Mura CMS

Anatomy of a Plugin

blueRiver

Grant Shepert

- Plugin developer
  - Gallery
  - Forums
  - Google Sitemap Generator
  - …
- Mura CMS Trainer
- ColdFusion developer for ~15 years

Grant Shepert

- Plugin developer
  - Gallery
  - Forums
  - Google Sitemap Generator
  - …
- Mura CMS Trainer
- ColdFusion developer for ~15 years
- **Opening Blue River Europe this summer!**

What is a plugin?

- A way to safely extend the Mura CMS
- Installable, distributable package (feature, not a reason)
- Contains all the necessary elements to build powerful custom applications within Mura CMS
  - Custom Administrator
  - Event Handlers
  - Display Objects

mura

Our Mission Today:

Build a Plugin from Scratch!

Easy as 1, 2, 3

mura

Our Mission Today:

Build a Plugin from Scratch!

Easy as 1, 2, 3 (er … 4, 5, 6)

mura

1: An Empty Directory

- Empty directory
- Add **/plugin** directory
- Add **/plugin/config.xml.cfm** file
- Add **/index.cfm**
- (Optional /plugin/license.txt)
- (Optional /plugin/config.cfm)
- (Optional /plugin/plugin.cfc)

config.xml.cfm

- Plugin configuration file
  - Setup variables (base & custom)
  - Display object registration
  - Event object registration

config.xml.cfm

```
<cfoutput>
<plugin>
	<name>My First Plugin</name>
	<package>MyFirstPlugin</package>
	<directoryFormat>packageOnly</directoryFormat>
	<loadPriority>5</loadPriority>
	<version>0.0.0.1</version>
	<provider>Blue River Interactive Group</provider>
	<providerURL>http://getmura.com</providerURL>
	<category>Application</category>
	<settings>
	</settings>
	<eventHandlers>
	</eventHandlers>
	<displayobjects location="global">
	</displayobjects>
</plugin>
</cfoutput>
```

config.xml.cfm

```
<cfoutput>
<plugin>
    …
    <ormcfclocation>
        Location within plugin root where you orm cfcs are located (can be a list)
    </ormcfclocation>
    <customtagpaths>
        Location within the plugin root where you would like the app to look for
        custom tags. (can be a list)
    </customtagpaths>
    <mappings>
        <mapping name="mycustlib" location="path of of plugin root"/>
    </mappings>
    …
</plugin>
</cfoutput>
```

2: Install

- Mura Plugins are .zip packages
- **UPDATE:** zip the contents of the folder **or** the folder itself

3: Connect To Mura

- The Mura Scope, a.k.a. **‘$’**
- The Plugin configuration, a.k.a. **‘pluginConfig’**

Mura Scope: $

- Direct access to the Bean Factory via
  $.getBean()
- Mura events: $.event(), $.announceEvent(), etc.
- Sub-scopes: $.content(), $.currentUser(),
  $.globalConfig(), $.siteConfig(), etc.
- Helpers: $.getContentRenderer(), $.getPlugin(), etc.

pluginConfig

- .getDirectory(), .getPackage(), etc.
- .getApplication(), .getSession()
- .getAssignedSites(), etc.

config.cfm

- Adding **/plugin/config.cfm**
- File name/location is best practice, not really required

config.cfm

```
<cfsilent>

<cfif not isDefined("$")>
	<cfset $ = application.serviceFactory.getBean('muraScope').init(session.siteid) />
</cfif>

<cfif not isDefined("pluginConfig")>
	<cfset pluginConfig = $.getBean('pluginManager').getConfig(packageName) />
</cfif>

</cfsilent>
```

3: Pretty

- The Mura admin ‘wrapper’

```
#$.getBean('pluginManager').renderAdminTemplate(
        body=bodyVariable,
        pageTitle=pluginConfig.getName()
)#
```

4: Display Objects

- These are the plugin's 'Content Objects'
- Inserted on a page manually via
  Site Manager ➔ *Content Objects*
- Two types: .cfm and .cfc

Via .cfm

- Simple ‘include’ format
- Keep in mind you are in the general Mura ‘scope’ of things

```
<cfoutput>Hello from my first plugin (via cfm)!</cfoutput>
```

Via .cfc

- Much more flexible
- Variable scope safe
- Hidden gem (tease!)

```
<cfcomponent extends='mura.plugin.pluginGenericEventHandler'>

	<cffunction name="myCFCDisplay" output="true">
		<cfargument name="$" required='false' />

		<cfoutput>Hello from my first plugin (via cfc)!</cfoutput>
		<!---<cfreturn “Hello from my first plugin (via cfc)!” /> --->
	</cffunction>

</cfcomponent>
```

Display Objects

- Create **/display**
- Create **/display/my_display.cfm**
- Create **/display/myDisplay.cfc**

Display Objects

- Back to the config.cfm.xml

```
<cfoutput>
<plugin>
        …
        <displayobjects location="global">
                <displayobject name="Display via cfm"
                  displayobjectfile="display/my_display.cfm"/>

                <displayobject name="Display via cfc"
                  displaymethod="myCFCDisplay"
                  component="display.myDisplay" persist="false"/>
        </displayobjects>
</plugin>
</cfoutput>
```

OptionsRender

- By convention, *OptionsRender* added to display method name
- Dynamic display objects

OptionsRender

```
<cffunction name="myCFCDisplayOptionsRender" returntype="string">
	<cfargument name="$">
	<cfset var str = "">

	<cfsavecontent variable="str"><cfoutput>
	<select name="availableObjects" id="availableObjects" class="multiSelect" size="14" style="width: 310px;">
		<option value='plugin~Gallery One~#$.event().getValue("ObjectID")#~{"id":"1234"}'>Gallery
		One</option>
		<option value='plugin~Gallery Two~#$.event().getValue("ObjectID")#~{"id":"5678"}'>Gallery
		Two</option>
	</select>
	</cfoutput></cfsavecontent>

	<cfreturn str>
</cffunction>
```

