**Request:**

Information or message send by client to server is known as request.

The request object is an instance of the System.Web.Httprequest class.

This object represents the values and properties of the http request that cause your page to be loaded.

It contains all the URL parameters and all other information sent by a client.

**Http request properties:**

1. **Application path and Physical path:-**

Application path gets the ASP.Net applications virtual directory (URL). While physical path gets the real directory.

1. **Browser:-**

This provides a link to an http browser capabilities object which contains properties describing various browser features, such as supports for activates control, cookies,

VB script and frames.

1. **Cookies:-**

This gets the collection of cookies sent with this request.

1. **Form:-**

This represents the collection of form variable that were posted back to the page. In almost all cases, you will retrieve this information from control properties instead of using this collection.

1. **IsLocal:-**

This returns true, if the user is requesting the page from the current computer.

1. **Querystring:-**

This provides the parameters that were passed along with the Querystring.

1. **URL and URL Reffer:-**

This provides a URL object that represent the current address for the page and the page were the user is coming from (the previous page that link to this page)

1. **User Host address and User Host name:-**

This get the IP address and the DNS name of the remote client.

You could also access this information the server variables collection. However, this information may not always be available.

**Response:**

Information send by server to client is known as Response.

The response object is a instance of the system.web.httpresponse class and it represents the web server response to a client request.

The http response does till provide important functions namely cookie features and the redirect method. The redirect method allows you to send the user to another page.

Here is an example,

You can redirect to a file in the current directory Response.Redirect(“default2.aspx”)

You can redirect to other website Response.Redirect(“<http://www.google.com>”)

The Redirect() method requires a round-trip. Essentially, it sends a message to the browser that instructs it to request a news page.

If you want to transfer the user to another page in the same web application, you can use a faster approach with the Server.Transfer() method.

**Http response members:**

1. **Cookies:-**

This is the collection of cookies send with the response. You can use this property to add additional cookies.

1. **IsClientConnected:-**

This is a Boolean value indicating whether the client is still connected to the server. If it is not, you might want to stop a time consuming operation.

1. **Write(), BinaryWrite() and WriteFile():-**

This method allows you to write the text or binary content directory to the response string. You can even write the content of a file.

1. **Redirect:-**

This method transfers the user to another page in your application or a different website.

**Server:**

The server object is an instance of the System.Web.HttpServerUtility class.

**Http server utility methods:**

1. **MachineName:-**

A property representing the computer name of the computer on which the page is running. This is the name of webserver computer. Uses to identify itself to rest of the network.

1. **GetLastError:-**

Retrieves the exception object for the most recently encountered error, (all or a null reference if there is not one). This error must have occurred while processing the current request and it must not have been handled.

1. **HTML Encode and HTML Decode:-**

Changes an ordinary string with a legal HTML characters.

1. **URL Encode and URL Decode:-**

Changes an ordinary string into string with legal URL character.

1. **MapPath():-**

Returns the physical file path the co-responds to specified virtual file path on the web server.

1. **Transfer():-**

The transfer execution to another webpage in the current application. This is similar to Response.Redirect(). But, it is faster.

It cannot be used to transfer the page to a site on another web server or to a non ASP.Net page (such as an HTML page or an ASP page)

The transfer method is quickest to redirect user to another page in your application.When you use this method a round-trip is not involved. Instead the ASP.Net engine simply loads the new page and begins processing it.

As a result the URL i.e. displayed in the client browsers won’t change.

You can transfer to a file in the current web application.

**i.e.**Server.Transfer(“newpage.aspx”)

You can’t redirect to another website. This attempt will cause an error.

**i.e.**Server.Transfer(“<http://www.google.com>”)

The MapPath() is another useful method of the server object.

**For e.g**. Imagine you want to load a file name info.txt from the current virtual directory.

Instead of hard coding path, you can use Request.ApplicationPath to get the current relative virtual directory and Server.MapPath to convert this to an absolute physical path.

