

Thursday, April 19, 2007

Complex Web Pages with the Zend Framework?

27 March 2008:

With the inclusion of Zend_View Enhanced as first documented, discussed and publicised in this blog series, in the Zend Framework as of 1.5.0 I'd like to thank everyone involved in the process. A few of you pioneered it's implementation and provided immense feedback which tailored the proposal to some specific needs, others kept bringing it up on the mailing lists and to the attention of the Zend reviewers, and others still spent a lot of time converging ideas towards a unified proposal. It just goes to show that the community really does have the power to influence the big picture and get stuff done!

The rest of this series can be accessed as follows:

<http://blog.astrumfutura.com/archives/282-Complex-Views-with-the-Zend-Framework-Part-2-View-Helper-Pattern.html>

<http://blog.astrumfutura.com/archives/283-Complex-Views-with-the-Zend-Framework-Part-3-Composite-View-Pattern.html>

<http://blog.astrumfutura.com/archives/285-Complex-Views-with-the-Zend-Framework-Part-4-The-View-Factory.html>

<http://blog.astrumfutura.com/archives/288-Complex-Views-with-the-Zend-Framework-Part-5-The-Two-Step-View-Pattern.html>

<http://blog.astrumfutura.com/archives/291-Complex-Views-with-the-Zend-Framework-Part-6-Setting-The-Terminology>

Recently I've been involved in a long discussion about the Zend Framework on the [PHP Developers' Network forum](#). Our approach was to pick a simple application (we decided to borrow the Java BluePrints Pet Shop for J2EE) and starting from a basic "Hello World!" example for the Zend Framework work towards a fully functional example. Of course, one of our goals wasn't just to "do it", we wanted to explore the framework in greater detail, and identify how best to use, misuse, subclass, and where it was logical to even replace components should they prove deficient for our needs.

After wrangling about configuration, the advantage/disadvantage of build tools (I love Phing and cannot survive without it!), the location of the bootstrap file, and a few other odds and ends we finally put up the "Hello World!" example in subversion. Many thanks to [Chris Corbyn](#) of [Swiftmailer](#) fame for contributing the repository!

<http://w3style.co.uk/devnet-projects/pet-store/trunk/>

We then decided to look at how the Zend Framework implements Views. In essence, the framework isn't as developed in that respect as its peers. A simple page is easily built using a Zend_View instance, a PHP/HTML template for a list of entries, and a few view helpers and filters. After that however, a complex page becomes progressively more difficult. This is complicated in part by the growing practice of instantiating Views using a helper function on the Controller - unfortunately this is unusable since it introduces coupling making re-use more difficult in other applications where the View has been subclassed.

Back on track, the main problem of a complex View, is that the current Controller is only aware of a subset of its own required Model (data) and the current View. So how do you get the View to include extra sections - for example, details from Technorati for your blog - which are common to ALL pages?

The Zend Framework currently suggests using a Controller's `_forward()` method - basically if your current View needs data from Technorati, then forward from the current Controller to a "TechnoratiController" to fetch it and assign it to the View, and then `_forward()` back. This works for simple things - but what if there are three, or even more sections? What if some sections, have additional embedded sections? How in heaven's name are you supposed to track all those `_forward()`'s?

While pondering the problem, we all agreed using the Controller `_forward()` was not a good idea - it's prohibitively complex, forces Controllers to become aware of other (possibly unrelated) controllers, and in general doesn't promote reuse. I haven't measured the performance impact of such a tactic but it can't be good. We also determined the Zend Framework's view helpers were of limited use being simple classes designed to offer a single public method to templates.

Finally we decided to visit two design patterns to help name the problem and offer a possible solution: Composite View and View Helper. So far we've just implemented a very basic Composite View (KISS) to

further the brainstorming. But it shows a lot of promise in offering a simple, effective solution to building complex web pages with the Zend Framework - granted it breaks the normal practice, and takes subclassing to introduce, but it's a reusable solution.

