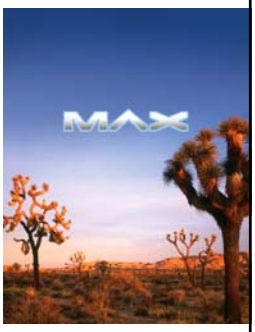
MAX 2006
Beyond Boundaries

Simon Horwith

ColdFusion MX7 Application Framework

Session WD204H-1

AboutWeb, LLC



1

Biography – Who am I?

Simon Horwith (shorwith@aboutweb.com)

- **Chief Information Officer, AboutWeb, LLC**
- **Editor-in-Chief, ColdFusion Developer's Journal**
- **Member of Team Macromedia**
- **Macromedia Certified Master Instructor**
- **Frequent Speaker at Conferences and User Groups**
- **Contributing Author of Several Books**
- **Developer, just like you!**

2

Audience – Who are you?

Target audience for this presentation:

- **CFMX 7 Developers interested in learning how and why to use the new Application framework**
- **Developers using versions of CF prior to CF MX 7 that:**
  - **will be migrating existing applications to MX 7 in the near future**
  - **are curious about how the application framework has changed in ColdFusion MX 7**

3

Overview and Objectives

In this session you will:

- Learn how to use the new application framework to define application settings and handle application events
- Learn the differences between the new application framework and the application framework that existed before CF MX 7
- Practice defining an application and handling application events in hands-on exercises
- Have Fun!!

4

Application Framework Overview

- Before ColdFusion MX 7
  - Application.cfm ran before each request
  - OnRequestEnd.cfm ran after each request
  - When a page is requested, ColdFusion:
    - looked in the local directory for Application.cfm
    - If no Application.cfm is found it traverses up the directory structure to the root drive, looking for one on each pass
    - The first time Application.cfm is found, it is processed before the page being requested and CF stops looking
    - If an Application.cfm was found, after the requested page runs, CF looks in the same directory as Application.cfm for a file named OnRequestEnd.cfm and, if found, runs it

5

Application Framework Overview cont'd

In ColdFusion MX 7

- ColdFusion still looks in the current directory and then traverses to the root, but now it first looks for Application.cfc
- If no Application.cfc is found, it looks in that directory for Application.cfm
- If Application.cfc is found, ColdFusion creates an instance of the component and uses it's data and methods
- If no Application.cfc is found but Application.cfm is found, ColdFusion behaves as it did in prior versions (including Application.cfm, then the requested page, then OnRequestEnd.cfm if one exists in that directory)

6

Defining Application Settings

- Prior to ColdFusion MX 7 application settings were defined with the <CFAPPLICATION> tag

```
<cfapplication
name="myApp"
applicationtimeout="#createTimeSpan(2,0,0,0)#"
sessionmanagement="true"
sessiontimeout="#createTimeSpan(0,0,20,0)#">
```


Defining Application Settings cont'd

- In CFMX 7 these settings retain the same names as the <CFAPPLICATION> attributes but are defined in the Application.cfc "this" scope outside of any methods

```
<cfcomponent>
    <cfset this.name = "myApp" />
    <cfset this.applicationtimeout = createTimeSpan(2,0,0,0) />
    <cfset this.sessionmanagement = true />
    <cfset this.sessiontimeout = createTimeSpan(0,0,20,0) />
</cfcomponent>
```


Application Settings Walkthrough

Objectives

After completing this lab, you should be able to:

- Browse the pre-existing sample and solution directories
- Create an Application.cfc component
- Define application settings in Application.cfc

Application Events

- Application Starts for the first time
- User sends first request – session starts
- Request begins
- Request is processed
- Request ends
- Eventually, the session expires
- Eventually, the application expires due to inactivity or a server restart
- During any of the above, errors can occur in pages!

Application Event Handlers

- A special function name is used for a method to handle each of the application events:
  - onApplicationStart()
  - onSessionStart()
  - onRequestStart()
  - onRequest()
  - onRequestEnd()
  - onSessionEnd()
  - onApplicationEnd()
  - onError()
- These methods are thread safe - locking is not necessary!

