

Re: What is/is not considered to be good OO programming

Source: <http://coding.derkeiler.com/Archive/PHP/comp.lang.php/2003-12/0609.html>

From: Tony Marston (tony_at_marston-home.demon.co.uk)

Date: 12/10/03

Date: 10 Dec 2003 06:02:48 -0800

googlemike@hotpop.com (Google Mike) wrote in message
news:<25d8d6a8.0312091001.27d1e483@posting.google.com>...

> *Oh what the hell. Here's my two cents...*

>

> *I agree with Tony Marston, if I read him right after skimming this
> thread. Brit programmers have common sense, unlike most of my American
> compatriots I have to interact with in the workplace.*

>

> *OO is way overblown. If you use it right, it's great. If you use it
> wrong, and many pseudo-senior devs I've seen use it unnecessarily, it
> can do nothing but introduce unnecessary complexity and even slow the
> app down.*

>

> *For that matter, the same argument can be made for using XSLT to
> abstract your presentation layer. XSLT is overblown and introduces
> nothing but delays, I find.*

Then your guys haven't learnt how to use XSL properly. It is a scripting language just like PHP, and as such you can make use of standard reusable modules. For example, in my sample application (which you can try online at <http://www.tonymarston.net/sample/index.html>) I have a family of 6 components for each entity/table – list, insert, update, enquire, delete, search – for which I need only 2 (two) XSL stylesheets. Within each of these stylesheets I make use of the `<xsl:include>` and `<xsl-call-template>` functions to invoke common code. When I want to paint a data field on a form all I use is the following:

```
<xsl-call-template name="datafield">  
  <xsl:with-param name="item" select="tree_type_desc"/>  
</xsl-call-template>
```

This template/subroutine then decides how to paint the field (display-only, single-line textbox, multi-line textbox, dropdown list, radiogroup, checkbox, whatever) using information supplied either in the XML file or as a parameter supplied at runtime to the XSL

transformation process.

I use similar template calls to build my action buttons, navigation buttons, column headings, etc. They are documented at <http://www.tonymarston.net/xml-xsl/xml-and-xsl.html>.

Using this technique, coupled with my own method of using classes/objects which so many of you find objectionable, I can define a database table and build the 6 components to maintain/display its contents in 60 minutes. That is an average of 10 (ten) minutes per component. How's about THEM apples!!

> *The same arguments going on here in PHP are being argued like mad on
> the .NET platform too.*

>

> *Take for example a simple web-based checkbook app for a series of
> users. You login, see your checkbook, and can make entries to it.
> Here's what a series of developers might think:*

>

> *psuedo-senior dev 1: Let's make a User object, Checkbook object, Login
> object, Logout object, and an Entry object! Let's do objects for
> anything we can think of!*

>

> *psuedo-senior dev 2: Let's make the business layer push out nothing
> but XML, then use XSLT to style this output for the end user.*

>

> *psuedo-senior dev 3: I think we should make the data layer emit
> nothing but XML.*

>

> *psuedo-senior dev 4: I think we should make everything in stored
> procedures and make it receive and emit XML. The presentation code can
> be written in PHP to read the XML and style it with XSLT.*

The simplest way to deal with people like that is to lock them in a room and see who can produces results the fastest. The winner gets to keep his job. THAT tends to sort out the men from the monkeys.

> *No. Not me on any of this, even if it were something as complex as a
> Near Earth Object Tracking System, and I've been programming since
> 1979 and am 36 years old. My title says "Senior Developer". My take on
> objects, XML, and XSLT would be:*

>

> *1. I'd use an associate array (better known as a hash table by us old
> f*cks), not an object, for the checkbook entry record.*

>

> *2. Thinking about code-reuse for future projects, I'd bundle my
> frequently used functions in a web class, strings class, db class, and
> a settings class. That way, when I call a function (er, "method") like
> Redirect, I can understand where this method came from, and so can
> others who read my code, because I would have it listed in my code as
> \$web->Redirect('page.php'). If Redirect breaks, you instantly know*

- > *it's in the "_web.php" page (if you saw my Requires statement).*
- >
- > *3. If an object isn't necessary, I avoid it like the plague. Not by*
- > *inexperience, but by /experience/.*

That goes for a whole host of different "thingies". I have seen programmers try to use every "thingy" in the book just to prove they know how to use it, which is not the same as using it effectively.

- > *4. When I have to work in teams, we decide on a set of pages and a*
- > *very small set of objects that we're going to assign to each team*
- > *member, and then we iron these out individually for a bit with test*
- > *harnesses. Eventually, we start sharing and interacting with each*
- > *others stuff when these start to stabilize.*
- >
- > *5. I like XML, but I dislike XSLT. XSLT is way, way too difficult of*
- > *an implementation of a superbly fantastic idea. I'll wait until*
- > *something better comes out to replace XSLT.*

I suggest you learn to use XSLT effectively because it is here to stay. Version 2.0 is in the pipeline, and the recently-released standard for XFORMS (see <http://www.w3.org/MarkUp/Forms/>) which will do away with all that non-standard javascript, is based quite heavily on XML and XSL.

- > *For now, you'll see me*
- > *doing and <% {code} %> and <%= \$var %> in my code, although I try as*
- > *much as possible to reduce interruptions in the HTML because of the*
- > *speed hit. That's why you see me doing most of my code in the top of*
- > *the page before the HTML tags.*
- >
- > *6. I like XML, but I don't want to make every thing a particular OO*
- > *layer run through it, such as making the business layer receive XML*
- > *and emit XML.*

My PHP development environment manages to mix a lot of different ideas – procedural scripts, 'included' scripts, objects, 3-tier architecture (with separate presentation, business and data access layers), XML and XSL transformations, and I like to think that I use each technique to its best advantage. If an old fart like me can do it, then why can't the rest of you?

Tony Marston
<http://www.tonymarston.net/>