Here, is an example

Dim physicalpath as string

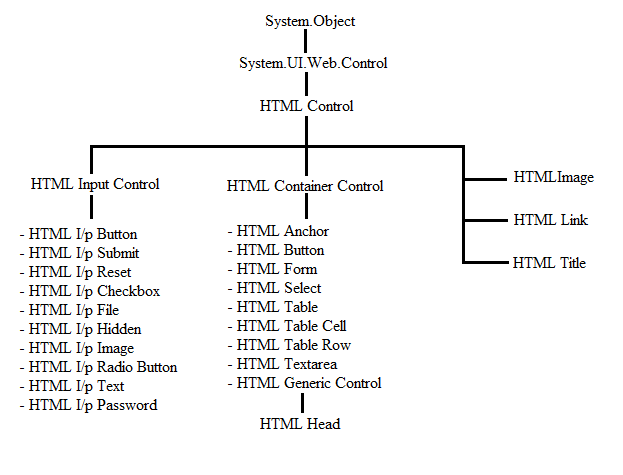
Physicalpath=Server.MapPath(“~/data/info.txt”)

**Difference between Server.Transfer and Response.Redirect:**

|  |  |
| --- | --- |
| **Response.Redirect** | **Server.Transfer** |
| Response.Redirect involves a round-trip to the server. | Server.Transfer avoids the round-trip.  It just changes the focus of the web server to different page and transforms the page processing to a different page. |
| Response.Redirect can be used for both .aspx and HTML pages. | Server.Transfer can be used only for .aspx page. |
| Response.Redirect can be used to redirect a user to an external website. | Server.Transfer can be used only on sites running on the same server.  You can’t use Server.Transfer to redirect the user to a page running on different server. |
| Response.Redirect changes the URL in the browser. So they can be bookmark. | Serever.Transfer retains the original URL in the browser.  It just replaces the content of the previous page with new page. |

**HTML Server Control:**

This are controls which are defined in the namespace System.Web.UI.HtmlControls

There are 20 different HTML server control. They are divided into different catagories based on whether they are input control or container control. Following diagram shows this hierarchy.

**Fig**: HTML Server Control

**The HTML Control Class:**

All the HTML server controls derives from the HTML base class HTML control. The following are set of common properties of HTML control class.

1. **Attribute:-**

Allow to access or add attribute in the control tag.

1. **Disabled:-**

It sets or gets the control disabled state. If true then the control Is usually grayed and not usable.

1. **Style:-**

Returns a collection of CSS attributes that are applied to the control.

1. **Tagname:-**

Returns the control tag name.

**The HTML Container Control Class:**

Any HTML tag that has both an opening and closing tag can contain other HTML content or controls i.e. anchor tag <a> which usually wraps text or an image with the text. <a>…..</a>

There are other tag like <div>….</div> which is also use as a container tag.

In addition to this we have bold tag. <b>….</b>

In addition to this we can use this tag to map the HTML server control class by using the attribute runat=”server”.

In this case we can interact with this tag using the HTML generic control.

The following are the 2 main properties of HTML container control:-

1. **InnerHTML:-**

Returns or sets the HTML tags inside the opening and closing tags. When you use the property, all characters are left as it is. This means you can embedded HTML markup.

1. **InnerText:-**

Returns or sets the text inside the opening and closing tags. When you use this property, any characters that would be interacted as special HTML syntax are automatically replaced with the HTML entity equivalents.

**The HTML Input Control Class:**

The HTML input control class allow for user interaction. It include checkboxes, textboxes, button and list boxes. The type attribute indicate the type of input control as in

<input type=”text”> (a textbox), <input type=”file”> (control for uploading file).

**The HTML Input Control properties:**

1. Name:-

Gets the unique identifier name for the HTML input control.

1. Type:-

Gets or sets the type of an HTML input control. For e.g. If this property is set to text, the HTML input control is textbox for data entry.

1. Value:-

Gets or sets the value associated with input control.

**The HTML Server Control Classes:**

HTML server controls and the specific properties and events that each one adds to the base class.

Runat=”server” will allow to access particular HTML control at coding file.

