

# Extending ASP.NET

Brian Noyes

Principal Software Architect

IDesign, Inc. ([www.idesign.net](http://www.idesign.net))



*Ride the wave*



## About Brian

- Principal Software Architect, IDesign Inc. ([www.idesign.net](http://www.idesign.net))
- Microsoft MVP in ASP.NET
- Writing
  - MSDN Magazine, asp.netPRO, Visual Studio Magazine, .NET Developer's Journal
  - Building Windows Forms Data Applications with .NET 2.0, Addison-Wesley, expected release spring 2005
- Speaking
  - Microsoft TechEd, Visual Studio Connections, DevEssentials, VSLive!, INETA Speakers Bureau
- Participate in Microsoft design reviews
- E-mail: [brian.noyes@idesign.net](mailto:brian.noyes@idesign.net)
- Blog: <http://www.softinsight.com/bnoyes>



*Ride the wave*

# Agenda

- ASP.NET Extensibility Overview
- Request Processing Pipeline
- Configuring ASP.NET Extensibility
- Handlers
- Applications
- Modules
- Advanced Handler Topics



*Ride the wave*

# ASP.NET Extensibility Overview

## Extension Points

- Application
  - Global.asax, code-behind
- Module
  - Custom classes
- Handler
  - Pages, custom classes, asynchronous handlers, handler factories
- ASP.NET 2.0
  - Custom providers



*Ride the wave*

# Agenda

- ASP.NET Extensibility Overview
- Request Processing Pipeline
- Configuring ASP.NET Extensibility
- Handlers
- Applications
- Modules
- Advanced Handler Topics



*Ride the wave*

# Request Processing Pipeline Overview

- IIS and ISAPI De-emphasized
- ASP.NET Manages:
  - Worker processes
  - AppDomains
  - Threads
  - Recycling
  - Application, Module, Handler instances



*Ride the wave*

# Request Processing Pipeline

## Servicing Requests

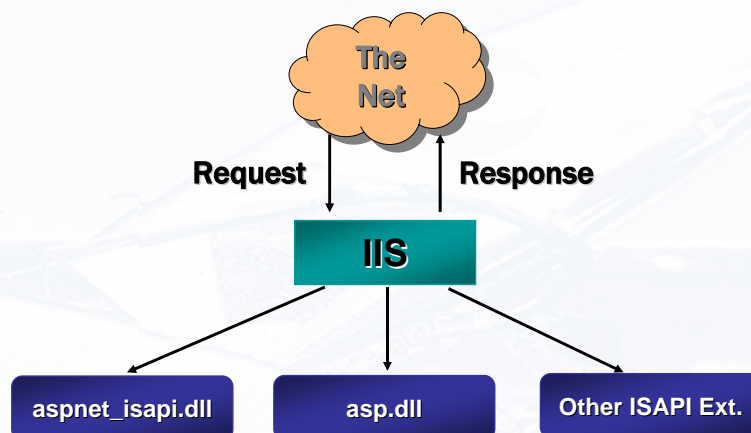
- Ultimate goal of a web application
  - Process request
  - Return response
- Internet Information Server (IIS) front end
  - Usually
  - Could be Cassini, Apache, etc.



