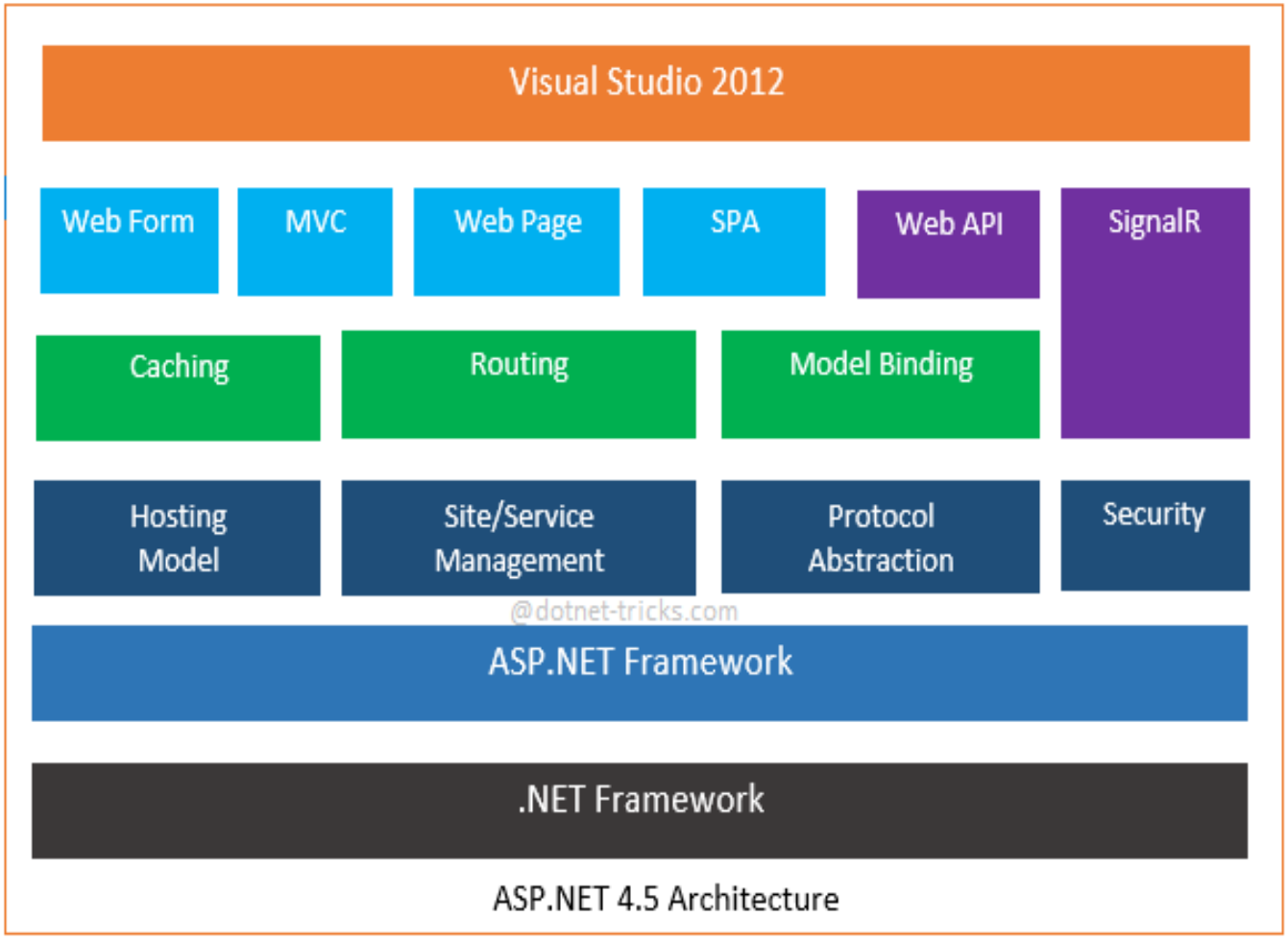
ASP.NET

- ASP.NET is an server-side web application framework designed for web development to produce dynamic web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, web applications and web services.
- It was first released in January 2002 with version 1.0 of the .NET Framework, and is the successor to Microsoft's Active Server Pages (ASP) technology.

Code-behind model

- Microsoft recommends dealing with dynamic program code by using the code-behind model, which places this code in a separate file or in a specially designated script tag.
- ASP.NET's code-behind model encourages developers to build applications with separation of presentation and content in mind.
- In theory, this would allow a Web designer, for example, to focus on the design markup with less potential for disturbing the programming code that drives it.



ASP.NET 4.5 Architecture

Components of Asp.NET 4.5 Architecture

1 .NET Framework

.Net framework is an integrated component of windows operating system that supports development and execution of next generation applications, Windows store apps and services.

2.ASP.NET Framework

- ASP.Net Framework is used to create dynamic website, web application and web services. It is built on the top of .NET Framework.
- Asp.NET Framework provides you various capabilities like Hosting Model, Site/Service Management, Protocol Abstraction, Security, Caching capability, Routing and Model Binding etc.

3.Asp.NET Site

There are following flavours of Asp.NET Site -

Web Forms:

This is the traditional event driven development model. It has drag and drop server controls, server events and state management techniques. This best for rapid application development (RAD) with powerful data access.

MVC

This is a lightweight and MVC (Model, View, Controller) pattern based development model. It provides full control over mark-up and support many features that allow fast & agile development. This best for developing lightweight, interactive and device oriented web application with latest web standards.

Web Pages

This is also a lightweight and Razor syntax based development model. It has built-in template and helpers also provide full control over mark-up. It is best for developing beautiful web application with latest web standards. You can also use WebMatrix which is a free tool and has built-in template; for developing Asp.Net Web Page.

SPA

SPA stands for Single Page Application which helps you to build web applications that include significant client-side interactions using HTML5, CSS3 and JavaScript. It is best to make highly interactive single page dashboard web applications.

4.Asp.NET Services

There are two ways to make Asp.Net Services as given below –

Web API

Asp.Net Web API is a framework for building HTTP services that can be consume by a broad range of clients including browsers, mobiles, iphone and tablets.

SignalR

ASP.NET SignalR is a library that simplifies the process of adding real-time web functionality to applications. Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data.

5.Visual Studio 2012

The Visual Studio IDE offers a set of tools that help you to write and modify the code for your programs, and also detect and correct errors in your programs. Using Visual Studio 2012 you can build Windows Store apps, desktop apps, mobile apps, ASP.NET web apps, and web services.

ASP.NET Page Life Cycle

- **Page request** A request to an ASPX page starts the life cycle of that page.
- **Start** The start phase for the page, where it gets access to properties like Request and Response.In addition,during this phase the PreInit event is raised to signal that the page is about to go into the initialization phase.
- **Page initialization** During this phase, the controls you have set up in your page or added programmatically become available. During this phase the Page class fires three events: Init, InitComplete, and PreLoad. Also during this phase,the control properties are loaded from ViewState again during a postback.
- **Load** During this phase the page raises the Load event.
- **Validation** During the validation phase, the Validation controls used to validate user input are processed

ASP.NET Page Life Cycle

- **Postback event handling:**During this phase, the controls in your page may raise their own events.For example, the DropDownList may raise a SelectedIndexChanged event when the user has chosen a different option in the list. When all event processing is done, the page raises the LoadComplete event.During this phase the PreRender event is raised to signal that the page is about to render to the browser. Shortly after that, SaveStateComplete is raised to indicate that the page is done storing all the relevant data for the controls in ViewState.
- **Rendering** Rendering is the phase where the controls (and the page itself) output their HTML to the browser.
- **Unload** The Unload phase is really a clean-up phase. This is the moment where the page and controls can release resources like database connections.

ViewState

- A Web application is stateless. A new instance of the Web page class is created every time that the page is requested from the server. This would ordinarily mean that all information in the page and in its controls would be lost with each round trip. For example, by default if a user enters information into a text box on an HTML Web page, that information is sent to the server.
- To overcome this limitation of Web programming, the ASP.NET page framework includes several state-management features to preserve page and control values between round trips to the Web server. One of these features is view state.

View State

- View state is the way that the ASP.NET page framework uses to preserve page and control values between round trips.
- View state is used automatically by the ASP.NET page framework to persist information that must be preserved between postbacks.
- When the HTML markup for the page is rendered, the current state of the page and values that must be retained during postback are serialized into base64-encoded strings. This information is then put into the view state hidden field or fields.
- View state is a repository in an ASP.NET page that can store values that have to be retained during postback.
- ViewState is HTML hidden field which maintain the data even after postback.

View State

Disadvantages of View State

- Security Risk: The Information of View State can be seen in the page output source directly. You can manually encrypt and decrypt the contents of a Hidden Field, but It requires extra coding. If security is a concern then consider using a Server-Based state Mechanism so that no sensitive information is sent to the client.
- Performance: Performance is not good if we use a large amount of data because View State is stored in the page itself and storing a large value can cause the page to be slow.
- It can store values for the same page only.
- View state provides state information for a specific ASP.NET page. If you need to use information on more than one page, or if you need the information to persist across visits to the Web site, you must use another method for maintaining state.

cross page posting

- ASP.NET by default submit the forms to the same pages, cross page posting is submitted the form to a different page.
- cross page posting enables you to post the WebPage and WebPage’s control values to another WebPage.
- "To use cross-page posting, you have to use the "postBackUrl" attribute to specify the page we want to post".

Web server controls

- Web server controls include traditional form controls such as buttons and text boxes as well as complex controls such as tables. They also include controls that provide commonly used form functionality such as displaying data in a grid, choosing dates, displaying menus, and so on.
- When the ASP.NET Web page runs, the Web server control is rendered on the page using appropriate markup, which often depends not only on the browser type but also on settings that you have made for the control. For example, a TextBox control might render as an **input** tag or a**textarea** tag, depending on its properties.

