
ASP+ Configuration and Deployment

Rob Howard
Microsoft Corporation

Agenda

- ASP+ application
- Global.asax
- Application configuration
- Deployment and versioning
- Tips and tricks
- Resources/questions

WFO

ASP+ Application

- Contains several file types
 - ASP+ pages, Web Service files, controls, assemblies, and any other application-related code (COM, etc.)
 - May contain a global application file (**global.asax**)
 - May contain a configuration file (**config.web**)
 - If **config.web** is not present, relies on default **config.web**
 - May contain custom assemblies

Global.asax Overview

- **global.asax** is the functional replacement for **global.asa**
- Only a single **global.asax** file may exist per application domain
 - Application defined in ASP+ host application
 - E.g. with IIS 5, it is an IIS virtual root or web root
- 12 new events for better application control

Global.asax Overview (cont'd)

- Parsed and dynamically compiled at runtime
 - IIS does not have to be stopped to update **global.asax**
- Configured so that any direct requests are rejected (**config.web** setting to be discussed)
- Generates no user interface
 - When ASP and ASP+ are run in parallel, applications are separate – no sharing of information/data between application models

Global.asax Events (1 of 4)

- **Application_Start/End**
 - Raised once when the application first starts
- **Session_Start/End**
 - Raised once when the session first starts
 - Raised for each session
- **Application_Error**
 - Raised when an unhandled error occurs
 - Errors handled in error trapping code will not raise this event
- **Application_AuthorizeRequest**
 - Raised when the request is ready to be authorized
 - Security modules sink this event

Global.asax Events (2 of 4)

- **Application_ResolveRequestCache**
 - Raised when the cache needs to be flushed
- **Application_BeginRequest**
 - Raised each time an application request is begun
- **Application_AuthenticateRequest**
 - Raised as part of the security system and signals that the request is ready to be authenticated
 - You can sink this event for your own security modules

Global.asax Events (3 of 4)

- **Application_AcquireRequestState**
 - Raised to notify that per-request state should be obtained
- **Application_PreRequestHandlerExecute**
 - Signals that the RequestHandler (such as page or service) is about to execute
- **Application_PostRequestHandlerExecute**
 - Signals that the RequestHandler has completed execution

Global.asax Events (4 of 4)

- **Application_ReleaseRequestState**
 - Raised to notify that per-request state should be stored
- **Application_UpdateRequestCache**
 - Raised when code processing is complete and the output can be added to the ASP+ file cache
- **Application_EndRequest**
 - Raised when the request has completed execution

Event Execution Order



- Supports `#include` syntax

```
<!--#Include file="[path]"-->
```

- Supports object tag declarations for global application objects

```
<object id="id" runat="server" progid | class |  
  classid="[value]" scope="[application | session]">
```

- **ProgID** or **ClassId**
 - COM / COM+ objects by ProgID or GUID
- **Class** - .NET classes in assemblies
- **Scope** - determines if object is application or session scoped

- Using `global.asax`

Configuration (pre-ASP+)

- Configuring ASP 3.0 (and previous versions) meant modifying the metabase – either:
 - Programmatically through ADSI (IIS://xxx)
 - `mkwebsrv.js` (IIS configuration script)

