL

- ✓ Web Service Description Language.
- ✓ Its Interface description is in XML.
- ✓ Communication is done on the basis of existing protocol and server architecture.
  - o HTTP & Web Server

- SMTP & Mail server
- FTP & FTP Server

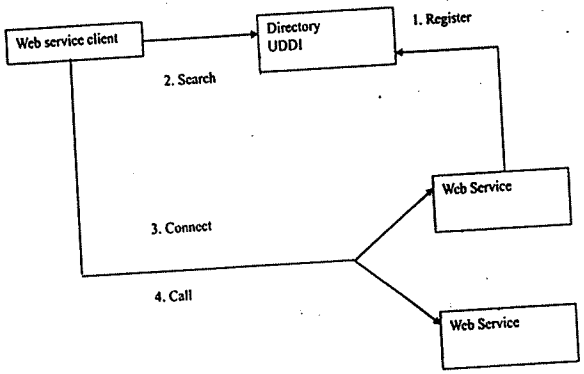
✓ It is XML based language used to define web services and to describe how to access them.

It is a file which is generated by Visual studio for each web service on the application. When you type the URL of a web service appended by ?WSDL parameter in web browser address bar, the asp.net application returns the WSDL file which contains the WSDL binding definition for the web service. We can disable WSDL file generation which is enabled by default for all web services on an asp.net application. Disabling automatic WSDL file generation makes it possible for you to override the default WSDL binding definition generated in the WSDL file for each web services.

UDDI

✓ Universal Description, Discovery and Integration
✓ It is a directory services where web services can be registered and search.

It is a specification that defines SOAP based web service for locating web services and programmable resources on a network. It offers a foundation for sharing across the enterprise and public service on the internet.



[Fig (11.a): Accessing Web Service]

Namespaces

Following are the namespaces that are used while implementing Web Services:

System.Web.Services;
System.Web.Services.configuration;
System.Web.Services.Description;
System.Web.Services.Protocols;
System.Web.Services.Discovery;
System.Xml.Serialization;

11.2 Interacting with Web Services

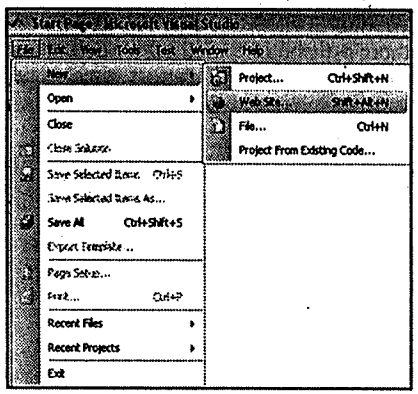
The extension of web service file in asp.net is .asmx. It contains the methods that are to be used as web services. To declare the method as web service <WebMethod> attribute is used before method declaration.

You need to create two applications:

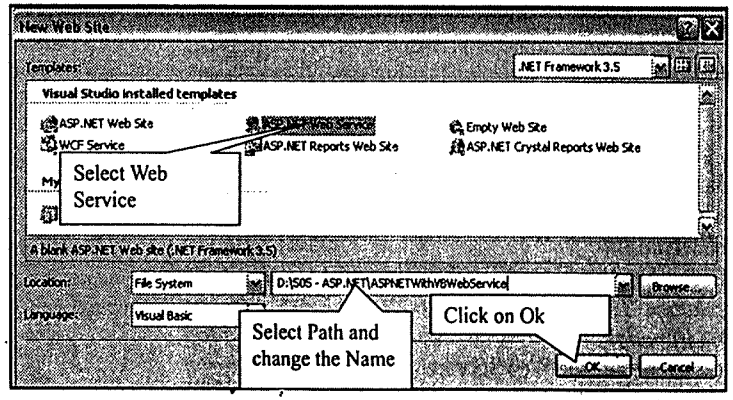
1. Web Service
2. Web Application to access Web Service

- **Creating Web Service**

✓ From File Menu → Select New → Select Web Site… → Select Web Service from the wizard → Give appropriate name → Click on Ok button. (Fig (11.b) and fig.(11.c))

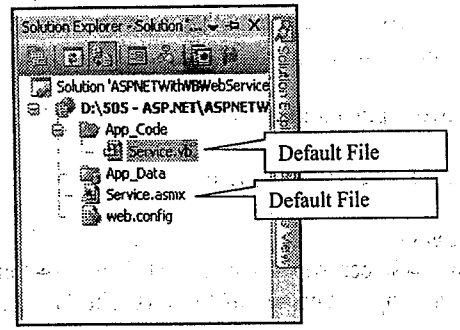


[Fig (11.b): First step to create Web Service]



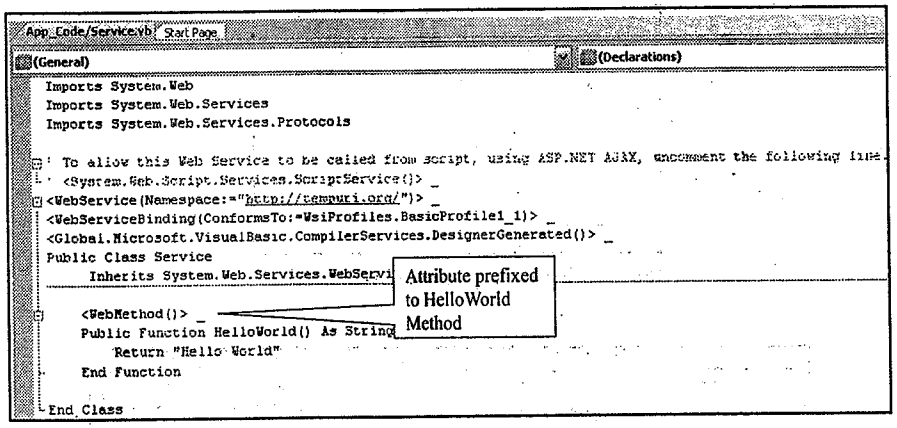
[Fig (11.c): Second step to create Web Service]

✓ There will be two default files one is service.asmx and other is service.vb which is stored in app_code folder. (Fig (11.d))



[Fig (11.d): Directory listing of Web Service]

✓ Now in service.vb, there will be one method by default which is prefixed with the attribute<WebMethod>. (Fig (11.e))



[Fig (11.e): Default Service.vb with <WebMethod>]

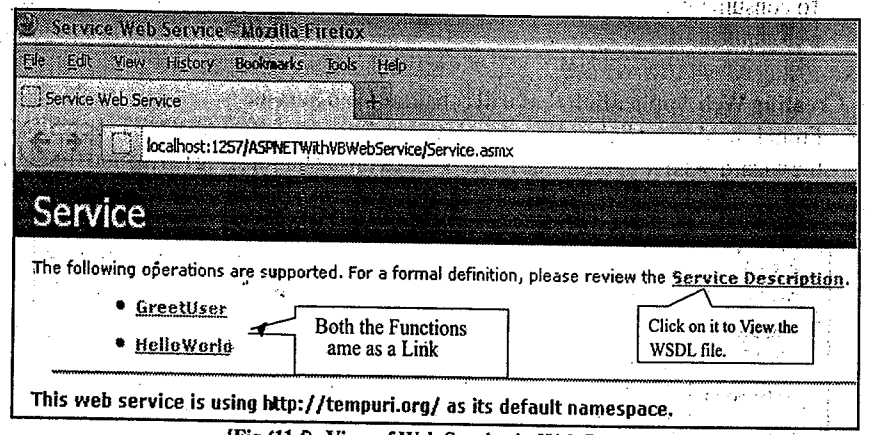
✓ Add another function below the HelloWorld Method of Service.vb file.

Example

```
<WebMethod()> _
  Public Function GreetUser(ByVal name As String) As String
    Return "Hello " & name
  End Function
```

✓ Save and execute the Web Service.

✓ Service.asmx file will be executed in web browser. (Fig (11.f))

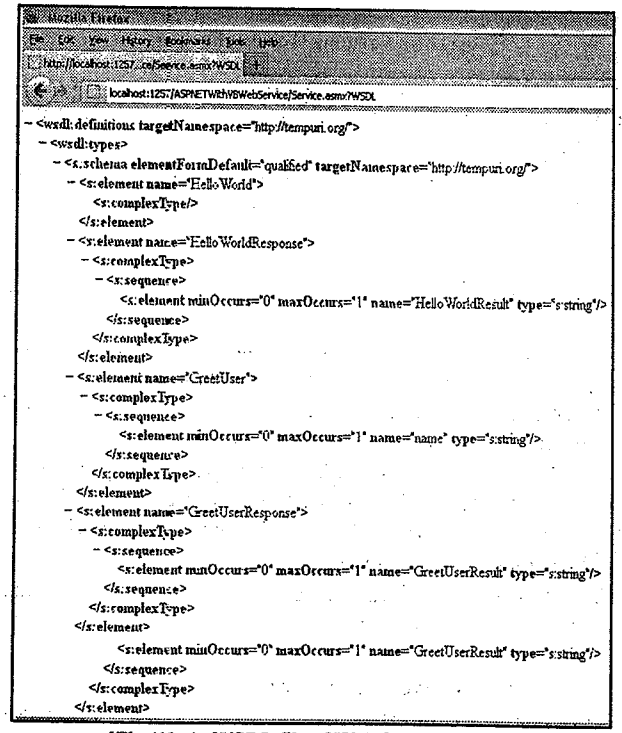


[Fig (11.f): View of Web Service in Web Browser]

✓ If you want to see the WSDL file click on Service Description link (Fig (11.f)) or Append ?WSDL in the Address Bar with the URL (Fig (11.g)).

Example

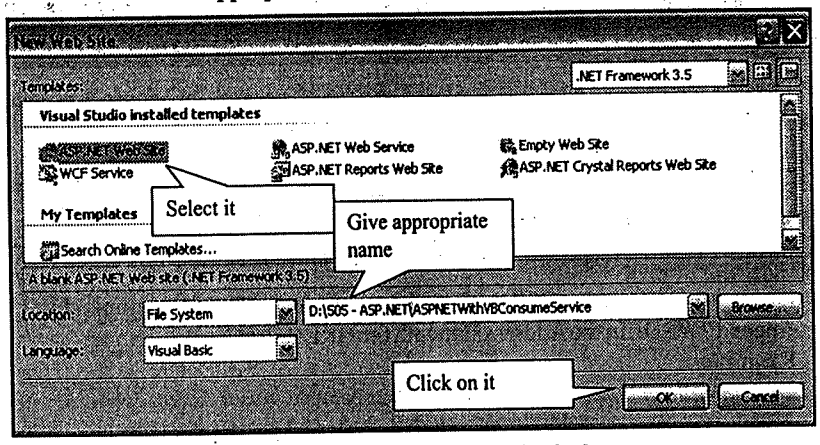
http://localhost:1257/ASPNETWithVBWebService/Service.asmx?WSDL



[Fig (11.g): WSDL file of Web Service]

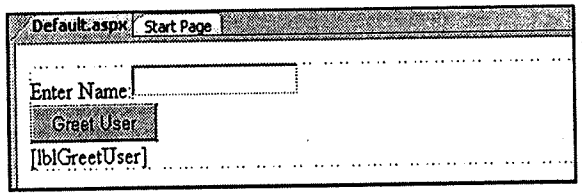
- ✓ To consume web service, you need to provide web reference of web service in your web application.

- **Creating Web Application that will consume Web Service**
  - ✓ From File Menu→Select New→Select Web Site...→ Select ASP.NET Web Site from the wizard→ Give appropriate name→Click on Ok button. (Fig (11.b) and fig.(11.h))



[Fig (11.h): Creating Web Application]

  - ✓ Place textbox (txtName) to take input, button (btnDisplayName) to call Web Service and label (lblName) for message and Label (lblGreetUser) to greet the user on the page. (Fig (11.i))



[Fig (11.i): Creating Web Application]

  - ✓ Add Web Reference in the Web Application:
    - o Right Click on the Web Application in Solution Explorer
    - o Select Add Web Reference from the Popup menu (Fig (11.j))
    - o Copy the URL of Web Service from the browser (Fig (11.k))
    - o Paste it in the URL bar of the Add Web Reference Wizard
    - o Click on go button (Fig (11.l)).

    Here the Web Reference is the reference of another Web Service that is added in the Web Application to access the Web Service.

