2011

.Net Interview Questions

Quick Reference and FAQ

General questions specifically from IT and .NET interviews point of view. Best for Fresher and students who want to have a feel of what .NET questions are asked in multinational companies.

Abhishek Goenka
CIBERSITES INDIA
Version 1.0

ASP.Net Interview Questions

From constructor to destructor (taking into consideration Dispose() and the concept of non-deterministic finalization), what are the events fired as part of the ASP.NET System.Web.UI.Page lifecycle. Why are they important? What interesting things can you do at each?

As all of us know a request comes from Client (Browser) and sends to Server (we call it as Web server) in turn server process the request and sends response back to the client in according to the client request.

But internally in the web server there is quite interesting process that happens. To get aware of that process we should first of all know about the architecture of the IIS

It mainly consists of 3 Parts/Files

- Inetinfo.exec
- ISAPI Filer (Container for Internet Server Application Interface dlls),
- Worker Process (aspnet_wp.exe)

Inetinfo.exe is the ASP.Net request handler that handles the requests from the client .If it's for static resources like HTML files or image files inetinfo.exe process the request and sent to client. If the request is with extension aspx/asp, inetinfo.exe processes the request to API filter. ISAPI filter will have several runtime modules called as ISAPI extensions. To process the request ISAPI filter takes the help of these runtime modules. The runtime module loaded for ASP page is asp.dll. And for ASP.NET page it's ASPNET_ISAPI.dll. From here the request is processed to the "worker process". Worker Process will have several application domains.

Worker process sends the request to HTTPPIPE line.(HTTP Pipeline is nonetheless collection of .net framework classes). HTTP Pipeline compiles the request into a library and makes a call to HTTP runtime and runtime creates an instance of page class

```
public class File : System.Web.UI.Page
{}
```

ASP.Net web page is a class derived from page class, this page class resides in system.web.dll

After creating instance pf page class HTTP Runtime immediately invokes process request method of page class

```
Page Req = new Page();
Req.ProcessRequest();
```


Prepared by Abhishek Goenka
Email – mailtoagoenka@gmail.com

Page Event	Typical Use
PreInit	Raised after the start stage is complete and before the initialization stage begins. **Use this event for the following:** • Check the IsPostBack property to determine whether this is the first time the page is being processed. The IsCallback and IsCrossPagePostBack properties have also been set at this time. • Create or re-create dynamic controls. • Set a master page dynamically. • Set the Theme property dynamically. • Read or set profile property values.
Init	Raised after all controls have been initialized and any skin settings have been applied. The Init event of individual controls occurs before the Init event of the page. Use this event to read or initialize control properties.
InitComplete	Raised at the end of the page's initialization stage. Only one operation takes place between the Init and InitComplete 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 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 event. Use this event to make changes to view state that you want to make sure are persisted after the next postback.
PreLoad	Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the Request instance.
Load	The Page object calls the OnLoad method on the Page object, and then recursively does the same for each child control until the page and all controls are loaded. The Load event of individual controls occurs after the Load event of the page. Use the OnLoad 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 control's Click event or a TextBox control's TextChanged event.
LoadComplete	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	Raised after the Page 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 object calls EnsureChildControls for each control and for the page.) The Page object raises the PreRender event on the Page object, and then recursively does the same for each child control. The PreRender event of individual controls occurs after the PreRender event of the page.

Prepared by Abhishek Goenka
Email – mailtoagoenka@gmail.com

	Use the event to make final changes to the contents of the page or its controls before the rendering stage begins.
PreRenderComplete	Raised after each data bound control whose DataSourceID property is set calls its DataBind method. For more information, see Data Binding Events for Data-Bound Controls later in this topic.
SaveStateComplete	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	This is not an event; instead, at this stage of processing, the Page object calls this method on each control. All ASP.NET Web server controls have a Render 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 method. For more information, see Developing Custom ASP.NET Server Controls. A user control (an .ascx file) automatically incorporates rendering, so you do not need to explicitly render the control in code.
Unload	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.

Although both Init and Load recursively occur on each control, they happen in reverse order. The Init event (and also the Unload event) for each child control occur before the corresponding event is raised for its container (bottom-up). However the Load event for a container occurs before the Load events for its child controls (top-down). Master pages behave like child controls on a page: the master page Init event occurs before the page Init and Load events, and the master page Load event occurs after the page Init and Load events.