*Ride the wave*

# Request Processing Pipeline

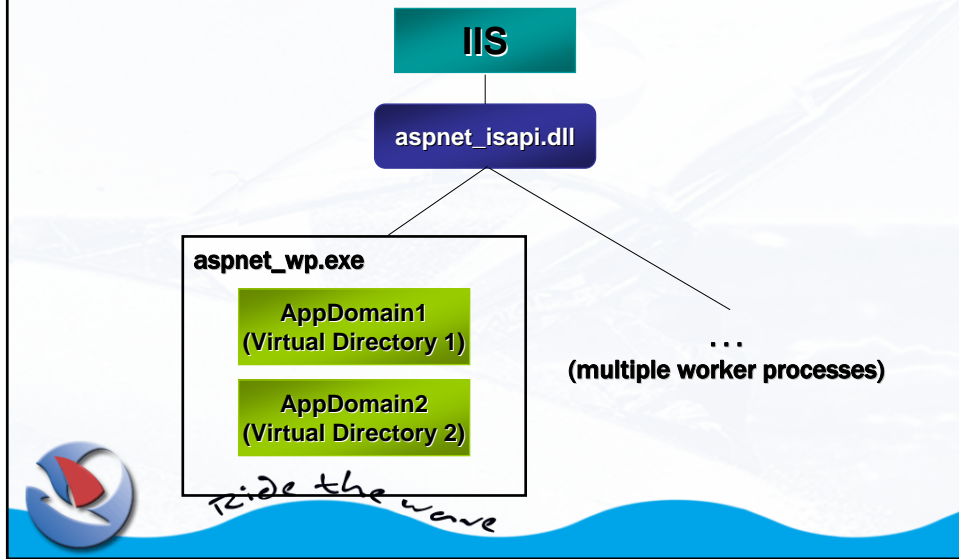
## IIS Processing



*Ride the wave*

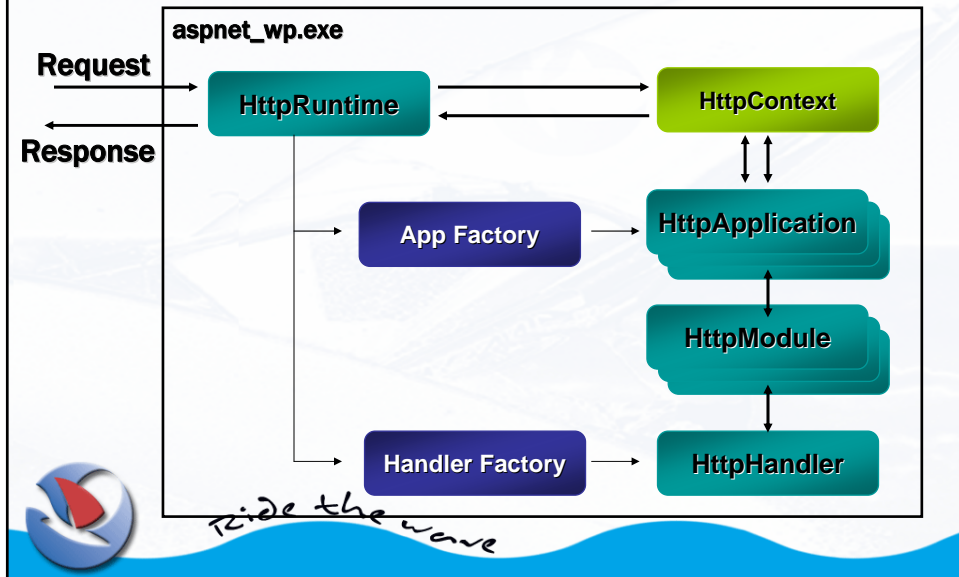
# Request Processing Pipeline

## ASP.NET Forwarding



# Request Processing Pipeline

## Runtime processing



# Request Processing Pipeline

## HttpContext

- Wraps HTTP request context
  - All associated state
- Members:
  - Request
  - Session
  - Items
  - Error
  - User
  - Response
  - Application
  - Server
  - AllErrors



*Ride the wave*

## Agenda

- ASP.NET Extensibility Overview
- Request Processing Pipeline
- **Configuring ASP.NET Extensibility**
- Handlers
- Applications
- Modules
- Advanced Handler Topics



*Ride the wave*

# Configuring ASP.NET Extensibility

## Configuration Files

**web.config****Application sub-folder****web.config****Virtual Directory****web.config****Site****machine.config****Machine***Ride the wave*

# Configuring ASP.NET Extensibility

## system.web elements

- ProcessModel
  - Configure worker process
  - machine.config only
- httpHandlers
  - Attach handlers to request and file type
  - All config levels
- httpModules
  - Attach modules for filtering
  - Machine, site, virtual directory levels

*Ride the wave*

# Agenda

- ASP.NET Extensibility Overview
- Request Processing Pipeline
- Configuring ASP.NET Extensibility
- **Handlers**
- Applications
- Modules
- Advanced Handler Topics



*Ride the wave*

# Handlers

## Overview

- Request target or endpoint
- Process request, return response
- Attached through config files
- Implements IHttpHandler interface