**HTML server control classes:-**

|  |  |  |
| --- | --- | --- |
| **Tag declaration** | **.Net class** | **Specific member** |
| <a runat=”server”> | HTML anchor | HREF, target, title, name, server click event. |
| <button runat=”server”> | HTML Button | CausesValidation, ValidationGroup, Server click event. |
| <Form runat=”server”> | HTML Form | Name, method, target, DefaultButton, DefaultFocus |
| <img runat=”server”> | HTML Image | Align, alt, border, height, src, width. |
| <input type=”button” runat=”server”> | HTML input button | Name, type, value, CausesValidation, ValidationGroup, server click event. |
| <input type=”reset” runat=”server”> | HTML input reset | Name, type, value. |
| <input type=”submit” runat=”server”> | HTML input submit | Name, type, value, CausesValidation, ValidationGroup, server click event |
| <input type=”checkbox” runat=”server”> | HTML input checkbox | Check, type, name, value, server click event |
| <input type=”file” runat=”server”> | HTML input file | Accept,maxlength, name, posted file, size, type, value. |
| <input type=”hidden” runat=”server”> | HTML input hidden | Name, type, value, server change event. |
| <input type=”image” runat=”server”> | HTML inputimablege | Align, alt, border, name, src, type, value, CausesValidation, ValidationGroup, sercer click event |
| <input type=”radio” runat=”server”> | HTML input radio button | Check, type, name, value, server change event |
| <input type=”text” runat=”server”> | HTML input text | Maxlength, name, type, value, serverChange event |
| <input type=”password” runat=”server”> | HTML input password | Maxlength, name, type, value, serverChange event |
| <select runat=”server”> | HTML select | Multiple, selectedindex, size, value, datasource, datatextfield, datavaluefield, items(collection), server change event |
| <table runat=”server”> | HTML table | Align, bgcolor, border, border-color, cellpadding, cellspacing, height, nowrap, width, rows(colloction). |
| <th runat=”server”> | HTML table cell | Align, bgcolor, border, colspan, rowspan, nowrap, valign |
| <tr runat=”server”> | HTML table row | Align, bgcolor, height, valign, cells (collection) |
| <textarea runat=”server”> | HTML text area | Cols, name, rows, value, server change event. |
| Any other <html> with runat=”server” attribute | HTML generic control | None. |

**ImageMap:**

ImageMap control is used to create an image that contains clickable hotspot region. When user click on the region, the user is either sent to a URL or a sub program is called. When it is rendered on the page, it is implemented through <img /> HTML tag.

Its properties like *BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc.* are implemented through style properites of <img>.

Following are some important properties that are very useful.

|  |  |
| --- | --- |
| *ImageUrl* | Url of image location. |
| *AlternetText* | Appears if image not loaded properly |
| *Tooltip* | Appears when on mouse over the image |
| *ImageAlign* | Used to align the Text beside image. |
| *HotSpotMode* | PostBack/Navigate .... When Navigate, the user is navigated to a different URL. In case of PostBack, the page is posted back to the server. |
| *OnClick* | Attach a server side event that fires after clicking on image when HostSpotMode is PostBack. |
| *PostBackValue* | You can access it in the server side click event through ImageMapEventArgs. (eg. e.PostBackValue) |
| Clicking on | |

<asp:ImageMap ID="ImageMap1" runat="Server" ImageUrl="controldata/gotocontrols.gif" OnClick="FireImageMapClick">

<asp:RectangleHotSpot AlternateText="Label" Left="10" Top="33" Right="75" Bottom="10" NavigateUrl="~/tutorials/controls/label.aspx" />

<asp:RectangleHotSpot AlternateText="Button" Left="80" Top="33" Right="150" Bottom="10" NavigateUrl="~/tutorials/controls/button.aspx" />

<asp:RectangleHotSpot AlternateText="ImageButton" Left="155" Top="33" Right="275" Bottom="10" NavigateUrl="~/tutorials/controls/imagebutton.aspx" />

<asp:RectangleHotSpot AlternateText="Fires server side Click Event. Postback value is ListBox" Left="300" Top="40" Right="400" Bottom="0" NavigateUrl="~/tutorials/controls/listbox.aspx" HotSpotMode="PostBack" PostBackValue="ListBox" /></asp:ImageMap>

**Asp Table:**

Table control is used to structure a web pages. In other words to divide a page into several rows and columns to arrange the information or images. When it is rendered on the page, it is implemented through <table> HTML tag.