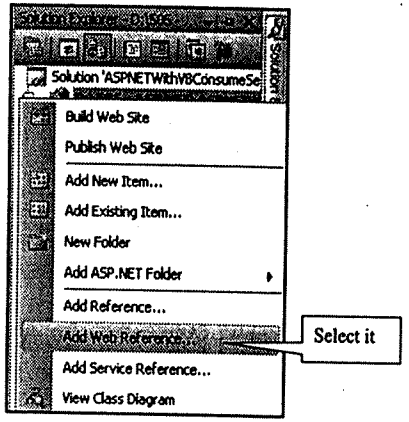
  - ✓ By default, you will get the Web Reference name as localhost.
  - ✓ Click on Add Reference Button to add reference to the Web Application. (Fig (11.l))

- ✓ View the directory listing of Web Application in Solution Explorer after adding the Web Reference. (Fig (11.m))
- ✓ Write the following code on the btnDisplayName_Click event of the Default.aspx.vb page:

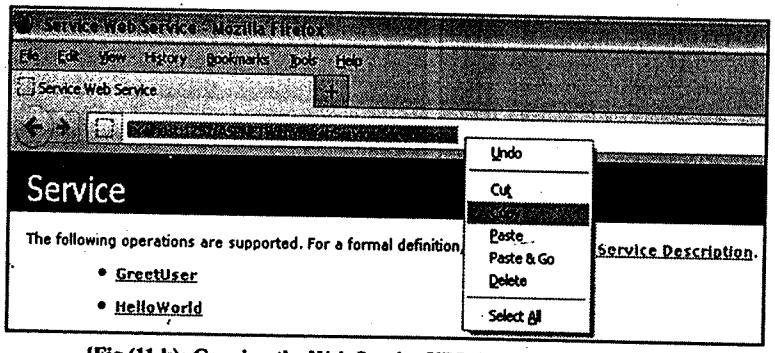
Example

```
Protected Sub btnDisplayName_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnDisplayName.Click
    Dim s1 As New localhost.Service
    lblGreetUser.Text = s1.GreetUser (txtName.Text)
End Sub
```

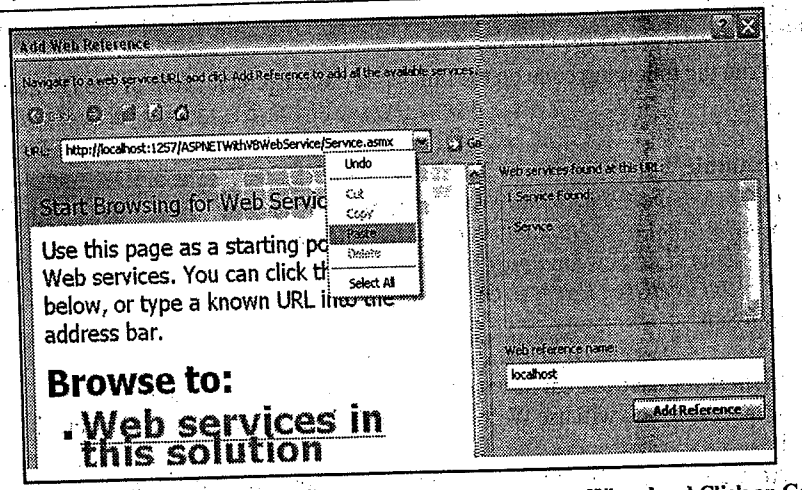
- ✓ In short you have to treat Service.vb file of web service as a class.
- ✓ To access it, you will use Web Reference (localhost) as a class and create the object of it.
- ✓ At last with the help of this object, access the methods of the web service.



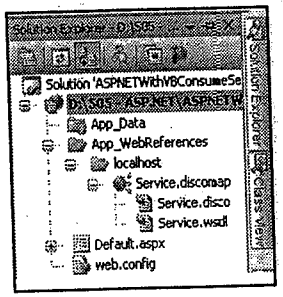
[Fig (11.j): Adding Web Reference]



[Fig (11.k): Copying the Web Service URL from the Web Browser]

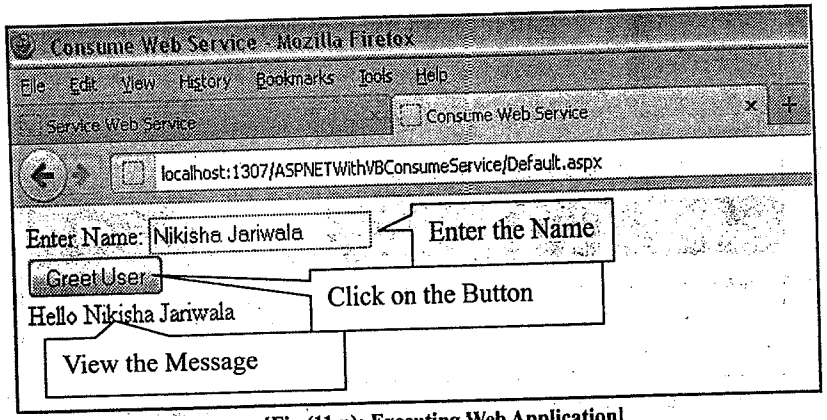


[Fig (11.l): Paste URL in the URL bar of the Add Web Reference Wizard and Click on Go]



[Fig (11.m): Directory listing of Web Application after adding the Web Reference]

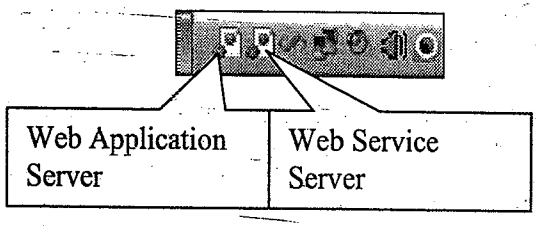
✓ Now, save and execute the Web Application.



[Fig (11.n): Executing Web Application]

Here the message is displayed with the help of GreetUser Web Method of the Web Service which is written in Service.vb file.

✓ While running the Web Application, you need to take care of one thing, is the Server of the Web Service must also be in running condition, then only you will be able to use the Web Service.



[Fig (11.o): Servers in notification area]

Exercise - 11

1. What is Web Service? Explain with example.
2. What is W3C?
3. What is the importance of SOAP in Web Service?
4. What is WSDL? How is it created in VS.NET?
5. To access Web Services from anywhere, where can it be stored?
6. List out the Namespaces used for developing Web Service.
7. Write the steps to create Web Service.
8. How to consume Web Service?
9. What is Web Reference?
10. What is the Default name of Web Reference?
11. How to add Web Reference in the Web Application?

12 Error Handling

Error

Error means bug that means when you compile or run a program, abnormal situation may raise which is known as error. There are different types of Error:

1. Syntax Error: The error raised by misspelled keywords or variables is called Syntax Error. Compiler will normally catch these errors while you are coding or after the compilation of program.
2. Logic Error: Due to the mistake or flaw present in the sequence of statements, code does not act as expected. This type of error is called Logic Error.
3. Runtime Error: The error which occurs when the code is executing, is called Runtime Error. It occurs if you do not handling an unexpected error.

Exception

Exceptions are run-time errors, which are raised when program is running. There are two ways to handle errors that occur at runtime:

(i) Unstructured exception handling
(ii) Structured exception handling

12.1 Unstructured Exception Handling

Unstructured Exception Handling is implemented with On Error GoTo statement, which is placed at the beginning of a code block to handle all possible exceptions that occurs during the execution of the code. Any exception that occurs is fatal and your program will stop.

Syntax:

On Error {GoTo [Line | Label | 0 | -1] | Resume Next | Line}

GoTo label | line - The argument is any line label or line number. If an exception occurs, program exception goes to the given location.
GoTo 0 - Disables enabled exception handler in the current procedure and reset it to nothing. It clears the Error object.
GoTo –1 - same as GoTo 0
Resume Next - It specifies that when an exception occurs, execution skips over the statement that cause the problem and goes to the statement immediately following and continues from that point.
Resume Line - line is the line number or label that specifies where to resume execution.

Example

```
Private Sub BtnModulus_Click()
    Dim n1, n2 As Integer
    On Error GoTo err_msg
    n1 = 12
    n2 = 5
    txtres.Text = n1 Mod n2
    Exit Sub
err_msg:
    MsgBox ("Division by zero error")
End Sub
```


Here, if Exception occurs then the execution is continued from the label specified with the On Error Goto Statement.

Err Object

When an error occurs, the Err object contains information about the error, which helps you to determine whether you can attempt to fix the error or ignore the error. The Err object also has method that allows you to raise errors or clear the state of the Err object.

Properties

Properties	Description
Number	It returns the numeric value of the error.
Description	It gets or sets a descriptive string for the current error number.
Erl	It returns an integer indicating the last line number of the last executed statement.
HelpFile	It contains the fully qualified path to the help file.
Source	It returns or sets a string that specifies the name of the object or application that originated the error.
HelpContext	It returns or sets an integer that contains the context ID for the topic, a Help file.

[Table:1 Properties of Err object]

Methods

Methods	Description
Clear()	It clears the Err object
Raise()	It raises an error.

[Table:2 Methods of Err object]

Example (Err Object)

```
Private Sub btnErr_Click()
    Dim n1, n2 As Integer
    On Error GoTo err_msg
    n1 = 12
    n2 = 5
    txtres.Text = n1 Mod n2
    Exit Sub
err_msg:
```

```
    MsgBox (Err.Number & Err.Description)
End Sub
```

Here, when the error occurs Err.Number will display Error Number and the Err.Description will display description of the error that occurs.

Example (Resume Next)

```
Private Sub btnResume_Click()
    Dim flag, n1, n2 As Integer
    On Error GoTo err_msg
    n1 = 12
    n2 = 0
    flag = 1
    txtres.Text = n1 Mod n2
    If flag = 0 Then
       txtres.Text = 0
    End If
    Exit Sub
err_msg:
    MsgBox (Err.Number & Err.Description)
    flag = 0
    Resume Next
End Sub
```

Here, when the error occurs the control goes to err_msg label as it is specified with On Error Goto Statement. When the Resume Next Statement is executed, the execution of the code starts from the next statement on which the error was occurred.

Example (On Error Goto 0)

```
Private Sub btnError0_Click()
    Dim n1, n2 As Integer
    On Error GoTo err_msg
    n1 = 12
    n2 = 0
    txtres.Text = n1 Mod n2
err_msg1:
    On Error GoTo 0                    'Turn Off error handling
    MsgBox ("Program completed")
    Exit Sub
err_msg:
    MsgBox (Err.Number & Err.Description)
    Resume Next                'Or    Resume Err_msg1
End Sub
```

Here, when the error occurs the control goes to err_msg label as it is specified with On

Error Goto Statement. When the Resume Next Statement is executed, the execution of the code starts from the next statement on which the error was occurred, where it On Error Goto 0 will turn off the Error handling.

12.2 Structured Exception Handling

It is implemented with Try…Catch…Finally statement, which is divided into a Try block, optional Catch blocks and an optional Finally block.

The Try block contains code where exceptions can occur; the Catch block contains code to handle the exceptions that occur.

If an exception occurs in the Try block the code throws the exception, so it can be caught and handled by the appropriate Catch statement. After the rest of the statement finishes, execution is always passed to the Finally block.

Syntax:

```
Try
        [Try statements]
        [Catch [exception1  [As type1]] [when expression1]
                Catch statement1
                [Exit Try]]
        [Catch [exception1  [As type2]] [when expression2]
                Catch statement2
                [Exit Try]]
        .....
        [Catch [exceptionN  [As typeN]] [when expressionN]
                Catch statementN
                [Exit Try]]
        [Finally
                [finally statement]]
End Try
```

Here type1, type2, …,typeN in the Catch block are the Exception type which can be name of the Exception class that you want to handle.

Exceptions are based on the Exception classes. The base class in the hierarchy of the Exception classes is System.Exception. It has two derived classes System.ApplicationException and System.SystemException.

```
System.Exception
    ├─System.ApplicationException
    └─System.SystemException
```

Each derived class itself has many derived classes. For example, OverflowException class, which is based on ArithmeticException class, which in turn based on SystemException Class, which in turn based on the System.Exception class.

System.Exception
└System.SystemException
└System.ArithmeticException
└System.OverFlowException

Exit Try: it is optional keyword that breaks out of the Try...Catch...Finally structure. Execution resumes with the code immediately following the End Try statement. The Finally statement will still be executed. It is not allowed in Finally blocks.

Example

```
Private Sub btnTryEx_Click()
    Dim n1, n2, n3 As Integer
    Try
      n1 = 12
      n2 = 0
      n3 = n1 Mod n2
      MsgBox ("Result: " & n3)
    Catch ex As Exception
      MsgBox ("Run-time Error occur")
    End Try
End Sub
```

Here, Try block contains the code that may raise the error and the Catch block contains the code that will handle the error. Now when the statement n3=n1 Mod n2 will be executed the error will be raised as n2=0. The control will pass to Catch block, where the Exception object ex is created and it will handle the error.

If a Try statement does not contain any Catch block, it must contain Finally block.

Example

```
Private Sub btnTryEx_Click()
    Dim n1, n2, n3 As Integer
      n1 = 12
      n2 = 0
    Try
      n3 = n1 Mod n2
      MsgBox ("Result: " & n3)
    Finally
      MsgBox ("Execution complete")
    End Try
End Sub
```

You can get more information about the exception by getting an exception object. You can do so by catching any exception based on the Exception class and using the exception object's ToString method or Message property to display information (error description, line no, filename and location of file etc.) about the exception.

Example

```
Private Sub btnExObj_Click()
  Dim n1, n2, n3 As Integer
  Try
    n1 = 12
    n2 = 0
    n3 = n1 Mod n2
    MsgBox ("Result: " & n3)
  Catch ex As Exception
    MsgBox (ex.Message)     'Or     MsgBox (ex.ToString)
  End Try
End Sub
```

Exception filtering in the Catch block

The process of handling different types of exception differently according to the nature of the exception that occurred is called filtering. There are two ways to filter exceptions with catch block.

(I) Filter on specific classes of exceptions.

(ii) To filter on any conditional expression, using When keyword.

(i) Filter on specific classes of exceptions

Example

```
Private Sub btnfilter_Click()
    Dim n1, n2, n3 As Integer
    Try
      n1 = 1223243551
      n2 = 123
      n3 = n1 * n2
      MsgBox("Result: " & n3)
    Catch ex As OverflowException
        MsgBox(ex.ToString)
    Catch ex1 As DivideByZeroException
      MsgBox(ex1.ToString)
    End Try
End Sub
```

Here the exceptions are filtered according to the Exception classes. If OverFlowException occurs, it is handled by first Catch block and if DivideByZeroException occurs, it is handled by second Catch block. Only, these two Exceptions are handled.