OptionsRender

```
<cffunction name="myCFCDisplayOptionsRender" returntype="string">
	<cfargument name="$">
	<cfset var str = "">

	<cfsavecontent variable="str"><cfoutput>
	<select name="availableObjects" id="availableObjects" class="multiSelect" size="14" style="width: 310px;">
		<option value='plugin~Gallery One~#$.event().getValue("ObjectID")#~{"id":"1234"}'>Gallery
		One</option>
		<option value='plugin~Gallery Two~#$.event().getValue("ObjectID")#~{"id":"5678"}'>Gallery
		Two</option>
	</select>
	</cfoutput></cfsavecontent>

	<cfreturn str>
</cffunction>
```

Accessing ‘params’

- Access our ‘params’ inside our registered display method

```
<cffunction name="myCFCDisplay" output="true">
	<cfargument name="$" required='false' />

	<cfset params = deserializeJSON( $.event().getValue("params")  ) />

	<cfoutput>
	<p>The gallery ID you have requested is #params.id#</p>
	</cfoutput>
</cffunction>
```

5: Events

- Mura has a strong event Model
- Front End Request, Contextual, Renderer & Handler events

Events

- Create **/event**
- Create **/event/eventHandler.cfc**

Events

- Back to the config.cfm.xml

```
<cfoutput>
<plugin>
	…
	<eventHandlers>
		<eventHandler event="onApplicationLoad" component="events.eventHandler"
		persist="false" />
	</eventHandlers>
	…
</plugin>
</cfoutput>
```

eventHandler.cfc

```
<cfcomponent extends="mura.plugin.pluginGenericEventHandler">

	<cffunction name="onApplicationLoad">
		<cfargument name="$">
		<cfset variables.pluginConfig.addEventHandler(this)>
	</cffunction>

	<cffunction name="onPageDefaultBodyRender“ output=“true”>
		<cfargument name="$">

		<cfset $.content().setValue('title','this is my cool page') />

		<cfoutput>
		<h3>#$.content().getTitle()#</h3>
		#$.content().getBody()#
		</cfoutput>
		<!--- <cfreturn “some string” /> --->
	</cffunction>
</cfcomponent>
```

eventHandler.cfc

```
<cfcomponent extends="mura.plugin.pluginGenericEventHandler">

    <cffunction name="onApplicationLoad">
        <cfargument name="$">
        <cfset variables.pluginConfig.addEventHandler(this)>
    </cffunction>

    <cffunction name="onPageDefaultBodyRender“ output=“true”>
        <cfargument name="$">

        <cfset $.content().setValue('title','this is my cool page') />

        <cfoutput>
        <h3>#$.content().getTitle()#</h3>
        #$.content().getBody()#
        </cfoutput>
        <!--- <cfreturn “some string” /> --->
    </cffunction>
</cfcomponent>
```

eventHandler.cfc

```
<cfcomponent extends="mura.plugin.pluginGenericEventHandler">

	<cffunction name="onApplicationLoad">
		<cfargument name="$">
		<cfset variables.pluginConfig.addEventHandler(this)>
	</cffunction>

	<cffunction name="onPageDefaultBodyRender“ output=“true”>
		<cfargument name="$">

		<cfset $.content().setValue('title','this is my cool page') />

		<cfoutput>
		<h3>#$.content().getTitle()#</h3>
		#$.content().getBody()#
		</cfoutput>
		<!--- <cfreturn “some string” /> --->
	</cffunction>
</cfcomponent>
```

6: plugin.cfc

- Install, Update, Delete

plugin.cfc

- Create **/plugin/plugin.cfc**

```
<cfcomponent>
	<cffunction name="init" returntype="any" access="public" output="false">
		<cfargument name="pluginConfig"  type="any" default="">
		<cfset variables.pluginConfig = arguments.pluginConfig>
	</cffunction>

	<cffunction name="install" returntype="void" access="public" output="false">
	</cffunction>

	<cffunction name="update" returntype="void" access="public" output="false">
	</cffunction>

	<cffunction name="delete" returntype="void" access="public" output="false">
	</cffunction>
</cfcomponent>
```

plugin.cfc

- Create **/plugin/plugin.cfc**

```
<cfcomponent>
	<cffunction name="init" returntype="any" access="public" output="false">
		<cfargument name="pluginConfig"  type="any" default="">
		<cfset variables.pluginConfig = arguments.pluginConfig>
	</cffunction>

	<cffunction name="install" returntype="void" access="public" output="false">
	</cffunction>

	<cffunction name="update" returntype="void" access="public" output="false">
	</cffunction>

	<cffunction name="delete" returntype="void" access="public" output="false">
	</cffunction>
</cfcomponent>
```

Advanced Options

- Frameworks: FW/1, ColdBox, Mach II, Fusebox & similar
- ORM friendly
- URL rewrites to bake your application into Mura
- Custom events
- Install: custom databases, bundles

Why Plugins?

- Update safe
- Packaged and organized
- Easily distributable
- Licensing exception
- Framework-friendly

When Plugins?

NO

- Small extensions and modifications to Mura CMS
- No chance you are distributing or extending

YES

- Distributed
- Related files
- Large or complex

Resources

- http://getMura.com/forum/
- http://getMura.com/app-store/mura-plugins/
- http://github.com/
- http://grantshepert.com/