What is EnableViewStateMAC?

Setting EnableViewStateMac=true is a security measure that allows ASP.NET to ensure that the viewstate for a page has not been tampered with. If on Postback, the ASP.NET framework detects that there has been a change in the value of viewstate that was sent to the browser, it raises an error - Validation of viewstate MAC failed.

Prepared by Abhishek Goenka
Email – mailtoagoenka@gmail.com

Use <%@ Page EnableViewStateMac="true"%> to set it to true (the default value, if this attribute is not specified is also true) in an aspx page.

But this has a side effect: it also prevents multiple servers from processing the same ViewState. One solution is to force every server in your farm to use the same key-- generate a hex encoded 64-bit or 128-bit <machineKey> and put that in each server's machine.config :

<!-- validation="[SHA1|MD5|3DES]" -->
<machineKey validation="SHA1"
validationKey="F3690E7A3143C185A6A8B4D81FD55DD7A69EEAA3B32A6AE813ECEEC" />

What is the difference between asp:Label and asp:Literal control?

asp:Label control

asp:Label control is used to write text on the page with different formatting options like bold, italic, underlined etc

asp:Literal control

Ideally Literal control is the rarely used control which is used to put static text on the web page. When it is rendered on the page, it is implemented just as a simple text.

Unlike asp:Label control, there is no property like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc. of Literal control. That makes it more powerful, you can even put a pure HTML contents into it.

What's a SESSION and APPLICATION object?

Viewstate - Viewstate is a hidden fields in an ASP.NET page, contains state of those controls on a page whose "EnableViewstate" property is "true".

You can also explicitly add values in it, on an ASP.NET page like:

```
Viewstate.Add( "TotalStudents", "87" );
```

Viewstate should be used when you want to save a value between different roundtrips of a single page as Viewstate of a page is not accessible by another page. Because Viewstate renders with the page, it consumes bandwidth, so be careful to use it in applications to be run on low bandwidth.

Session Variable - Session variables are usually the most commonly used. When a user visits a site, it's sessions starts and when the user become idle or leave the site, the session ends. Session variables should be used to save and retrieve user specific information required on multiple pages. Session variables consumes server memory, so if your may have a huge amount visitors, use session very carefully and instead of put large values in it try to put IDs and references

Application variables - Application variables are shared variables among all users of a web application. Application variables behave like static variables and they are substitute of static variables as static variables are stateless in web applications. Only shared values should be persisted in Application variables, and as soon as they are not in use they should be removed explicitly.

Cache - Cache is probably the least used state feature of ASP.NET. Cache is basically a resource specific state persistence feature, means unlike session it stick with resource instead of user, for instance: pages,

controls etc. Cache should be used or frequently used pages, controls, and data structures. Data cache can be used to cache frequently used list of values e.g. list of products

Cookies - Cookies are some values saved in browsers by the website to retrievable and use afterwards. Usually cookies are used to help dynamic websites to identify visitors and retrieve their saved preferences. Cookies are also used to facilitate auto login by persisting user id in a cookie save in user's browser. Because cookies have been saved at client side, they do not create performance issues but may create security issues as they can be hacked from browser.

What are the different types of caching?

CachingOutput Caching - We can use Page output for those page which content are relatively static. So rather than generating the page on each user request we can cached the page using Page output caching so that it can be access from cache itself. So, Instead of pages can be generated once and then cached for subsequent request. Page output caching allows the entire content of a given page to be stored in the cache.

<%@ Page Language="C#" %>

<%@ OutputCache Duration='300' VaryByParam='none' %>

Fragment caching - ASP.NET provides a mechanism for caching portions of pages, called page fragment caching. To cache a portion of a page, you must first encapsulate the portion of the page you want to cache into a user control. In the user control source file, add an OutputCache directive specifying the Duration and VaryByParam attributes. When that user control is loaded into a page at runtime, it is cached, and all subsequent pages that reference that same user control will retrieve it from the cache.

<!— UserControl.ascx —>

<%@ OutputCache Duration='60'

VaryByParam='none' %>

<%@ Control Language="'C#'" %>

Data Caching - Caching of data can dramatically improve the performance of an application by reducing database contention and round-trips. Simply data caching store the required data in cache so that web server did not send request to DB server every time for each and every request which increase the web site performance.

There are three Different ways to add data or object into cache. But based upon the situation we have to access it. These methods are Cache[], Cache.add(), cache.insert(). The following table will show you the clear difference of these methods.

