**What is ASP.NET**

ASP.NET is a **web application development** framework for building web sites and web applications that follows object oriented programming approach.

**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 same page. So, the whole page is refreshed.

**What is the difference between a post back and a call back?**

*A****callback****is generally a call for execution of a function after another function has completed.*

**Difference between ASP.NET WebForms and ASP.NET MVC?**

ASP.NET Web Forms uses Page controller pattern approach for rendering layout. In this approach, every page has it's own controller i.e. code-behind file that processes the request. On the other hand, ASP.NET MVC uses Front Controller approach. In this approach a common controller for all pages, processes the requests.

**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 update 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.

**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 couple 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.

**Please briefly explain the usage of Global.asax?**

Global.asax is basically ASP.NET Application file. It’s a place to write code for Application-level events such as Application start, Application end, Session start and end, Application error etc. raised by ASP.NET or by HTTP Modules.

There is a good list of events that are fired but following are few of the important events in Global.asax:

* Application\_Init occurs in case of application initialization for the very first time.
* Application\_Start fires on application start.
* Session\_Start fires when a new user session starts
* Application\_Error occurs in case of an unhandled exception generated from application.
* Session\_End fires when user session ends.
* Application\_End fires when application ends or time out.

**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 inherits 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: What are they?

* **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.

**What are HttpHandlers and HttpModules in ASP.NET?**

**HttpHandler** An ASP.NET HTTP handler is the process (frequently referred to as the "endpoint") that runs in response to a request made to an ASP.NET Web application. The most common handler is an ASP.NET page handler that processes .aspx files. When users request an .aspx file, the request is processed by the page through the page handler. You can create your own HTTP handlers that render custom output to the browser.

**Built-in HTTP Handlers in ASP.NET**

ASP.NET maps HTTP requests to HTTP handlers based on a file name extension. Each HTTP handler can process individual HTTP URLs or groups of URL extensions in an application. ASP.NET includes several built-in HTTP handlers, as listed in the following table.

ASP.NET page handler (\*.aspx) The default HTTP handler for all ASP.NET pages.

Web service handler (\*.asmx) The default HTTP handler for Web service pages created as .asmx files in ASP.NET.

Generic Web handler (\*.ashx) The default HTTP handler for all Web handlers that do not have a UI.

**What are some Typical uses for custom HTTPHandlers ?**

* **Image server**   If you want a Web application to serve images in a variety of sizes, you can write a custom handler to resize images and then send them to the user as the handler's response.
* **RSS feeds**   To create an RSS feed for a Web site, you can create a handler that emits RSS-formatted XML. You can then bind a file name extension such as .rss to the custom handler. When users send a request to your site that ends in .rss, ASP.NET calls your handler to process the request.

**What are some typical uses for HTTP modules?**

* **Security**
* **Statistics and logging**   Because HTTP modules are called on every request, you can gather request statistics and log information in a centralized module, instead of in individual pages.
* **Custom headers or footers**   Because you can modify the outgoing response, you can insert content such as custom header information into every page response.

**What are the different page navigation techniques in asp.net?**

**1. Hyperlink control**  **2. Response.Redirect**  
**3. Server.Transfer**  
**4. Server.Execute**  
**5. Cross-Page postback**  
**6. Window.Open**

**What is State Management?**

HTTP is a stateless protocol by nature. So, we need some mechanism to preserve state (i.e. state of a webpage, a control or an object etc.) between subsequent requests to server from one or more clients. And this mechanism is referred as **State Management**.

