Walkthrough: Creating and Registering a Custom HTTP Module

Other Versions

This walkthrough illustrates the basic functionality of a custom HTTP module. An HTTP module is called on every request in response to the BeginRequest andEndRequest events. As a result, the module runs before and after a request is processed.

If the ASP.NET application is running under IIS 6.0, you can use HTTP modules to customize requests for resources that are serviced by ASP.NET. This includes ASP.NET Web pages (.aspx files), Web services (.asmx files), ASP.NET handlers (.ashx files), and any file types that you have mapped to ASP.NET. If the ASP.NET application is running under IIS 7.0, you can use HTTP modules to customize requests for any resources that are served by IIS. This includes not just ASP.NET resources, but HTML files (.htm or .html files), graphics files, and so on. For more information, seeASP.NET Application Life Cycle Overview for IIS 5.0 and 6.0 and ASP.NET Application Life Cycle Overview for IIS 7.0.

The example module in this topic adds a message to the requested ASP.NET Web page at the beginning of any HTTP request. It adds another message after the page has been processed. The module includes code that makes sure that it does not add text to a request for any other file type.

Each event handler is written as a private method of the module. When the registered events are raised, ASP.NET calls the appropriate handler in the module, which writes information to the ASP.NET Web page.

Tasks illustrated in this walkthrough include the following:

- How to create the code for an HTTP module.
- How to register the module in the Web.config file.

Prequisites

In order to complete this walkthrough, you will need:

- Visual Studio or Visual Web Developer.

The walkthrough also assumes that you are working with IIS 6.0 or IIS 7.0. However, you can see the functionality of the module even if you run the ASP.NET Development Server.

Creating a Custom HTTP Module Class

To begin, you will create a class file that implements the module.

To create a custom HTTP module class

1. Create an ASP.NET Web site and name it Handler.

> **Note**
>
> You can choose any name for the Web site.

2. If the Web site does not already have an App_Code folder, create one under the root of the site.

3. In the App_Code directory, create a class file named HelloWorldModule.vb (for Visual Basic) or HelloWorldModule.cs (for C#).

> **Note**
>
> Alternatively, if you are using Visual Studio (not Visual Web Developer Express), you can create`HelloWorldModule` as a class library project, compile it, and put the resulting assembly in the Web application's Bin directory.

4. Add the following code to the class file:

C# | **VB**

```
using System;
using System.Web;
public class HelloWorldModule : IHttpModule
{
    public HelloWorldModule()
    {
    }

    public String ModuleName
    {
        get { return "HelloWorldModule"; }
    }

    // In the Init function, register for HttpApplication
    // events by adding your handlers.
    public void Init(HttpApplication application)
    {
        application.BeginRequest +=
            (new EventHandler(this.Application_BeginRequest));
        application.EndRequest +=
            (new EventHandler(this.Application_EndRequest));
    }

    private void Application_BeginRequest(Object source,
         EventArgs e)
    {
    // Create HttpApplication and HttpContext objects to access
    // request and response properties.
        HttpApplication application = (HttpApplication)source;
        HttpContext context = application.Context;
        string filePath = context.Request.FilePath;
        string fileExtension =
            VirtualPathUtility.GetExtension(filePath);
        if (fileExtension.Equals(".aspx"))
        {
```

```
            context.Response.Write("<h1><font color=red>" +
                "HelloWorldModule: Beginning of Request" +
                "</font></h1><hr>");
        }
    }

    private void Application_EndRequest(Object source, EventArgs e)
    {
        HttpApplication application = (HttpApplication)source;
        HttpContext context = application.Context;
        string filePath = context.Request.FilePath;
        string fileExtension =
            VirtualPathUtility.GetExtension(filePath);
        if (fileExtension.Equals(".aspx"))
        {
            context.Response.Write("<hr><h1><font color=red>" +
                "HelloWorldModule: End of Request</font></h1>");
        }
    }

    public void Dispose() { }
}
```

5. Save and close the class file.

6. In the **Build** menu, click **Build Web Site**.

   If the Web site does not build, correct any problems. The custom HTTP module must compile or you will not be able to register the module.

Registering the HTTP Module in IIS 6.0 and IIS 7.0 Classic Mode

After you have created the `HelloWorldModule`class, you register the module by creating an entry in the Web.config file. Registering the HTTP module enables it to subscribe to request-pipeline notifications.

In IIS 7.0, an application can run in either Classic or Integrated mode. In Classic mode, requests are processed basically the same as they are in IIS 6.0. In Integrated mode, IIS 7.0 manages requests by using a pipeline that enables it to share requests, modules, and other features with ASP.NET.

The procedure for registering a module is different in IIS 7.0 Classic mode and IIS 7.0 Integrated mode. This section describes the procedure for IIS 6.0 and IIS 7.0 Classic mode. The procedure for registering a module that is running in IIS 7.0 Integrated mode is described in the next section.

To register the module for IIS 6.0 and IIS 7.0 running in Classic mode

1. If the Web site does not already have a Web.config file, create one under the root of the site.

2. Add the following highlighted code to the Web.config file:

```
<configuration>
  <system.web>
<httpModules><add name="HelloWorldModule" type="HelloWorldModule"/></httpModules>
  </system.web>
</configuration>
```

The code registers the module with the class name and the module name of`HelloWorldModule`.

Registering the HTTP Module in IIS 7.0 Integrated Mode

The process for registering a module in IIS 7.0 Integrated mode is slightly different than the process for IIS 7.0 Classic mode.

To register the module for IIS 7.0 running in Integrated mode

1. If the Web site does not already have a Web.config file, create one under the root of the site.
2. Add the following highlighted code to the Web.config file:

```
<configuration>
<system.webServer><modules><add name="HelloWorldModule" type="HelloWorldModule"/></modu
</configuration>
```

Note

You can also register the module by using IIS Manager. For more information, see Configuring Modules in IIS 7.0.

The code registers the module with the class name and the module name of`HelloWorldModule`.

Testing the Custom HTTP Module

After you have created and registered your custom HTTP module, you can test it.

To test the custom HTTP module

1. Add a new ASP.NET page in the application.
2. Right click the page you just added and select **View in Browser**.

   The HTTP module appends a string to the start and end of the response. The module automatically runs during any request for a file whose extension is assigned to ASP.NET. For more information, see HTTP Handlers and HTTP Modules Overview.