

OO Principles: Encapsulating CFCs

Posted At : July 23, 2009 9:45 AM | Posted By : Steve

Related Categories: ColdFusion, OO Principles

I don't "do OO" development in ColdFusion. I'm starting with that statement not to spark another debate about whether to use OO in ColdFusion, but rather to clarify that while this post is about a principle of object oriented development, you don't need to "Do OO" in order to learn, use, and benefit from encapsulating CFCs.

When I first started using CFCs, I knew that **encapsulation and decoupling** were important, but this brought up new challenges. For example, if I had a method that queried a database, how would it know what datasource to use?

Before using CFCs, I would just use a request scoped variable to store the datasource. If, however, I access a request scoped datasource in a CFC then I am breaking encapsulation.

I could pass in the datasource with every method call, but that is a lot of extra code and it doesn't "feel" right. After all, if I am saving a record then the datasource isn't data for that record. Going down this path makes life more difficult and code less clear. But I don't want to break encapsulation either.

Fortunately, there is another way.

I should take a brief interlude to point out here that I am using CFCs as services, not objects. If you are using them as objects, then this is probably too basic for you. If you don't know the difference, don't worry about it just yet.

Instead of accessing the component by cfinvoke using the path of the component, I can instantiate it into a variable and use the instantiated component instead. The advantage here is that the instantiated component can store data within it (like a datasource, perhaps).

By convention, CFCs are instantiated using the "init" method (functions in a CFC are generally called methods).

So, I will write an init method that will take a datasource argument.

```
<cffunction name="init" access="public" returntype="any" output="no">
    <cfargument name="datasource" type="string" required="yes">

    <cfset variables.datasource = arguments.datasource>

    <cfreturn This>
</cffunction>
```

Most programmers put this method at the top of their component. Note that the datasource is copied from arguments scope (which lives only within the method itself) to variables scope (which is available to all methods in a component).

Assuming this is in a CFC named "Users.cfc", this is how I could load that into an Application - scoped component:

```
<cfset Application.Users = CreateObject("component","Users").init(datasource=request.dsn)>
```

The value of request.dsn is now stored in Application.Users. It is important to note here that the variables scope inside a CFC persists with the CFC instantiation itself. So, in this case, the variables scope in Users.cfc will exist for as long as the Application.Users variable does - even on subsequent page calls.

Now the Users component know about the datasource without breaking encapsulation and I don't have to pass it in to every method.