**State Management techniques used in ASP.NET can be categorized in two types:**

1. Client-Side State Management
   * View State
   * Control State
   * Hidden Fields
   * Cookies
   * Query String
2. Server-Side State Management
   * Application State
   * Session State
   * Profile Properties
   * Cache

**What is ViewState? or Explain ViewState as State Management Technique?**

ViewState is one of the **Client-Side State Management** techniques that provides page-level state management, which means state is preserved between subsequent requests to same page. By using this technique, state of the page along with its controls is stored in a hidden form field  i.e. “\_\_VIEWSTATE” and this field is again available on server when page is posted back with HTTP Request.  
You can find this hidden field by looking into view source of an .ASPX page as:

*<input type=”hidden” name=”\_\_VIEWSTATE” value=”wEPDwUKMTM4OTIxNTEzNA9kFgJmD2QWAgIBD2QWAgIDDxYCHgVzdHlsZQV” />*

ViewState data is encoded in Base64 String encoded format.

**Can we Enable/Disable ViewState?**

Yes, ViewState can be enabled or disable at different levels:

* Control Level  
  ViewState for a specific control can be enabled or disabled by setting*EnableViewState* property.

*Control.EnableViewState = false;*

* Page Level

We can enable/disable ViewState for a complete page using page directive.

*<%@ Page Language=”C#” EnableViewState=”false” %>*

* Application Level  
  For whole application, we can enable/disable views in configuration file as follows

*<pages enableViewState=”false”>*

**What is the difference between Session.Clear() and Session.Abandon() in ASP.NET?**

As we understand that Session is a Collection and it stores data as Key/Value pair. So,  
**Session.Clear()** clears all the session values but doesn’t destroy the Session. however,

**Session.Abandon()** destroys the session object.

In other words, Session.Clear() is like deleting all files inside a folder (say “Root”) but Session.Abandon() means deleting the “Root” folder.

You might use **Session.Abandon()** on a logout function.

**What is the difference between Application and Session State?**

Application state is a common data repository for an application and all of its users.

On the other hand, Session state is specific to a single user session.

**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.

**What is the difference between Label Control and Literal Control?**

A Label control in ASP.NET renders text inside <span> tags while a Literal Control renders just the text without any tags.  
With Label controls we can easily apply styles using it’s CssClass property, however, if we don’t want to apply style/formatting, it’s better to go for a Literal control.

**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 inherits 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.

#### HTML Server Controls Vs Web Server Controls, Please define?

**HTML Server Controls** are server-side mapped form of HTML elements. In order to make HTML elements programmable on server-side. Light weight but not feature rich.

**ASP.NET** framework added runat=”server” attribute, so it’s accessible in ASP.NET code-behind. A typical HTML Server Control is as follows: Feature rich but heavy.

**Web Server Controls** are more feature-rich as compared to HTML server controls and truly designed to provide Win Apps development experience for ASP.NET Web developers.

#### Difference between Response.Write() and Response.Output.Write()?

Difference between Response.Write() and Response.Output.Write() is that the later provide formatting capability through String.Format-Style which the former doesn’t have.

#### What is Caching and what are the benefits of using it?

Caching is a mechanism that improve performance for an application by storing data in memory for fast access. When the application will access data from Cache (i.e. in-memory) instead of fetching it from original data store (may be a database), it will definitely improve performance

 Load on server is reduced when data is fetched from Cache instead of original source, thus improving scalability of an application.

 Caching normally keep serving application data even if the original source is temporarily down, thus improving availability of an application.

**Name three types of cache Managment in ASP.NET**

**Page Output Cache**  
In case of Page Output Cache, the output of a complete web page is stored in a cache. So, when that web page is accessed again, it will be loaded from cache instead of fetching page data again from data source.

**Partial Page Cache**For Partial Page Cache (also known as Page Fragment Cache), a part or fragment of a web page is stored in Cache as opposed to complete page caching for Page Output Cache. For example, caching a user control on a web page that displays product categories using Page Fragment Cache.

**Data Cache**  
In some scenarios, we may store frequently used objects into cache using ASP.NET Cache API. So, later on, that object will be loaded from cache instead of instantiating object again and fetching data from original source for it.