You can use GetType method of Exception class to get the type of the exception as a string.

(ii) Filter using When keyword:

This option is often used to filter by exception number, which you can check with Err object's Number property.

Example

```
Private Sub btnFilter_Click()
    Dim n1, n2, n3 As Integer
    Try
      n1 = 1223243551
      n2 = 0
      n3 = n1 Mod n2
      MsgBox("Result: " & n3)
    Catch ex As OverflowException
      MsgBox(ex.ToString)
    Catch When Err.Number = 11
      MsgBox("divided by zero")
   Catch ex As System.ArgumentException
   MsgBox("Invalid argument value")
    End Try
End Sub
```

You can use multiple Catch statements when you filter exceptions. If you want to add a general exception handler to catch any exceptions not filtered, you can add a Catch block for the Exception class at the end of the other Catch blocks:

Example

```
Private Sub Button2_Click()
    Dim n1, n2, n3 As Integer
    Try
      n1 = 1223243551
      n2 = 0
      n3 = n1 Mod n2
      MsgBox("Result: " & n3)
    Catch ex As OverflowException
      MsgBox(ex.ToString)
    Catch When Err.Number = 11
      MsgBox("divided by zero")
   Catch ex As Exception
   MsgBox("Exception occurred ..")
    End Try
End Sub
```

Here, Exception is the general Exception handler in the last Catch block.

Finally

The code in the Finally block is always executed in a Try...Catch...Finally statement, even if there was no exception, and even if you execute an Exit Try statement. This allows you to deallocate resources and so on.

Example

```
Private Sub Button2_Click()
    Dim n1, n2, n3 As Integer
    Try
      n1 = 12
      n2 = 0
      n3 = n1 Mod n2
      MsgBox("Result: " & n3)
    Catch ex As Exception
      MsgBox(ex.Message)                    'Or    MsgBox(ex.ToString)
    Finally
        MsgBox("Execution completed..")
    End Try
End Sub
```

Throwing an Exception

You can throw an exception using the throw statement and you can also rethrow a caught exception using the Throw statement.

Example

```
Try
        Throw New OverflowException
    Catch ex As Exception
        MsgBox(ex.Message)
End Try
```

Throwing a custom exception

You can customize the exceptions by creating a new exception object based on the ApplicationException object.

Example

```
Try
    Throw New ApplicationException("This is new exception")
Catch ex As Exception
    MsgBox(ex.Message)
End Try
```

Example

```
Try
    If txtno1.Text = "" Then
    Throw New ApplicationException("Exception:please enter a string")
     Else
            MsgBox("You entered : " & txtno1.Text)
     End If
 Catch ex As Exception
        MsgBox(ex.Message)
End Try
```

Here, in the above example, the code written in the Try block will check that the txtno1 TextBox is empty or not. If it is empty, then it will throw the exception with the help of ApplicationException class object. Otherwise, it will display the inputted number in the message box.

12.3 Error handling at different levels in ASP.NET

Error in ASP.NET can be handled at two levels:

(1) Page Level
(2) Application Level

12.3.1 Page Level

To handle the Errors at Page Level you need to use the errorPage attribute in the webform. This attribute defines the page the user should be redirected to, when an unhandled exception occurs in that specific page. You can set the error page in the @Page Directive or the ErrorPage Property of the page.

Example

```
<%@ Page language="c#" Codebehind="WebForm1.aspx.cs"
  AutoEventWireup="false" Inherits="WebTest.WebForm1" errorPage=
"/WebTest/ErrorPages/PageError.html"%>
```

If you define the error page in the errorPage attribute, then it maps to the Page.ErrorPage property, and hence it may be set programmatically. The value may optionally include query string parameters. But, if no parameters are added, ASP.NET would automatically add one with the name aspxerrorpath. This parameter would hold the value of the relative URL to this page, so that the error page would be able to determine which page caused the error.

If a value is specified in this attribute (or property) and an unhandled exception occurs in the page, the Page class would automatically perform a redirect to the specified page. If a value is not specified, the exception is assumed to be unhandled, so new HttpUnhandledException object is created and then thrown, propagating it to the next higher level.

12.3.2 Application Level

Global.asax file will catch all handled ASP.NET errors while processing request.

Example

```
Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
 'Code that runs when an unhandled error occurs
     Dim exc as Exception =Server.getLastError()
     If exc.message.contains("nocatch") Then
          Return
     End If
     Server.Transfer ("Error.Aspx")
End Sub
```

Customizing error page

To customize the default error page, you will have to change the default configuration setting of the application in Web.Config file.

The mode attribute of <customErrors> tag in Web.config file specifies whether to show user-defined custom error pages or ASP.NET error pages. There are 3 modes:

- Off mode: when error attribute is set to "off", ASP.NET uses its default error page.
- On mode: ASP.NET uses user defined custom error page instead of default error page.
- Remote mode: Custom error pages are shown for all remote users. ASP.NET error pages with rich error information are displayed only for local users.

Example

```
<configuration>
     <system.web>
          <customErrors mode="Off"/>
     </system.web>
</configuration>
```

Here the <customErrors> mode is set to Off in Web.Config file.

Example

```
<customErrors mode="remote" defaultredirect="error.htm"/>
```

Here the <customerrors> mode is set to Remote with the default error page.

Example

```
<customErrors mode="On" defaultRedirect="/WebTest/ErrorPages/ AppError.html
    ">
     <error statusCode="404" redirect="/WebTest/ErrorPages/404.htm"/>
</customErrors>
```

Here the <customErrors> mode is set to On and the default error page is also set to "AppError.html". But, if 404 status code errors come then it should be redirected to another page "404.htm".

The defaultRedirect attribute specifies the path for generic error page. This error page would typically have a link that allows user go back to the home page or perform the request once again.

The settings specified in the page level (errorPage attribute) would override the settings specified in the <customErrors>. Errors in the page would be handled by the Page class first, that prevents the exception from being propagated to the application level. It is only when the Page class fails to handle the exception then <customErrors> will handle the exception.

Exercise - 12

1. What is Error? Explain its types.
2. What is Exception? List out the ways to handle exceptions.
3. Explain Unstructured Exception Handling in detail.
4. What is the significance of On Error Goto 0 statement?
5. Explain Err object in detail.
6. Explain Structured Exception Handling in detail.
7. What is the significance of Finally block?
8. Explain Exception Filtering in detail.
9. What is the importance of When clause?
10. Which attribute of page is used to handle the error at Page Level?
11. Explain Application Level Error Handling.
12. Explain <customError> of Web.config file.
13. What is the use of defaultRedirect attribute?
14. Explain various modes of <customError> tag.

13 Ajax

13.1 Introduction

Ajax (Asynchronous JavaScript and XML) is a group of web development methods that is used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to, and retrieve data from, a server in the background (asynchronously) without interrupting the display and behavior of the existing page.
HTML and CSS can be used in combination for mark up and style information respectively. The DOM (Document Object Model) is accessed with JavaScript to dynamically display, and allow the user to interact with the information on the page. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads. Only the user interface that contains new information is updated asynchronously. It is a data driven and not a page driven.
In short, with the help of Ajax you can exchange data with a server, and update parts of a web page without reloading the whole page.

XMLHttpRequest

It is a JavaScript object adopted by modern browsers, which communicates with the server via standard HTTP Get / Post method. It works in background for performing asynchronous communication with the server.

Properties

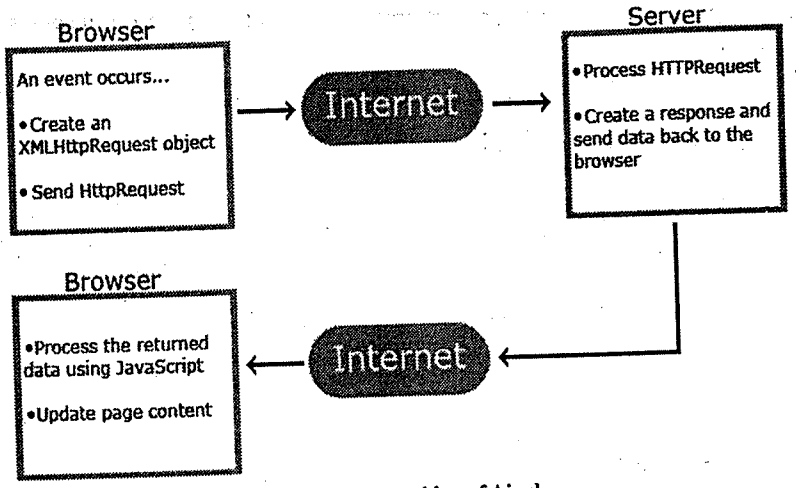
Properties	Description
OnReadyStateChange	Set with JavaScript event handler that fires at each state change.
ReadyState	It returns current status of request.
Status	It returns HTTP status returned from the server.
ResponseText	It gives the data returned from the server in the form of String.
ResponseXML	It gives the XML document of the data that is returned from the server.
StatusText	It gives the text returned from the server for the HTTP status.

[Table:1 Properties of XMLHttpRequest object]

Methods

Methods	Description
Open()	It opens the connection with the server.
Send()	It sends the request including the string and DOM object data.
Abort()	It terminates current request.
GetAllResponseHeaders()	It returns the headers as a string.
GetResponseHeader()	It returns value of the given header.
SetRequestHeader()	It sets the request headers before sending.

[Table:2 Methods of XMLHttpRequest object]



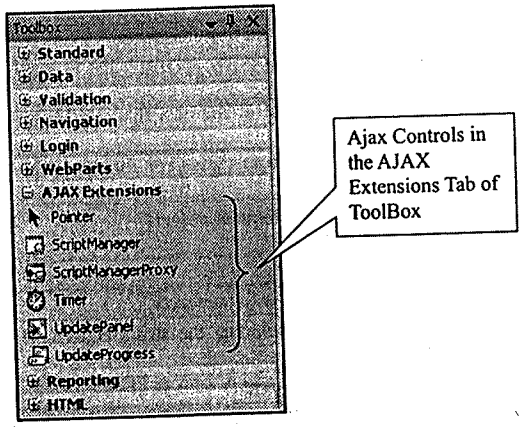
[Fig (13.a): working of Ajax]

13.2 Ajax in ASP.NET

If you want to use Ajax in ASP.NET, you don't need to worry about the code related to its implementation. In ASP.NET, there are various controls available with the help of which you can easily create Ajax enabled web pages.

- **Controls for Ajax**

  Ajax server controls consist of server and client code that integrate to produce rich client behavior. When you add an Ajax-enabled control to an ASP.NET Web page, the page automatically sends supporting client script to the browser for Ajax functionality. Each control is made up of different templates according to their functionality.



[Fig (13.b): Controls available in ASP.NET to Implement Ajax]

ScriptManager Control

It manages script resources for client components, partial-page rendering, localization, globalization, and custom user scripts. The ScriptManager control is required in order to use the UpdatePanel, UpdateProgress, and Timer controls. It must be the first control on the page, on which you want to implement Ajax.

An ASP.NET page supports partial-page rendering which is controlled by the following factors:

- The ScriptManager control's EnablePartialRendering property must be true (the default value).
- There must be at least one UpdatePanel control on the page.
- The SupportsPartialRendering property must be true (the default value). If the SupportsPartialRendering property is not set explicitly, its value is based on browser capabilities.

Properties

Properties	Description
ID	It gets or sets the programmatic identifier assigned to the server control.
Controls	It gets a ControlCollection object that represents the child controls for a specified server control in the UI hierarchy.
IsInAsyncPostBack	It gets a value that indicates whether the current postback is being executed in partial-rendering mode.
Page	It gets a reference to the Page instance that contains the server control.

[Table:3 Properties of ScriptManager Control]

Methods

Methods	Description
DataBind()	It binds a data source to the invoked server control and all its child controls.
Dispose()	It enables a server control to perform final clean up before it is released from memory.
Focus()	It sets input focus to a control.
HasControls()	It determines if the server control contains any child controls.

[Table:4 Methods of ScriptManager Control]

UpdatePanel Control

UpdatePanel control is an important part of AJAX functionality in ASP.NET. It is used with the ScriptManager control to enable partial-page rendering. Partial-page rendering reduces the need for synchronous postbacks and instead of complete page update, it updates only the part of the page. Partial-page rendering improves the user experience because it reduces the screen flicker that occurs during a full-page postback and improves Web page interactivity.

Refreshing UpdatePanel Content

When partial-page rendering is enabled, a control can perform a postback that updates the whole page or an asynchronous postback that updates the content of one or more UpdatePanel controls. Whether a control causes an asynchronous postback and updates an UpdatePanel control depends on the following:

- If the UpdateMode property of the UpdatePanel control is set to Always, the UpdatePanel control's content is updated on every postback that originates from the page. This includes asynchronous postbacks from the controls that are inside other UpdatePanel controls and postbacks from controls that are not inside UpdatePanel controls.
- If the UpdateMode property is set to Conditional, the UpdatePanel control's content is updated in the following circumstances:
  - When you call the Update method of the UpdatePanel control explicitly.
  - When the UpdatePanel control is nested inside another UpdatePanel control and the parent panel is updated.
  - When a postback is caused by a control that is defined as a trigger by using the Triggers property of the UpdatePanel control.
  - When the ChildrenAsTriggers property is set to true and a child control of the UpdatePanel control causes a postback.

The combination of setting the ChildrenAsTriggers property to false and the UpdateMode property to Always is not allowed and will throw an exception.