	Stored data in cache	Support Depedency	Support Expiration	Support Priority Settings	Return Object
cache[]	Yes	No	No	No	No
cache.insert()	Yes	Yes	Yes	Yes	No
cache.add()	Yes	Yes	Yes	Yes	Yes

Is it possible to prevent a browser from caching an ASPX page?

Just call SetNoStore on the HttpCachePolicy object exposed through the Response object's Cache property, as demonstrated here:

```
<%@ Page Language="C#" %>
<html>
  <body>
    <%
      Response.Cache.SetNoStore ();
      Response.Write (DateTime.Now.ToLongTimeString ());
    %>
  </body>
</html>
```

What does AspCompat="true" mean and when should I use it?

The AspCompat attribute forces the page to execute in STA mode. The runtime throws an exception if the compatibility tag is omitted and an STA component is referenced on the page. If you convert the STA component to an assembly using Tlbimp.exe, the runtime does not detect that the component uses the STA model and does not throw an exception, but your application can suffer from poor performance.
<%@Page AspCompat=true Language = VB%>

What are the main event handlers in Global.asax?

ASP.NET provides several modules that participate in each request and expose events you can handle in Global.asax. You can customize and extend these modules as you like, or develop completely new custom modules to process information for and about HTTP requests made to your ASP.NET-based application. Following are important events catered for in the Global.asax file.

- **Application_Init**: Fires when the application initializes for the first time.
- **Application_Start**: Fires the first time an application starts.
- **Session_Start**: Fires the first time when a user's session is started.
- **Application_BeginRequest**: Fires each time a new request comes in.
- **Application_EndRequest:** Fires when the request ends.
- **Application_AuthenticateRequest**: Indicates that a request is ready to be authenticated.
- **Application_Error:** Fires when an unhandled error occurs within the application.
- **Session_End:** Fires whenever a single user Session ends or times out.
- **Application_End:** Fires when the application ends or times out (Typically used for application cleanup logic).

What are different types of directives in .NET?

@Page: Defines page-specific attributes used by the ASP.NET page parser and compiler. Can be included only in .aspx files <%@ Page AspCompat="TRUE" language="C#" %>

@Control: Defines control-specific attributes used by the ASP.NET page parser and compiler. Can be included only in .ascx files. <%@ Control Language="VB" EnableViewState="false" %>

@Import: Explicitly imports a namespace into a page or user control. The Import directive cannot have more than one namespace attribute. To import multiple namespaces, use multiple @Import directives. <% @ Import Namespace="System.web" %>

@Implements: Indicates that the current page or user control implements the specified .NET framework interface.<%@ Implements Interface="System.Web.UI.IPostBackEventHandler" %>

@Register: Associates aliases with namespaces and class names for concise notation in custom server control syntax.<%@ Register Tagprefix="Acme" Tagname="AdRotator" Src="AdRotator.ascx" %>

@Assembly: Links an assembly to the current page during compilation, making all the assembly's classes and interfaces available for use on the page. <%@ Assembly Name="MyAssembly" %><%@ Assembly Src="MySource.vb" %>

@OutputCache: Declaratively controls the output caching policies of an ASP.NET page or a user control contained in a page<%@ OutputCache Duration="#ofseconds" Location="Any | Client | Downstream | Server | None" Shared="True | False" VaryByControl="controlname" VaryByCustom="browser | customstring" VaryByHeader="headers" VaryByParam="parametername" %>

@Reference: Declaratively indicates that another user control or page source file should be dynamically compiled and linked against the page in which this directive is declared.

What are ASHX files? What are HttpHandlers? Where can they be configured?

ASP.NET programming supports the creation of custom HttpHandler components, which provide a flexible and efficient way to process requests that don't return standard HTML-based pages. For example, HttpHandler components are great for situations in which you want to return simple text, XML, or binary data to the user.