#### How to use Page Output Cache in ASP.NET?

Implementing Page Output Cache in ASP.NET is simple. For Page Output Caching, **@ OutputCache** directive is used on an ASP.NET page as follows:

*<%@ OutputCache Duration=”50″ VaryByParam=”None” %>*

Duration value is in seconds and it tells the page that how long to cache the contents?  
Now, when we will access the page, it will verify that either it exists in Cache? if Yes, then verify that is it expired? If not then fetch it from Cache and render otherwise create a new instance of the page and put it back to Cache.

The other parameter of this directive is “VaryByParam”. If it’s value is specified to something as follows:

*<%@ OutputCache Duration=”50″ VaryByParam=”ProductId” %>*

Now, Cache is dependent on the value of this parameter, If the value of parameter remains same, page will be fetched from Cache otherwise it will be refreshed again.

#### How to use Page Fragment or Partial Page Cache in ASP.NET?

Page Fragment Caching uses the same @ OutputCache directive with VaryByControl parameter as follows:  
*<%@ OutputCache Duration=”50″ VaryByParam=”None” VaryByControl=”ControlName” %>*  
  
In this case, Cache is dependent on the value of Control specified in VaryByControl parameter. For example, content on a page are dependent on the selected values of a dropdownlist, so, VaryByControl will have the dropdownlist control name as value.

#### How to use Data Cache in ASP.NET?

We have already explained the usage of Data Cache above in this series of ASP.NET Interview Questions that in particular situations, we need to store objects into cache. Adding an object to Cache and accessing it from Cache is simple.

We can use “*Add*” method to add an object to Cache as:

*Cache.Add(key, value, dependencies, absoluteExpiration, slidingExpiration, priority, onRemoveCallback); if (Cache[“ProductKey”] == null)  
Cache.Add(“ProductKey”,  
objProduct,  
null,  
DateTime.Now.AddSeconds(60),  
Cache.NoSlidingExpiration,  
CacheItemPriority.High,  
null);*

***NOTE:***

*System.Runtime.Caching 4.0 >*

*System.Web.Caching 3.5 <*

#### Authentication Vs Authorization?

Authentication and Authorization are two key security related concepts that are independent but normally go together.

Authentication is a process that verifies the identity of a user. On ther hand, Authorization is the process of assigning rights/privileges to already authenticated user.

#### What is the difference between Windows Authentication and Forms Authentication in ASP.NET?

**Windows Authentication** is a way to authenticate a user against Windows accounts. Windows authentication mode is suitable for corporate users in windows environment.

In case of **Forms Authentication**, a separate list of users is maintained for authentication. For example, we can maintain the list in database and authenticate user against it.

We can set authentication mode in web.config as follows:

*<authentication mode=”Forms”>*

#### What is Protected Configuration in ASP.NET?

While developing an ASP.NET application, we normally store a number of important sensitive information in our config files like encryption keys, connection strings etc. Application vulnerability increases if this sensitive information is stored as plane text. So ***Protected Configuration*** is an ASP.NET feature that enables to encrypt such sensitive information in configuration files.

#### What is Passport Authentication?

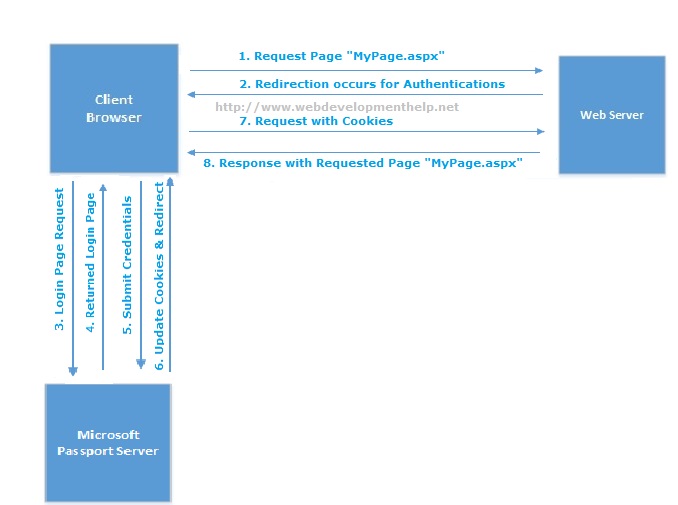
There are three types of authentications in ASP.NET i.e.