Properties

Properties	Description
ID	It gets or sets the programmatic identifier assigned to the server control.
Controls	It gets a ControlCollection object that represents the child controls for a specified server control in the UI hierarchy.
IsInAsyncPostBack	It gets a value that indicates whether the current postback is being executed in partial-rendering mode.
Page	It gets a reference to the Page instance that contains the server control.

[Table:5 Properties of UpdatePanel Control]

Methods

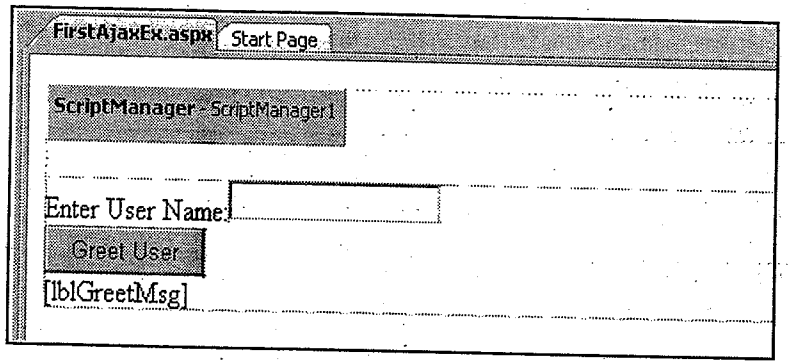
Methods	Description
Update()	It causes an update of the content of an UpdatePanel control.
DataBind()	It binds a data source to invoked server control and all its child controls.
Dispose()	It enables a server control to perform final clean up before it is released from memory.
Focus()	It sets input focus to a control.
HasControls()	It determines if the server control contains any child controls.

[Table:6 Methods of UpdatePanel Control]

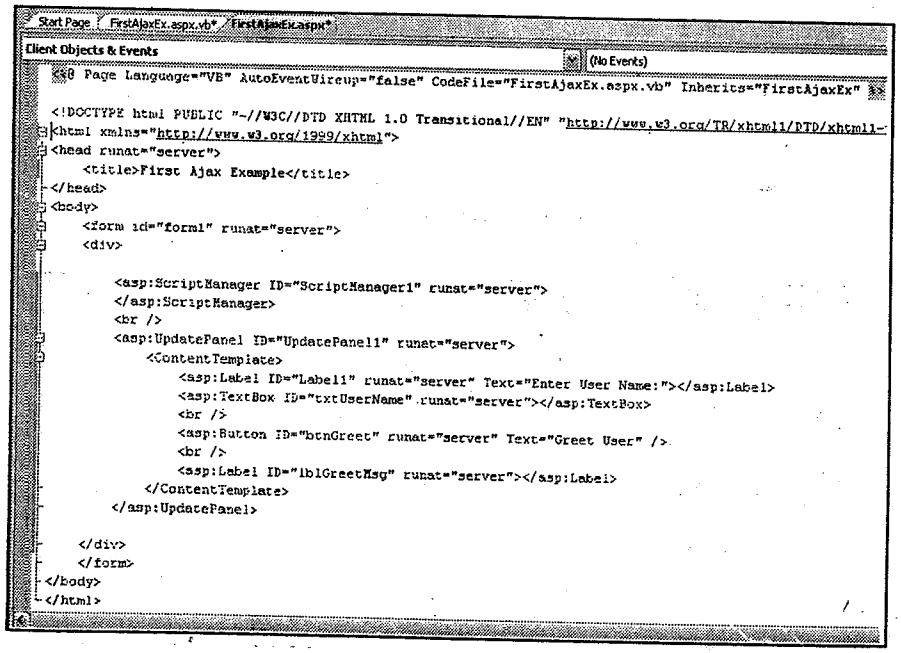
Example

Create a Page to greet the user whose name is entered in the TextBox, using Ajax.

- Place the ScriptManager Control. It must be the first control on the Web Page.
- Place the UpdatePanel Control below the ScriptManager Control.
- Place TextBox, Button and Label Control in the UpdatePanel Control (Fig (13.c),and fig. (13.d))

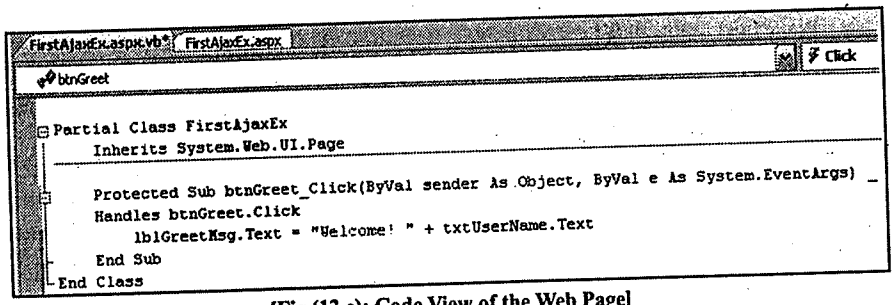


[Fig (13.c): Design View of the Web Page]



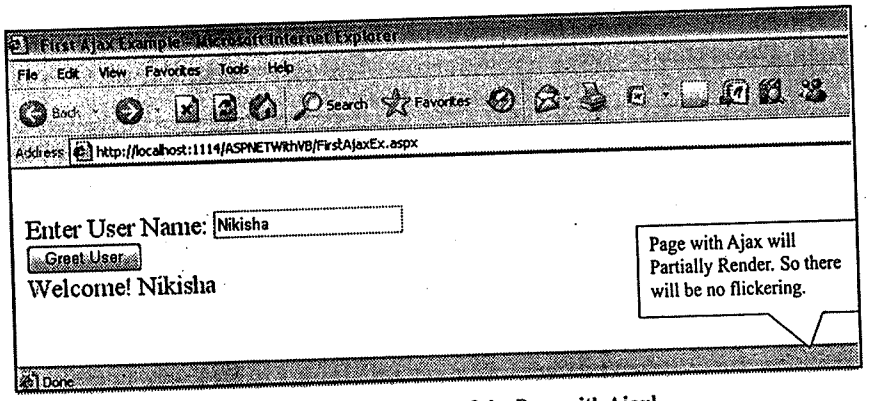
[Fig (13.d): Source View of the Web Page]

- Write the following code on the Click() Event of "Greet User" button. (fig (13.e))

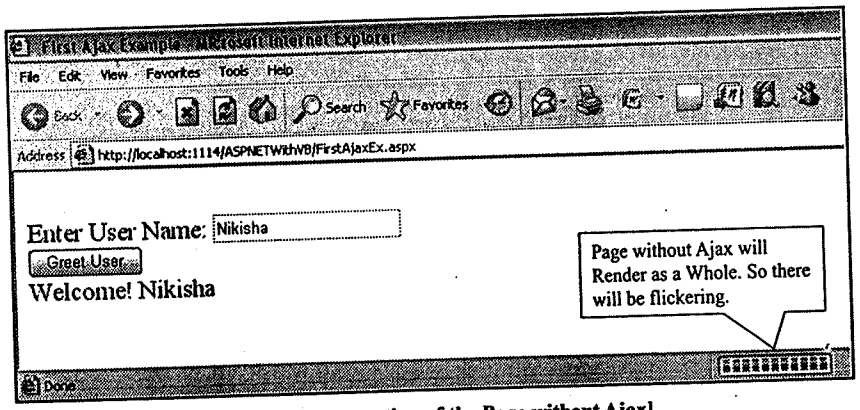


[Fig (13.e): Code View of the Web Page]

- Save and execute the Web page. (fig (13.f))



[Fig (13.f): Execution of the Page with Ajax]



[Fig (13.g): Execution of the Page without Ajax]

UpdateProgress Control

The UpdateProgress control provides status information about partial-page updates in UpdatePanel controls. You can customize the default content and the layout of the UpdateProgress control. To prevent flashing, when a partial-page update is very fast, you can specify a delay before the UpdateProgress control is displayed.

If a partial-page update is slow, you can use the UpdateProgress control to provide visual feedback about the status of the update. You can put multiple UpdateProgress controls on a page, each associated with a different UpdatePanel control. Alternatively, you can use one UpdateProgress control and associate it with all UpdatePanel controls on the page.

You can associate an UpdateProgress control with an UpdatePanel control by setting the AssociatedUpdatePanelID property of the UpdateProgress control. When a postback event originates from an UpdatePanel control, any associated UpdateProgress controls are displayed. If you do not associate the UpdateProgress control with a specific UpdatePanel control, the UpdateProgress control displays progress for any asynchronous postback.

If the ChildrenAsTriggers property of an UpdatePanel control is set to false and an asynchronous postback is generated inside the UpdatePanel control then it will display associated UpdateProgress control(s).

Properties

Properties	Description
ID	It gets or sets the programmatic identifier assigned to the server control.
Controls	It gets a ControlCollection object that represents the child controls for a specified server control in the UI hierarchy.
IsViewStateEnabled	It gets a value indicating whether view state is enabled for this control.
Page	It gets a reference to the Page instance that contains the server control.
AssociatedUpdatePanelID	It sets / gets the ID of the UpdatePanel Control to which it is associated.
DisplayAfter	It specify the time in microseconds after which the ProgressTempate of the control is displayed.

[Table:7 Properties of UpdateProgress Control]

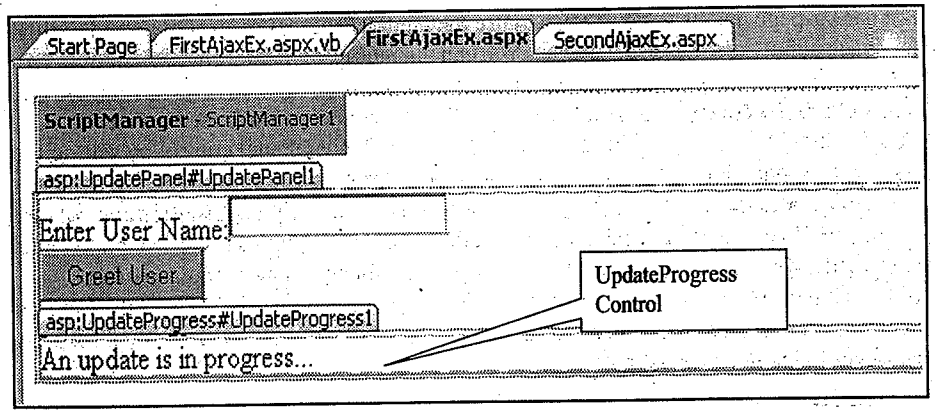
Methods

Methods	Description
DataBind()	It binds a data source to invoked server control and all its child controls.
Dispose()	It enables a server control to perform final clean up before it is released from memory.
Focus()	It sets input focus to a control.
HasControls()	It determines if the server control contains any child controls.

[Table:8 Methods of UpdateProgress Control]

Example

Create same example as specified in UpdatePanel control Example, but add one more control UpdateProgress to it after the lblGreetMsg Label. (Fig (13.h))



[Fig (13.h): UpdateProgress Control Example]

Make changes in the Source View of the Page as shown in the Fig (13.i). The ProgressTemplate is used to specify the text to be displayed when the UpdatePanel rendering is in progress.