Application Events – onApplicationStart()

- Code that is run only when an application initializes is contained inside a function named "onApplicationStart"
- ReturnType: Boolean (optional if no returntype is specified in the function definition)
- Arguments: None
- Application doesn't start if false is returned or error thrown
- Popular Uses
  - Set global variables and cached queries in the application scope
  - Contact Administrator (Email/SMS)?

onApplicationStart() Code Sample

```
<cffunction name="onApplicationStart"
  returntype="boolean">
  <cfset var done = false />
  <!--- set some application scope variables --->
  <cfset application.dsn = "myDSN“ />
  <cfset application.dbUser = "myDBUser“ />
  <cfset application.dbPass = "myDBPass“ />
  <cfset done = true />
  <cfreturn done />
</cffunction>
```


Application Events – onApplicationEnd()

- Code that is run only when an application expires (due to activity timeout or elegant server shutdown) is contained inside a function named “onApplicationEnd”
- ReturnType: void
- Arguments: ApplicationScope (struct)
- This method cannot access session or request scopes and can only access application scope as an argument – server scope is accessible
- Popular Uses
  - Clean-up application data and/or processes
  - Contact Administrator (Email/SMS)?

onApplicationEnd() Code Sample

```
<cffunction name="onApplicationEnd" returntype="void">
  <cfargument name="appScope" type="struct" required="true“ />
  <cfset var logMsg = "Application shut down with the following
  keys in the application scope: " &
  structKeyList(arguments.appScope) />
  <cflog file="appEvents" text="#logMsg#" application="true“ />
  <cfmail to="webmaster@site.com" from="webmaster@site.com"
  subject="Application shut down">
  #logMsg#
  </cfmail>
  <cfreturn />
</cffunction>
```


Application Events Walkthrough

Objectives

After completing this lab, you should be able to:

- Define Application event handlers in Application.cfc

Session Events – onSessionStart()

- Code that is run the first time a user makes a request is contained inside a function named “onSessionStart”
- ReturnType: None
- Arguments: None
- Popular Uses
  - Initialize session variables
  - Authentication and other user security-related code
  - Write to log file or database

onSessionStart() Code Sample

```
<cffunction name="onSessionStart" returntype="void">
  <cfset session.objUser = createObject("component",
  "user") />
  <cfreturn />
</cffunction>
```

Session Events – onSessionEnd()

- Code that is run when a session ends just prior to the session being removed from memory (due to activity timeout or closing the browser if using J2EE sessions) is contained inside a function named "onSessionEnd"
- ReturnType: void
- Arguments: SessionScope (struct), ApplicationScope (struct)
- Application and session scopes only accessible as arguments, server scope accessible, request scope unavailable
- Popular Uses
  - Clean-up session data and/or processes
  - Write to log file or database

onSessionEnd() Code Sample

```
<cffunction name="onSessionEnd" returntype="void">
  <cfargument name="sessionScope" type="struct"
  required="true“ />
  <cfargument name="appScope" type="struct"
  required="true“ />
  <cfset var logMsg = "Session ID " &
  arguments.sessionScope.jSessionID & "ended" />
  <cflog file="sessionEvents" text="#logMsg#"
  application="true“ />
  <cfreturn />
</cffunction>
```


Session Events Walkthrough

Objectives

After completing this lab, you should be able to:

- Define Session event handlers in Application.cfc

Request Events – onRequestStart()

- Code that is run when a request begins (just prior to the requested page being executed) is contained inside a function named "onRequestStart"
- ReturnType: Boolean (optional if no returntype is specified in the function definition)
- Arguments: targetPage (string)
- Request isn't processed if false is returned or error thrown
- Popular Uses
  - Any business logic that needs to be performed on every request, such as user authorization or setting request specific variables

onRequestStart() Code Sample

```
<cffunction name="onRequestStart" returntype="boolean">
  <cfargument name="target" type="string" required="true“
  />
  <cfset var allow =
  session.objUser.allowAccess(arguments.target) />
  <cfreturn allow />
</cffunction>
```


Request Events – onRequestEnd()