* Windows Authentication
* Forms Authentication
* Passport Authentication

Windows and Forms Authentications are already explained.  
Passport Authentication actually validates against a centralized authentication service i.e. Microsoft Passport Service. We don’t need to implement our own custom authentication mechanism if implementing .NET Passport Single Sign-In (SSI) service.

#### Can you briefly explain how Passport Authentication works?

As discussed above that Passport Authentication is a central service. It just authenticate (validate the credentials), no authorization (grant or deny access to a site). So, implementing application will check for the Passport Authentication Cookie. In case of unavailability of Passport Cookie, user is redirected to passport Sign-In page. User provides the credentials on Sign-In page, if validated,  Authentication Cookie is stored on client machine and redirected to the requested page.



#### What are the advantages of using Passport Authentication?

Advantages of Passport Authentication are:

* We don’t need to care of authentication mechanism our self, Passport SSI does this for us.
* Single login credentials can be used to access multiple sites. User don’t need to remember separate credentials for individual site.

#### What is Role-based Security?

Role Based Security is a technique we use to implement authorization on the basis of user’s roles within an   organization. It’s more granular approach to grant or revoke permissions on resources through user’s roles.

#### What are the different Security Controls in ASP.NET?

ASP.NET provides several security controls which are actually Web Server controls. You can find those in your Visual Studio Toolbox.

**Login Control:**  
In almost every application we need to take user credentials on a typical login page. Login control provides the same standard functionality and reduces the effort for building it from scratch.

**LoginName:**  
After a user successfully logged in to an application, we normally display his/her username to top right or some other place on the page. Now, this functionality is provided by LoginName control.

**LoginView Control:**  
LoginView control displays different view for different users. Using AnonymousTemplate and LoggedInTemplate, different information can be presented to different users.

**LoginStatus Control:**  
LoginStatus control implies whether a user is authenticated or not. For an unathenticated user, it displays a link to login page. On the other hand, for authenticated user, a logout link is displayed.

**LoginRecovery Control:**  
Password recovery is another important functionality simplified through PasswordRecovery control. It sends an email with login credentials to registered user email.

#### What is Impersonation in ASP.NET?

Impersonation is an act of a user to pretend itself to be another user. By default, ASP.NET executes application code using the same user account as that of ASP.NET process i.e. Network Service. But with impersonation enabled, it executes code with the windows identity of the user making the request.

For example, if a user ‘user1′ logged in and IIS is setup to run as Network Service. If ‘user1′ call a piece of code on another computer (may be a web service call), the other computer will see the IIS user instead of ‘user1′. But we can enable impersonation to allow ‘user1′ to access the web service using its windows identity instead of Network Service.

#### How to configure Impersonation in ASP.NET?

config file. default is false

#### Define AJAX?

AJAX stands for “Asynchronous JavaScript and XML”. It’s basically a technique for creating Rich Internet Applications (RIA) that are faster as well as more interactive, using a combination of commonly used techniques as HTML/XHTML, CSS, Document Object Model (DOM), JavaScript, XML/XSLT and XMLHttpRequest object.

**XMLHttpRequest** object is the key basis of AJAX and makes it possible to communicate with the web server asynchronously and exchange data.

#### Please elaborate XMLHttpRequest Object further?

XMLHttpRequest is the core object in AJAX technology regardless of any implementation. XMLHttpRequest object is used to exchange data with a server seamlessly. Basically JavaScript uses this Object to exchange XML as well as text data between client and server. An AJAX implementation uses this object and communicate with server but it doesn’t require the complete page to be refreshed.

