# Why I Don't Do OO

Posted At : July 16, 2008 7:30 AM | Posted By : Steve
Related Categories: ColdFusion, OO Principles

This is an entry that I have been intending to write for some time. I keep putting it off. Perhaps because I don't think I will explain myself well, perhaps because I don't want to be flame-bait.

**T.J. Downes** recent entry "**Why Aren't More CF Developers Using OO Methodologies and Frameworks?**" made me decide that now is the time.

He asked those of don't use OO to explain why, so I will do my best to explain why I don't.

**I Don't Need It**

This is a bit hard to explain. More accurately, I have yet to run into a problem that I feel would be solved by moving all of my development to OO. I do use object from time to time. For example, my code generator makes pretty heavy use of objects. They make sense there and so I use them.

Generally, however, I hear about the benefits of OO and I think that I get all of the benefits that I care about without doing pure OO development (more on that later).

**It Adds Complexity**

For years, I have had great success in talking to developers who use other languages in getting them to take another look at ColdFusion. I have had two standby examples that I have used for years.

The first is <cfquery>. Any developer from ASP/ASP.NET or PHP that has to write SQL (all of them so far) that I have talked to has been really impressed with the direct simplicity of <cfquery>.

The next is <cfoutput>. Want to loop over a recordset and output an HTML list?

```
<ul>
<cfoutput query="qMyRecords">
   <li>#MyValue#</li>
</cfoutput>
</ul>
```

Done!

Want to make nested lists for categories?:

```
<ul>
<cfoutput query="qMyRecords" group="MyCategory">
   <li>#MyCategory#
      <ul>
      <cfoutput>
         <li>#MyValue#</li>
      </cfoutput>
      </ul>
   </li>
</cfoutput>
</ul>
```

Done again!

This compares with having to do a manual loop with some sort of check to see if you are at the end of the recordset. The nesting requires extra conditional logic to compare to previous values. It isn't hard, but it makes the code just a little messier. I see the same thing now in ColdFusion when people try to add OO-ness to the output instead of just using queries.

**I Can Follow OO Principles without Using OO**

I know this sounds strange and I don't think I will explain this part well, but give me a chance. I do use a service layer. I am very strict about encapsulation. I take advantage of inheritance and (occasionally) polymorphism within my service layer. I just pass around data instead of objects.

So, for example, I don't (usually) have a "User" object. Instead, I have a "Users" service. This would have a "getUsers" method that returns a recordset.

One of the criticisms about this is "What if need a property of a user that can't be calculated in SQL?" This belies an assumption that a recordset must come directly from the database. My view code doesn't care how my service puts together the recordset. It just knows that it will get a recordset from the service with the data it needs.

So, for example, it may have an "Age" column in the recordset. That column may be calculated and added outside of a query. The view code simply doesn't care how the column was created. It just knows what data it gets from the service.

**Extra Rules I Can Enforce**

One advantage that I get my using a service-based approach instead of an object-based approach is that I can enforce a rule of "no method calls in the view". The view can call UDFs, but it cannot call service methods directly. That is the controller's job. The view can only use the data that it gets from the controller.

The view does looping but it doesn't know what happens under the hood for any given method. Consequently, I don't what the view calling methods. The worst performance problems I have seen in ColdFusion come from looping over queries. How can the view knowingly avoid this when it doesn't know what happens under the hood for any method?

**The Starting Over Problem**

One criticism I hear of ColdFusion programmers who refuse to move to OO is that they don't want to give up their procedural expertise and start over with a new way of doing things. I certainly have to concede that is a factor for me. My development seems to be going well and I am not keen on going through a major slow-down to dramatically shift my development style.

This is especially true as I have a large tool-set built around my current practices. Currently, I use my tool set to quickly build stable but flexible solutions for my clients. It offers me a competitive advantage. In order for me to give that up, I would have to see major benefits for the cost.

That being said, if I do see those benefits, I am not opposed to going through growing pains to get them.

**This is ColdFusion Specific**

It is important to note here that I am not anti-OO. I remember being in a job interview where the people interviewing me said "OO is just a fad". I thought they were sticking their heads in the sand. Of course, they also told me that CSS slowed down web sites, so it clearly wasn't a good match for me.

When I write JavaScript, I do (try) to write OO code. It makes sense there. When I practice Flex coding (I am just barely learning), I definitely do OO there. It just makes sense there. When I write a façade for ColdFusion to talk to Flex, I will likely have ColdFusion return objects to Flex. Again, that just makes sense.

**This is Me Specific**

I am not here trying to advocate that everyone avoid OO. I'm just trying to provide my reasons for not using OO. Hopefully they show how a reasonable programmer can avoid OO and that OO need not be "The One True Path". Maybe this just shows that I am missing the boat completely.

OK. Those are my reasons for avoiding OO.

Flame On!