I'll visit Composite View in more depth soon - for the moment here's the J2EE BluePrints page for the design pattern: [Composite View](#). If anyone knows of a really good example in PHP or Ruby, it would be great to hear about it!


On the View Helper bit - the Zend Framework seems to persist a belief that only Controllers should interface with the Model layer (i.e. Database and associated Model classes). This Model-Push strategy can be complemented with an equally valid Model-Pull strategy - where View Helper classes have the ability to directly access and read data from the Model to pass to the requesting View. This completely avoids the fickle tactic of Controller forwarding and the complexity it tends to introduce into an application.

Stay tuned for a deeper look at both these patterns - specifically how they can be brought to bear on the Zend Framework with a little subclassing of `Zend_View`.

Posted by Pádraic Brady in PHP General, PHP Security, Zend Framework at 20:36

Very interesting post. I'm looking forward to hearing more. I'd like to track down the forum threads and read those too. I am especially interested in seeing a ZF example with pagination. Anonymous on Apr 20 2007, 02:57


Thanks for your post & keep up the great work! We're currently evaluating ZF to port our site to. Anonymous on Apr 20 2007, 04:13

Sounds very interesting, I'm looking forward to reading more about your findings 

Anonymous on Apr 20 2007, 05:17

I've posted about this to the ZF fw-general mailing list. So far not much luck - Matthew posted a dispatch plugin for implementing Two-Step View (or a variation thereof), and there's a JIRA issue suggesting a helper to add nested controller forwarding. These don't really address the root problem though - we want to nest Views into a tree structure (Composite) of any arbitrary depth, not nest numerous dispatch cycles with all the complexity and overhead each involves - that idea seems too smelly to be taken seriously.

All I need is a View, and it's Model - and the simplest solution is a View Helper with the necessary methods a View template can use to access that Model. Then a View tree which just needs a single method call to render a View composite - typical Composite Pattern.

I doubt it would take very much to implement - a few of us are looking at that now starting with a very simple composite design. I'll post here as it progresses and becomes useful (or at least interesting enough to blog about 

). Anonymous on Apr 20 2007, 16:14

Great post, I am currently building a large commercial project using Zend framework (I think the risk is paying off.) and I also became quickly frustrated in how to integrate common functionality without breaking from best practise. I look forward to what you come up with as a solution. Anonymous on Apr 20 2007, 16:48

Great introduction, Pádraic. As well as Marcus, I'm eagerly waiting for a follow-up. Anonymous on Apr 20 2007, 19:13

im also having the same problem as you!

it seems like ZF kind of has this huge bottleneck with it's current MVC approach.

Most professional web developer needs a view structure that will let you mix and match different parts of the site, whether it'll be on the sidebar, on on the body, or anywhere...

It seems the current approach only lets you do a very simple website, where header and footer barely changes.

The view helper doesn't really help much either, they just make you code the same thing twice.

So i found this post very interesting and relieved that I wasn't the only one with this kind of problem.


Let us know of any progress you have on the solution.

It will be a great help to all of us (ZF users) out there. Anonymous on Apr 21 2007, 06:00


Part 2 looks at the View Helper pattern - primarily it pushes the point the View can directly access the Model (read-only to keep data manipulation within Controllers). This makes View Helpers more useful in bypassing the Controller setup.

I'll close in on nested Views soon - myself and Chris Thompson are looking at it and I'll blog about the progress that we have in zeroing in on a solution that's easy to add in to the ZF with a little subclassing. Anonymous on Apr 21 2007, 06:47


what do you mean by part 2? where can i view this? and what is it that you are discussing? Anonymous on Apr 28 2007, 10:40

I am discussing "complex views". Basically how to build a web page from any number of reusable elements (headers, footer, menu, widgets, etc.) which tend to common to many pages in a web applications. Read the entry 


. Anonymous on Apr 28 2007, 17:32

I posted basically same problems on fw-general mailinglist but people tend to say is just our own case and is np on doing such complex things with ZF 


Anonymous on May 1 2007, 14:10

The key is to remain vocal 

. Like any open source project, you can only bring about change (well, realistically a compromise) if you're persistent, persuasive and push a logical point of view in a reasonable manner. Plus we both have low karma in the ZF project - that always takes a little extra effort to overcome.