#### How to send a request to server using XMLHttpRequest Object?

We can send a request to server using HTTP GET and POST methods as follows:

**//Simple GET Request**

var xmlHttp = new XMLHttpRequest();

xmlHttp.open(“GET”, “TestFile.txt”, true);

xmlHttp.send();

**//Simple POST Request**

var xmlHttp = new XMLHttpRequest();

xmlHttp.open(“POST”, “TestFile.txt”, true);

xmlHttp.send();

#### What is ASP.NET AJAX?

Microsoft provided an implementation of AJAX functionality known as**ASP.NET AJAX**.  
As we discussed in above interview question that AJAX is a combination of various techniques, so Microsoft simplified the usage of these techniques with its own implementation. ASP.NET AJAX is a set of extensions to ASP.NET and comes with reusable AJAX controls. Using ASP.NET AJAX, we can develop applications that can update partial page instead of a complete page refresh.

#### Difference between Synchronous and Asynchronous Postback?

In Synchronous postback, complete web page is sent to server and in return rendering the output (i.e. complete page), whereas in case of Asynchronous postback, partial page goes to the server and renders only partial (required) part of the page.

Normally a method making Synchronous call always waits for response to do next task (i.e. might be another call). On the other hand, for Asynchronous, no waiting required

#### What are the basic controls in ASP.NET AJAX?

Following controls can be considered as core AJAX controls in ASP.NET.

* ScriptManager
* ScriptManagerProxy
* UpdatePanel
* UpdateProgress
* Timer

#### What is a ScriptManager in ASP.NET AJAX?

In order to use AJAX functionality on a web page, we add a ScriptManager control to the page in most of the scenarios, because ScriptManager control register AJAX library scripts to that particular web page. We can have only one ScriptManager per page.

*<asp:ScriptManager ID=”ScriptManager1″ runat=”server”></asp:ScriptManager>*

ScriptManager manages all ASP.NET AJAX resources of a web page, creates proxies for asynchronous web service call and also manages partial page updates… etc.

#### ScriptManager Vs ScriptManagerProxy?

As we understand that we can have only one **ScriptManager** control on a page but we can have multiple **ScriptManagerProxy** controls.  
Consider a scenario that we have ScriptManager in our MasterPage that is available for all content pages. Now, we wanted to register a web service in a particular page. So, we will not add another ScriptManager to that page instead we will add ScriptManagerProxy to it in order to avoid error.

#### What is the role of UpdatePanel in ASP.NET AJAX?

UpdatePanel is the control that facilitate the partial page rendering functionality in an ASP.NET application. As discussed earlier that using ASP.NET AJAX, we can communicate with a web server asynchronously and update a part of a page without a complete page postback. In order to apply partial page update/rendering, we can add one or more UpdatePanel controls to our ASP.NET Page

#### What are the limitations of AJAX?

* AJAX on an application will not work if JavaScript is disabled.
* In some scenarios, it exposes vulnerability.
* It will always be difficult to bookmark application state.
* Application behavior may be slow in some scenarios, because of different loading time of controls on a single page.

#### What is ASP.NET Web API?

ASP.NET Web API is a framework that simplifies building HTTP services for broader range of clients (including browsers as well as mobile devices) on top of .NET Framework. Using ASP.NET Web API we can create non-SOAP based services like plain XML or JSON strings etc. with many other advantages including:

* Create resource-oriented services using the full features of HTTP.
* Exposing services to a variety of clients easily like browsers or mobile devices etc.

#### What are the advantages of using ASP.NET Web API?

Using ASP.NET Web API has a number of advantages, but core of the advantages are:

* It works the HTTP way using standard HTTP verbs like GET, POST, PUT, DELETE etc for all CRUD operations.
* Complete support for routing.
* Response generated in JSON or XML format using MediaTypeFormatter.
* It has the ability to be hosted in IIS as well as self-host outside of IIS.
* Supports Model binding and Validation.
* Support for OData.
* and more….

