**Explain the Global.asax file in ASP.NET**

The Global.asax, also known as the ASP.NET application file, is located in the root directory of an ASP.NET application. This file contains code that is executed in response to application-level and session-level events raised by ASP.NET or by HTTP modules. You can also define ‘objects’ with application-wide or session-wide scope in the Global.asax file. These events and objects declared in the Global.asax are applied to all resources in that web application.

**Note 1:** The Global.asax is an optional file. Use it only when there is a need for it.

**Note 2:** If a user requests the Global.asax file, the request is rejected. External users cannot view the file.

The Global.asax file is parsed and dynamically compiled by ASP.NET. You can deploy this file as an assembly in the \bin directory of an ASP.NET application.

**How to create Global.asax**

Adding a Global.asax to your web project is quiet simple.

Open Visual Studio 2005 or 2008 > Create a new website > Go to the Solution Explorer > Add New Item > Global Application Class > Add.

**Examining the methods related to the events in Global.asax**

There are 2 ‘set’ of methods that fire corresponding to the events. The first set which gets invoked on each request and the second set which does not get invoked on each request. Let us explore these methods.

**Methods corresponding to events that fire on each request**

Application\_BeginRequest() – fired when a request for the web application comes in.

Application\_AuthenticateRequest() –fired just before the user credentials are authenticated. You can specify your own authentication logic over here.

Application\_AuthorizeRequest() – fired on successful authentication of user’s credentials. You can use this method to give authorization rights to user.

Application\_ResolveRequestCache() – fired on successful completion of an authorization request.

Application\_AcquireRequestState() – fired just before the session state is retrieved for the current request.

Application\_PreRequestHandlerExecute() - fired before the page framework begins before executing an event handler to handle the request.

Application\_PostRequestHandlerExecute() – fired after HTTP handler has executed the request.

Application\_ReleaseRequestState() – fired before current state data kept in the session collection is serialized.

Application\_UpdateRequestCache() – fired before information is added to output cache of the page.

Application\_EndRequest() – fired at the end of each request

**Methods corresponding to events that do not fire on each request**

Application\_Start() – fired when the first resource is requested from the web server and the web application starts.

Session\_Start() – fired when session starts on each new user requesting a page.

Application\_Error() – fired when an error occurs.

Session\_End() – fired when the session of a user ends.

Application\_End() – fired when the web application ends.

Application\_Disposed() - fired when the web application is destroyed.

**Show me an example!!**

Let us see an example of how to use the Global.asax to catch unhandled errors that occur at the application level.

To catch unhandled errors, do the following. Add a Global.asax file (Right click project > Add New Item > Global.asax). In the Application\_Error() method, add the following code:

 VB.NET

Sub Application\_Error(ByVal sender As Object, ByVal e As EventArgs)

        ' Code that runs when an unhandled error occurs

        Dim objErr As Exception = Server.GetLastError().GetBaseException()

        Dim err As String = "Error in: " & Request.Url.ToString() & ". Error Message:" & objErr.Message.ToString()

  End Sub

Here we make use of the Application\_Error() method to capture the error using the Server.GetLastError().