I made a post about this weeks ago and didn't get a convincing reply - hence the blog series. Without good karma you really need to get out there and grab attention with something concrete to show off as a solution and point a finger at. Otherwise you're just ignored - not from ignorance or anything personal, it's just that folk are busy and don't have time to commit deep thought to every hair brained scheme 

. Always a mistake to take criticism or lack of attention personally - it's never meant that way at all. Anonymous on May 1 2007, 19:41

Your're right, and is good you started this series (And is better to be a series rather than one tutorial, much more "noise" 

).

Now some comments...

You used View Helpers to get data from Model...

Other possible approach is to use controllers instead of helpers. How ? Multiple controllers to be called on bootstrapper, each controller for a view...

Basically instead of calling one controller for each page we can have one controller for each view... And NOT with `_forward`, rather directly from bootstrapper if possible, each one controller to be dispatched one by one... Anonymous on May 1 2007, 19:51

Yep that's all possible - I just haven't looked at the process personally. The current practice leans heavily on `_forward()`. I would

assume a front-to-back system would place reliance on some configuration files to assess what any specific "route" requires in terms of a Model (the controllers to call to retrieve it) for its collective templates, call each in turn, and then aggregate the View output from each at a similar level.


I know at least one person who believes the Response object (being in the View layer) is a high level component where View aggregation can be centralised within the bootstrap. Would require subclassing, or possibly some helper class the Response would delegate to.

I would also guess the process of dispatching each controller in such a theoretical system I have not implemented ;), would be handled by an Application Controller. In a way what you're describing is straight from the MVC approach in that you're looking for the layer that sits between Front Controller and Action Controller which is responsible for calling controllers one by one, and aggregating the resulting View.

If the "Application Controller" makes sense and fits your thinking then you are likely on the right track. The only thing being the ZF doesn't have application controllers per se. It skips that layer in favour of compositing. The App Controller is only zeroing in on a possible solution - it's probably not an exact match for what you mean.

The app controller is largely a refactoring of Controller/View pairs. Once a number of related controllers in a specific workflow/process are duplicating model calls or decision making it's time to look at an application controller to encapsulate the workflow for that process. It's a little abstract to describe in a short comment - but a Google for the pattern might help with some longer explanations that put it better. Anonymous on May 1 2007, 22:39

Thanks for your comments.

Actually atm i am using just one View (but with distinct zones - subviews, tree like organized - which which are rendered by view), not multiple (but must be multiple in the near future, waiting for your conclusions ).

). Instead for

And now what i said it works, i mean now i am having controllers for each subview, and there are launched/ dispatched on bootstrapper instead of dispatching controller of page:

```
foreach($subviews as $sv)
{
$request->setControllerName($sv["controller"]);
$request->setActionName($sv["action"]);
$frontController = $frontController->setRequest($request);

$frontController->dispatch();
}
```

So, each subview has a controller (news controller let's suppose), and action (news list, news detail, news archive, aso).

Is true that i wiped basically normal Controller/Action approach for now, i am actually having a xml database with pages, on that database is for each page described briefly each subview:

```
general/navbar
general/
products/list
```

Once i am integrating multiple views objects instead of using just one is closer to a good solution... Anonymous on May 1 2007, 23:05

[geshi lang=xml]

general/navbar

general/menu

products/list

news/list

[/lang] Anonymous on May 1 2007, 23:09

What about this Module-Controller-Action talked about on the Zend Community Site? Anonymous on Aug 20 2007, 09:56