Web server controls offer the following advantages:

1. Make it easier for manufacturers and developers to build tools or applications that automatically generate the user interface.
2. Simplify the process of creating interactive Web forms, which requires less knowledge of how HTML controls work and make the task of using them less prone to errors.

Basic ASP.NET Sever Controls

To use a Web server control, use the following syntax (which uses the **TextBox** control as an example):

```
<asp:textbox text="hello world" runat=server />
```

Basic ASP.NET Sever Controls

Button Control :The **Button** control allows you to create a push button on the Web Forms page. By default, a **Button** control is a **submit** button

CheckBox Control: The CheckBox control provide a way for users to select multiple options. Individual CheckBox controls raise the CheckedChanged event when users click the control. By default, this event does not cause the page to be posted to the server. However, you can force the control to perform an immediate postback by setting the AutoPostBack property to true.

HyperLink Control: The HyperLink control creates links on a Web page that enables users to move from page to page in an application.

Image Control: The Image Web server control enables you to display images on an ASP.NET Web page and manage these images in your own code.

ImageButton Control: Enables you to handle user clicks in an image

Basic ASP.NET Sever Controls

Label Control: The label lets you programmatically set text in an ASP.NET Web page.

Panel Control : You can use the Panel control as a container for other controls.

DropDownList: The DropDownList Web server control enables users to select a single item from a predefined drop-down list.

RadioButton Control: The RadioButton control enable users to select from a small set of mutually exclusive, predefined choices.

Table Control :The table control enables you to create tables on ASP.NET pages that you can program in server code.

TextBox Control :The TextBox control provides a way for users to type information into an ASP.NET Web page, including text, numbers, and dates.

ListBox : The ListBox Web server control enables users to select one or more items from a predefined list.

CheckBoxList: It is list of multiple checkbox to select multiple option.

Basic ASP.NET Sever Controls

RadioButtonList: The RadioButtonList control is list of multiple radiobutton enable users to select from a small set of mutually exclusive, predefined choices.

BulletedList: The BulletedList control creates an unordered or ordered (numbered) list of items, which render as HTML ul or ol elements, respectively.

Calendar : The Calendar Web server control can be used to display selectable dates in a calendar and to display data associated with specific dates.

ASP.NET validation controls

- ASP.NET validation controls allow you to check valid user input on a web page.
- A validator is a control that checks one input control for a specific type of error condition and displays a description of that problem.
- Validation controls enable you to rapidly create Web Forms with validation that prohibit users from entering invalid data.
- The beauty of the validation controls is that they can execute both on the client and the server, enabling you to create responsive and secure web applications.
- To prevent your system from receiving invalid data, it's important to validate this data before you allow your system to work with it.
- The validation controls are extremely helpful in validating the data that a user enters in the system.
- When you add a validation control to a web page, the control renders JavaScript that validates the associated control at the client.

validation controls

ASP.NET provides the following validation controls:

- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator
- ValidationSummary

validation controls

Properties

- Text : Gets or sets the text displayed in the validation control when validation fails.
- ErrorMessage : Gets or sets the text for the error message displayed in a ValidationSummary control when validation fails.
- ControlToValidate : This property contains the ID of the control that needs to be validated.
- Display : This property determines whether the hidden error message takes up space or not. With the Display set to Static, the error message takes

  up screen estate, even when it is hidden.With the Display set to Dynamic, the error message does not takes up screen estate, when it is hidden.

RequiredFieldValidator

- The RequiredFieldValidator control ensures that the required field is not empty.
- Makes the associated input control a required field.
- It is generally tied to a text box to force input into the text box.

E.g.

<asp:RequiredFieldValidator ID="ReqVal1" runat="server" ControlToValidate="TextBox1" Display="Dynamic" ErrorMessage="Please enter your name" Text="*"></asp:RequiredFieldValidator>

RangeValidator

- The RangeValidator control enables you to check whether a value falls within a certain range.
- The control is able to check data types like strings, numbers, dates, and currencies.
- For example, you can use it to make sure a number is between 1 and 10, or a selected date falls between today and the next two weeks.

Properties

MinimumValue : This property determines the lowest acceptable value.

MaximumValue : This property determines the highest acceptable value.

Type : This property determines the data type that the validation control checks. This value can be set to String, Integer, Double, Date, or Currency to check the respective data types.

CompareValidator

- The CompareValidator can be used to compare the value of one control to another value.
- This is often used in sign-up forms where a user has to enter a password twice to make sure they type the same password both times.
- Alternatively, instead of comparing to another control, you can also compare against a constant value.

Properties

ControlToCompare : This property contains the ID of the control that the validator compares against. When this property is set, ValueToCompare has no effect.

Operator : This property determines the type of compare operation. Operators are Equal , NotEqual , GreaterThan , and GreaterThanEqual,LessThan,LessthanEqual,DataTypeCheck to perform different validation operations.

CompareValidator

Type : This property determines the data type that the validation control checks. This value can be set to String, Integer, Double, Date, or Currency to check the respective data types.

ValueToCompare: This property allows you to define a constant value to compare against. When this property is set, make sure you clear the ControlToCompare property as that will otherwise take precedence.

RegularExpressionValidator

- The RegularExpressionValidator control allows you to check a value against a *regular expression.*
- Regular expressions offer a compact syntax that allows you to search for patterns in text strings.
- Visual Web Developer comes with a few built-in expressions that make it easy to validate values like e-mail addresses and zip codes.
- You set the regular expression in the ValidationExpression property as Internet Email Address for email validations.

CustomValidator

- Performs user-defined validation on an input control.
- This validator allows you to write custom validation functions for both the client (in JavaScript) and the server (using VB.NET or C#).
- The client side validation is accomplished through the ClientValidationFunction property. The client side validation routine should be written in a scripting language, such as JavaScript or VBScript, which the browser can understand.
- The server side validation routine must be called from the control's ServerValidate event handler. The server side validation routine should be written in any .Net language, like C# or VB.Net.

ValidationSummary

- It shows a summary of all validation errors in the page.
- The ValidationSummary control allows you to summarize the error messages from all validation controls on a Web page in a single location.
- The error message displayed in the ValidationSummary control for each validation control on the page is specified by the ErrorMessage property of each validation control.
- It can display these errors in three different ways: using a list embedded in the page, using a JavaScript alert box, or using both at the same time.

DisplayMode : The summary can be displayed as a list, a bulleted list, or a single paragraph.

HeaderText : specify a custom title in the heading section of the ValidationSummary control.

ShowSummary : You can control whether the ValidationSummary control is displayed or hidden

ShowMessageBox: The summary can also be displayed in a message box.

Master Pages

- It allow you to define the look and feel of all the pages in your site in a single location.
- ASP.NET master pages allow you to create a consistent layout for the pages in your application.
- A single master page defines the look and feel and standard behavior that you want for all of the pages (or a group of pages) in your application.
- You can then create individual content pages that contain the content you want to display.
- A master page is an ASP.NET file with the extension .master (for example, MySite.master) with a predefined layout that can include static text, HTML elements, and server controls.

Master Pages

ContentPlaceHolder

To create regions that content pages can fill in, you need to define ContentPlaceHolder control. In turn, the replaceable content is defined in content pages.

```
<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
```

Content

The page-specific content is then put inside an <asp:Content> control that points to the relevant ContentPlaceHolder.

```
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
    Runat="Server"></asp:Content>
```

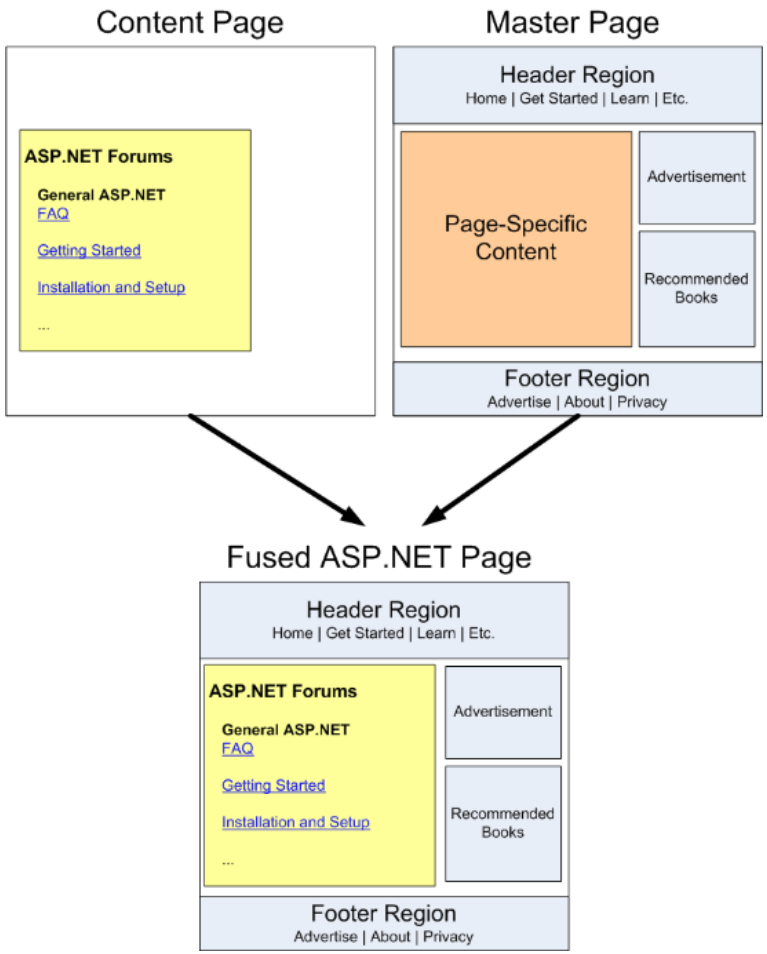
Advantages of master page

- Master pages provide functionality that developers have traditionally created by copying existing code, text, and control elements repeatedly; using framesets; using include files for common elements
- They allow you to centralize the common functionality of your pages so that you can make updates in just one place.

master page

At runtime, when the page is requested, the markup from the master page and the content page are merged, processed, and sent to the browser.

master page


Content Page
Master Page
ASP.NET Forums
General ASP.NET
FAQ
Getting Started
Installation and Setup
...
Header Region
Home | Get Started | Learn | Etc.
Page-Specific Content
Advertisement
Recommended Books
Footer Region
Advertise | About | Privacy
Fused ASP.NET Page
Header Region
Home | Get Started | Learn | Etc.
ASP.NET Forums
General ASP.NET
FAQ
Getting Started
Installation and Setup
...
Advertisement
Recommended Books
Footer Region
Advertise | About | Privacy

Theme

- A *theme is a collection of files that define the looks of a page.*
- A theme is a collection of property settings that allow you to define the look of pages and controls, and then apply the look consistently across pages in a Web application
- It can include skin files, CSS files, and images.
- You define themes in the special App_Themes folder in the root of your website. Within this folder you need to create one or more subfolders that define the actual themes.
- A link to each CSS file in the theme folder is added to your page's <head> section automatically whenever the theme is active.