The easiest way to create a custom HttpHandler component is to create a source file with an .ashx extension. You must then add a @WebHandler directive to the top of the .ashx file, along with a class definition that implements the IHttpHandler interface. Any class that implements the IHttpHandler interface must provide an implementation of the IsReusable method and the ProcessRequest method.

```
<%@ Assembly Name="Microsoft.SharePoint, [full assembly name]" %>
<%@ WebHandler Language="C#" Class="HelloHttpHandler" %>
using System;
using System.Web;
using Microsoft.SharePoint;

public class HelloHttpHandler : IHttpHandler {
  public bool IsReusable {
    get { return false; }
  }
  public void ProcessRequest(HttpContext context) {
    SPSite siteColl = SPContext.Current.Site;
    SPWeb site = SPContext.Current.Web;
    context.Response.ContentType = "text/plain"
    context.Response.Write("Hello HttpHandler from the site " +
                           site.Title +
                           " at " +
                           site.Url);
  }
}
```


Prepared by Abhishek Goenka
Email – mailtoagoenka@gmail.com

After you deploy your .ashx file within a directory nested within the \LAYOUTS directory, it is accessible to any site in the farm by using a site-relative path.

```
http://MyWebServer/sites/Sales/_layouts/Litware/HelloHttpHandler.ashx
```

What is needed to configure a new extension for use in ASP.NET? For example, what if I wanted my system to serve ASPX files with a *.jsp extension?

It is possible to configure new extension for use in ASP.Net. This as to be configured in IIS actually in order for IIS to route your pages to the proper ISAPI

Follow this: http://blogs.msdn.com/gduthie/archive/2007/03/14/custom-file-extensions-in-asp-net-2-0.aspx

What events fire when binding data to a data grid? What are they good for?

ItemCreated: The ItemCreated event is fired when an item in a DataGrid control is created. This means that at the time the event is fired, the DataGrid does not yet know about the data that will be bound to it. So, if the logic of your method depends on this data being available to the control, you're better off using the ItemDataBound event. Other than that, the ItemCreate event differentiates itself in one other way from the ItemDataBound event: the ItemCreated event is raised when data is bound to the control and during round-trips (postbacks). These qualities make the event especially well-suited to add custom attributes to a DataRow (such as onmouseover or other javascript events) or to control the appearance in ways that do not depend on the data within the DataRow (such as making every 10th row a different color).

ItemDataBound: The ItemDataBound event is fired after after an item in a DataGrid control is bound. This means that (unlike the ItemCreated event) you can add special formatting to a DataRow that is dependent upon the data contained within that row. Since ItemDataBound is fired after the ItemCreated event, it is within this event that you are presented with the final opportunity to access the data before it is rendered to the client. These qualities make the event well-suited for changing the appearance of a row or cell based on the data within that row or cell (such as highlighting outliers or other important information).Example:

Assume we have the following DataGrid declared on our .aspx page:

```
<asp:DataGrid ID="MainDataGrid"
   runat="server"
   AutoGenerateColumns="true"
   OnItemDataBound="MainDataGrid_ItemDataBound"
   OnItemCreated="MainDataGrid_ItemCreated" />
```

On the code behind page then, we can create the following two methods to handle adding titles to header row, to specify more descriptive headers, and to change the row background color based on an employee's salary:

```
protected void MainDataGrid_ItemCreated(object sender, DataGridItemEventArgs e)
{
   //If the item is in the header
   if (e.Item.ItemType == ListItemType.Header)
   {
      //Iterate through each cell
      foreach(TableCell item in e.Item.Cells)
      {
         //Add the title attribute — we could just as easily
         //add a javascript onmouseover event here
         item.Attributes.Add("title", item.Text);
      }

      //Since the header values are set before we have access
      //to the data, we can modify the third column header to
      //be a bit more descriptive
      e.Item.Cells[2].Text = "Salary (in US$)";
   }
}


protected void MainDataGrid_ItemDataBound(object sender, DataGridItemEventArgs e)
{
   //Since DataGrid differentiates between Items and AlternatingItems, you sometimes
have to check
   //for one *or* the other

if (e.Item.ItemType == ListItemType.Item || e.Item.ItemType == ListItemType.Alternating
Item)
   {
      //Here we will modify the row color based on the salary
      //We can only do this within ItemDataBound since it relies
      //on the data being available from the data source
      if (Convert.ToInt32(e.Item.Cells[2].Text) < 10000)
      {
         e.Item.BackColor = System.Drawing.Color.LightPink;
      }
      else if (Convert.ToInt32(e.Item.Cells[2].Text) < 1000000)
      {
         e.Item.BackColor = System.Drawing.Color.LightBlue;
      }
      else
      {
         e.Item.BackColor = System.Drawing.Color.LightGreen;
      }
   }
}
```