- Code that is run after a request is processed but before the output is returned to the client is contained inside a function named "onRequestEnd"
- ReturnType: void
- Arguments: TargetPage (string)
- Popular Uses
  - Any business logic that needs to be performed after every request, such as performing data clean-up or performance metrics analysis of some sort

onRequestEnd() Code Sample

```
<cffunction name="onRequestEnd" returntype="void">
  <cfargument name="target" type="string" required="true“
  />
  <cfinclude template="footer.cfm“ />
  <cfreturn />
</cffunction>
```


Request Events – onRequest()

- Code that must intercept a request is contained inside a function named “onRequest”
- ReturnType: void
- Arguments: TargetPage (string)
- Using this method intercepts the requested page and makes the CFC the request – i.e. you must manually include any resource to send back to the client.
- Using this method makes the variables scope of any requested page that’s included visible to other Application.cfc methods
- Do not use this method if there are CFCs accessed via SOAP or remoting that are effected by Application.cfc
- Popular Uses
  - In order to intercept and/or manipulate the output stream (the output generated by the requested page) generated by a requested resource – for example, you could append/prepend text, replace specific characters or words, etc. by including the resource requested in a <CFSAVECONTENT> and could manipulate that content before returning it to the client

onRequest() Code Sample

```
<cffunction name="onRequest" returntype="void">
  <cfargument name="target" type="string" required="true“
  />
  <cfset var retText = “” />
  <cfsavecontent variable="retText"><cfinclude
  template="#arguments.target#"></cfsavecontent>
  <cfset retText = replaceNoCase(retText, "Simon", "King
  Simon", "All") />
  <cfset writeOutput(retText) />
  <cfreturn />
</cffunction>
```


Request Events Walkthrough

Objectives

After completing this lab, you should be able to:

- Define Request event handlers in Application.cfc

Error Events – onError()

- Code that is run whenever an unhandled exception occurs is contained inside a function named “onError”
- ReturnType: void
- Arguments: Exception (struct), EventName (string)
- Event name argument holds name of Application.cfc event that threw exception – is an empty string if onRequest() isn’t used and the error wasn’t thrown by an event method
- Overrides any <CFERROR> or CFAdmin error settings
- Cannot display anything to users if the error occurred in onApplicationEnd() or onSessionEnd()
- If onError() throws an exception, it’s handled by <CFERROR> tags in the Application.cfc initialization block and by CFAdmin error pages
- Popular Uses
  - Error handling

onError() Code Sample

```
<cffunction name="onError" returntype="void">
  <cfargument name="objErr" type="struct" required="true“
  />
  <cfargument name="guiltyEvent" type="string"
  required="true“ />
  <cfif trim(arguments.guiltyEvent) is "">
   <cfinclude template="error_template.cfm“ />
  <cfelse>
   <cflog file="eventErrs" text="An error occured in the
  following event: #arguments.guiltyEvent#“ />
  </cfif>
  <cfreturn />
</cffunction>
```

Other Information

- You can add other methods to Application.cfc
- Application.cfc can extend other components
- You can manually call Application.cfc methods – one great use for this is manually resetting a session
- If you manually call an event – the scope is not locked

Inheritance Walkthrough

Objectives

After completing this lab, you should be able to:

- Use inheritance with Application.cfc
- Extend an existing Application.cfc in order to secure a directory

Summary

- The new application framework in ColdFusion MX 7 is simple to use and gives developers access to events that were previously unavailable
- In Application.cfc you can capture events by creating methods with the following names:
  - onApplicationStart(), onApplicationEnd()
  - onSessionStart(), onSessionEnd()
  - onRequestStart(), onRequest(), onRequestEnd()
  - onError()
- Any code valid in a CFC is valid in Application.cfc

Resources

- Macromedia Live Docs
  http://livedocs.macromedia.com/coldfusion/7/htmldocs/
- CFCZone.org
  http://www.cfczone.org/
- CFDJ and countless blogs
- Google!
  http://www.google.com

Q&A

Feel free to send questions and/or comments to shorwith@aboutweb.com

Please don't forget to fill out your survey:

Session ID: WD204H-1

Session Title: ColdFusion Application Framework

Speaker: Simon Horwith

Thanks!


Better by Adobe™
2006 Adobe Systems Incorporated. All Rights Reserved.
37
Adobe