*Ride the wave*

# Handlers

## Built-in handlers

- PageHandlerFactory (\*.aspx)
- SimpleHandlerFactory (\*.ashx)
- WebServiceHandlerFactory (\*.asmx)
- HttpRemotingHandlerFactory (\*.rem, \*.soap)
- HttpForbiddenHandler (\*.cs, \*config, etc.)
- StaticFileHandler (\* GET,HEAD)
- HttpMethodNotAllowedHandler (\*)



*Ride the wave*

# Handlers

## Custom Handler Examples

- Dynamic image generation
- Dynamic content generation
  - HTML, XML/XSLT, etc.
- Distributed computational process invocation
- Sound file alerts



*Ride the wave*

# Handlers

## Custom Handler Requirements

- Implement IHttpHandler
- Compile assembly and place in bin folder
- Configure file extension in web.config
- Configure file extension in IIS



*Ride the wave*

# Custom Handlers

## Implement IHttpHandler

```
public class Simple Handler : IHttpHandler
{
    public void ProcessRequest(HttpContext context) { }

    public bool IsReusable { get { } }
}
```



*Ride the wave*

# Custom Handlers

## Configuring Handlers

- `<httpHandlers>` section
  - `<add>`
    - verb (GET,POST,HEAD,PUT,\*)
    - path (file spec)
    - type (namespace.type,assembly)
  - `<remove>`
    - verb
    - path
  - `<clear>`



*Ride the wave*

# Custom Handlers

## Configuring Handlers

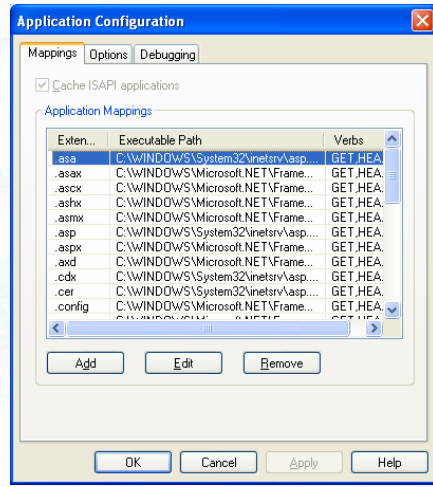
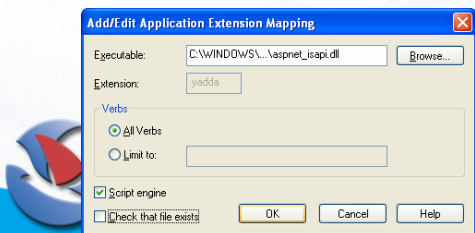
```
<system.web>  
  <httpHandlers>  
    <add verb="GET"  
      path="*.myext"  
      type="CustomHandlerLib1.CustomHandler,  
      CustomHandlerLib1" />  
  </httpHandlers>  
</system.web>
```



*Ride the wave*

# Custom Handlers Configuring IIS

- Virtual Directory Properties
  - Directory Tab
    - Application Settings – Configuration Button
      - Application Mappings



## Demo

# Custom Handlers

Brian Noyes  
 IDesign, Inc.  
[www.idesign.net](http://www.idesign.net)



*Ride the wave*

# Custom Handlers

## SimpleHandlers

- Created by the SimpleHandlerFactory
- .ashx file containing:
  - Handler class definition
  - WebHandler directive
- Compiled and run first time it is called



*Ride the wave*

# Custom Handlers

## SimpleHandlers

- Advantages:
  - No IIS config required
  - No entries in .NET config files required
  - xcopy deployment
- Disadvantages
  - One time compilation penalty
  - Errors not detected until runtime
  - No IntelliSense or color coding



*Ride the wave*

# Demo

## Simple Handlers

Brian Noyes  
IDesign, Inc.