ASP.NET Code Block Types

Interaction with the page from these blocks is limited, since the code is executed during the render phase of the page life cycle

```
<% Trace.Warn("Embedded Code Block", "Hello World"); %>
```

Let's have a look at the different types of syntax we can use for code blocks in ASP.NET pages. There are really four types of code blocks, and the first one is different from the others:

- <%$ %>
- <%# %>
- <% %>
- <%= %>

ASP.NET Expression Syntax

First of all we have ASP.NET expressions which look like <%$ AppSettings:Key %>

```
<asp:Label runat="server" Text="<%$ AppSettings:Key %>" />
```

ASP.NET Data-Binding syntax

The next code construct is the data-binding syntax: <%# Eval("Value") %> which is used to bind to properties to data on demand.

Statement and Expression/Evaluated Code Blocks

Display some values

```
<%
    string message = "Hello World!";
    Response.Write(message);
%>
```

These are delimited by <%= and %> and the content of this code block becomes the parameter to the HtmlTextWrite.Write() method. Therefore, the code inside this type of code block should be an expression, and not a statement.

```
<%= String.Format("The title of this page is: {0}", this.Title ?? "n/a") %>
```

What method do you use to explicitly kill a user s session?

Session.Abandon

Which two properties are on every validation control?

1. ControlToValidate
2. ErrorMessage

What are the validation controls in asp.net?

There are 5 validation controls in asp.net

1. RequiredFieldValidator
2. RangeValidator
3. RegularExpressionValidator
4. CompareValidator
5. CustomValidator

Prepared by Abhishek Goenka
Email – mailtoagoenka@gmail.com

ValidationSummary is not a validation control but a control that displays summary of all error occurs while validating the page.

How to load a user control dynamically in runtime?

```
Control c = (Control)Page.LoadControl("~/usercontrol/MyUserControl.ascx");
Page.Form.Controls.Add(c);
```

How to get the authentication mode from web.config file programmatically at runtime?

```
System.Web.Configuration.AuthenticationSection section =
        (AuthenticationSection)WebConfigurationManager.GetSection("system.web/authentication");
Label1.Text = section.Mode.ToString();
```

What's the difference between Response.Write() and Response.Output.Write()?

Response.Output.Write() allows you to write formatted output.

What's a bubbled event?

When you have a complex control, like DataGrid, writing an event processing routine for each object (cell, button, row, etc.) is quite tedious. The controls can bubble up their event handlers, allowing the main DataGrid event handler to take care of its constituents.

ASP.NET Compilation Tool (Aspnet_compiler.exe)

The ASP.NET Compilation tool (Aspnet_compiler.exe) enables you to compile an ASP.NET Web application, either in place or for deployment to a target location such as a production server. In-place compilation helps application performance because end users do not encounter a delay on the first request to the application while the application is compiled.

Compilation for deployment can be performed in one of two ways: one that removes all source files, such as code-behind and markup files, or one that retains the markup files.

What is different between WebUserControl and in WebCustomControl?

Web user controls :- Web User Control is Easier to create and another thing is that its support is limited for users who use a visual design tool one good thing is that its contains static layout one more thing a separate copy is required for each application.

Web custom controls: - Web Custom Control is typical to create and good for dynamic layout and another thing is that it has full tool support for user and a single copy of control is required because it is placed in Global Assembly cache.

What is smart navigation?

Enable smart navigation by using the Page.SmartNavigation property. When you set the Page.SmartNavigation property to true, the following smart navigation features are enabled:

- The scroll position of a Web page is maintained after postback.
- The element focus on a Web page is maintained during navigation.
- Only the most recent Web page state is retained in the Web browser history folder.
- The flicker effect that may occur on a Web page during navigation is minimized.

Note: Smart navigation is deprecated in Microsoft ASP.NET 2.0 and is no longer supported by Microsoft Product Support Services

How many types of cookies are there in ASP.NET ?

- In-memory cookies: An in-memory cookie goes away when the user shuts the browser down.
- Persistent cookies: A persistent cookie resides on the hard drive of the user and is retrieved when the user comes back to the Web page.