#### What new features are introduced in ASP.NET Web API 2.0?

More new features introduced in ASP.NET Web API framework v2.0 are as follows:

* Attribute Routing
* External Authentication
* CORS (Cross-Origin Resource Sharing)
* OWIN (Open Web Interface for .NET) Self Hosting
* IHttpActionResult
* Web API OData

#### WCF Vs ASP.NET Web API?

Actually, **Windows Communication Foundation** is designed to exchange standard SOAP-based messages using variety of transport protocols like HTTP, TCP, NamedPipes or MSMQ etc. On the other hand, **ASP.NET API** is a framework for building non-SOAP based services over HTTP only

#### is it true that ASP.NET Web API has replaced WCF?

It’s a misconception that ASP.NET Web API has replaced WCF. It’s another way of building non-SOAP based services, for example, plain XML or JSON string etc.

Yes, it has some added advantages like utilizing full features of HTTP and reaching more clients such as mobile devices etc.

But WCF is still a good choice for following scenarios:

* If we intended to use transport other than HTTP e.g. TCP, UDP or Named Pipes.
* Messag Queuing scenario using MSMQ.
* One-way communication or Duplex communication

#### How to return View from ASP.NET Web API method?

(A tricky Interview Question) No, we can’t return view from ASP.NET Web API Method. As we discussed in earlier interview question about difference between ASP.NET MVC and Web API that ASP.NET Web API creates HTTP services that renders raw data. Although, it’s quite possible in ASP.NET MVC application.  
[Back to top](http://www.webdevelopmenthelp.net/2014/05/asp-net-web-api-interview-questions.html#top)

#### How to restrict access to Web API method to specific HTTP Verb?

Attribute programming plays it’s role here. We can easily restrict access to an ASP.NET Web API method to be called using a specific HTTP method. For example, we may required in a scenario to restrict access to a Web API method through HTTP POST only as follows:

**[HttpPost]** public void UpdateStudent(Student aStudent) { StudentRepository.AddStudent(aStudent); }

[Back to top](http://www.webdevelopmenthelp.net/2014/05/asp-net-web-api-interview-questions.html#top)

#### Can we use Web API with ASP.NET Web Form?

Yes, ASP.NET Web API is bundled with ASP.NET MVC framework but still it can be used with ASP.NET Web Form. It can be done in three simple steps as follows:

1. Create a Web API Controller.
2. Add a routing table to Application\_Start method of Global.asax.
3. Make a jQuery AJAX Call to Web API method and get data.

[jQuery call to Web API](http://www.webdevelopmenthelp.net/2014/01/performing-crud-operations-using-asp-net-web-api-part-2.html) for all CRUD (Create, Retrieve, Update, Delete) operations can be [found here](http://www.webdevelopmenthelp.net/2014/01/performing-crud-operations-using-asp-net-web-api-part-2.html).

[Back to top](http://www.webdevelopmenthelp.net/2014/05/asp-net-web-api-interview-questions.html#top)

#### How we can provide an alias name for ASP.NET Web API action?

We can provide an alias name for ASP.NET Web API action same as in case of ASP.NET MVC by using “ActionName” attribute as follows:

[HttpPost] **[ActionName(“SaveStudentInfo”)]** public void UpdateStudent(Student aStudent) { StudentRepository.AddStudent(aStudent); }

[Back to top](http://www.webdevelopmenthelp.net/2014/05/asp-net-web-api-interview-questions.html#top)

In this ASP.NET Tutorial, we covered most important Interview Questions on ASP.NET Web API framework. Hopefully, it will be helpful for Web API developer Interview but along with these questions, do the practical implementation as much as you can. In [**Practical guide to ASP.NET Web API**](http://www.webdevelopmenthelp.net/2014/02/a-practical-guide-to-asp-net-web-api.html), you can find a good step by step approach for understanding and implementing ASP.NET Web API services

**Session Vs Cache**

One important difference is, that items in the cache can expire (will be removed from cache) after a specified amount of time. Items put into a session will stay there, until the session ends.