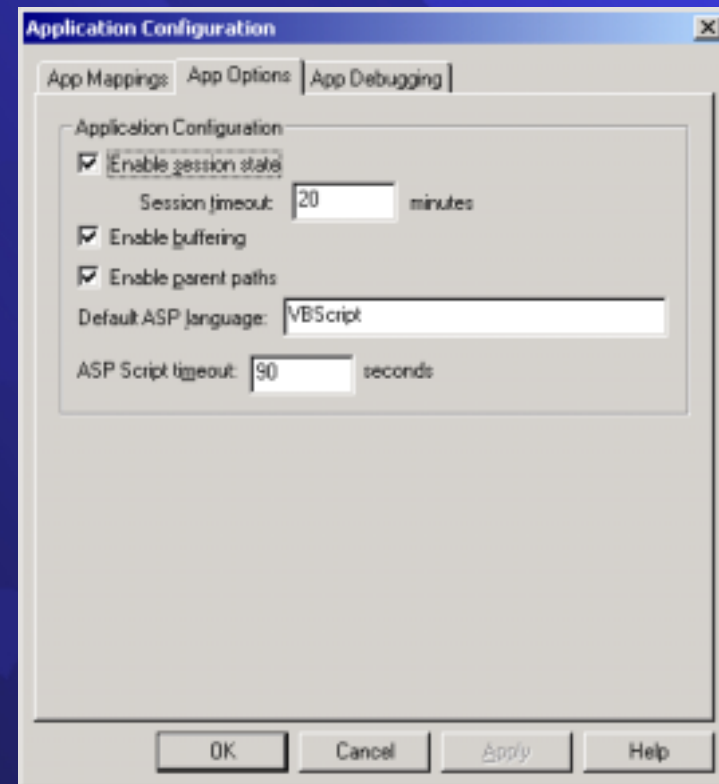
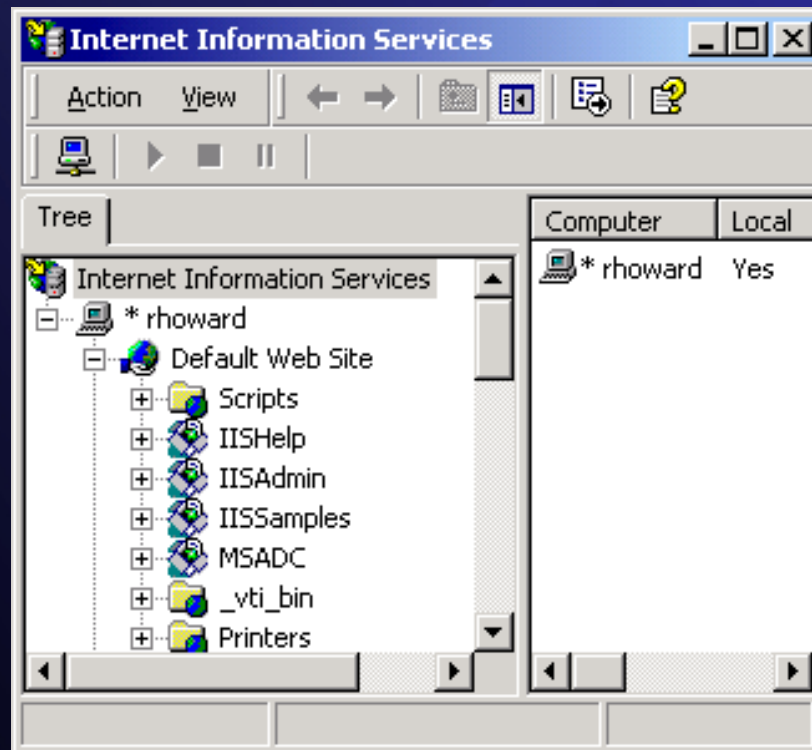
```
// First, create instance of Web service
ServiceObj = GetObject("IIS://Localhost/W3SVC");

// Second, create a new virtual server at the service
ServerObj = ServiceObj.Create("IISWebServer", WNumber);

...
```

Configuration (pre-ASP+)

- Or GUI based via Internet Services Manager



Configuration (pre-ASP+)

- Metabase is binary and location dependent
- Can be programmatically copied and/or updated
- Not human readable/writable
- Difficult to restore damaged metabase
- Black-box accessibility
- Difficult to replicate settings to multiple machines in a server farm
- Changes to metabase required server restarts
 - Application unavailable to clients
 - Loss of session state

ASP+ Configuration

- All configuration data is in an XML file
 - **config.web**
- Human readable format
 - Edit via notepad or XML based tools
- Multiple **config.web** files can be used to apply settings to a web application
 - Only 1 per directory
 - A union of the setting is computed
- System automatically detects and applies changes
 - No reboots
- Text based file can easily be replicated throughout the system via FTP, xcopy, DAV, etc.

Config.web (1 of 3)

- Extensible XML configuration file that supports an open schema that others can extend
- File syntax:

```
<configuration>  
  <configsections>  
    [config section handlers]  
  </configsections>  
  [config section settings]  
</configuration>
```

Config.web (2 of 3)

- Config section handlers: Identify .NET classes that interpret config section settings
- Config section settings: Specify settings for the .NET classes
- Together these two sections of **config.web** will be used to generate configuration settings for the web application

```
<configuration>
  <configsections>
    <add name="sessionstate"
      type="System.Web.SessionState.SessionStateModule$
      SectionHandler" />
  </configsections>
  <sessionstate inproc="true" usesqlserver="false"
    cookieless="false" timeout="20" server="localhost" port="42424" />
</configuration>
```

Config.web (3 of 3)

- Sessionstate **<configsections>** defines the .NET class to apply the settings found in the definition of **<sessionstate>**
- This configuration class is then applied to the application files in the application domain
 - this includes web services (**.asmx**) and web pages (**.aspx**)

WFO

Common Config Section Settings (1 of 3)

SessionState

- `Inproc = [true | false]`
 - Default setting is true
 - Indicates whether session state should be stored "in-process" or "out-of-process"
 - In-process: No process boundaries to cross, state limited to single machine
 - Out-of-process: Process boundaries, state available to all machines

WFO

Common Config Section Settings (2 of 3)

- **Cookieless = [true | false]**
 - Determines whether or not cookieless sessions is used
- **Timeout=[number of minutes]**
 - Determines how long a session will last with no activity