Theme

- **Setting the theme at the page level:** Setting the Theme property at the page level by set the relevant attribute in the Page directive of the page:

  <%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" **Theme="DarkGrey" %>**

- **Setting the theme at the site level:**To enforce a theme throughout the entire web site, you can set the theme in the web.config file. To do this, open the web.config file, locate the <pages> element,and add a theme attribute to it:

  <pages theme="DarkGrey">

  ...

  </pages>

- **Setting themes programmatically:** The third and final way to set a theme is through code. Because of the way themes work, you need to do this early on in the page's life cycle.

skin file

- Skins are simple text files that contain markup for controls.
- Placed in a theme folder under App_Themes,they are an integral part of the ASP.NET themes feature.
- A skin file (with a .skin extension) contains the server side presentational elements of a control. These settings are then applied to all the controls to which the skin applies.
- A skin file has the file name extension .skin and contains property settings for individual controls such as Button, Label, TextBox, or Calendarcontrols.
- E.g.

<asp:Button BackColor=”#cccccc” ForeColor=”#308462” runat=”server” />

With this skin definition, all the buttons in your site will get a BackColor of #cccccc and a ForeColor of #308462.

skin file

- Not all properties of a control are skinnable.
- Properties that influence the appearance (BackColor, ForeColor, BorderColor,and so on) can be skinned while properties that influence behavior (Enabled, EnableViewState,ID, and more) cannot be set.
- For example,

1. you can’t set the Enabled property of the Button through a .skin.
2. The control in the .skin file cannot have an ID attribute.

Named Skins

- Named skins are identical to normal skins with one exception: they have a SkinID set that allows you refer to that skin by name.
- Controls in your ASPX pages can then use that SkinID to apply that specific skin to the control.
- The named skin works almost exactly the same as normal skins. However, with a named skin a control can point to a specific skin in one of the skin.

Creating Named Skin

```
<asp:Button SkinID="RedButton" BackColor="Red" ForeColor="Black" runat="server" />
```

Using a Named skin

```
<asp:Button ID="Button2" runat="server" Text="Button" SkinID="RedButton" />
```

ASP.NET State Management

ASP.NET State Management Overview

- A new instance of the Web page class is created each time the page is posted to the server. In traditional Web programming, this would typically mean that all information associated with the page and the controls on the page would be lost with each round trip. For example, if a user enters information into a text box, that information would be lost in the round trip from the browser or client device to the server.
- To overcome this inherent limitation of traditional Web programming, ASP.NET includes several options that help you preserve data on both a per-page basis and an application-wide basis. These features are as follows:

ASP.NET State Management

Client Side : They storing data on the client side.

- View state
- Hidden fields
- Cookies
- Query strings

Server Side : They storing data in memory on the server side.

- Application state
- Session state

View State

- View state is the way that the ASP.NET page framework uses to preserve page and control values between round trips.
- View state is used automatically by the ASP.NET page framework to persist information that must be preserved between postbacks.
- When the HTML markup for the page is rendered, the current state of the page and values that must be retained during postback are serialized into base64-encoded strings. This information is then put into the view state hidden field or fields.
- View state is a repository in an ASP.NET page that can store values that have to be retained during postback.
- ViewState is HTML hidden field which maintain the data even after postback.

View State

Disadvantages of View State

- Security Risk: The Information of View State can be seen in the page output source directly. You can manually encrypt and decrypt the contents of a Hidden Field, but It requires extra coding. If security is a concern then consider using a Server-Based state Mechanism so that no sensitive information is sent to the client.
- Performance: Performance is not good if we use a large amount of data because View State is stored in the page itself and storing a large value can cause the page to be slow.
- It can store values for the same page only.
- View state provides state information for a specific ASP.NET page. If you need to use information on more than one page, or if you need the information to persist across visits to the Web site, you must use another method for maintaining state.

Hidden Field

- ASP.NET allows you to store information in a HiddenField control, which renders as a standard HTML hidden field. A hidden field does not render visibly in the browser, but you can set its properties just as you can with a standard control.
- When a page is submitted to the server, the content of a hidden field is sent in the HTTP form collection along with the values of other controls.
- A hidden field acts as a repository for any page-specific information that you want to store directly in the page.

Cookie

- A cookie is a small amount of data that is stored either in a text file on the client file system.
- Cookies can be temporary (with specific expiration times and dates) or persistent.
- The cookies are saved on the client device, and when the browser requests a page, the client sends the information in the cookie along with the request information.
- Cookies may be used for authentication, identification of a user session, user's preferences, shopping cart contents, or anything else that can be accomplished through storing text data.
- ***Advantages of using cookies***

  Occupies less memory, do not require any server resources and are stored on the user's computer so no extra burden on server.

Limitations of Cookie

- Cookies are domain specific i.e. a domain cannot read or write to a cookie created by another domain. This is done by the browser for security purpose.
- · Cookies are browser specific. Each browser stores the cookies in a different location. The cookies are browser specific and so a cookie created in one browser(e.g in Google Chrome) will not be accessed by another browser(Internet Explorer/Firefox).
- · Most of the browsers store cookies in text files in clear text. So it's not secure at all and no sensitive information should be stored in cookies.
- · Most of the browsers have restrictions on the length of the text stored in cookies. It is 4096(4kb) in general but could vary from browser to browser.
- · Some browsers limit the number of cookies stored by each domain(20 cookies). If the limit is exceeded, the new cookies will replace the old cookies.
- · Cookies can be disabled by the user using the browser properties. So unless you have control over the cookie settings of the users (for e.g. intranet application), cookies should not be used.

QueryString

- Querystring is string which append to the URL, which is used to transfer the information from one page to other page.
- Querystring is way to transfer information from one page to another through the URL.
- QueryString is attached to the URL with "?".

Eg.http://www.contoso.com/listwidgets.aspx?category=basic&price=100

- In order for query string values to be available during page processing, you must submit the page using an HTTP GET command. That is, you cannot take advantage of a query string if a page is processed in response to an HTTP POST command.
- **Disadvantages:**

  - All the attributes and values are visible to the end user. Therefore, they are not secure.
  - There is a limit to URL length of 255 characters.

Session

Background of session

- Web is **stateless**, which means a new instance of a web page class is re-created each time the page is posted to the server.
- As we all know, HTTP is a stateless protocol, it can't hold client information on a page.
- If the user inserts some information and move to the next page, that data will be lost and the user would not be able to retrieve that information. What do we need here? We need to store information.
- Session provides a facility to store information on server memory.
- For every client, session data is stored separately, which means session data is stored on a per client basis.

Session

- Use ASP.NET session state to store and retrieve values for a user.
- ASP.NET session state enables you to store and retrieve values for a user as the user navigates ASP.NET pages in a Web application.
- It is secure, transparent from users, and we can store any kind of object in it.
- Along with these advantages, some times session can cause performance issues in high traffic sites because it is stored in server memory and clients read data from the server.

Advantages and disadvantages of Session

Advantages:

- It helps maintain user state and data all over the application.
- It is easy to implement and we can store any kind of object.
- Stores client data separately.
- Session is secure and transparent from the user.

Disadvantages:

- Performance overhead in case of large volumes of data/user, because session data is stored in server memory.

Application State

- Application State is a state management technique. Application State is stored in the memory of the the server and is faster than storing and retrieving information in a database.
- Session state is specific for a single user session, but Application State is for all users and sessions.
- Application State does not have a default expiration period. When we close the worker process the application object will be lost.
- Technically the data is shared amongst users by a HTTPApplcationState class and the data can be stored here in a key/value pair. It can also be accessed using the application property of the HTTPContext class.

Global.asax file

- **T**he Global.asax file is used for handling application events or methods.
- The Global.asax file, also known as the ASP.NET application file, is an optional file that contains code for responding to application-level events raised by ASP.NET.
- At run time, Global.asax is parsed and compiled into a dynamically generated .NET Framework class derived from the HttpApplication base class.
- The Global.asax file itself is configured so that any direct URL request for it is automatically rejected; external users cannot download or view the code written within it.

The events of the Global.asax

- **Application_Start() :** This method is invoked initially when first application domain is created.
- **Session_Start() :** This method is called every time a session is start.
- **Application_BeginRequest() :** After an application has started the first method Application_BeginRequest() is executed for every user.
- **Application_AuthenticateRequest() :** It checks to determine whether or not the user is valid.
- **Application_Error() :** Whenever an unhandled exception occurs then this event will be called.
- **Session_End() :** When a user session is ended and all the data related to a specific user is cleared then the Session_End() event is called.
- **Application_End() :** This method is called before the application ends. This can take place if IIS is restarted or the application domain is changing.
- **Application_Disposed() :** This event is called after the application will be shut down and the .NET GC is about to reclaim the memory it occupies. Although this is very late to perform any clean-up but we can use it for safety purposes.