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<br />
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Label ID="Label1" runat="server" Text="Enter User Name:"></asp:Label>
        <asp:TextBox ID="txtUserName" runat="server"></asp:TextBox>
        <br />
        <asp:Button ID="btnGreet" runat="server" Text="Greet User" />
        <br />
        <asp:Label ID="lblGreetMsg" runat="server"></asp:Label>
        <br />
        <asp:UpdateProgress ID="UpdateProgress1" runat="server"
            AssociatedUpdatePanelID="UpdatePanel1">
            <ProgressTemplate>
 An update is in progress...
</ProgressTemplate>
        </asp:UpdateProgress>
    </ContentTemplate>
</asp:UpdatePanel>
```

UpdateProgress Control ProgressTemplate

[Fig (13.i): Source View of UpdateProgress Control (ProgressTemplate)]

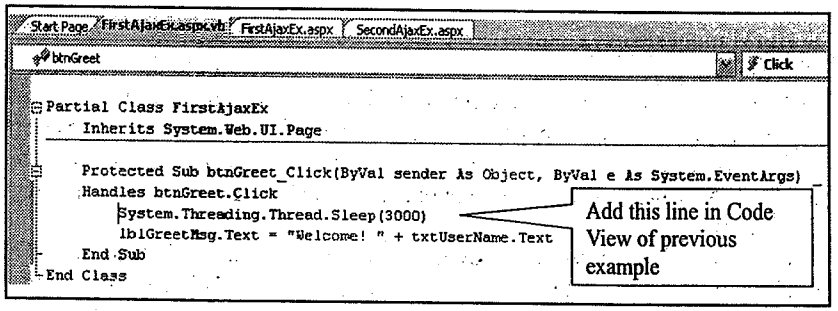
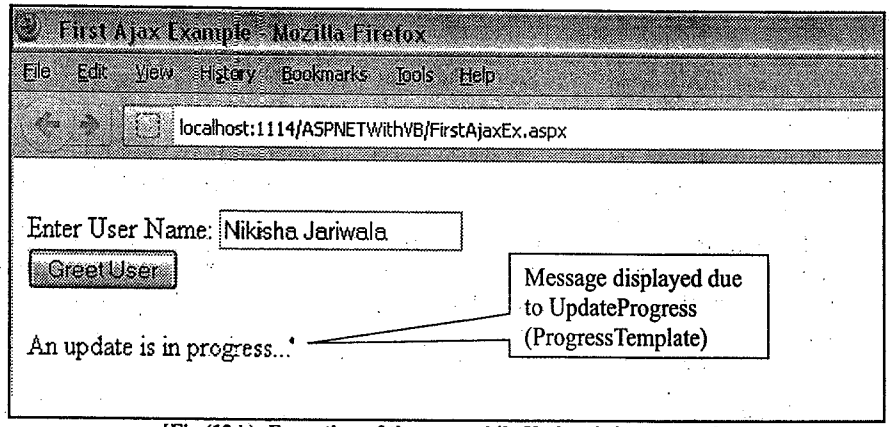
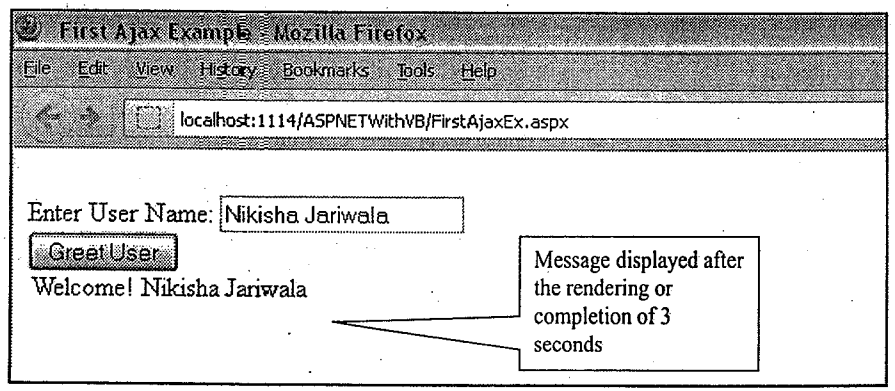


Fig (13.j): Code View of UpdateProgress Control Example]

By adding the specified line in Fig (13.j), the processing of the application will sleep for 3 seconds. So, you will able to see the effect of UpdateProgress Control.
Save and execute the application. (Fig (13.k)and fig.(13.l))



[Fig (13.k): Execution of the page while Update is in progress]

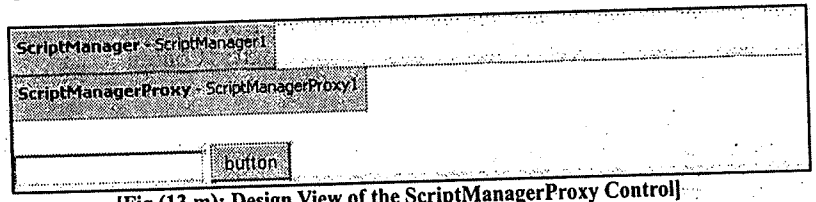


[Fig (13.l): Execution of the page after the 3 seconds is complete]

ScriptManagerProxy Control

You can add only one instance of the ScriptManager control to the page. The page can include the control directly or indirectly inside a nested component such as a user control, content page for a master page, or nested master page. In case, where a ScriptManager control is already on the page but a nested or parent component needs additional features of the ScriptManager control, the component can include a ScriptManagerProxy control. (Fig (13.m))

The ScriptManagerProxy works by detecting the main ScriptManager on your page at runtime, making sure that any references given to it are also given to the real ScriptManager.



[Fig (13.m): Design View of the ScriptManagerProxy Control]

Timer Control

Timer control allows you to do postbacks at certain intervals. If you use it together with UpdatePanels, it updates your page partially as well as it can also be used for the entire page postback.

The Timer control is a server control that embeds a JavaScript component into the Web page. The JavaScript component initiates the postback from the browser when the interval in the Interval property has elapsed. If you set the properties for the Timer control in code that runs on the server then those properties are passed to the JavaScript component. It has Tick() Event, which is raised each time the duration specified in the Interval property is reached.

Properties

Properties	Description
ID	It gets or sets the programmatic identifier assigned to the server control.
Interval	It specify the duration of the Tick() event in milliseconds.

[Table:9 Properties of Timer Control]

Methods

Methods	Description
DataBind()	It binds a data source to invoked server control and all its child controls.
Dispose()	It enables a server control to perform final clean up before it is released from memory.
Focus()	It sets input focus to a control.
HasControls()	It determines if the server control contains any child controls.

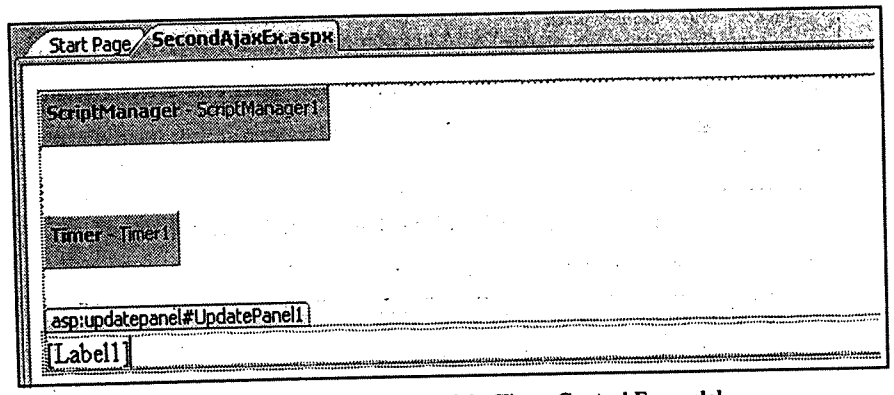
[Table:10 Methods of Timer Control]

- Default event:Tick()

Example

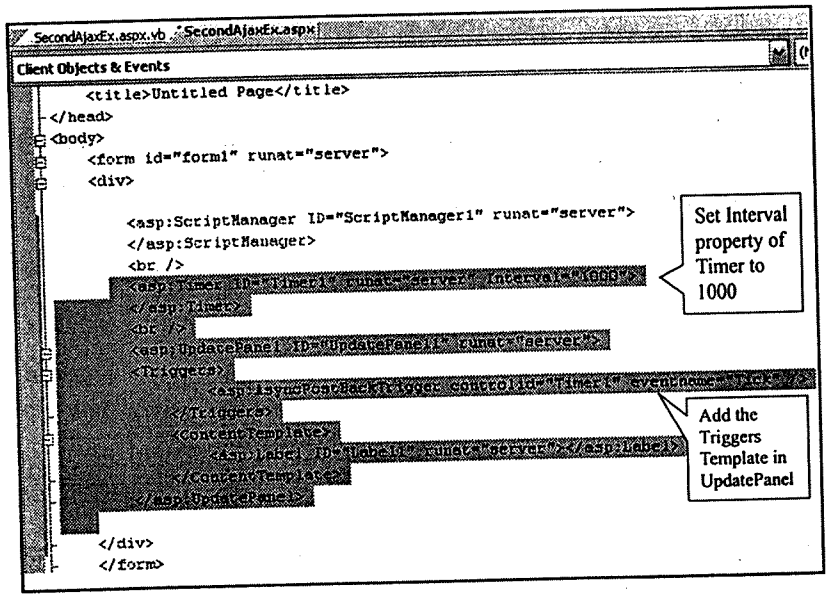
Create a web page which displays date and time. The time is updated every second.

- Create a design View of the page. (Fig (13.n))



[Fig (13.n): Design View of the Timer Control Example]

- Update the Source View (Fig (13.o))



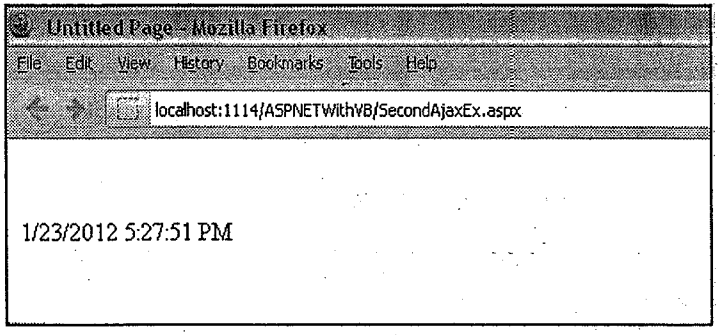
[Fig (13.o): Source View of the Timer Control Example]

- Write the code in Tick() Event of Timer control. (Fig (13.p))

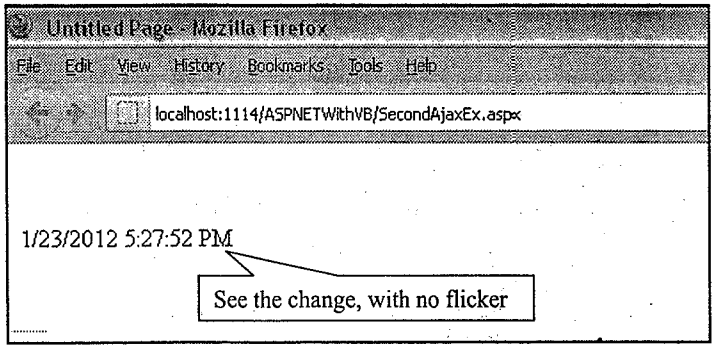
```
SecondAjaxEx.aspx.vb*  SecondAjaxEx.aspx
Timer1                                  Tic

