INFO: ASP.NET HTTP Modules and HTTP Handlers Overview

Summary

This article provides an introduction to the ASP.NET HTTP modules and HTTP handlers.

For additional ASP.NET overviews, refer to the following Microsoft Knowledge Base article:
305140 INFO: ASP.NET Roadmap

More Information

HTTP modules and HTTP handlers are an integral part of the ASP.NET architecture. While a request is being processed, each request is processed by multiple HTTP modules (for example, the authentication module and the session module) and is then processed by a single HTTP handler. After the handler has processed the request, the request flows back through the HTTP modules.

This article is divided into the following sections:
HTTP Modules
Available Events
Configuring HTTP Modules
Creating HTTP Modules
HTTP Handlers
Configuring HTTP Handlers
Creating HTTP Handlers

HTTP Modules

Modules are called before and after the handler executes. Modules enable developers to intercept, participate in, or modify each individual request. Modules implement the IHttpModule interface, which is located in the System.Web namespace.

Available Events

An HttpApplication class provides a number of events with which modules can synchronize. The following events are available for modules to synchronize with on each request. These events are listed in sequential order:
BeginRequest: Request has been started. If you need to do something at the beginning of a request (for example, display advertisement banners at the top of each page), synchronize this event.
AuthenticateRequest: If you want to plug in your own custom authentication scheme (for example, look up a user against a database to validate the password), build a module that synchronizes this event and authenticates the user how you want to.
AuthorizeRequest: This event is used internally to implement authorization mechanisms (for example, to store your access control lists (ACLs) in a database rather than in the file system). Although you can override this event, there are not many good reasons to do so.

ResolveRequestCache: This event determines if a page can be served from the Output cache. If you want to write your own caching module (for example, build a file-based cache rather than a memory cache), synchronize this event to determine whether to serve the page from the cache.
AcquireRequestState: Session state is retrieved from the state store. If you want to build your own state management module, synchronize this event to grab the Session state from your state store.
PreRequestHandlerExecute: This event occurs before the HTTP handler is executed.
PostRequestHandlerExecute: This event occurs after the HTTP handler is executed.
ReleaseRequestState: Session state is stored back in the state store. If you are building a custom session state module, you must store your state back in your state store.
UpdateRequestCache: This event writes output back to the Output cache. If you are building a custom cache module, you write the output back to your cache.
EndRequest: Request has been completed. You may want to build a debugging module that gathers information throughout the request and then writes the information to the page.
The following events are available for modules to synchronize with for each request transmission. The order of these events is non-deterministic.
PreSendRequestHeaders: This event occurs before the headers are sent. If you want to add additional headers, you can synchronize this event from a custom module.
PreSendRequestContent: This event occurs when the Response.Flush method is called. If you want to add additional content, you can synchronize this event from a custom module.
Error: This event occurs when an unhandled exception occurs. If you want to write a custom error handler module, synchronize this event.

Configuring HTTP Modules

The <httpModules> configuration section handler is responsible for configuring the HTTP modules within an application. It can be declared at the computer, site, or application level. Use the following syntax for the <httpModules> section handler:

```
<httpModules>
   <add type="[COM+ Class], [Assembly]" name="[ModuleName]" />
   <remove type="[COM+ Class], [Assembly]" name="[ModuleName]" />
   <clear />
</httpModules>
```

Creating HTTP Modules

To create an HTTP module, you must implement the IHttpModule interface. The IHttpModule interface has two methods with the following signatures:

```
void Init(HttpApplication);
void Dispose();
```

For additional information about creating HTTP modules, click the article numbers below to view the articles in the Microsoft Knowledge Base:
308000 HOW TO: Create an ASP.NET HTTP Module by Using Visual Basic .NET
307996 HOW TO: Create an ASP.NET HTTP Module by Using Visual C# .NET

HTTP Handlers

Handlers are used to process individual endpoint requests. Handlers enable the ASP.NET framework to process individual HTTP URLs or groups of URL extensions within an application. Unlike modules, only one handler is used to process a request. All handlers implement the IHttpHandler interface, which is located in the System.Web namespace. Handlers are somewhat analogous to Internet Server Application Programming Interface (ISAPI) extensions.

Configuring HTTP Handlers

The <httpHandlers> configuration section handler is responsible for mapping incoming URLs to the IHttpHandler or IHttpHandlerFactory class. It can be declared at the computer, site, or application level. Subdirectories inherit these settings.

Administrators use the <add> tag directive to configure the <httpHandlers> section. <Add> directives are interpreted and processed in a top-down sequential order. Use the following syntax for the <httpHandler> section handler:

```
<httpHandlers>
   <add verb="[verb list]" path="[path/wildcard]" type="[COM+ Class], [Assembly]" val
   <remove verb="[verb list]" path="[path/wildcard]" />
   <clear />
</httpHandlers>
```

Creating HTTP Handlers

To create an HTTP handler, you must implement the IHttpHandler interface. The IHttpHandler interface has one method and one property with the following signatures:

```
void ProcessRequest(HttpContext);
bool IsReusable {get;}
```

NOTE: If session state is required in your HTTP handler, you also need to implement the IRequiresSessionState interface.
For additional information about creating HTTP handlers, click the article numbers below to view the articles in the Microsoft Knowledge Base:

308001 HOW TO: Create an ASP.NET HTTP Handler by Using Visual C# .NET
For additional information, click the article numbers below to view the articles in the Microsoft Knowledge Base:
307997 HOW TO: Create an ASP.NET HTTP Handler by Using Visual Basic .NET

Last Updated: Jun 10, 2019