Building Pages with a Page Object

Posted At : May 20, 2009 11:34 PM | Posted By : Jon Hartmann
Related Categories: ColdFusion, Experiments, Frameworks

I was reading through **PHP and MySQL Web Development** at the local bookstore the other night when I noticed that their example code for explaining object oriented concepts was a Page class that basically built a page based on its properties. While the concept of directly building the return code string by string is a little foreign to ColdFusion, I was very interested in the idea of an extensible Page object that could help me manage my applications. Read more to see what I came up with.

The Idea

The basic concept is that I want to come up with a Page class that I can reuse and extend to handle the basics of putting a page together. As an added bonus it should also probably be able to handle rendering different versions of the same page, such as when working with "themes" like in PHP. I'm going to have to limit myself from going beyond that for space reasons... my original list of specifications for this project had turned it into a mini-framework, and I think thats a little beyond the scope of what I'm looking to do right now.

Iteration One: The Basics

Ok, so the first thing I'm going to do is a very basic Page.cfc. It should act like my final version, but it wont be dynamic for now.

Page.cfc

```
<cfcomponent output="false">


    <cffunction name="Display" description="Displays the contents of the current page." access="public" output="false" returntype="string">
        <cfargument name="theme" type="string" required="false" />


        <cfsavecontent variable="local.returnValue">
<html>
    <head>
        <title>Test</title>
    </head>
    <body>
        Hello World!
    </body>
</html>
        </cfsavecontent>


        <cfreturn trim(local.returnValue) />
    </cffunction>


</cfcomponent>
```

Back in my root I drop in an index.cfm, and add the test code:

index.cfm

```
<cfscript>
    Page = CreateObject("Page");


    WriteOutput(Page.display());
</cfscript>
```

Run that and I get a simple hello world response. So far so good, but its not nearly dynamic enough.

Iteration 2: Make It Dynamic

The next step is to make this thing more dynamic since currently all of the content is static. This isn't really hard, I just added some variables scope properties to the Page.cfc, added the Getters and Setters, and updated display() to use those values:

Page.cfc

```
<cfcomponent output="false">

    <cfset variables.title = "Default" />
    <cfset variables.content = "" />

    <cffunction name="getTitle" access="public" output="false" returntype="string">
        <cfreturn variables.title />
    </cffunction>

    <cffunction name="setTitle" access="public" output="false" returntype="void">
        <cfargument name="title" type="string" required="true" />

        <cfset variables.title = arguments.title />

        <cfreturn />
    </cffunction>

    <cffunction name="getContent" access="public" output="false" returntype="string">
        <cfreturn variables.content />
    </cffunction>

    <cffunction name="setContent" access="public" output="false" returntype="void">
        <cfargument name="content" type="string" required="true" />

        <cfset variables.content = arguments.content />

        <cfreturn />
    </cffunction>

    <cffunction name="Display" description="Displays the contents of the current page." access="public" output="false" returntype="string">

        <cfsavecontent variable="local.returnValue">
<html>
    <head>
        <title><cfoutput>#getTitle()#</cfoutput></title>
    </head>
    <body>
        <cfoutput>#getContent()#</cfoutput>
    </body>
</html>
        </cfsavecontent>

        <cfreturn trim(local.returnValue) />
    </cffunction>

</cfcomponent>
```

Now my page isn't getting content by default, so I have to update my index.cfm as well.

index.cfm

```
<cfscript>
    Page = CreateObject("Page");

    Page.setContent("Hello from another world!");

    WriteOutput(Page.display());
</cfscript>
```

Ok, thats better, but I'm still putting my display code into my Page object, and what about those themes?

Iteration 3: Themes

For this iteration I want to pull the code for the display out of my Page.cfc into its own template, and I also want to make the whole thing work with themes so that its a little more dynamic. To do this I created a "themes/" folder in my root, and added two folders "red/" and "blue/" and added the following index.cfm files to them.

/themes/red/index.cfm

```
<html>
    <head>
        <title><cfoutput>#getTitle()#</cfoutput></title>
        <style>
            body {
                color: red;
            }
        </style>
    </head>
    <body>
        <h1>Red Theme</h1>
        <cfoutput>#getContent()#</cfoutput>
    </body>
</html>
```

/themes/blue/index.cfm

```
<html>
    <head>
        <title><cfoutput>#getTitle()#</cfoutput></title>
        <style>
            body {
                color: blue;
            }
        </style>
    </head>
    <body>
        <h1>Blue Theme</h1>
        <cfoutput>#getContent()#</cfoutput>
    </body>
</html>
```

Now I can switch between a red and a blue theme for my application, but I need a mechanism to control which theme to use, so I'll create an Application.cfc as well.

Application.cfc

```
<cfcomponent output="false">

    <cfset this.name = "PageFramework" />

    <cffunction name="OnApplicationStart">
```

```
        <cfset OnInitialization() />

    </cffunction>

    <cffunction name="OnRequestStart">

        <cfparam name="url.reinit" default="0" />

        <cfif url.reinit>
            <cflock scope="application" type="exclusive" timeout="10">
                <cfset OnInitialization() />
            </cflock>
        </cfif>

    </cffunction>

    <cffunction name="OnInitialization">

        <cfset application.theme = "red" />

    </cffunction>

</cfcomponent>
```

And finally, you'll beed to update your Page.cfc to work with this new setup:

Page.cfc

```
<cfcomponent output="false">

    <cfset variables.title = "Default" />
    <cfset variables.content = "" />
    <cfset variables.template = "index.cfm" />

    <cffunction name="getTitle" access="public" output="false" returntype="string">
        <cfreturn variables.title />
    </cffunction>

    <cffunction name="setTitle" access="public" output="false" returntype="void">
        <cfargument name="title" type="string" required="true" />

        <cfset variables.title = arguments.title />

        <cfreturn />
    </cffunction>

    <cffunction name="getContent" access="public" output="false" returntype="string">
        <cfreturn variables.content />
    </cffunction>

    <cffunction name="setContent" access="public" output="false" returntype="void">
        <cfargument name="content" type="string" required="true" />

        <cfset variables.content = arguments.content />

        <cfreturn />
    </cffunction>

    <cffunction name="Display" description="Displays the contents of the current page." access="public" output="false" returntype="string">
        <cfargument name="theme" type="string" required="false" default="#application.theme#" />
```

```
        <cfsavecontent variable="local.returnValue">
            <cfinclude template="/themes/#arguments.theme#/#variables.template#" />
        </cfsavecontent>

        <cfreturn trim(local.returnValue) />
    </cffunction>

</cfcomponent>
```

Now you should be able to switch your theme by updating your application.theme value and reinitalizing your application.

The End?

From here there is a lot more that can be done with this code, like adding in controller logic, handling the use of objects that extend Page.cfc (notice that I pulled the template out to a separate variable?), making the Getters and Setters work with a more dynamic data set so that more data can be passed through the Page object, and all kinds of things. This is all for now, but expect me to come back with some updates soon.