.NET Framework Developer's Guide

**Global.asax File**

The Global.asax file, also known as the ASP.NET application file, is an optional file that contains code for responding to application-level events raised by ASP.NET or by HttpModules. The Global.asax file resides in the root directory of an ASP.NET-based application. At run time, Global.asax is parsed and compiled into a dynamically generated .NET Framework class derived from the [HttpApplication](http://msdn.microsoft.com/en-us/library/system.web.httpapplication(VS.71).aspx)base class. The Global.asax file itself is configured so that any direct URL request for it is automatically rejected; external users cannot download or view the code written within it.

The ASP.NET Global.asax file can coexist with the ASP Global.asax file. You can create a Global.asax file either in a WYSIWYG designer, in Notepad, or as a compiled class that you deploy in your application's \Bin directory as an assembly. However, in the latter case, you still need a Global.asax file that refers to the assembly.

The Global.asax file is optional. If you do not define the file, the ASP.NET page framework assumes that you have not defined any application or session event handlers.

When you save changes to an active Global.asax file, the ASP.NET page framework detects that the file has been changed. It completes all current requests for the application, sends the [Application\_OnEnd](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.endrequest(VS.71).aspx) event to any listeners, and restarts the application domain. In effect, this reboots the application, closing all browser sessions and flushing all state information. When the next incoming request from a browser arrives, the ASP.NET page framework reparses and recompiles the Global.asax file and raises the [Application\_OnStart](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.beginrequest(VS.71).aspx) event.

For detailed information about using the Global.asax file with your applications, see the following topics:

[Working with HttpApplication Instances](http://msdn.microsoft.com/en-us/library/a0xez8f2(VS.71).aspx)

[Using Modules with the Global.asax File](http://msdn.microsoft.com/en-us/library/szzd570s(VS.71).aspx)

.NET Framework Developer's Guide

**Working with HttpApplication Instances**

During the lifetime of your application, ASP.NET maintains a pool of Global.asax-derived [HttpApplication](http://msdn.microsoft.com/en-us/library/system.web.httpapplication(VS.71).aspx) instances. When your application receives an HTTP request, the ASP.NET page framework assigns one of these instances to process that request. That particular **HttpApplication** instance is responsible for managing the entire lifetime of the request it is assigned to, and the instance can only be reused after the request has been completed.

It is through the **HttpApplication** class that you have programmatic access to the [HttpApplication.Init](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.init(VS.71).aspx) and [HttpApplication.Dispose](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.dispose(VS.71).aspx) methods, as well as the [Application\_OnStart](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.beginrequest(VS.71).aspx) and [Application\_OnEnd](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.endrequest(VS.71).aspx) events. You also have access to any events exposed by any HttpModule.

**Overriding the Init and Dispose Methods**

You invoke the [HttpApplication.Init](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.init(VS.71).aspx) method immediately after you create an instance of the [HttpApplication](http://msdn.microsoft.com/en-us/library/system.web.httpapplication(VS.71).aspx) class. You can use this method to create and configure any object that you want to use across all handling events, as shown in the following example.

[Visual Basic]

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl08_code');" \o "Copy Code)

<script language="VB" runat="server">

Public Overrides Sub Init()

'Insert something interesting here.

End Sub

</script>

[C#]

<script language="C#" runat="server">

public override void Init() {

// Insert something interesting here.

}

</script>

The **Init** method is different from the **Application\_OnStart** event, because it is always invoked on all **HttpApplication** instances within an application. **Application\_OnStart** is raised only once during an application's lifetime, on the first instance of **HttpApplication**. Use **Application\_OnStart** only to create or modify states shared by all pipeline instances, such as the use of the **ApplicationState** object. Do not use it to create local variables, because local variables are not shared by multiple **HttpApplication** instances. Invoke the [HttpApplication.Dispose](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.dispose(VS.71).aspx) method immediately before destroying an instance of the **HttpApplication** class. You can use it to clean up any local resources.