ASP.NET can also remove items from cache when the amount of available memory gets small.

Another difference: the session state can be kept external (state server, SQL server) and shared between several instances of your web app (for load balancing). This is not the case with the cache.

Besides of these differences (as others have noted): session is per user/session while cache is per application.

## The XMLHttpRequest Object

The XMLHttpRequest object is used to exchange data with a server behind the scenes.

The XMLHttpRequest object is **a developer's dream**, because you can:

* Update a web page without reloading the page
* Request data from a server after the page has loaded
* Receive data from a server after the page has loaded
* Send data to a server in the background
* **XMLHttpRequest** (**XHR**) is an API available to web browser scripting languages such as JavaScript. It is used to send HTTP or HTTPS requests to a web server and load the server response data back into the script.

The most obvious question which comes to my mind is then what is the difference between callback and postback. Both hit the server and gets data back to the client. Callbacks differ from postback in which the former only sends relevant information back to the server and preserves the **Viewstate**and the form data. All the ASP.NET server page cycles occur along with client side events. This helps in loading the page faster because in the rendering phase only the **relevant**contentis sent to the browser. For example in AJAX, only the contents ofthe UpdatePanel are rendered in the browser.

## Response Object

**The ASP Response object is used to send output to the user from the server.**

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

### Collections

|  |  |
| --- | --- |
| **Collection** | **Description** |
| [Cookies](http://www.w3schools.com/asp/coll_cookies_response.asp) | Sets a cookie value. If the cookie does not exist, it will be created, and take the value that is specified |

### Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| [Buffer](http://www.w3schools.com/asp/prop_buffer.asp) | Specifies whether to buffer the page output or not |
| [CacheControl](http://www.w3schools.com/asp/prop_cachecontrol.asp) | Sets whether a proxy server can cache the output generated by ASP or not |
| [Charset](http://www.w3schools.com/asp/prop_charset.asp) | Appends the name of a character-set to the content-type header in the Response object |
| [ContentType](http://www.w3schools.com/asp/prop_contenttype.asp) | Sets the HTTP content type for the Response object |
| [Expires](http://www.w3schools.com/asp/prop_expires.asp) | Sets how long (in minutes) a page will be cached on a browser before it expires |
| [ExpiresAbsolute](http://www.w3schools.com/asp/prop_expiresabsolute.asp) | Sets a date and time when a page cached on a browser will expire |
| [IsClientConnected](http://www.w3schools.com/asp/prop_isclientconnected.asp) | Indicates if the client has disconnected from the server |
| [Pics](http://www.w3schools.com/asp/prop_pics.asp) | Appends a value to the PICS label response header |
| [Status](http://www.w3schools.com/asp/prop_status.asp) | Specifies the value of the status line returned by the server |

### Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| [AddHeader](http://www.w3schools.com/asp/met_addheader.asp) | Adds a new HTTP header and a value to the HTTP response |
| [AppendToLog](http://www.w3schools.com/asp/met_appendtolog.asp) | Adds a string to the end of the server log entry |
| [BinaryWrite](http://www.w3schools.com/asp/met_binarywrite.asp) | Writes data directly to the output without any character conversion |
| [Clear](http://www.w3schools.com/asp/met_clear.asp) | Clears any buffered HTML output |
| [End](http://www.w3schools.com/asp/met_end.asp) | Stops processing a script, and returns the current result |
| [Flush](http://www.w3schools.com/asp/met_flush.asp) | Sends buffered HTML output immediately |
| [Redirect](http://www.w3schools.com/asp/met_redirect.asp) | Redirects the user to a different URL |
| [Write](http://www.w3schools.com/asp/met_write_response.asp) |  |