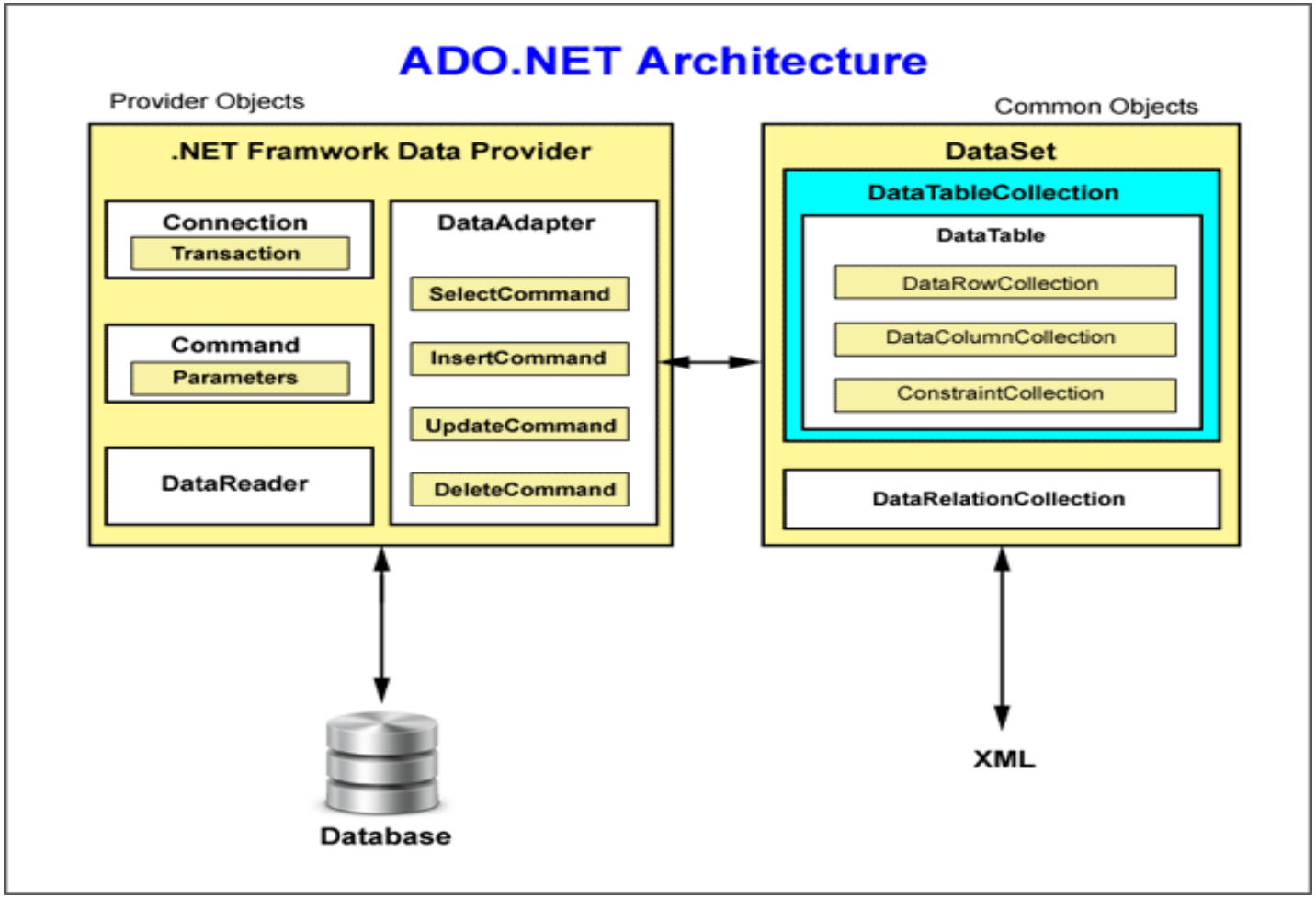
ADO.NET

- ADO stands for ActiveX Data Objects.
- ADO.NET is a database technology of .NET Framework used to connect application system and database server.
- ADO.NET consists of a set of classes used to handle data access
- ADO.NET uses XML to store and transfer data among applications, which is not only an industry standard but also provide fast access of data for desktop and distributed applications.
- ADO.NET data is cached and transferred in XML (EXtensible Markup Language) format. XML provide fast access of data for desktop and distributed applications. XML is plain text designed to transport and store data and is self-descriptive.
- Performance and scalability are two major factors when developing web-based application and services. Disconnected cached data in XML help in performance and scalability.

ADO.NET

- ADO.NET provides consistent access to data sources such as SQL Server and XML, and to data sources exposed through OLE DB and ODBC.
- ADO.NET includes .NET Framework data providers for connecting to a database, executing commands, and retrieving results.
- ADO.NET uses data in a disconnected fashion. When you access data, ADO.NET makes a copy of the data using XML. ADO.NET only holds the connection open long enough to either pull down the data or to make any requested updates. This makes ADO.NET efficient to use for Web applications. It's also decent for desktop applications.
- You can work on connected and disconnected manner.
- In disconnected model you will get old data as you are editing it. Outlook is an example of disconnected model. We work on offline object model and when connection is required it is connected.
- System.Data namespace is the core of ADO.NET, it contains classes used by all data providers.

ADO.NET Architecture



ADO.NET Providers

- .NET Framework Data Provider for SQL Server : Provides data access for Microsoft SQL Server. Uses the System.Data.SqlClient namespace.

- .NET Framework Data Provider for OLE DB : For data sources exposed by using OLE DB. Uses the System.Data.OleDb namespace.

- .NET Framework Data Provider for ODBC : For data sources exposed by using ODBC. Uses the System.Data.Odbc namespace.

- .NET Framework Data Provider for Oracle : For Oracle data sources. The .NET Framework Data Provider for Oracle supports Oracle client software version 8.1.7 and later, and uses the System.Data.OracleClient namespace.

ADO.NET

ADO.NET provides the following two models for accessing data from a Data Source:

- Connected Architecture
- Disconnected Architecture

1. **Connection Oriented Architecture:**
   In this case we require a continuous connection with the Data Source for accessing data in it.Here the “DataReader” class holds the data.

   DataReader is Forward only and Read Only.

2. **Disconnected Oriented Architecture**
   In this case we do not require a continuous connection with the Data Source for accessing data.
   Here the “DataSet” class holds the data in the client machines.

data providers

- A .NET Framework data provider is used for connecting to a database, executing commands, and retrieving results. Those results are either processed directly, placed in a DataSet in order to be exposed to the user as needed, combined with data from multiple sources, or remoted between tiers. .NET Framework data providers are lightweight, creating a minimal layer between the data source and code, increasing performance without sacrificing functionality.
- A data provider contains Connection, Command, DataAdapter, and DataReader objects. These four objects provides the functionality of Data Providers in the ADO.NET.

Objects of ADO.NET

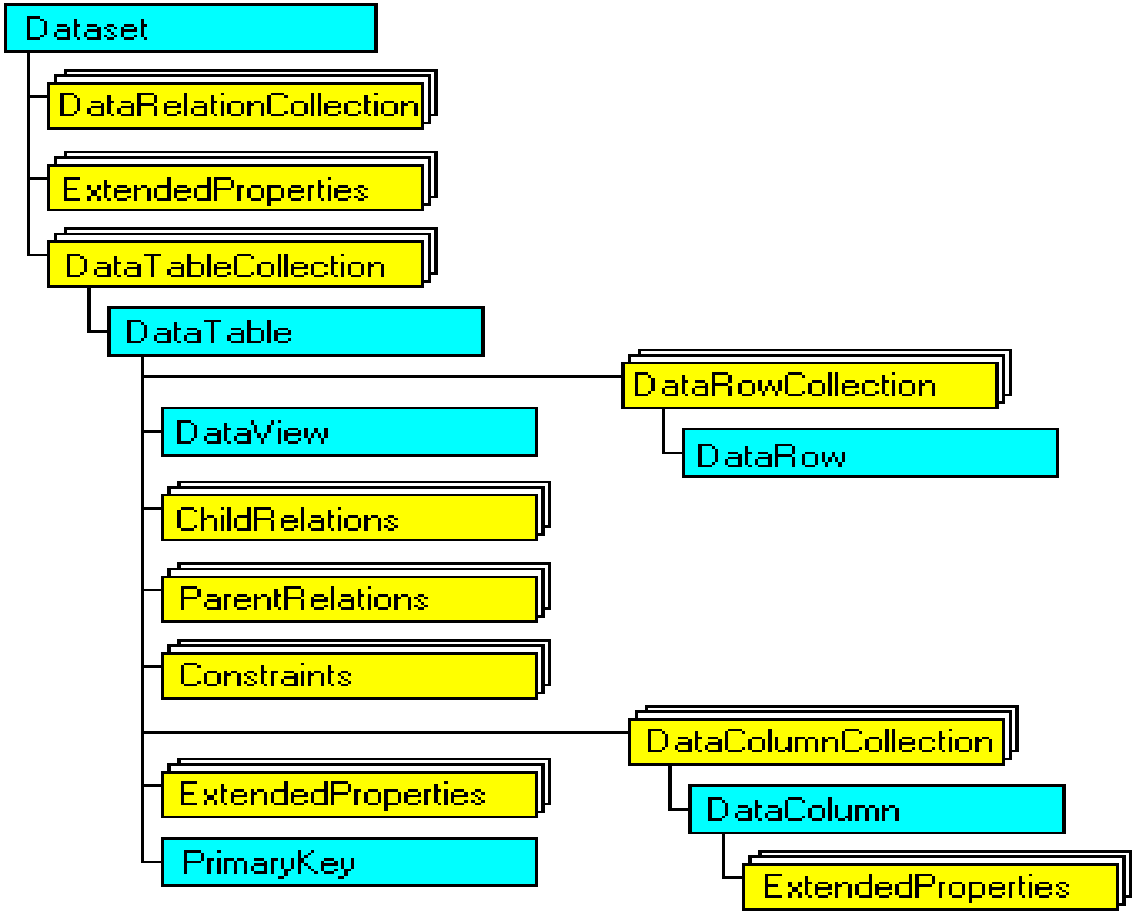
- **Connection** : Establishes a connection to a specific data source. The connection helps identify the database server, the database name, user name, password, and other parameters that are required for connecting to the data base. A connection object is used by command objects so they will know which database to execute the command on.
- **Command** : Executes a command against a data source. You use a command object to send SQL statements to the database. A command object uses a connection object to figure out which database to communicate with.
- **DataReader** : Reads a forward-only, read-only stream of data from a data source. It represent the Connected Architecture.