Partial Class SecondAjaxEx
    Inherits System.Web.UI.Page

    Protected Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Timer1.Tick
        Label1.Text = DateTime.Now.ToString()
    End Sub
End Class
```

Fig (13.p): Code View of the Timer Control Example]

- Save and execute the page. (Fig (13.q) & (13.r))



[Fig (13.q): Page when executed]



[Fig (13.r): Page when executed after 1 second]

The controls explained in this chapter are the basic controls of the Ajax in ASP.NET. But Ajax Control ToolKit is available which provides more advanced functionality such as AutoComplete, Rating, TabControl, PasswordStrength and so on.

Exercise - 13

1. What is Ajax?
2. What is partial – page rendering?
3. Which object is used to implement Ajax?
4. Explain controls that are supported by ASP.NET for Ajax.
5. What is the importance of ScriptManager control?
6. What is the purpose of UpdatePanel and UpdateProgress control?
7. List out the templates of UpdatePanel control.
8. What is the significance of ScriptManagerProxy control?
9. Explain Timer control with example.

Acronyms

ABBR	Full Form	ABBR	Full Form
ABC	Abstract Base Class	HTML	HyperText Markup Language
ADO	ActiveX Data Object	HTTP	HyperText Transfer Protocol
ADT	Abstract Data Type	IDE	Integrated Development Environment
AJAX	Asynchronous Javascript And Xml	IIS	Internet Information Service
API	Application Programming Interface	IP	Internet Protocol
ASP	Active Server Pages	JIT	Just In Time
ATM	Automatic Teller Machine	JVM	Java Virtual Machine
BCL	Base Class Library	MAC	Message Authentication Code
BDP	Borland Data Provider	MSDE	Microsoft Data Engine
CLR	Common Language Runtime	MSIL	Microsoft Intermediate Language
CLS	Common Language Specification	ODBC	Open DataBase Connectivity
COM	Component Object Model	OLEDB	Object Linking & Embedding DataBase
CPU	Central Processing Unit	OS	Operating System
CSS	Cascading Style Sheet	RDO	Remote Data Object
CTS	Common Type System	SMTP	Simple Message/Mail Transfer Protocol
DAO	Data Access Object	SOAP	Simple Object Access Protocol
DNS	Domain Name Server	SQL	Structured Query Language
DOM	Document Object Model	UDDI	Universal Discovery and Description Integration
EDP	Event Driven Program	UI	User Interface
FCL	Framework Class Library	URI	Universal Resource Identifier
FTP	File Transfer Protocol	URL	Universal Resource Locator
GAC	Global Assembly Cache	VS	Visual Studio
GUI	Graphical User Interface	W3C	World Wide Web Consortium
		WAT	Website Administration Tool
		WFC	Windows Foundation Class
		WSDL	Web Service Description Language
		XML	eXtensible Markup Language

MCQ Questions

1. _____ is the engine that compiles and run the application.
   a. CLS b. CLR c. CTS d. JIT

2. Microsoft Intermediate Language is also called _____.
   a. Common Intermediate Language. b. Common Language Runtime
   c. Common Language Specification.

3. Which block of .NET framework provides Classes that encapsulates common function?
   a. CLS b. XML c. CTS d. BCL

4. Which block of .NET framework is also called Data Access Layer?
   a. XML b. FCL c. ADO.NET & XML d. BCL

5. _____ provides integrated and unified way of developing GUI.
   a. HTMLForms b. WebForms c. WinForms

6. Which component of .NET framework is browser-based User Interface?
   a. HTMLForms b. WebForms c. WinForms

7. _____ Standard used for storing, carrying and exchanging data over the internet.
   a. HTML b. XML c. FTP d. HTTP

8. Which are the applications that run on a Web Server and communicate with other applications?
   a. HTMLForms b. WebForms c. WinForms d. Web Services

9. The component of .NET framework that provides unification & interoperability is _____.
   a. CLS b. CLR c. CTS d. JIT

10. Which language category of CLS can use classes for inheritance?
    a. Consumer b. Extender c. Complaint Provider

11. Which is the smallest executable unit of code?
    a. Metadata b. DLL c. Assembly d. PE file

12. _____ is formed by collecting core components of Assembly.
    a. Metadata b. DLL c. Assembly d. Manifest

13. In .NET, Metadata is the ....
    a. Smallest executable unit. b. Declarative information which is produced at compile time.
    c. Logical grouping of classes.

14. Which component is formed within the Assembly?
    a. Manifest b. DLL c. Assembly d. Metadata

15. In .NET, version information is divided into _____ parts.
    a. 1 b. 2 c. 3 d. 4

16. The code that targets to CLR is called Managed code.
    a. True b. False

17. Which component of CLR provides CAS model?
    a. Type checker b. Debugger c. Security Engine d. Exception Manager

18. BCL is built on _____ nature of runtime.
    a. Object oriented b. Function oriented
    c. Procedural oriented

19. What is namespace?
    a. Logical grouping of function. b. Logical grouping of Assembly.
    c. Logical grouping of classes.

20. Which component of .NET framework provides Cross Language Integration?
    a. CLS b. CLR c. CTS d. JIT

21. Which is the protocol used to transfer information between Server and browser?
    a. HTML b. XML c. FTP d. HTTP

22. Which status code indicates redirection of the client to another URL?
    a. 5xx b. 4xx c. 3xx d. 2xx

23. Which status code indicates an error occur on server port?
    a. 5xx b. 4xx c. 3xx d. 2xx

24. For separating server-side code from client-side code on ASP.NET page, what programming model should you use?
    a. Seperation Model b. Code-Behind Model
    c. In-Line Model d. ClientServer Model

25. Which programming approach is supported by ASP.NET?
    a. Event Driven Programming b. Linear Programming

26. The first event triggered in an aspx page is_____.
    a. Page_Init() b. Page_Load() c. Page_Click()

27. What is the extension of the Page in ASP.NET
a. .asp b. .web c. .aspx

28. Which one of the following is the last stage of the Web forms lifecycle?
a. Page Load b. Page validations
c. Page Render d. Page Unload

29. In which event controls are fully loaded?
a. Page_Init() b. Page_Load()
c. Control Events d. Page_Unload()

30. In which of the following format, output will be rendered to browser when an .aspx page is requested from the web server?
a. ASP b. JSP c. XML d. HTML

31. Which of the following languages are used to write server side scripting in ASP.NET?
a. C# b. VB c. Both C# & VB d. C++

32. Can you prevent your class from being inherited and becoming a base class for some other classes?
a. True b. False

33. _____ Keyword in base class and _____ keyword in derived class is used to implement virtual functions in VB.NET.
a. Overloaded, Overloads b. Overridable, Overrides
c. Shared, Sharable

34. _____ allow objects to perform actions whenever a specific occurrence takes place.
a. Property b. Method c. Events d. Fields

35. Which method is overridden to create Destructor?
a. Finalize() b. Final() c. Finally()

36. What does an Interface can contain?
a. Property b. Method c. Events d. All

37. What is Abstract Base Class?
a. A class that cannot be instantiated b. A specifications for class members
c. A class used for Multiple Inheritance

38. With the help of _____ we can implement Multiple Inheritances.
a. Class b. Interface c. Abstract class d. Inherits

39. _____ Keyword is used to implement Multiple versions of the methods.
a. Overridable b. Overrides c. Overloads d. Shared

40. Which feature helps you to split single class to multiple classes?
a. Page Subclassing b. Partial Class
c. Multi Class d. Both a & b

41. Which event causes web page to be sent back to the server for immediate processing?
a. Validation Events b. Cached events
c. PostBack Events

42. Postback occurs in which of the following forms.
a. HTMLForms b. WebForms c. WinForms

43. Which is the base class of ASP.NET Web Form Class?
a. System.Web.UI.Page b. System.Web.UI.Form
c. System.Web.GUI.Page d. System.Web.Form

44. _____ is the DataType return in IsPostback property.
a. Bit b. Boolean c. Int d. Object e. String

45. In the Design view in Visual Studio of an ASP.NET web page, what is the easiest way to create an event handler for the default event of an ASP.NET server control?
a. Open the code-behind page and write the code b. Right-click the control and select Create Handler
c. Drag an event handler from the ToolBox to the desired control d. Double-click the control

46. How to identify that the page is Postback?
a. IsPostBack property b. Smart Navigation property
c. AutoPostBack property d. By using session

47. _____ allows executing ASP.NET application level events and setting application-level variables.
a. Application object b. Global.asax file
c. Impersonation d. Web.config file

48. How to force all the validation control to run?
a. Set EnableClientScript true b. Page.IsValid()
c. Page.Validate()

49. From which class global.asax file derived?
a. HTTPApplication class b. HTTPSessionState class

50. Why Global.asax is used?
a. Declare Global variables b. Implement application and session level events
c. No use

51. You can have only one Global.asax file per project.
a. True b. False

52. Which event is fired when a new user visits the application web site?
a. Application_Init b. Application_Start
c. Application_BeginRequest d. Session_Start

53. Which event procedure resides in the Global.asax file?
a. Page_Error b. Global_Error
c. Application_ServerError d. Application_Error

54. Which window lists the solution name, application name, and all the pages that are used in the application?
a. Project Explorer b. Solution Explorer
c. Server Explorer d. Object Browser

55. Which directive can be used to implement Cross Page Posting?
a. @Page b. @PageType
c. @PreviousPageType d. @Reference

56. Which directive is used to associate namespaces and classes?
a. @Page b. @Register
c. @PreviousPageType d. @Reference

57. Which property of Page Class is used to check, that validations are successfully validated or not?
a. IsValid b. Valid c. ErrorPage d. IsPostBack

58. _____ Property of the Page is used to define the page that handles error at page level.
a. IsValid b. Valid c. ErrorPage d. IsPostBack

59. Which Method of Page Class is used to search server control in current container?
a. FindControl() b. Find() c. Search() d. SearchControl()

60. Which of the following is used for Server-side comment?
a. <!-- --> b. <%-- --%> c. <-- -->

61. In ASP.NET, controls must appear in _____ tag.
a. <form > b. <control Runat="Server">
c. <control > d. <form Runat="Server">

62. How do you get information from a form that is submitted using the "post" method?
a. Request.QueryString b. Request.Form
c. Response.Write d. Response.WriteLn

63. What is the difference between Response.Write() andResponse.Output.Write()?
a. Response.Output.Write() allows you to buffer output
b. Response.Output.Write() allows you to write formatted output
c. Response.Output.Write() allows you to flush output
d. Response.Output.Write() allows you to stream output

64. Which ASP.NET object is used to get information about the web servers?
a. Application b. Response c. Request d. Server

65. What is the significance of Server.MapPath()?
a. Returns the Virtual Path of the web folder
b. Maps the specified virtual path to Physical path
c. Returns the physical file path that corresponds to virtual specified path
d. All the above

66. In case of _____, there is a round trip and hence puts a load on server.
a. Server.Transfer() b. Response.Redirect()

67. Which one of the following has a parameter called as "preserveForm"?
a. Server.Transfer() b. Response.Redirect()

68. _____ is the message sent from the web server to the client.
a. Application b. Response c. Request d. Server

69. When a browser asks for a page from a server, it is called _____.
a. Application b. Response c. Request d. Server

70. Which property of the Request object gives host address?
a. HostAddress b. HostName c. UserAddress d. UserHostAddress

71. Which property of the Request object gives information of user's computer?
a. HostAgent b. HostInfo c. UserAgent d. UserInfo

72. Which is the instance of System.Web.HttpServerUtility class?
a. Application b. Response c. Request d. Server

73. Which property of Server object changes an ordinary string into a string with legal URL characters?
a. UrlEncode() b. UrlDecode() c. HtmlEncode() d. HtmlDecode()

74. Which of the following can transfer the execution control to different domain?
a. Server.Transfer() b. Response.Redirect()

75. Within which namespace, Control Class is available?
a. System.Web.UI b. System.Web.UI.WebControl
c. System.Object d. System.Web

76. Which method of Control Class is used to bind Data Source to the controls?
a. DataBound() b. DataBind() c. BindingSource() d. SourceBind()

77. What is the return type of HasControls() method of Control class?
a. Byte b. Bit c. Boolean d. Object

78. Which is the namespace for Html Server control classes?
a. System.Web.UI.HtmlControl b. System.Web.UI.Control
c. System.Object d. System.Web

79. Which is the base class of all the Input Html Server control Classes?
a. HtmlInput b. HtmlInputControl

80. Which property of the HtmlControl class provides name of the tag of particular Html control?
a. Tag b. HtmlTag c. TagName d. HtmlName

81. Which of the following are the common events of Html Server Control?
a. ServerClick() b. ServerChange() c. Both a & b

82. Which is the base class of all the Web Server Control Classes?
a. WebControl b. Control c. Object

83. Which property of WebControl class specifies that the code is to be run on server and not on client?
a. Server b. RunAt c. RunAtServer

84. Which is the base class for all the Standard Control classes?
a. WebControl b. Control c. Object

85. Which controls are used to only display text on the Page?
a. Label b. Literal c. TextBox d. Both a & b

86. Which property of TextBox is used to change behavior of the control?
a. TextType b. Mode c. TextMode d. Behavior

87. It is possible to set TextBox control to Read Only.
a. True b. False

88. Which property of Button control can be used to prevent Page Validation?
a. Validate b. CausesValidation c. IsValid d. PreventValidation

89. Which property of Button control is used to set the Page to which it is redirected when Button is clicked?
a. PostUrl b. IsPostBack c. PageUrl d. PostBackUrl

90. Which property of Button control is not available with LinkButton control?
a. TextAlign b. UseSubmitBehavior c. Text d. PostBackUrl

91. Which property of ImageButton control provides path to an image?
a. Src b. Path c. Url d. ImageUrl

92. Which property of ImageButton control is used to display text when image is not available?
a. Alt b. AlternateText c. Text d. AltText

93. Which Html tag is similar to HyperLink control of ASP.NET?
a. <a> b. <Link> c. <Href>

94. Which property of HyperLink control set the Url to link page?
a. Href b. Link c. Url d. NavigateUrl

95. What is the difference between CheckBox control & RadioButton control?
a. CheckBox allows to select only one value, while RadioButton allows to select multiple value
b. CheckBox allows to select multiple value, while RadioButton allows to select only one value

96. Which property of CheckBox & RadioButton is used to determine, the selected value?
a. Checked b. Selected c. Clicked d. IsChecked

97. Which property of RadioButton indicates the Group within which it belongs?
a. Group b. GroupName c. GrpName

98. Which is the default event of Button Control?
a. Click() b. ButtonClick() c. DblClick() d. DoubleClick()

99. Which property of Web Server Control indicates that an automatic PostBack to server should occur or not?
a. IsPostBack b. AutoPostBack c. PostBack d. IsAutoPostBack

100. Which property of List Controls allows it to interact with Database?
a. DataSource b. DataTextField c. DataValueField d. All

101. Which Html tag is similar to <asp:ListItem/> in ASP.NET?
a. <Select> b. <Option/> c. Both a & b

102. Which property of ListItem class is used to indicate whether the Item is selected or not?
a. Selected b. IsChecked c. Checked d. IsSelected

103. It is possible to select multiple items from the ListBox control.
a. True b. False

104. Which of the following List controls is single selection drop down list?
a. RadioButtonList b. ListBox c. DropDownList d. BulletedList

105. Which of the following is true for RadioButtonList & CheckBoxList controls?
