A ColdFusion Page Controller

Posted At : March 14, 2007 2:26 PM | Posted By : Steve
Related Categories: ColdFusion, Page Controller

A hate to admit it, but ever since I stopped using Fusebox a few years ago, I have been without a controller. I had my view separated from my model, but no controller to be seen. In fact, I even questioned the need for a controller on most of my projects (while seeing how it could help on larger projects). I was all ready to write a "Do I need a Controller" blog entry.

And then...

I discovered the Page Controller.

Most ColdFusion frameworks - at least the so-called MVC frameworks - are really Front Controllers. That is, they handle incoming requests and coordinate processing among multiple requests. Per Martin Fowler, a **Front Controller** is "A controller that handles all requests for a Web site."

A **Page Controller** is "An object that handles a request for a specific page or action on a Web site." (again, per Fowler).

My implementation of a page controller starts with a **PageController.cfc** that can be extended by each page controller. This has following methods that can be used by any page controller:

- go (basically just cflocation)
- param (similar to cfparam, but default kicks in instead of error for incorrect data type)
- require (similar to cfparam, but no default or error - uses redirect for not defined or wrong data type)

The page controller itself is integrally tied to a given page. Therefore, I name it the same as the page (except the extension) and put it in the same folder). So, "about.cfm" would have a page controller called "about.cfc" (noting that this would only be for pages that need a page controller).

My view files have access to shared-scope variables (such as Application scope) and can do output, but don't perform any business logic or run any queries. My model components don't have access to shared-scope variables and can't do output, but can perform business logic and run queries. I think this is standard in both cases.

My page controller can acess shared-scope variables, but only in the pseudo-constructor. They cannot do output or perform business logic. Instead, they get data from the model and send data from the view to the model. The advantage of a page controller is that while a method of the model cannot know about the structure of a form in the view, the page controller can. This allows it to translate the form structure of a page to the API of a model method.

Here is an example of the top of a view page ("sample-page.cfm") that is using a page controller:

```
<cfset Controller = CreateObject("component","sample-page")>
<cfset StructAppend(variables,Controller.loadData())>
```

The load loadData() method returns a structure of variables that should be available to the page. You can take a look at the example **sample-page.cfc** to see what the page controller itself could look like.

The page controller can also have methods to respond to form posts. These methods can then translate the form structure into data appropriate for the API of one or more methods in the model. The saveQuiz() method in sample-page.cfc is an example of this. You can see how passing in the Form structure to a method in the model wouldn't have worked here, but the code to do the translation doesn't really fit in a view file either.

I often make these methods access="remote". This allows me to submit a form directly to the controller (though I have yet to do so). This seems to open up a security issue, but as anyone could post any form to your page anyway, it really just makes clear the entry-point for external data that already exists.

Although I don't have time to cover it in this post, this use of a page controller also makes AJAX much easier (hopefully I can cover that in a post next week).

My one warning about my use of page controllers is that I have only been using them for a few weeks. So, I haven't run into any trouble with them, but nor have I used them for very long.