- **DataAdapter** : Sometimes the data you work with is primarily read-only and you rarely need to make changes to the underlying data source Some situations also call for caching data in memory to minimize the number of database calls for data that does not change. The data adapter manage data in a disconnected mode. Populates a DataSet and resolves updates with the data source.
- **DataSet** : DataSet objects are in-memory representations of data. They contain multiple Datatable objects, which contain columns and rows, just like normal database tables. You can even define relations between tables to create parent-child relationships. The DataSet is specifically designed to help manage data in memory and to support disconnected operations on data. The DataSet is an object that is used by all of the Data Providers, which is why it does not have a Data Provider specific prefix.

DataSet

- The DataSet object is supporting disconnected architecture that doesn't require a permanent connection with a Data Source for holding data. The DataSet is a memory-resident representation of data that provides a consistent relational programming model regardless of the data source. It can be used with multiple and differing data sources, with XML data, or to manage data local to the application. The DataSet represents a complete set of data, including related tables, constraints, and relationships among the tables. The following illustration shows the DataSet object model.
- It provides scrollable navigation to data, that allows us to move in any direction.
- It is updatable, in other words changes can be performed to data present in it and also send changes back to the DB.

DataSet


Dataset
DataRelationCollection
ExtendedProperties
DataTableCollection
DataTable
DataRowCollection
DataRow
DataView
ChildRelations
ParentRelations
Constraints
DataColumnCollection
DataColumn
ExtendedProperties
ExtendedProperties
PrimaryKey

DataTable

- An ADO.NET DataSet contains a collection of zero or more tables represented by DataTable objects. The DataTableCollection contains all theDataTable objects in a DataSet.
- A **DataTable** is defined in the System.Data namespace and represents a single table of memory-resident data. It contains a collection of columns represented by a DataColumnCollection, and constraints represented by a ConstraintCollection, which together define the schema of the table.
- A **DataTable** also contains a collection of rows represented by the DataRowCollection, which contains the data in the table. Along with its current state, a DataRow retains both its current and original versions to identify changes to the values stored in the row.

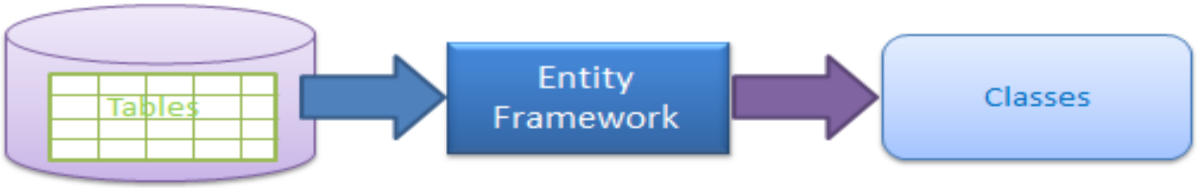
DataRelation

- A **DataSet** contains relationships in its DataRelationCollection object. A relationship, represented by the DataRelation object, associates rows in one**DataTable** with rows in another **DataTable**. A relationship is analogous to a join path that might exist between primary and foreign key columns in a relational database. A **DataRelation** identifies matching columns in two tables of a **DataSet**.
- Relationships enable navigation from one table to another in a **DataSet**. The essential elements of a **DataRelation** are the name of the relationship, the name of the tables being related, and the related columns in each table. Relationships can be built with more than one column per table by specifying an array of DataColumn objects as the key columns. When you add a relationship to the DataRelationCollection, you can optionally add a **UniqueKeyConstraint** and a **ForeignKeyConstraint** to enforce integrity constraints when changes are made to related column values.

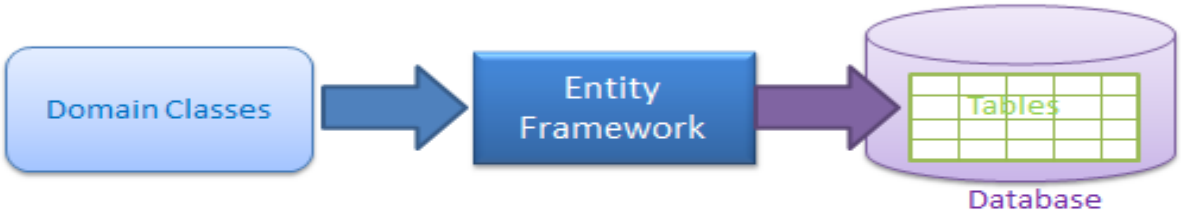
Entity Framework

- Microsoft has provided an O/RM framework called "Entity Framework" to automate database related activities for your application.
- *The Microsoft ADO.NET Entity Framework is an Object/Relational Mapping (ORM) framework that enables developers to work with relational data as domain-specific objects, eliminating the need for most of the data access plumbing code that developers usually need to write.*
- *Using the Entity Framework, developers issue queries using LINQ, then retrieve and manipulate data as strongly typed objects.*
- *The Entity Framework's ORM implementation provides services like change tracking, identity resolution, lazy loading, and query translation so that developers can focus on their application-specific business logic rather than the data access fundamentals.*

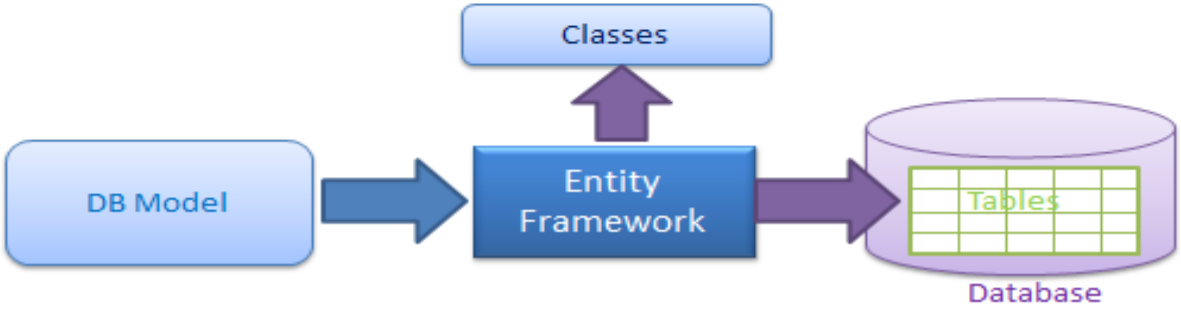
- Entity framework is useful in three scenarios. First, if you already have existing database or you want to design your database first than other parts of the application. Second, you want to focus on your domain classes and then create the database from your domain classes. Third, you want to design your database schema on the visual designer and then create the database and classes.
- The following figure illustrates the above scenarios.