a. CheckBoxList allows to select only one value, while RadioButtonList allows to select multiple value
b. It is possible to set the direction in which the values are displayed in both the controls.

106. Which property of BullettedList Control indicates the type of the list?
a. BulletStyle b. BulletType c. BulletMode d. ListType

107. Which property of the BullettedList control determines the type of rendering of the item in the list?
a. DisplayMode b. BulletMode c. DisplayStyle d. BulletStyle

108. Is it possible to attach data source with List control?
a. No b. Yes

109. An alternative way of displaying text on web page is using
a. <asp:Label> b. <asp:ListItem> c. <asp:Button>

110. For ASP.NET web application a graphics designer created elaborate images that show the product lines of the company. Some graphics of the product line are rectangular, circular, and others are having complex shapes. If these images are used as a menu on Web site. What is the best way of incorporating these images into Web site?
a. Use ImageButton and use the x- and y-coordinates that are returned when the user clicks to figure out what product line the user clicked.
b. Use the Table, TableRow, and TableCell controls, break the image into pieces that are displayed in the cells, and use the TableCell control's Click event to identify the product line that was clicked.
c. Use the MultiView control and break up the image into pieces that can be displayed in each View control for each product line. Use the Click event of the View to identify the product line that was clicked.
d. Use an ImageMap control and define hot spot areas for each of the product lines. Use the PostBackValue to identify the product line that was clicked.

111. While developing ASP.NET web application, if there is a need to display a list of parts in a master/detail scenario where the user can select a part number using a list that takes a minimum amount of space on the Web page. When the part is selected, a DetailsView control displays all the information about the part and allows the user to edit the part. Which Web control is the best choice to display the part number list for this scenario?
a. TextBox b. FormView c. RadioButtonList d. DropDownList

112. ImageMap control is used to create _____.
a. Map b. ImageSpots c. HotSpot

113. Which object is used by PostedFile property of FileUpload control to work with Uploaded file?
a. HttpPostedFile b. HttpFile c. PostedFile d. Files

114. Panel control is similar to _____ tag of HTML.
a. b. c. <panel> d. <pan>

115. Panel control can have grouping caption on it.
a. False b. True

116. Which control does not give any visible output?
a. PlaceHolder b. Panel c. GroupBox d. FileUpload

117. Which of the following are container controls?
a. PlaceHolder b. Panel c. MultiView d. All

118. Which property of MultiView control is used to set the active View?
a. ActiveViewIndex b. ViewIndex c. ActiveIndex d. None

119. Which control can be used to create Wizard like layout?
a. PlaceHolder b. Panel c. MultiView d. All

120. Which class contains RowSpan & ColumnSpan property?
a. Table b. TableRow c. TableCell d. All

121. Which of the following are Rich Controls?
a. Calender b. AdRotator c. GridView d. Both a & b

122. Is it possible to select whole week or month in Calendar control?
a. Yes b. No

123. Which is the default event of Calendar control?
a. SelectedDateChanged() b. SelectionChanged()
c. SelectedIndexChanged() d. SelectionDateChanged()

124. _____ Property of AdRotator control contains path to XML file for Advertisement information.
a. AdvertisementFile b. XMLFile c. AdsFile d. None

125. Which type of expression is used to validate complex string patterns like email-address?
a. Extended Expression b. Basic Expression c. Regular Expression d. Irregular Expression

126. **Which of the following control is used to validate that two fields are equal or not?**
a. RegularExpressionValidator b. ConpareValidator
c. RequiredFieldValidator d. None

127. **_____ Validation control is used for "PatternMatching".**
a. RegularExpressionValidator b. ConpareValidator
c. RequiredFieldValidator d. None

128. **_____ is a property common to every validation control.**
a. ValidationExpression b. InitialValue
c. ValueToCompare d. ControlToCompare
e. ControlToValidate

129. **Which control display validation errors in a central location or display a general validation error description?**
a. RegularExpressionValidator b. ConpareValidator
c. RequiredFieldValidator d. ValidationSummary

130. **Which is the base class of all the Validation Control classes?**
a. BaseValidator b. ValidatorControl
c. ValidationControl d. None

131. **Which property of BaseValidator class display validation error description in ValidationSummary control?**
a. Description b. ErrorDescription c. ErrorMessage d. Text

132. **Which property of BaseValidator class indicates that the input control passes validation or not?**
a. IsValid b. Valid c. ErrorPage d. IsPostBack

133. **Which of the following types are supported by Type property of RangeValidator control?**
a. String b. Integer c. Currency d. All

134. **Which of the following can be used to specify the range in RangeValidator control?**
a. MinimumValue, MaximumValue b. Min, Max
c. Minimum, Maximum d. MinVal, MaxVal

135. **Is it possible to check equality of value within the control and some predefined value?**
a. Yes b. No

136. **Which of the following operators can be used with CompareValidator control?**
a. Equal b. GreaterThan c. DataTypeCheck d. All

137. **CustomValidator control can work with both Client & Server side validation Script.**
a. Yes b. No

138. **Which event of CustomValidator control is used to perform validation at Server Side?**
a. ServerClick() b. ServerChange() c. ServerValidate()

139. **Which of the following are Navigation Controls?**
a. Menu b. TreeView c. SiteMapPath d. All

140. **Which of the following file is required to create Site Map with SiteMapPath control?**
a. Web.config b. Web.sitemap c. XMLfile

141. **Which of the following Navigation Control is used to create hierarchical structure as table of content?**
a. Menu b. TreeView c. SiteMapPath d. All

142. **Nodes within the TreeView control are represented by _____ object.**
a. TreeNode b. TreeNodes c. Nodes Collection

143. **_____ Property of TreeNode object gives the path from root node to current node.**
a. Path b. TreePath c. Value d. ValuePath

144. **It is possible to set web.sitemap file as a DataSource for TreeView control.**
a. False b. True

145. **SiteMapPath control nodes can also be rendered as HyperLink to the page.**
a. True b. False

146. **It is not possible to display Items in Horizontal direction with Menu control.**
a. True b. False

147. **When you use built – in Login control it automatically creates its own database.**
a. True b. False

148. **It is possible to customize the Login controls.**
a. True b. False

149. **Which is the default event of Login control?**
a. LoggedIn() b. Authenticate() c. LoggedOut()

150. **Which of the following control contains LoggedInTemplate & AnonymousTemplate?**
a. Login b. LoginStatus c. LoginView d. LoginName

151. **Which of the following control displays the name contained in the User property of the page?**
a. Login b. LoginStatus c. LoginView d. LoginName

152. If non-reversible encryption is used for password encryption then PasswordRecovery control generates new password.
a. True b. False

153. Which property of PasswordRecovery control defines the characteristics of the email message used to send new or recovered password?
a. Mail b. MailInfo c. Email d. MailDefinition

154. Which of the following control have SuccessPageUrl Property?
a. PasswordRecovery b. ChangePassword c. Both a & b d. None

155. Which of the following control have DestinationPageUrl Property?
a. CreateUserWizard b. Login c. Both a & b d. None

156. By default all login controls works with Froms Authentication mode.
a. True b. False

157. Which of the following allows you to create consistent layout of the page?
a. Master Page b. Theme c. CSS d. None

158. It is possible to execute Master Page individually.
a. True b. False

159. One website can have more than one Master Pages.
a. True b. False

160. Master Page must have at least one _____.
a. ContentPlaceHolder b. PlaceHolder c. Header d. Footer

161. It is possible to bind .aspx page with the Master Page.
a. True b. False

162. Which property of the page is set when Master Page is bind with page?
a. MasterPage b. MasterFile c. MasterPageFile d. None

163. _____ Control is added in the Standard tab when the Master Page is created.
a. ContentPlaceHolder b. PlaceHolder c. Table d. HyperLink

164. Which event of Page is used to attach Master Page at runtime?
a. PreInit() b. Load() c. PreRender() d. UnLoad()

165. It is possible to create one master page from another master.
a. True b. False

166. _____ are skin templates that allow you to define the look of pages and controls.
a. MasterPage b. Themes c. CSS d. None

167. _____ is a set of properties and templates that can be used to standardize the size, font, and other characteristics of controls of a page.
a. MasterPage b. Themes c. CSS d. Skin

168. The skin with SkinID is called _____ Skin.
a. Default b. IDSkin c. Named d. None

169. Which of the following folder is created, when skin is added to the website?
a. App_Templates b. App_Themes c. App_Skins d. All

170. Runat attribute is not written within .skin file, when skin is created.
a. True b. False

171. Which property of the page is set to attach .skin file with Page?
a. SkinID b. Theme c. SkinFile d. Skin

172. It is possible to set theme at runtime.
a. True b. False

173. It is possible to apply theme globally within web.config file.
a. True b. False

174. Which property of the control is set to attach Named Skin with it?
a. Skin b. Theme c. SkinFile d. SkinID

175. Which template is added to website to create CSS?
a. Css file b. Style Sheet c. XML file d. None

176. Which property of the Page can be set to apply Style Sheet to it?
a. StyleSheet b. Theme c. StyleSheetTheme d. CSSTheme

177. Which property of the control is set to specify style sheet class?
a. StyleClass b. CssClass c. StyleSheetClass d. None

178. Which of the following model provides the framework for accessing any type of data that can be used with ASP.NET pages?
a. ADO b. DAO c. ADO.NET d. RDO

179. _____ was the first data access model created for local databases.
a. ADO b. DAO c. ADO.NET d. RDO

180. Which of the following model is Disconnected Data Model?
a. ADO b. DAO c. ADO.NET d. RDO

181. Which of the following model uses XML while connecting with database?
a. ADO.NET b. DAO c. ADO d. RDO

182. Which type of library is present in ADO.NET?
a. Procedural oriented b. Function oriented c. Object oriented d. All

183. _____ is used for providing and maintaining the connection to the database.
a. Data Provider b. Data Set c. Data Adapter d. XML

184. Which of the following class are available with System.Data.SqlClient namespace?
a. Data Reader b. Data Set c. Data Adapter d. Both a & c

185. Which object is used by command object, so that it will know the database on which it has to execute query?
a. Data Reader b. Connection c. Data Adapter d. None

186. BeginTransaction() method is used to create Transaction object.
a. True b. False

187. Which of the following object is used to retrieve subset of data from database?
a. Data Reader b. Connection c. Data Adapter d. Command

188. _____ Property of Command is used to determine how to interpret command text.
a. CommandText b. Connection c. CommandType d. Transaction

189. _____ is used to specify the placeholders instead of direct values in command object.
a. CommandText b. Parameters c. Command d. Transaction

190. Which of the following method of Command object will execute the SQL statement that returns singleton value?
a. ExecuteNonQuery() b. ExecuteReader() c. Prepare() d. ExecuteScalar()

191. Which of the following method of Command object will create DataReader object?
a. ExecuteNonQuery() b. ExecuteReader() c. Prepare() d. ExecuteScalar()

192. Which character is used as prefix for defining Parameter in SQL Server?
a. @ b. : c. # d. None

193. To associate parameter with command object, you need to add parameter in the Parameters collection of Command object.
a. True b. False

194. Which of the following object is forward-only, read-only record set?
a. DataSet b. DataReader c. DataAdapter d. Command

195. Which method of DataReader object is used to read next record within Data Reader?
a. NextResult() b. Next() c. Read() d. ReadNext()

196. Which property of DataReader object specifies that rows are selected or not?
a. SelectedRows b. GetRows c. IsRowsSelected d. HasRows

197. Which method of DataReader object is used to navigate from one record set to another?
a. NextResult() b. Next() c. Read() d. ReadNext()

198. Which of the following object acts as a bridge between data source and Dataset?
a. DataSet b. DataReader c. DataAdapter d. Command

199. Which method of DataAdapter object is used to populate DataSet object with data retrieved from data source?
a. Populate() b. Fill() c. FillData() d. PopulateData()

200. DataAdapter object will Open connection implicitly by itself; there is no need to explicitly open connection.
a. True b. False

201. Which of the following object is used to implement changes in the database that is done in DataSet?
a. CommandBuilder b. Command c. DataAdapter d. DataReader

202. It is mandatory to pass DataAdapter object within CommandBuilder object when it is created?
a. True b. False

203. Which of the following object is in-memory representation of the data?
a. CommandBuilder b. DataSet c. DataAdapter d. DataReader

204. DataSet object is made up of DataTable, Constraints and DataRelation objects.
a. True b. False

205. It is possible to create relationship between tables in the DataSet object.
a. True b. False

206. Which method of DataRow object is used to retrieve child rows as per rows of parent table?
a. HasRows() b. GetRows() c. ChildRows() d. GetChildRows()

207. DataView object is only used to create View from the DataSet.
a. True b. False

208. Which of the following object provide custom View of DataTable?
a. DataView b. DataSet c. DataAdapter d. DataReader

209. DataView can be used for sorting and searching data from DataSet.
a. True b. False

210. Which of the following property of DataView is similar to Where clause of SQL query?
a. Where b. RowFilter c. Filter d. WhereClause

211. Which keyword is needed to perform sort in descending order in DataView Sort property?
a. Descend b. Dsc c. Desc d. Descending

212. It is possible to perform search on only single column within DataView.
a. True b. False

213. Which of the following folder is created automatically when the database is created?
a. App_Database b. Database c. Data d. App_Data

214. Which of the following window of Visual Studio is used to work with Database?
a. Server Explorer b. Properties c. Solution Explorer d. Document Explorer

215. Which of the following is used to implement Single Value Data Binding?
a. <%#Expression %> b. <%$Expression %> c. Both a & b d. None

216. To evaluate #Expression, it is mandatory to use DataBind() method of the Page.
a. True b. False

217. What is the use of $Expression Single Value Data Binding?
a. To bind Application Setting b. To bind Connection String
c. Both a & b d. None

218. Which of the following controls can be used with Repeated Value Data Binding?
a. RadioButtonList b. BulletedList c. ListBox d. All

219. Which of the following properties of controls can be used for Repeated Value Data Binding?
a. DataTextField b. DataValueField c. DataSource d. All

220. Which of the following collections can be used with controls for Repeated Value Data Binding?
a. ArrayList b. SortedList c. DataSet d. All

221. Which of the following events are available with all Data Controls?
a. ItemCreated b. CancelCommand c. EditColumn d. Both a & b

222. Which Data Controls are Read only?
a. GridView b. ListBox c. Repeater d. Both b & c

223. Which Data Controls supports selection, Deletion, Editing operations?
