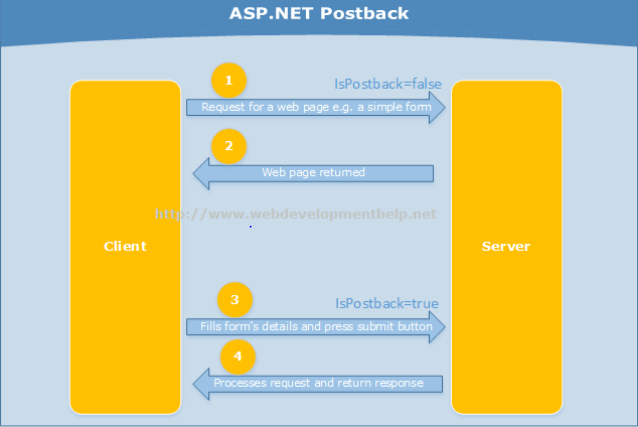
### What is the concept of Postback in ASP.NET?

A postback is a request sent from a client to server from the same page user is already working with. ASP.NET was introduced with a mechanism to post an HTTP POST request back to the same page. It’s basically posting a complete page back to server (i.e. sending all of its data) on the same page. So, the whole page is refreshed.

In the below diagram, when first request is made from client to server for a web page (say a simple registration form), IsPostback property value will be false. It will be a GET request that may be initiated by:

* typing a web page URL in a browser window or
* using JavaScript window.open method
* or, clicking a hyperlink on a webpage page.

After user filled the returned form and presses a button (say submit button, or use JavaScript to submit form), an HTTP Post request is made to server on same page with data as shown in point 3. This time IsPostbackproperty value will be true. Server processes the request and returns a response back to client.



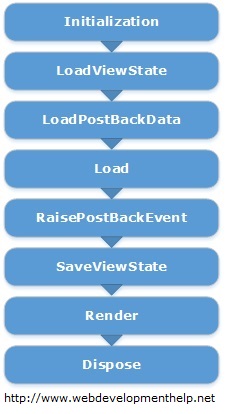
Another concept related to this approach is "Callback" that is also asked sometimes during a technical interview question. Click here to understand [Postback Vs Callback in ASP.NET](http://www.webdevelopmenthelp.net/2013/11/difference-between-postback-and-callback.html" \t "_blank).

### Please briefly explain ASP.NET Page life Cycle?

ASP.NET page passes through a series of steps during its life cycle. Following is the high-level explanation of life cycle stages/steps.

Initialization: Controls raise their Init event in this stage. Objects and variables are initializes for complete lifecyle of request.

LoadViewState: is a post back stage and loads the view state for the controls that enabled its view state property.



LoadPostBackData: is also a post back stage and loads the data posted for the controls and updates them.

Load: In this stage, page as well as all the controls raise their Load event. Till this stage, all the controls are initialized and loaded. In most of the cases, we are coding this event handler.

RaisePostBackEvent: is again a postback stage. For example, it's raise against a button click event. We can easily put our code here to perform certain actions.

SaveViewState: Finally, controls state is saved in this stage before Rendering HTML.

Render: This is the stage where HTML is generated for the page.

Dispose: Lastly, all objects associated with the request are cleaned up.

For very detailed explanation, Page Life Cycle is explained [here](http://www.codeproject.com/Articles/73728/ASP-NET-Application-and-Page-Life-Cycle).

In order to get other ASP.NET related details as per Microsoft Certification Exams, you can [follow here](http://www.webdevelopmenthelp.net/microsoft-certification-exams).

### What is the difference between custom controls and user controls?

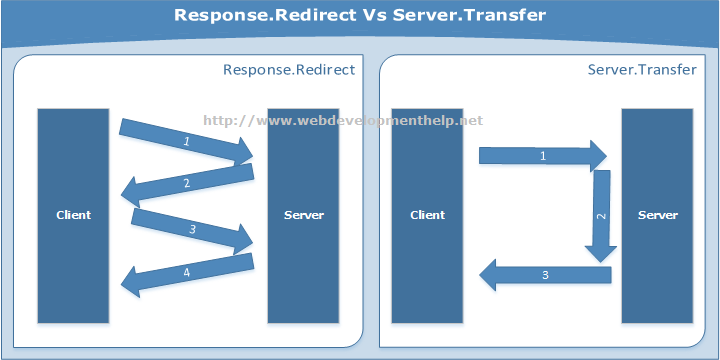
Custom controls are basically compiled code, i.e., DLLs. These can be easily added to toolbox, so it can be easily used across multiple projects using drag and drop approach. These controls are comparatively hard to create.

But User Controls (.ascx) are just like pages (.aspx). These are comparatively easy to create but tightly coupled with respect to User Interface and code. In order to use across multiple projects, we need to copy and paste to the other project as well.

### Difference between Response.Redirect and Server.Transfer?

In case of Response.Redirect, a new request is generated from client-side for redirected page. It's a kind of additional round trip. As new request is generated from client, so the new URL is visible to user in browser after redirection.

While in case of Server.Transfer, a request is transferred from one page to another without making a round trip from client. For the end user, URL remains the same in browser even after transferring to another page.



### What are the different types of Validation controls in ASP.NET?

In order to validate user input, ASP.NET provides validation server controls. All validation controls inherit from BaseValidator class which contains the common validation properties and methods like ControlToValidate, Enabled, IsValid, EnableClientScript, ValidationGroup,Validate(), etc.

ASP.NET provides a range of validation controls:

* RequiredFieldValidator validates compulsory/required input.
* RangeValidator validates the range. Validates that input falls between the given range values.
* CompareValidator validates or compares the input of a control with another control value or with a fixed value.
* RegularExpressionValidator validates input value against a defined regular expression pattern.
* CustomValidator allows to customize the validation logic with respect to our application logic.
* ValidationSummary displays all errors on page collectively.

### What are the types of Authentication in ASP.NET?

There are three types of authentication available in ASP.NET:

* **Windows Authentication**: This authentication method uses built-in windows security features to authenticate user.
* **Forms Authentication**: authenticate against a customized list of users or users in a database.
* **Passport Authentication**: validates against Microsoft Passport service which is basically a centralized authentication service.

### What are Session state modes in ASP.NET?

ASP.NET supports different session state storage options:

* **In-Process** is the default approach. It stores session state locally on same web server memory where the application is running.
* **StateServer** mode stores session state in a process other than the one where application is running. Naturally, it has added advantages that session state is accessible from multiple web servers in a Web Farm and also session state will remain preserved even web application is restarted.
* **SQLServer** mode stores session state in SQL Server database. It has the same advantages as that of StateServer.
* **Custom** modes allows to define our custom storage provider.
* **Off** mode disables session storage.