Generate Data Access Classes for Existing Database

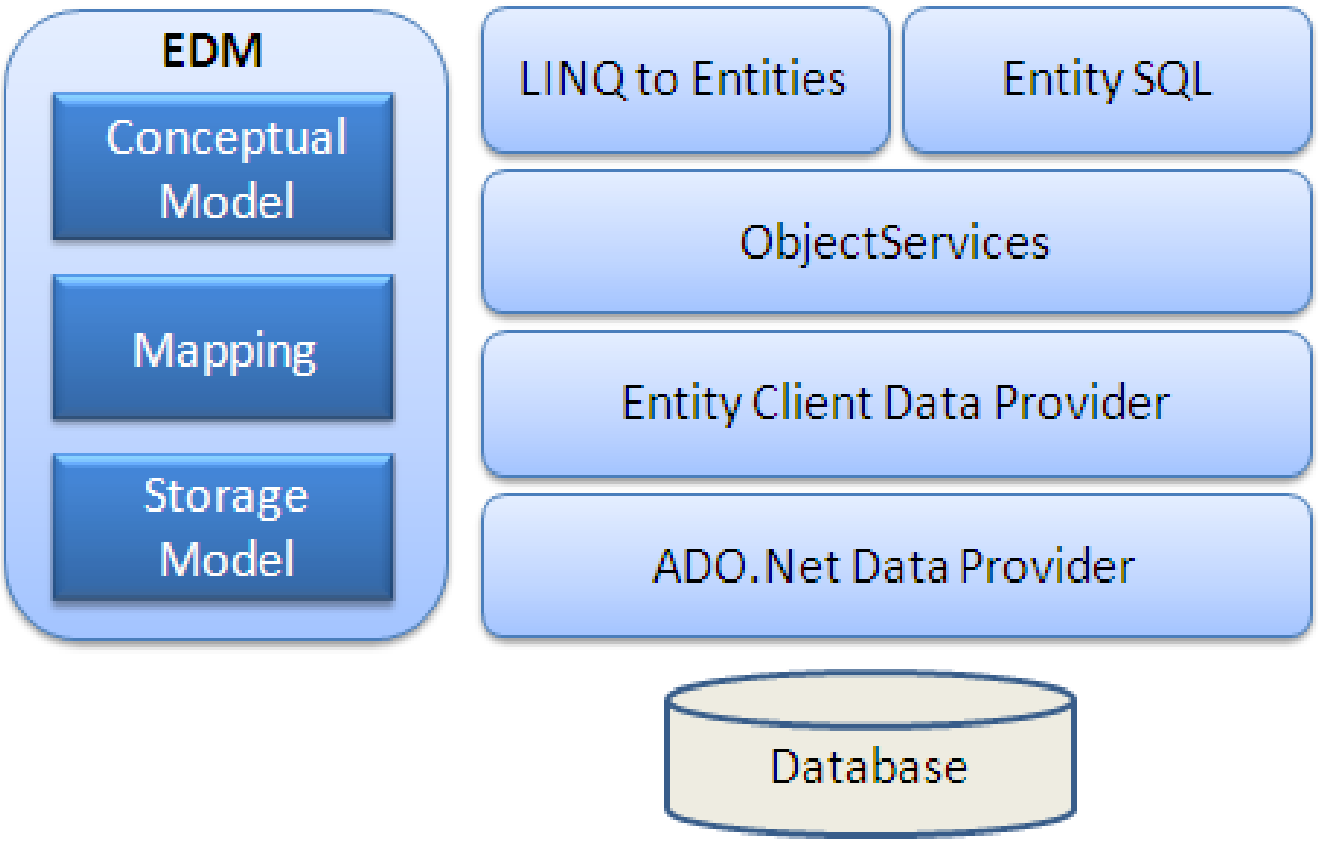


Create Database from the Domain Classes



Create Database and Classes from the DB Model design

Entity Framework Architecture



EDM (Entity Data Model)

EDM consist three main parts- Conceptual model, Mapping and Storage model.

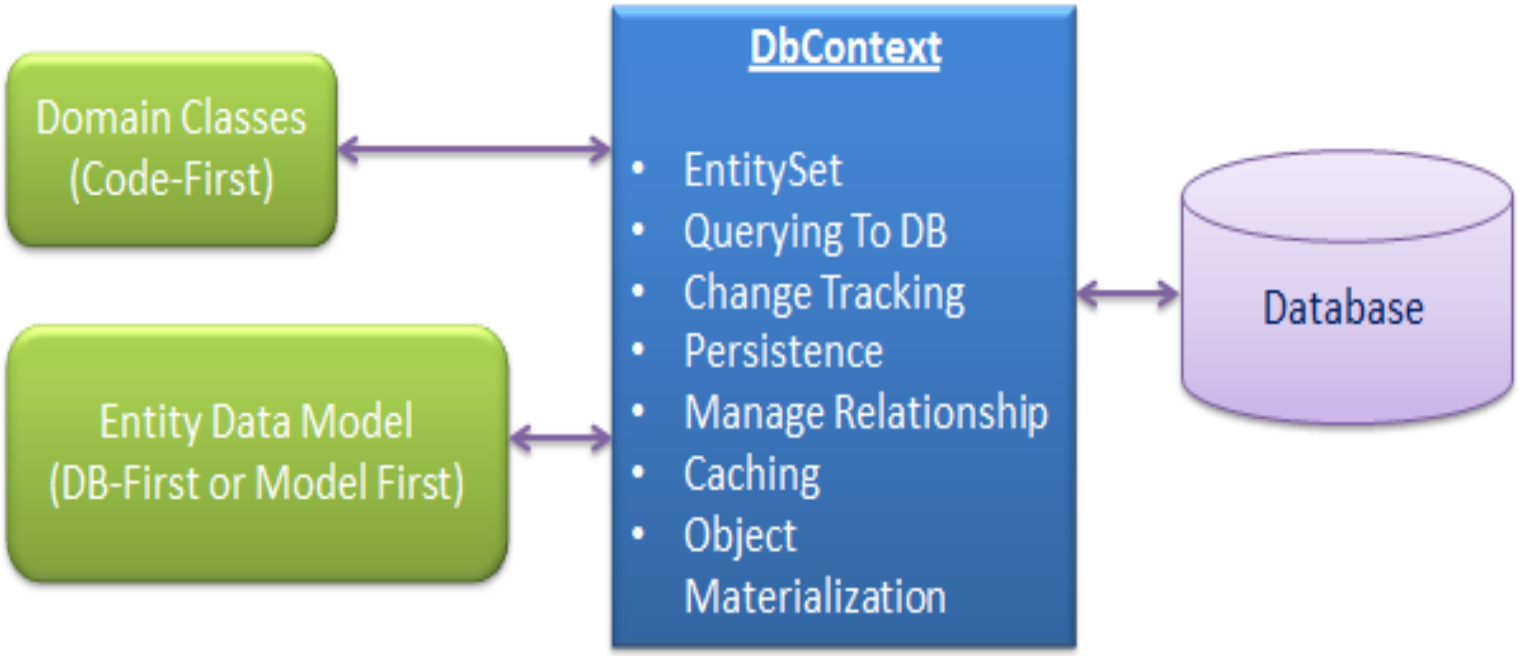
- **Conceptual Model:** The conceptual model contains the model classes and their relationships. This will be independent from your database table design.
- **Storage Model:** Storage model is the database design model which includes tables, views, stored procedures and their relationships and keys.
- **Mapping:** Mapping consist information about how the conceptual model is mapped to storage model.
- **LINQ to Entities:**LINQ to Entities is a query language used to write queries against the object model. It returns entities, which are defined in the conceptual model. You can use your LINQ skills here.
- **Entity SQL:** Entity SQL is another query language just like LINQ to Entities. However, it is a little more difficult than L2E and also the developer will need to learn it separately.

EDM (Entity Data Model)

- **Object Service:**Object service is a main entry point for accessing data from the database and to return it back. Object service is responsible for materialization, which is process of converting data returned from entity client data provider (next layer) to an entity object structure.
- **Entity Client Data Provider:**The main responsibility of this layer is to convert L2E or Entity SQL queries into a SQL query which is understood by the underlying database. It communicates with the ADO.Net data provider which in turn sends or retrieves data from database.
- **ADO.Net Data Provider:**This layer communicates with database using standard ADO.Net.

DbContext

- DbContext is an important part of Entity Framework. It is a bridge between your domain or entity classes and the database.



DbContext

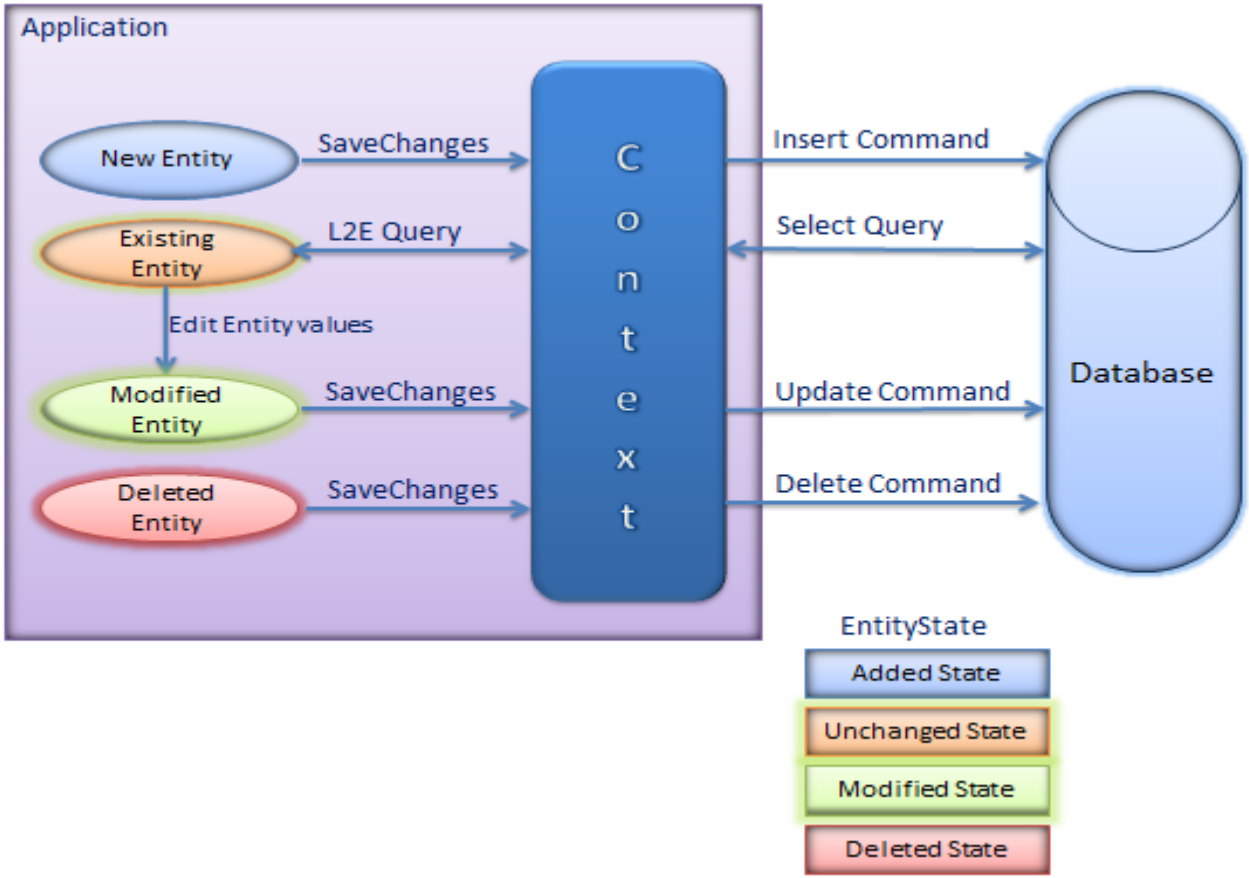
DbContext is the primary class that is responsible for interacting with data as object. DbContext is responsible for the following activities:

- **EntitySet:** DbContext contains entity set (DbSet<TEntity>) for all the entities which is mapped to DB tables.
- **Querying:** DbContext converts LINQ-to-Entities queries to SQL query and send it to the database.
- **Change Tracking:** It keeps track of changes occurred in the entities after it has been querying from the database.
- **Persisting Data:** It also performs the Insert, update and delete operations to the database, based on the entity states.
- **Caching:** DbContext does first level caching by default. It stores the entities which has been retrieved during the life time of a context class.
- **Manage Relationship:** DbContext also manage relationship in DB-First or Model-First approach in Code-First approach.
- **Object Materialization:** DbContext converts raw table data into entity objects.