Its properties like *BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc.* are implemented through style properites of <table> tag.

We can simply use HTML <table> control instead of using asp:Table control. However many of one benefits of using asp:Table control is we can dynamically add rows or columns at the runtime or change the appearance of the table.   
You can skip ID property of the TableRow or TableCell, however it is advisable to write these property otherwise you will not be able to play with these controls.

Following are some important properties that are very useful.

|  |  |
| --- | --- |
| *BackImageUrl* | Used to Set background image of the table |
| *Caption* | Used to write the caption of the table. |
| |  |  | | --- | --- | | Demo of asp:Table control | | | Row 1 - Cell 1 | Row 1 - Cell 2 | | Row 2 - Cell 1 | Row 2 - Cell 2 | | [Change Table Back Color](javascript:__doPostBack('ctl00$PlaceHolderForContents$lnl1','')) |

<asp:Table ID="Table2" runat="Server" CellPadding="2" CellSpacing="1"

BorderColor="CadetBlue" Caption="Demo of asp:Table control" BorderWidth="1" BorderStyle="Dashed">

<asp:TableRow ID="TableRow2" runat="Server" BorderWidth="1">

<asp:TableCell ID="TableCell4" runat="Server" BorderWidth="1">

Row 1 - Cell 1

</asp:TableCell>

<asp:TableCell ID="TableCell5" runat="Server">

Row 1 - Cell 2

</asp:TableCell>

</asp:TableRow>

<asp:TableRow ID="TableRow3" runat="Server">

<asp:TableCell ID="TableCell6" runat="Server">

Row 2 - Cell 1

</asp:TableCell>

<asp:TableCell ID="TableCell7" runat="Server">

Row 2 - Cell 2

</asp:TableCell>

</asp:TableRow>

</asp:Table>

**BulletedList :**

BulletedList control is used to display the data in a list prefixed with bullet characters. The item can be statically written or can be bound with the datasource. When it is rendered on the page, it is implemented through <table> HTML tag.

Its properties like *BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc.* are implemented generally through style properites of <ul> tag, However it depends on BulletStyle property.

Following are some important properties that are very useful.

|  |  |
| --- | --- |
| *DisplayMode* | HyperLink/LinkButton/Text. Determines how to display the items. |
| *FirstBulletNumber* | Sets a starting number for Bulleted list when BulletStyle is set to Numbering. |
| *Items* | Gets the colleciton of the items in the list control. |
| *BulletStyle* | Circle/CustomImage/Disc/LowerAlpha/LowerRoman/Numbered/Square/UpperAlpha/UpperRoman. Determines the style of the bullet. |
| *AppendDataBoundItems* | Determines whether statically defined items should remain and shown when adding  items dynamically. |
| *DataTextField* | Name of the field to set as items text. Used when DisplayMode is Hyperlink or LinkButton. |
| *DataValueField* | Name of the field to set as items value. Used when DisplayMode is Hyperlink or LinkButton. |
| *BulletImageUrl* | Used to set the Bullet Image when BulletStyle is CustomImage. |

**Literal:**

Ideally Literal control is the rarely used control which is used to put static text on the web page. When it is rendered on the page, it is implemented just as a simple text.   
Unlike asp:Label control, there is no property like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc. of Literal control. That makes it more powerful, you can even put a pure HTML contents into it.

|  |  |
| --- | --- |
| Select color to change the background color the cell | Ex. Just a text inside Literal Control |
| // Set the background color of the cell from server side event  <td> <asp:Literal ID="Literal2" runat="Server" />  Ex. <asp:Literal ID="Literal3" runat="Server" Text="Just a text inside Literal Control"></asp:Literal>  </td>  </tr>    // CODE BEHIND  // Fires when Button is clicked    Literal1.Text = " bgcolor='" + dropStatic.SelectedValue + "'";  litText.Text = "<div style='background-color:white;color:#000000'>Literl Control is powerful</div>"; | |