If you create a cookie without specifying an expiration date, you are creating an in-memory cookie, which lives for that browser session only. The following illustrates the script that would be used for an in-memory cookie:

```
Response.Cookies("SiteArea") = "TechNet"
```

The following illustrates the script used to create a persistent cookie:

```
Response.Cookies("SiteArea") = "TechNet"
Response.Cookies("SiteArea").Expires = "August 15, 2000"
```

TODO List

Explain how PostBacks work, on both the client-side and server-side. How do I chain my own JavaScript?
How does ViewState work and why is it either useful or evil?
What is the OO relationship between an ASPX page and its CS/VB code behind file in ASP.NET 1.1? In 2.0?
What happens from the point an HTTP request is received on a TCP/IP port up until the Page fires the On_Load event?
How does IIS communicate at runtime with ASP.NET? Where is ASP.NET at runtime in IIS5? IIS6?
What is an assembly binding redirect? Where are the places an administrator or developer can affect how assembly binding policy is applied?
Compare and contrast LoadLibrary(), CoCreateInstance(), CreateObject() and Assembly.Load().

ASP.NET MVC 3 Interview Questions

What is MVC?

MVC is a framework methodology that divides an application's implementation into three component roles: models, views, and controllers.

"**Models**" in a MVC based application are the components of the application that are responsible for maintaining state. Often this state is persisted inside a database (for example: we might have a Product class that is used to represent order data from the Products table inside SQL).

"**Views**" in a MVC based application are the components responsible for displaying the application's user interface. Typically this UI is created off of the model data (for example: we might create an Product "Edit" view that surfaces textboxes, dropdowns and checkboxes based on the current state of a Product object).

"**Controllers**" in a MVC based application are the components responsible for handling end user interaction, manipulating the model, and ultimately choosing a view to render to display UI. In a MVC application the view is only about displaying information – it is the controller that handles and responds to user input and interaction.

Which are the advantages of using MVC Framework?

MVC is one of the most used architecture pattern in ASP.NET and this is one of those ASP.NET interview question to test that do you really understand the importance of model view controller.

It provides a clean separation of concerns between UI and model.

1. UI can be unit test thus automating UI testing.
2. Better reuse of views and model. You can have multiple views which can point to the same model and also vice versa.
3. Code is better organized.

What is Razor View Engine?

Razor view engine is a new view engine created with ASP.Net MVC model using specially designed Razor parser to render the HTML out of dynamic server side code. It allows us to write Compact, Expressive, Clean and Fluid code with new syntaxes to include server side code in to HTML.

What is namespace of asp.net MVC?

ASP.NET MVC namespaces and classes are located in the System.Web.Mvc assembly.

System.Web.Mvc namespace

Contains classes and interfaces that support the MVC pattern for ASP.NET Web applications. This namespace includes classes that represent controllers, controller factories, action results, views, partial views, and model binders.

System.Web.Mvc.Ajax namespace

Contains classes that support Ajax scripts in an ASP.NET MVC application. The namespace includes support for Ajax scripts and Ajax option settings.

System.Web.Mvc.Async namespace

Contains classes and interfaces that support asynchronous actions in an ASP.NET MVC application

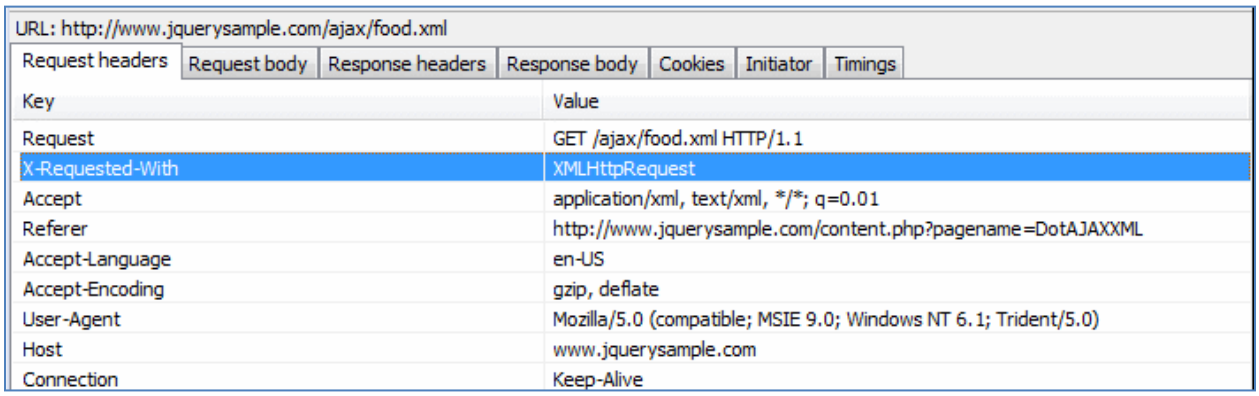
System.Web.Mvc.Html namespace

Contains classes that help render HTML controls in an MVC application. The namespace includes classes that support forms, input controls, links, partial views, and validation.