[www.idesign.net](http://www.idesign.net)



*Ride the wave*

## Custom Handlers

### Best practices

- Indicate reusability correctly
- Store state where it belongs
- Process requests quickly
- Prefer custom handlers for invisible processing
- Prefer compiled handlers over simple handlers



*Ride the wave*

# Agenda

- ASP.NET Extensibility Overview
- Request Processing Pipeline
- Configuring ASP.NET Extensibility
- Handlers
- Applications
- Modules
- Advanced Handler Topics



*Ride the wave*

# Custom Applications

## Overview

- Declared through global.asax file
- Class derived from `HttpApplication`
- Code behind or script block
- Wire up event handlers
  - Explicit or implicit
- Any additional methods, properties, fields, events
- Multiple instances used to process requests



*Ride the wave*

# Custom Applications

## Lifecycle Events

- BeginRequest
- AuthenticateRequest
- AuthorizeRequest → User Property valid
- ResolveRequestCache
- AcquireRequestState → Handler Created
- PreRequestHandlerExecute
- PostRequestHandlerExecute → ProcessRequest() Called
- ReleaseRequestState
- UpdateRequestCache
- EndRequest



*Ride the wave*

# Custom Applications

## Unordered Events

- Disposed
- Error
- PreSendRequestContent
- PreSendRequestHeaders



*Ride the wave*

# Custom Applications

## Application Scope Events

- Application\_Start
- Application\_End
- Session\_Start
- Session\_End



*Ride the wave*

# Demo

## Simple Custom Application

Brian Noyes  
IDesign, Inc.

[www.idesign.net](http://www.idesign.net)



*Ride the wave*

# Agenda

- ASP.NET Extensibility Overview
- Request Processing Pipeline
- Configuring ASP.NET Extensibility
- Handlers
- Applications
- **Modules**
- Advanced Handler Topics



*Ride the wave*

# HTTP Modules

## Overview

- Request processing filters
- Plug into pipeline between application and handler
- Added through config file entries
- Pre-process requests coming in
- Post-process requests going out
- Handle application events



*Ride the wave*

# HTTP Modules

## Built-in Modules

- OutputCacheModule
- SessionStateModule
- WindowsAuthenticationModule
- FormsAuthenticationModule
- PassportAuthenticationModule
- UrlAuthorizationModule
- FileAuthorizationModule



*Ride the wave*

# HTTP Modules

## Module Examples

- Custom security
- Response Stream filtering
- User tracking and logging
- Custom caching/state maintenance
- Front end redirect and URL parsing



*Ride the wave*

# Custom Modules

## Implementation Requirements

- Class that implements IHttpModule
- Handle events of interest
- Compile and place in bin folder
- Configure in machine.config or web.config



*Ride the wave*

# Custom Modules

## Implement IHttpModule

```
public class Simple Handler : IHttpModule
{
    public void Init(HttpApplication app)
    {
        // Subscribe to events of interest
    }

    public void Dispose() { // clean up }
}
```



*Ride the wave*

# Custom Handlers

## Configuring Handlers

- `<httpModules>` section
  - `<add>`
    - name
    - type (namespace.type,assembly)
  - `<remove>`
    - name
  - `<clear>`



*Ride the wave*

# Custom Handlers

## Configuring Handlers

```
<httpModules>
  <remove name="FormsAuthentication"/>
  <add name="MyFormsAuthentication"
    type="MyModules.MyFormsAuthenticationModule, Module1"/>
</httpModules>
```



*Ride the wave*

# Demo

## Custom Modules

Brian Noyes  
IDesign, Inc.

[www.idesign.net](http://www.idesign.net)



*Ride the wave*

## Custom Modules

### Gotchas

- Know the request lifecycle
- Be careful with redirects
- Stubbing Event Handling
- Module processing order
- Don't be greedy



*Ride the wave*

# Agenda

- ASP.NET Extensibility Overview
- Request Processing Pipeline
- Configuring ASP.NET Extensibility
- Handlers
- Applications
- Modules
- **Advanced Handler Topics**



*Ride the wave*

# Advanced Handler Topics

## Asynchronous Handlers

- Perform processing on a self-managed thread
  - Don't tie up threads from the thread pool
- Class that implements IHttpAsyncHandler
- Uses standard .NET Async pattern
  - BeginProcessRequest
  - EndProcessRequest
  - Register a callback method



*Ride the wave*

# Asynchronous Handlers

- Normal request handling performed on thread from process-wide thread pool
  - 25 threads by default
  - configurable through `<processModel>` in `machine.config`
- Asynchronous handlers allow you to perform lengthy processing on a separate (self-managed) thread