a. GridView b. ListBox c. DetailsView d. Both a & c

224. Which of the following control can generate records in horizontal & vertical layout?
a. GridView b. ListBox c. DataList d. Both b & c

225. Which of the following Data Control is "Lookless" control?
a. Repeater b. ListBox c. DataList d. Both b & c

226. Which of the following Template is mandatory to display data in Repeater control?
a. HeaderTemplate b. ItemTemplate c. FooterTemplate d. Both b & c

227. Which of the following Data Control allows you to add new record within the database?
a. FormView b. ListBox c. Repeater d. None

228. Which of the following control display single record at a time?
a. FormView b. ListBox c. Repeater d. GridView

229. It is possible to bind Where clause with the Data Source of the Data Control were value can be retrieved from QueryString or any control.
a. True b. False

230. If insert, update, delete is to be performed with the help of Data Control, the data source bound with the control can automatically generate these queries.
a. True b. False

231. Which of the following is not a member of Command object?
a. ExecuteScalar b. ExecuteReader c. Open d. ExecuteNonQuery

232. Which of the following is faster and consume lesser memory?
a. DataReader b. DataSet

233. Which of the following does not have any visible interface?
a. GridView b. FormView c. Repeater d. DataList

234. In ASP.NET web application, you want to display a list of employee on a Web page. The employee list displays 10 records at a time, and you require the ability to edit the record. Which Web control is the best choice for this scenario?
a. GridView b. FormView c. Repeater d. Table

235. What is the best way to store the connection strings?
a. Config file b. Database c. Text Files d. Session

236. In ASP.NET web application, you have a DataSet containing a Customer DataTable and an Order DataTable. You want to easily navigate from an Order DataRow to the Customer who placed the order. What object will allow you to easily navigate from the Order to the Customer?
a. DataColumn object b. DataTable c. DataRow d. DataRelation

237. What is the property name of a GridView that regulates the sorting?
a. EnableSorting b. AllowSorting c. Sorting d. DisableSorting

238. Which of the following is not true about ADO.NET?
a. ADO.NET enables to create distributed data sharing applications
b. ADO.NET uses XML to transfer data across applications and data source
c. ADO.NET doesn't support disconnected architecture
d. The classes of ADO.NET are defined in the System.Data namespace

239. Which of the following illustrates the benefit of ADO.NET?
a. Interoperability
b. It uses DataSet to represent data in memory that can store data from multiple tables and multiple sources.
c. Disconnected data access
d. All of the above

240. DataAdapter object populates a DataSet and resolves updates with the data source.
a. Yes b. No

241. Which of the following is not true for ADO.NET DataSet?
a. DataSet provides a disconnected view of a data source
b. Dataset enables to store data from multiple tables and multiple sources
c. We can create relationship between the tables in a DataSet
d. All of the above are true

242. Which of the following method of the command object is best suited when you have aggregate functions in a SELECT statement?
a. ExecuteNonQuery b. ExecuteScalar c. Open d. ExecuteReader

243. Which of the following is not the method of DataAdapter?
a. FillSchema b. ReadData c. Fill d. Update

244. Which of the following is the method provided by the DataSet object to generate XML?
a. ReadXML b. WriteXML c. GetXML d. All

245. From performance point of view which is the fastest?
a. DataList b. GridView c. Repeater d. DetailsView

246. Which one of the class contains the objects that we use to connect to a data source via an OLE-DB provider?
a. System.XML b. System.Data.SQLClient
c. System.Data d. System.Data.OLEDB

247. Which method of command object doesn't return any row?
a. ExecuteNonQuery b. ExecuteScalar c. ExecuteReader

248. We can have multiple machine.config file on our computer.
a. True b. False

249. Machine.config file is used to configure all the application according to a particular machine.
a. True b. False

250. Which of the following is used to set the base configuration for all .Net assemblies running on the server?
a. Web.config b. App.config c. Machine.config

251. Which element of web.config setting is used to set the build type to debug or release?
a. Authentication b. Authorization c. Trace d. Compilation

252. Which of the following class provides properties to access information of <appSettings> element of web.config file?
a. ConfigurationManager b. WebManager c. Configuration d. None

253. Which tag of web.config file contains entire configuration specific to ASP.NET?
a. <appSettings> b. <System.web> c. <compilation> d. None

254. Which of the following is the graphical configuration tool of ASP.NET?
a. Config b. DLL c. GAC d. WAT

255. How many tabs are available in Website Administration Tool?
a. 1 b. 3 c. 4 d. 2

256. It is possible to create users through WAT?
a. True b. False

257. WAT allows you to set default error page?
a. True b. False

258. Which of the following is used to manage states in asp.net application?
a. Session b. Application c. ViewState d. All

259. Which of the following Cache types are supported by ASP.NET?
a. Output Caching b. Data Caching c. Both a & b d. None

260. Default Session data is stored in _____ in ASP.NET.
a. StateServer b. InProcess c. Session d. All

261. Which of the following object help you to maintain data across users?
a. Application b. Session c. Response d. Request

262. Which of the following ASP.NET object encapsulates the state of the client?
a. Application b. Session c. Response d. Request

263. Which of the following is the Mode of storing ASP.NET session?
a. InProc b. StateServer c. Sql Server d. All

264. Which of the following is not the way to maintain state?
a. Cookies b. ViewState c. Hidden Fields d. Request

265. Which OutputCache attribute is used to cache multiple responses for a single web form based on an HTTP post parameter or QueryString?
a. VaruByParam b. VaryByHeader c. VaryByCustom d. Shared

266. Which one is not accessed from all your pages on your website?
a. Cookies b. ViewState c. Session d. Application

267. Which of the following holds true for caching?
a. Caching an item incurs considerable overhead
b. Cache a web page if it is frequently used and does not contain data that frequently changes
c. It stores frequently used items in memory so that they can be accessed more quickly
d. All of the above are true

268. How do you preserve persistent data in a web application?
a. Cookies b. Application c. Session d. All

269. By default, ASP.NET store SessionIDs in _____.
a. Cookies b. Application c. Session d. DataBase

270. _____ method(s) are used with Application object to ensure only one process accesses a variable at a time.
a. Synchronize() b. Lock() & UnLock() c. ThreadLock d. All

271. You need to store state data that is accessible to any user who connects to your Web application. Which object should you use?
a. Response.Cookies b. Response.ViewState c. Session d. Application

272. Saving ViewState information adds to the amount of data that must be transmitted back to the server with each request.
a. True b. False

273. Session_End is fired in which of these SessionState modes?
a. StateServer b. InProc c. SQLServer d. None

274. A project on which you are working calls for you to store small amount of frequently changing information about a page on the client. For this project, security is not a worry. Which is the best method to use?
a. Hidden Field b. URL c. Cookie d. QueryString

275. Where is the ViewState information stored?
a. Hidden Field b. URL c. Cookie d. QueryString

276. In which mode of ASP.NET session, session state is serialized and stored in a separate process.
a. InProc b. StateServer c. SQLServer d. Cookie

277. The small text stored on client machine by web application is called _____.
a. Hidden Field b. URL c. Cookie d. QueryString

278. What does Count property of Session Specifies?
a. Get the number of items within the Session
b. Get Count of Session created for user
c. Get number of users using Session
d. None of the above

279. Which of the following ends the Session?
a. If user closes the browser b. If TimeOut property is reached
c. If Abandon () method is called d. All

280. _____ allows you to search the information about all profiles.
a. ProfileManager b. User Profile c. Session d. Cookies

281. By default, Anonymous user tracking is _____.
a. On b. Off

282. A base _____ class defines the interface, or contract, that all profile providers must implement.
a. Profile b. UserProfile c. ProfileProvider d. SqlProvider

283. It is possible to nest User Profile group under another User Profile group.
a. True b. False

284. Which of the following Output Caching configuration are implemented in ASP.NET?
a. Page Level Configuration b. Application Level Configuration
c. Machine Level Configuration d. Both a & b

285. What is the extension of a web user control file?
a. .asmx b. .aspx c. .ashx d. .ascx

286. _____ Directive informs the compiler of any custom server control added to the page.
a. Register b. Page c. Master d. Custom

287. Web User control can be executes individually, without any page.
a. True b. False

288. _____ Directive is used for Web User Control.
a. Register b. WebControl c. UserControl d. Control

289. <GreetControl:Welcome ID ="W1" runat ="server" />
In the above line GreetControl is _____.
a. TagPrefix b. TagName c. Src d. Control Name

290. The user control does not have html, body, or form elements in it.
a. True b. False

291. Choose the correct option for web services.
a. Web service can be consumed by clients who are not .NET Platform
b. For web service, you need the client to be .NET compliant
c. Web service is faster than remoting
d. None of the above

292. _____ is the Web Service interface description.
a. W3C b. SOAP c. HTTP d. WSDL

293. WSDL is in _____ Language.
a. HTML b. XML c. XAML d. UDDI

294. _____ is a directory service where web services can be registered.
a. W3C b. SMTP c. XAML d. UDDI

295. Which of the following is the extension of Web Service?
a. .asmx b. .aspx c. .ashx d. .ascx

296. <WebMethod> attribute is used to _____.
a. Declare web service.
b. Declare method as web service.
c. Attach web service with any of the web application.
d. None of the above

297. Which of the following is the default files created, on the creation of web service?
a. Service.asmx b. Service.vb c. Default.aspx d. Both a & b

298. To use web service, we need to add web reference of web service in web application.
a. True b. False

299. The error which occurs when the code is executing, is called _____.
a. Logic Error b. Runtime Error c. Syntax Error d. None

300. What is the meaning of "On Error Goto 0"?
a. Disables exception handler in the current procedure
b. It Clears Error object.
c. If error occurs, it set the program pointer to line no. 0
d. Both a & b
e. None of the above

301. Which object contains information about the error, when it occurs?
a. Error b. Err c. Er d. None

302. What is the significance of Resume Next?
a. Continues execute from next line exactly below the line in which error occurs.
b. Suspend program execution, wait to change the code & then resume execution.
c. Resumes execution from particular line.
d. None of the above.

303. It is possible to have multiple Catch blocks in Structured Exception Handling.
a. True b. False

304. It is possible to use Err object with When clause in Structured Exception Handling.
a. True b. False

305. Which of the following block of Structured Exception Handling is always executes, if it is written?
a. Try b. Catch c. Finally d. All

306. Which of the following keyword is used to throw exception?
a. Try b. Catch c. Finally d. Throw

307. _____ Attribute of @page directive is used to set the page to be shown when any error occur.
a. errorPage b. Error c. PageError d. None

308. Which of the following can be used to handle Application level Exception?
a. Global.asax b. Web.config c. Both a & b d. None

309. It is possible to customize Error page according to the status code of error.
a. True b. False

310. Can we use multiple ScriptManager on the page?
a. No b. Yes

311. Can you nest UpdatePanel within each other?
a. Yes b. No

312. Describe the technology that makes up AJAX?
a. XMLHttpRequest b. DOM c. JavaScript d. HTML e. All

313. Which of the following are server controls that come up with ASP.NET AJAX?
a. ScriptManager b. UpdatePanel c. Timer d. All

314. What is the difference between synchronous postback and asynchronous postback?
a. Asynchronous postback renders only the required part of the page; whereas, synchronous postback renders the entire page for any postback.
b. Asynchronous postback executes only one postback at a time, that is, if you have two buttons doing asynchronous postback, the actions will be performed one by one; whereas, synchronous postback executes all the actions at once.
c. Asynchronous postback only modifies the update panel that raises the postback; whereas, synchronous postback modifies the entire page.
d. All of the above.

315. Which of the following control provides status information of partial page rendering?
a. ScriptManager b. UpdatePanel c. UpdateProgress d. Timer

316. If the UpdateMode property is set to Conditional, then content of UpdatePanel Control is updated under which circumstances?
a. When you call the Update method of the UpdatePanel control explicitly.
b. When the UpdatePanel control is nested inside another UpdatePanel control and the parent panel is updated.
c. When a postback is caused by a control that is defined as a trigger by using the Triggers property of the UpdatePanel control.
d. All of the above.

317. Which property of UpdateProgress control specify the time after which the ProgressTemplate is displayed?
a. Time b. Interval c. DisplayAfter d. None

318. What is the significance of ScriptManagerProxy control of Ajax?
a. It works by detecting ScriptManager control on the page.
b. It is the Parent control of ScriptManager control.
c. Both a & b
d. None of the above

MCQ Answer

1. b	2. a	3. d	4. c	5. c	6. b
7. b	8. d	9. a	10. b	11. c	12. d
13. b	14. a	15. d	16. a	17. c	18. a
19. c	20. c	21. d	22. c	23. a	24. b
25. a	26. a	27. c	28. d	29. b	30. d
31. c	32. a	33. b	34. c	35. a	36. d
37. a	38. b	39. c	40. d	41. c	42. b
43. a	44. b	45. d	46. a	47. b	48. c
49. a	50. b	51. a	52. d	53. d	54. b
55. c	56. b	57. a	58. c	59. a	60. b
61. d	62. b	63. b	64. d	65. c	66. b
67. a	68. b	69. c	70. d	71. c	72. d
73. a	74. b	75. a	76. b	77. c	78. a
79. b	80. c	81. c	82. a	83. b	84. a
85. d	86. c	87. a	88. b	89. d	90. b
91. d	92. b	93. a	94. d	95. b	96. a
97. b	98. a	99. b	100.d	101.b	102. a
103.a	104.c	105.b	106.a	107.a	108.b
109.a	110.d	111.d	112.c	113.a	114.b
115.b	116.a	117.d	118.a	119.c	120.c
121.d	122.a	123.b	124.a	125.c	126.b

127.a					
133.d	134.a	135.a	136.d	137.a	138.c
139.d	140.b	141.b	142.a	143.d	144.b
145.a	146.b	147.a	148.a	149.b	150.c
151.d	152.a	153.d	154.c	155.a	156.b
157.a	158.b	159.a	160.a	161.a	162.c
163.a	164.a	165.a	166.b	167.d	168.c
169.b	170.a	171.b	172.a	173.a	174.d
175.b	176.c	177.b	178.c	179.b	180.c
181.a	182.c	183.a	184.d	185.b	186.a
187.d	188.c	189.b	190.d	191.b	192.a
193.a	194.b	195.c	196.d	197.a	198.c
199.b	200.a	201.a	202.a	203.b	204.a
205.a	206.d	207.b	208.a	209.a	210.b
211.c	212.b	213.d	214.a	215.c	216.a
217.c	218.d	219.d	220.d	221.d	222.d
223.d	224.c	225.a	226.b	227.a	228.a
229.a	230.a	231.c	232.a	233.c	234.a
235.a	236.d	237.b	238.c	239.d	240.a
241.d	242.b	243.b	244.d	245.c	246.d
247.a	248.b	249.a	250.c	251.d	252.a
253.b	254.d	255.c	256.a	257.a	258.d
259.c	260.b	261.a	262.b	263.d	264.d
265.a	266.b	267.d	268.d	269.a	270.b
271.d	272.a	273.b	274.a	275.a	276.b
277.c	278.a	279.d	280.a	281.b	282.c
283.b	284.d	285.d	286.a	287.b	288.d
289.a	290.a	291.a	292.d	293.b	294.d
295.a	296.b	297.d	298.a	299.b	300.d
301.b	302.a	303.a	304.a	305.c	306.d
307.a	308.c	309.a	310.a	311.a	312.e
313.d	314.d	315.c	316.d	317.c	318.a

88493 87887