Entity Lifecycle

- During an entity's lifetime, each entity has an entity state based on the operation performed on it via the context (DbContext). The entity state is an enum of type *System.Data.Entity.EntityState* that includes the following values:
- Added
- Deleted
- Modified
- Unchanged
- Detached

Entity Lifecycle



Entity Lifecycle

- As you can see in the above figure, new entity in context has Added entity state. So the context will execute insert command to the database. The same way, when you retrieve an existing entities using L2E queries, it will have Unchanged state because you have just retrieved an entity and hasn't performed any operation on it yet. When you modify values of existing entity, it changes its state to Modified which in-tern will execute update command on SaveChanges. Deleted entity from context will have Deleted state which in-tern will execute delete command to the database.

Security

Security revolves around three important questions:

- ❑ Who are you?
- ❑ How can you prove that?
- ❑ What are you allowed to do in the system?

Security

Identity: Who Are You?

- An identity is what makes you, *you.*
- For a web site your identity may be your name and e-mail address.

Authentication: How Can You Prove Who You Are?

- Authentication is about providing evidence about who you are.
- eg. With a web site you need to provide an e-mail address and a password. Together these two pieces form the evidence that prove your identity.
- There are many other mechanisms used for authentication, including high-tech fingerprint or iris scans, smart cards.

Security

Authorization: What Are You Allowed to Do?

- Depending on who you are, a system grants you more or fewer privileges to access certain areas.
- The permissions for the user are based on its user name (the identity it represents) and the *roles (or security* groups) the user is optionally assigned to.
- Similarly, resources can be opened up or blocked for specific users or roles.
- When there is a match between the current user and the access rules for the resource a user is trying to access, the user is granted access. If the user is blocked specifically, then access is denied.

ASP.NET Application Services

ASP.NET ships with a number of services, of which the most important ones Are:

- **Membership**: Enables you to manage and work with user accounts in your system.
- **Roles:** Enables you to manage the roles that your users can be assigned to.
- **Profiles:** Allows you to store user-specific data in a back-end database.

Login Controls

- The login controls take away much of the complexity usually associated with writing a security layer in a web application.
- The available login controls effectively encapsulate all the code and logic you need to validate and manage users. These controls work by communicating with the configured provider through the application services, instead of talking to a database directly.

Login:

- Login control allows a user to log in to the site.
- Under the hood it talks to the configured membership provider through the application services to see if the user name and password represent a valid user in the system.

Login Control Properties

- **DestinationPageUrl** :This property defines the URL the user is sent to after a successful login attempt.
- **CreateUserText :** This property controls the text that is displayed to invite users to sign up for a new account.
- **CreateUserUrl :** This property controls the URL where users are taken to sign up for a new account.
- **DisplayRememberMe** : This property specifies whether the control displays the Remember Me option. When set to False or when the checkbox is not set when logging in, users need to reauthenticate every time they close and reopen the browser.
- **RememberMeSet :** This property specifies whether the Remember Me option is initially checked.

Properties of login control

- **PasswordRecoveryText** : This property controls the text that is displayed to tell users they can reset or recover their password.
- **PasswordRecoveryUrl** This property specifies the URL where users are taken to get their (new) password.

In addition to these properties, the control has a range of Text properties, such as LoginButtonText,RememberMeText, TitleText, and UserNameLabelText that are used to control the text that appears in the control

LoginView

- The LoginView is a handy control that lets you display different data to different users.
- It allows you to differentiate between anonymous and logged-in users, and you can even differentiate between users in different roles.
- The LoginView is template driven and as such lets you define different templates that are shown to different users.

AnonymousTemplate : The content in this template is shown to unauthenticated users only.

LoggedInTemplate :The content in this template is shown to logged-in users only.This template is mutually exclusive with the AnonymousTemplate.

Only one of the two is visible at any time.

RoleGroups : This template can contain one or more RoleGroups elements that in turn contain a ContentTemplate element that defines the content for the specified role. The RoleGroups element is mutually exclusive with the LoggedInTemplate. That means that if a user is a member of one of the roles for the RoleGroup, the content in the LoggedInTemplate is not visible

LoginStatus control:

- It provides information about the current status of the user.
- It provides a Login link when the user is not authenticated and a Logout link when the user is already logged in.
- You control the actual text being displayed by setting the LoginText and LogoutText properties.
- Finally, you can set the LogoutAction property to determine whether the current page refreshes if the user logs out, or whether the user is taken to another page after logging out. You determine this destination page by setting the LogoutPageUrl.

LoginName :

- It display the name of the logged-in user.
- To embed the user's name in some text you can use the FormatString property. If you include {0} in this format string, it will be replaced with the user's name.

CreateUserWizard:

This is control is used to sign up for new account.

control exposes a number of useful properties :

- **ContinueDestinationPageUrl** This property defines the page where users are taken when they click Continue.
- **DisableCreatedUser** Whether or not the user is marked as disabled when the account is created. When set to True, users cannot log in to the site until their account has been enabled.
- **LoginCreatedUser** Whether or not the user is logged in automatically after the account has been created.
- **MailDefinition** This property contains a number of subproperties that allow you to define the (optional) e-mail that gets sent to users after they sign up.

PasswordRecovery :

- The PasswordRecovery control allows users to retrieve their existing password (when the system supports it) or get a brand new auto-generated password. In both cases, the password is sent to the e-mail address that the user entered when signing up for an account.
- The PasswordRecovery also has a MailDefinition element that allows you to point to a file that you want to send as the mail body.

ChangePassword :

- The ChangePassword control allows existing and logged-in users to change their passwords.
- It also has a MailDefinition element that allows you to send a confirmation of the new password to the user's e-mail address.

AJAX(Asynchronous JavaScript And Xml)

- Ajax, allows your client-side web pages to exchange data with the server through asynchronous calls.
- popular feature driven by Ajax is the flicker-free page that allows you to perform a postback to the server without refreshing the entire page.
- Create flicker-free pages that allow you to refresh portions of the page without a full reload and without affecting other parts of the page.
- Provide feedback to your users during these page refreshes.
- Update sections of a page and call server-side code on a scheduled basis.
- Access server-side Web Services and work with the data they return.

Creating Flicker-Free Pages

- To avoid full postbacks in your ASPX pages and update only part of the page, you can use the UpdatePanel server control.
- For this control to operate correctly, you also need a ScriptManager control.

UpdatePanel Control :

- The UpdatePanel control is a key component in creating flicker-free pages.
- Whenever one of the controls within the UpdatePanel causes a postback to the server, only the content within that UpdatePanel is refreshed.

Property of UpdatePanel

- Triggers: The Triggers collection contains PostBackTrigger and AsyncPostBackTrigger elements. The first is useful if you want to force a complete page refresh, whereas the latter is useful if you want to update an UpdatePanel with a control that is defined outside the panel.
- ContentTemplate It's the container in which you place controls as children of the UpdatePanel.

ScriptManager Control

- The ScriptManager control serves as the bridge between the client page and the server.
- It manages script resources (the JavaScript files used at the client), takes care of partial-page updates and handles interaction with your web site for things like Web Services and the *ASP.NET application services* like membership, roles, and profile.

Property of ScriptManager :

- Services : The <Services> element allows you to define Web Services that are accessible by your client side pages.

UpdateProgress Control

- To tell your users to hold on for a few seconds while their request is being processed, you can use the UpdateProgress control.
- You connect the UpdateProgress control to an UpdatePanel using the AssociatedUpdatePanelID property.
- Its contents, defined in the <ProgressTemplate> element, are then displayed whenever the associated UpdatePanel is busy refreshing.
- You usually put text like “Please wait” or an animated image
- in this template to let the user know something is happening, although any other markup is acceptable as well.

Timer Control

- Timer control is used for executing server-side code on a repetitive basis.
- At a specified interval, the control fires its Tick event. Inside an event handler for this event you can execute any code you see fit.

Properties of Timer Control :

Enabled : This property determines whether the Timer control currently ticks. When Enabled is True, the control raises its Tick event at the interval specified in the Interval property. When Enabled is False, the control does nothing and raises no events.

Interval :This property determines the interval in milliseconds between the Tick events that the control raises. For example, if you want the control to fire an event every minute, set this property to 60,000.

ScriptManagerProxy Control

- The ScriptManagerProxy control enables nested components such as content pages and user controls to add script and service references to pages when a ScriptManager control is already defined in a parent element.
- A Web page can contain only one ScriptManager control, either directly on the page itself or indirectly inside a nested or parent component. The ScriptManagerProxy control enables you to add scripts and services to content pages and to user controls where the master page or host page already contains a ScriptManager control.

Accordion

- The Accordion is a web control that allows you to provide multiple panes and display them one at a time. It is like having several CollapsiblePanels where only one can be expanded at a time. Each AccordionPane control has a template for its Header and its Content.