*Ride the wave*

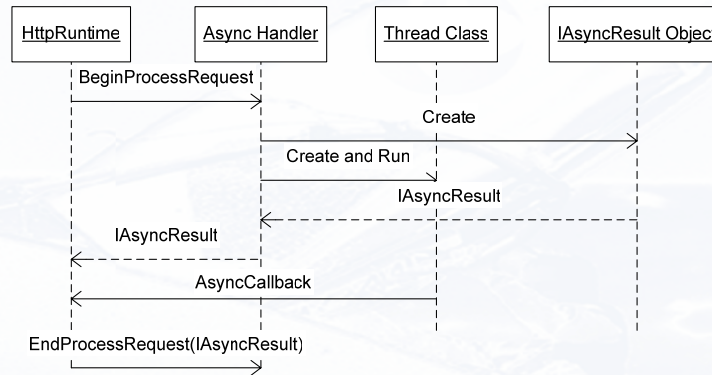
# Asynchronous Handlers

- Requirements:
  - Same as normal handlers, except implement `IHttpAsyncHandler` (derives from `IHttpHandler`)
- Basics:
  - `ProcessRequest()` will never be called
  - Spin your own worker thread in `BeginProcessRequest()` and use passed in context object
    - Optionally can pass a state object
    - Optionally create an object derived from `IAsyncResult` and return
  - Notify ASP.NET when processing is complete through `AsyncCallback`
  - Clean up resources when `EndProcessRequest()` is called if necessary



*Ride the wave*

# Async Handler Processing



Ride the wave

# Implement IHttpAsyncHandler

```

public class MyAsyncHandler : IHttpAsyncHandler
{
    public System.IAsyncResult BeginProcessRequest(System.Web.HttpContext context,
        System.AsyncCallback cb, object extraData)
    {
        // Spin worker thread - signal ASP.NET on cb object when done
        // Optionally return an IAsyncResult object
    }

    public void EndProcessRequest(System.IAsyncResult result)
    {
        // Clean up
    }

    public void ProcessRequest(System.Web.HttpContext context)
    {
        // Do nothing
    }

    public bool IsReusable
    {
        get {return false;}
    }
}
  
```



Ride the wave

# Handler Factories

- Handler factories create handler instances for the application
- Can be used to perform custom handler pooling or for custom creation semantics
- Requirements
  - Same as for custom handlers, except:
    - Define class that implements IHttpHandlerFactory
    - Configure that as the handler in machine.config or web.config



*Ride the wave*

# Implement IHttpHandlerFactory

```
public class MyHandlerFactory : IHttpHandlerFactory
{
    public IHttpHandler GetHandler(HttpContext context, string requestType,
        string url, string pathTranslated)
    {
        // Construct handler and return it
    }
    public void ReleaseHandler(IHttpHandler handler)
    {
        // Destroy or pool handler based on IsReusable
    }
}
```



*Ride the wave*

## Session Summary

- Multiple extensibility points in ASP.NET
- Handlers for custom processing of requests
- Applications for custom lifecycle event handling
  - Non-portable across apps
- Modules for custom lifecycle event handling and request filtering
  - Portable across apps
- Async Handlers and Handlers Factories for advanced scenarios



*Ride the wave*

## Resources

“Extend ASP.NET With HTTP Modules”, asp.netPRO Feb 2003

“Handle Requests Asynchronously”, Brian Noyes,

DotNetDashboard,

<http://www.dotnetdashboard.com/DesktopDefault.aspx?tabindex=0&tabid=68>

“Securely Implement Request Processing, Filtering, and Content Redirection with HTTP Pipelines in ASP.NET”, Tim Ewald and Kieth Brown, MSDN Magazine Sep 2002

“Essential ASP.NET with Examples in C#” or “Essential ASP.NET with Examples in Visual Basic .NET”, Fritz Onion, Addison-Wesley (ISBN 0-201-76040-1 / 0-201-76039-8)

E-mail: [brian.noyes@idesign.net](mailto:brian.noyes@idesign.net)

Blog: <http://www.softinsight.com/bnoyes>



*Ride the wave*