The **Dispose** method is different from the **Application\_OnEnd** event because it is always invoked on all **HttpApplication** instances within an application. **Application\_OnEnd** is raised only once during an application's lifetime, on the last instance of **HttpApplication** that is torn down. Use **Application\_OnEnd** only to clean up states or resources shared by all pipeline instances, such as the use of the **ApplicationState** object. Do not use it to clean up local variables, because local variables are not shared by multiple **HttpApplication** instances.

You cannot use the [Request](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.request(VS.71).aspx), [Response](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.response(VS.71).aspx), and [Session](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.session(VS.71).aspx) properties of an **HttpApplication** instance during either the **Init** or the **Dispose** stage of request execution.

The following example demonstrates how you can override the two life cycle methods provided by the **HttpApplication** base class.

[Visual Basic]

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl13_code');" \o "Copy Code)

<script language="VB" runat=server>

Public Overrides Sub Init()

' Init override code goes here.

End Sub

Public Overrides Sub Dispose()

' Init override code goes here.

End Sub

</script>

[C#]

<script language="C#" runat=server>

public override void Init() {

// Init override code goes here.

}

public override void Dispose() {

// Init override code goes here.

}

</script>

**Handling HttpApplication Events**

You can use the Global.asax file to synchronize any event that is exposed by the [HttpApplication](http://msdn.microsoft.com/en-us/library/system.web.httpapplication(VS.71).aspx) base class. To do this, you must use the following naming pattern to author methods:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl15_code');" \o "Copy Code)

Application\_EventName(AppropriateEventArgumentSignature)

For example, if you want code that responds to the [OnStart](http://msdn.microsoft.com/en-us/library/system.web.sessionstate.sessionstatemodule.start(VS.71).aspx), [BeginRequest](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.beginrequest(VS.71).aspx), and [OnEnd](http://msdn.microsoft.com/en-us/library/system.web.sessionstate.sessionstatemodule.end(VS.71).aspx) events of your application, the code you include in the Global.asax file might look something like the following.

[Visual Basic]

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl19_code');" \o "Copy Code)

<Script language="VB" runat="server">

Sub Application\_OnStart()

'Application start-up code goes here.

End Sub

Sub Application\_BeginRequest()

'Application code for each request could go here.

End Sub

Sub Application\_OnEnd()

'Application clean-up code goes here.

End Sub

</script>

[C#]

<Script language="C#" runat="server">

public void Application\_OnStart() {

// Application start-up code goes here.

}

public void Application\_BeginRequest() {

// Application code for each request could go here.

}

public void Application\_OnEnd() {

// Application clean-up code goes here.

}

</script>

.NET Framework Developer's Guide

**Using Modules with the Global.asax File**

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. For example, you could create an output cache module that implements output-caching behaviors for your entire application.

All modules, whether custom or provided by the .NET Framework, must implement the [IHttpModule](http://msdn.microsoft.com/en-us/library/system.web.ihttpmodule(VS.71).aspx) interface. As long as these modules are registered with your application, you can easily interact with the HTTP requests coming in to your application.

**Handling HttpModule Events**

You can use the Global.asax file to handle any event exposed by the modules in the request. For example, you might create a custom authentication module for your ASP.NET Web application in which you might expose an [OnAuthenticateRequest](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.authenticaterequest(VS.71).aspx) event. The code that you write to handle the events exposed by an HttpModule must conform to the following naming pattern:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl03_code');" \o "Copy Code)

FriendlyModuleName\_EventName(AppropriateEventArgumentSignature)

For example, if you want to include event-handling code for the beginning and end of a session, as well as for an **OnAuthenticateRequest** event, it could look like the following.

[Visual Basic]

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl04_code');" \o "Copy Code)

<Script language="VB" runat="server">

Sub Session\_OnStart()

'Session start-up code goes here.

End Sub

Sub Session\_OnEnd()

'Session clean-up code goes here.

End Sub

Sub Application\_OnAuthenticateRequest(Source As Object, Details as EventArgs)

'Authentication code goes here.

End Sub

</script>

[C#]

<Script language="C#" runat="server">

void Session\_OnStart() {

// Session start-up code goes here.

}

void Session\_OnEnd() {

// Session clean-up code goes here.

}

void Application\_OnAuthenticateRequest(Object Source, EventArgs Details) {

// Authentication code goes here.

}

</script>