How to identify AJAX request with C# in MVC.NET?

The solution is in depended from MVC.NET framework and universal across server-side technologies. Most modern AJAX applications utilize XmlHTTPRequest to send async request to the server. Such requests will have distinct request header:

X-Requested-With = XMLHTTPREQUEST



MVC.NET provides helper function to check for Ajax requests which internally inspects X-Requested-With request header to set IsAjax flag.

HelperPage.IsAjax Property

Gets a value that indicates whether Ajax is being used during the request of the Web page.

Namespace: System.Web.WebPages

Assembly: System.Web.WebPages.dll

However, same can be achieved by checking requests header directly:

Request["X-Requested-With"] == “XmlHttpRequest”

What is Repository Pattern in ASP.NET MVC?

Repository pattern is useful for decoupling entity operations form presentation, which allows easy mocking and unit testing.

“*The Repository will delegate to the appropriate infrastructure services to get the job done. Encapsulating in the mechanisms of storage, retrieval and query is the most basic feature of a Repository implementation*”

“*Most common queries should also be hard coded to the Repositories as methods.*”

Which MVC.NET to implement repository pattern Controller would have 2 constructors on parameterless for framework to call, and the second one which takes repository as an input:

```
class myController: Controller
{
   private IMyRepository repository;

   // overloaded constructor
   public myController(IMyRepository repository)
   {
      this.repository = repository;
   }

   // default constructor for framework to call
   public myController()
   {
      //concreate implementation
      myController(new someRepository());
   }
...

   public ActionResult Load()
   {
      // loading data from repository
      var myData = repository.Load();
   }
}
```

What is difference between MVC (Model-View-Controller) and MVP(Model-View-Presenter)?

The main difference between the two is how the manager (controller/presenter) sits in the overall architecture.

All requests go first to the Controller

MVC pattern puts the controller as the main 'guy' in charge for running the show. All application request comes through straight to the controller, and it will decide what to do with the request.

Giving this level of authority to the controller isn't an easy task in most cases. Users interaction in an application happen most of the time on the View.

Thus to adopt MVC pattern in a web application, for example, the url need to become a way of instantiating a specific controller, rather than 'simply' finding the right View (webform/ html page) to render out. Every requests need to trigger the instantiation of a controller which will eventually produce a response to the user.

This is the reason why it's alot more difficult to implement pure MVC using Asp.Net Webform. The Url routing system in Asp.Net webform by default is tied in to the server filesystem or IIS virtual directory structure. Each of these aspx files are essentially Views which will always get called and instantiated first before any other classes in the project. (Of course I'm overgeneralizing here. Classes like IHttpModule, IHttpHandler and Global.asax would be instantiated first before the aspx web form pages).

MVP (Supervising Controller) on the other hand, doesn't mind for the View to take on a bigger role. View is the first object instantiated in the execution pipeline, which then responsible for passing any events that happens on itself to the Presenter.

The presenter then fetch the Models, and pass it back to the view for rendering.

What is the 'page lifecycle' of an ASP.NET MVC?

Following process are performed by ASP.Net MVC page:
1) App initialization
2) Routing
3) Instantiate and execute controller
4) Locate and invoke controller action
5) Instantiate and render view

How to call javascript function on the change of Dropdown List in ASP.NET MVC?

Create a java-script function:

```
<script type="text/javascript">
        function selectedIndexChanged() {
        }
</script>
```

Call the function:

```
<%:Html.DropDownListFor(x => x.SelectedProduct,
new SelectList(Model.Products, "Value", "Text"),
"Please Select a product", new { id = "dropDown1",
onchange="selectedIndexChanged()" })%>
```

How route table is created in ASP.NET MVC?

When an MVC application first starts, the Application_Start() method is called. This method, in turn, calls the RegisterRoutes() method. The RegisterRoutes() method creates the route table.

How do you avoid XSS Vulnerabilities in ASP.NET MVC?

Use the syntax in ASP.NET MVC instead of using in .net framework 4.0.