It also supports three AutoSize modes so it can fit in a variety of layouts.

1. **None** - The Accordion grows/shrinks without restriction. This can cause other elements on your page to move up and down with it.
2. **Limit** - The Accordion never grows larger than the value specified by its Height property. This will cause the content to scroll if it is too large to be displayed.
3. **Fill** - The Accordion always stays the exact same size as its Height property. This will cause the content to be expanded or shrunk if it isn't the right size.

Calendar

- Calendar is an ASP.NET AJAX extender that can be attached to any ASP.NET TextBox control. It provides client-side date-picking functionality with customizable date format and UI in a popup control. You can interact with the calendar by clicking on a day to set the date, or the "Today" link to set the current date.
- In addition, the left and right arrows can be used to move forward or back a month. By clicking on the title of the calendar you can change the view from Days in the current month, to Months in the current year. Another click will switch to Years in the current Decade. This action allows you to easily jump to dates in the past or the future from within the calendar control.

CascadingDropDown

- CascadingDropDown is an ASP.NET AJAX extender that can be attached to an ASP.NET DropDownList control to get automatic population of a set of DropDownList controls. Each time the selection of one the DropDownList controls changes, the CascadingDropDown makes a call to a specified web service to retrieve the list of values for the next DropDownList in the set.
- CascadingDropDown enables a common scenario in which the contents of one list depends on the selection of another list and does so without having to embed the entire data set in the page or transfer it to the client at all.

CascadingDropDown Properties

- Below are some important properties of the AJAX CascadingDropDown

1. **TargetControlID** – Here we need to set the ID of the DropDownList control for which you want to make an AJAX Cascading DropDownList.
2. **ServicePath** – Here we set the URL of the Web Service that will act as source of data for the AJAX CascadingDropDown DropDownList.
3. **ServiceMethod** – Here we set the name of the Web Method that will be used to populate the AJAX CascadingDropDown DropDownList.

CascadingDropDown

4. **PromptText** – This property is the Text part that of the first or the default item that will be displayed in the AJAX CascadingDropDown DropDownList.
5. **PromptValue** – This property is the Value part that of the first or the default item that will be displayed in the AJAX CascadingDropDown DropDownList.
6. **Category** – This property is used to specify the Category for the AJAX CascadingDropDown DropDownList, Category value is passed as parameter to the Child or dependent AJAX CascadingDropDown DropDownList ServiceMethod i.e. Web Method.
7. **ParentControlID** – This property is used to set the ID of the DropDownList on whose selection the DropDownList will trigger the data population process.
8. **LoadingText**– This property used to display the loading text when the call is made to the Web Method until the data is populated in the AJAX CascadingDropDown DropDownList.

CollapsiblePanel

- The CollapsiblePanel is a very flexible extender that allows you to easily add collapsible sections to your web page. This extender targets any ASP.NET Panel control. The page developer specifies which control(s) on the page should be the open/close controller for the panel, or the panel can be set to automatically expand and/or collapse when the mouse cursor moves in or out of it, respectively.
- The panel is also post-back aware. On a client postback, it automatically remembers and restores its client state. This demonstrates the ability of these extenders to have some communication between the client and the server code.

Properties:

1. **TargetControlID** - the Panel to operate expand and collapse.

CollapsiblePanel

- **Collapsed** - Specifies that the object should initially be collapsed or expanded.
- **AutoCollapse** - True to automatically collapse when the mouse is moved off the panel.
- **AutoExpand** - True to automatically expand when the mouse is moved over the panel.
- **ScrollContents** - True to add a scrollbar if the contents are larger than the panel itself. False to just clip the contents.
- **ExpandControlID/CollapseControlID** - The controls that will expand or collapse the panel on a click, respectively. If these values are the same, the panel will automatically toggle its state on each click.
- **CollapsedText** - The text to show in the control specified by TextLabelID when the panel is collapsed.

CollapsiblePanel

- **ExpandedText** - The text to show in the control specified by TextLabelID when the panel is opened.
- **ImageControlID** - The ID of an Image control where an icon indicating the collapsed status of the panel will be placed. The extender will replace the source of this Image with the CollapsedImage and ExpandedImage urls as appropriate. If the ExpandedText or CollapsedText properties are set, they are used as the alternate text for the image.
- **CollapsedImage** - The path to an image used by ImageControlID when the panel is collapsed
- **ExpandedImage** - The path to an image used by ImageControlID when the panel is expanded
- **ExpandDirection** - can be "Vertical" or "Horizontal" to determine whether the panel expands top-to-bottom or left-to-right.

FilteredTextBox

FilteredTextBox is an extender which prevents a user from entering invalid characters into a text box.

Properties :

- **TargetControlID** - The ID of the text box for this extender to operate on.
- **FilterType** - A the type of filter to apply, as a comma-separated combination of**Numbers**, **LowercaseLetters**, **UppercaseLetters**, and **Custom**. If Custom is specified, the ValidChars field will be used in addition to other settings such as Numbers.
- **FilterMode** - A the filter mode to apply, either **ValidChars** (default) or**InvalidChars**. If set to InvalidChars, FilterType must be set to Custom; if set to ValidChars, FilterType must contain Custom.
- **ValidChars** - A string consisting of all characters considered valid for the text field, if "Custom" is specified as the filter type. Otherwise this parameter is ignored.
- **InvalidChars** - A string consisting of all characters considered invalid for the text field, if "Custom" is specified as the filter type and "InvalidChars" as the filter mode. Otherwise this parameter is ignored.

NumericUpDown

NumericUpDown is an ASP.NET AJAX extender that can be attached to an ASP.NET TextBox control to add "up" and "down" buttons that increment and decrement the value in the TextBox. The increment and decrement can be simple +1/-1 arithmetic, they can cycle through a provided list of values (like the months of the year).

ModalPopup :

The ModalPopup extender allows a page to display content to the user in a "modal" manner which prevents the user from interacting with the rest of the page. The modal content can be any hierarchy of controls and is displayed above a background that can have a custom style applied to it.

Popup

PopupControl is an ASP.NET AJAX extender that can be attached to any control in order to open a popup window that displays additional content. This popup window will probably be interactive and will probably be within an ASP.NET AJAX UpdatePanel, so it will be able to perform complex server-based processing (including postbacks) without affecting the rest of the page. The popup window can contain any content, including ASP.NET server controls, HTML elements, etc. Once the work of the popup window is done, a simple server-side call dismisses it and triggers any relevant script on the client to run and update the page dynamically.

Properties :

1. **TargetControlID:** The ID of the control you want to attach the popup control with.
2. **PopupControl:** The ID of the control to display as popup window.

Popup

3. **Position:** This is optional property and can be set to Left, Right, Top, Bottom or Center. It specifies the relative position of the popup control with respect to target control.
4. **CommitProperty:** This property is also optional and specifies which property will return the result of the popup window.
5. **OffsetX and OffsetY:** These two properties specify the number of pixels to offset the popup window horizontally or vertically from its default position as specified by Position property.

Sending mails

SmtpClient Class

Allows applications to send e-mail by using the Simple Mail Transfer Protocol (SMTP).

MailMessage Class

Provides properties and methods for constructing an e-mail message.

Properties of MailMessage :

- **Attachments:** Specifies the collection of attachments that are transmitted with the message.
- **Bcc :** Gets or sets a semicolon-delimited list of email addresses that receive a blind carbon copy (BCC) of the e-mail message.

Properties of MailMessage

- **Body** : Gets or sets the body of the e-mail message.
- **Cc :** Gets or sets a semicolon-delimited list of e-mail addresses that receive a carbon copy (CC) of the e-mail message.
- **From :** Gets or sets the e-mail address of the sender.
- **Subject :** Gets or sets the subject line of the e-mail message.
- **To :** Gets or sets a semicolon-delimited list of recipient e-mail addresses.

Web Services

- Web services are components on a Web server that a client application can call by making HTTP requests across the Web.
- Web Services are essentially methods that you can call over the Internet.
- This makes them ideal for exchanging data between different systems.
- They make it easy to exchange data between different types of platforms.
- For example, with a Web Service it's easy to exchange data between an ASP.NET web site running on Microsoft Windows and a PHP-based site running on Linux.
- it's also possible to exchange data between an ASP.NET web site and a client browser using JavaScript.
- Web service uses the Web protocols is the Simple Object Access Protocol (SOAP). SOAP is a W3C submitted note (as of May 2000) that uses standards based technologies (XML for data description and HTTP for transport) to encode and transmit application data.

Web Services

- ASP.NET enables you to create Web services can be accessed from client script in Web pages. The pages communicate with the server through a Web service communication layer that uses AJAX technology to make Web service calls. Data is exchanged asynchronously between client and server.
- In AJAX-enabled Web pages, the browser makes an initial request to the server for the page, and then makes subsequent asynchronous requests to Web services for data.
- Web Service is an application that is designed to interact directly with other applications over the internet. In simple sense, Web Services are means for interacting with objects over the Internet. The Web serivce consumers are able to invoke method calls on remote objects by using SOAP and HTTP over the Web. WebService is language independent and Web Services communicate by using standard web protocols and data formats, such as

1. HTTP
2. XML
3. SOAP

Web Services

What is SOAP?

- SOAP (simple object access protocol) is a remote function calls that invokes method and execute them on Remote machine and translate the object communication into XML format. In short, SOAP are way by which method calls are translate into XML format and sent via HTTP.

Web Services

Examples for Web Service

- National weather forecasters use them to supply data to web-sites and news organizations.
- Stock prices are provided this way by major exchanges and corporations.

When to use Web Services

- Cross platform – i.e. Communicate between a Java app and a .NET app
- Cross trust boundaries – between two unrelated organizations
- Future considerations – if there is a possibility that the logic may have to support third party integration