WFO

Common Config Section Settings (3 of 3)

AppSettings

- Special section that creates entries into a dictionary available in application files (pages and web services)
- Useful for storing data such as passwords and database settings

```
<configuration>
  <appsettings>
    <add key="DSN" value="Server=rhoward;
      Database=advworks; UID=AdvWorksUsr; PWD=4xdf89R"
    </appsettings>
  ...
```

- Retrieve the data

```
[String] = (String)
  ((Hashtable)Context.GetConfig("appsettings"))["DSN"];
```

-
- Examining the root `config.web`
 - Using cookieless sessions

wipro

Application Deployment Today

- Typical ASP-based web application includes
 - ASP code pages
 - COM / COM+ components (both system components and custom components)
- Deployment requires
 - Physical access to the server
 - Server is unavailable while the updates take place
 - Stop the web server
 - Update ASP pages
 - Register components
 - Restart the server

Application Deployment Today

- Component versioning - all components are global
 - Registry settings
 - ProgID naming conflicts
 - Interfaces are immutable
 - Although applications are independent, versions are not
- Bottom line
 - deployment and versioning is not easy

ASP+ Deployment

- Goal - no local server access
 - Enable deployment by simply copying bits to server
 - FTP, DAV, etc.
 - No admin or registration utilities (**regsvr32**)
 - No web server restarts
 - Support process model that can re-cycle
 - Works for page and services
 - Changes can be made and the server spins up a new process behind the scenes
- Configuration data is an XML text file
 - **Config.web**

ASP+ Deployment (1 of 2)

- `\bin` directory is where all assemblies are "registered" in ASP+
- Copy assemblies to `\bin` directory
 - Application marked via IIS
 - Each application has own `\bin` directory which you must create
 - Assemblies contain own metadata
 - No locked DLLs
 - No utilites (such as `regsvr32.exe`)

ASP+ Deployment (2 of 2)

- Web applications are isolated
 - Components can be private (no naming collisions)
 - Other applications can use different versions of the same component
 - One app uses ADO 2.5, another uses ADO 2.6, no collisions
- Uninstall means del *.*
 - No dangling registry entries
 - No registry clean up

Backwards Compatibility (1 of 2)

- COM / COM+ in managed code
 - **TlbImp.exe** – Creates a RCW (runtime callable wrapper) for an unmanaged object
 - Deploy wrapper to `\bin` directory and use as a normal .NET assembly
 - Downside:
 - Deployment still requires registration (**regsvr32.exe**) for the COM/COM+ component
 - Marshalling overhead

Backwards Compatibility (2 of 2)

- .NET assemblies in unmanaged code
 - **TlbExp.exe** – Creates a CCW (COM callable wrapper) for a managed class
 - Requires registration – **RegAsm.exe**
- Tools found in
 - [system]\program files\NGWSSDK\
– (this naming/location will change)

- Building an assembly
- Assembly deployment
- Interoperability

wro

Tips and Tricks

- Do not generate user interfaces within **global.asax**
 - No **Response.Write()**'s
- Turn off session state for pages that don't need it

```
<%@ Page SessionState=False %>
```

- Use AppSettings section of **config.web** to store application specific data
- Don't confuse **global.asax** and **config.web**
 - **global.asax** – application events and code
 - **config.web** – application settings

Tips and Tricks

- Application is cycled (but available) when application file updates are made
 - Cycling the application means application and session state is lost
 - Event in **global.asax** allows you to persist data before application is cycled
- Interoperability with unmanaged code
 - Static class methods are unavailable
- Consider porting frequently used COM / COM+ components to managed code
 - Eliminate marshalling overhead
 - XCOPY deployment advantage

Summary

- **global.asax** supports 12 new events that you can participate in
- ASP+ uses an XML based configuration file: **config.web**
- **config.web** is an extensible format
- Easy deployment
 - No registration
 - No starting and stopping services

Resources

- Wrox Press: *"A Preview of ASP+"*
 - Chapter 6 covers Configuration and Deployment
- Web sites
 - <http://msdn.microsoft.com>
 - <http://www.asptoday.com>
 - <http://www.develop.com/dm/default.asp>
 - <http://www.aspnet.com/aspnet/index.aspx>

Definitions

- Assembly
 - Unit of deployment in .NET, containing namespaces and classes
- Base Class Library (BCL)
 - Data, ASP+, XML classes, etc. are part of the BCL

WFO

Definitions

- Common Language Runtime (CLR)
 - Platform that provides a common type system and allows language integration
 - Managed code – code that is executed within the CLR
 - Un-managed code – native platform code
- Microsoft Interpreted Language (MSIL)
 - Instruction set that CLR understands

Questions?

Rob Howard
Microsoft Corporation