# How I Use Transfer - Part V.1 - A Comment and Response

Posted At : July 8, 2008 10:28 AM | Posted By : Bob Silverberg
Related Categories: OO Design, How I Use Transfer, ColdFusion, Transfer

Brain Kotek wrote a lengthy and informative comment to my last blog post. I started composing a response as a comment of my own, but it got very long, and I think that both Brian's comment and my response contribute greatly to the series, so I've turned my response into a blog post. I'm including Brian's comments in quotes throughout this post.

> Bob, great series so far! I had a couple of comments and opinions about what you're doing here that I wanted to bring up. I may create a blog entry on it if you or anyone else would like me to expand, but in brief:

Hi Brian, thanks for the comments. I think that most of your points are excellent, and will probably result in changes to the way I'm doing some things. Part of why things are the way they are is that this has been an organic process for me. When I started trying to write a service object I realized that it was pretty generic, so I started parameterizing things. I wasn't really thinking much about OO principles, rather just strictly trying to write as little code as possible.

> I'm curious about how much benefit you feel you get by having to specify the name of a specific Transfer object in your ColdSpring config. I mean, you already know that this is a "ReviewService", and you've said that you may need to interact with other Transfer objects by name, so this mixing of a dynamically supplied object name with hardcoded object names would seem to be making things more complex than they might need to be.

I do find that I get quite a bit of benefit from having a "main" transfer object defined to my service. I end up having to write a lot less code because most of the operations for that object are dealt with via the AbstractService. I could easily move that info from Coldspring into a hardcoded variable in my Concrete Service, which would eliminate the mixing to which you refer. Would you see that as an improvement?

I have also thought, as I was writing this, that maybe I should move from inheritance to composition, and inject a "Modifier" object (for lack of a better name) into my service, which would basically do the same thing (the Modifier would have the same methods as the AbstractService).

> My take is that the service layer should be less about wrapping a single Transfer object or database table and more about creating a useful external API for your applications to use.

Interesting point, so you don't see the service layer as "doing much", is that right? You see it more as a facade? It's sounding like much of what I put into the service layer you'd put into a gateway. N'est pas?

> My opinion would also be that this service is too dependent on Transfer. I try to make my services as "dumb" as possible. This ReviewService is tightly coupled to Transfer and its implementation. Even the method signature is exposing the fact that Transfer is in use with the needsClone argument, meaning stuff "outside" of the service has to "know" about Transfer and under what conditions a clone is required.
>
> I try to keep the database and Transfer-specific stuff down in a Gateway object. Gateways are meant to encapsulate interaction with an external resource, which is just what a database is. That way, if I ever need to switch out the ORM, or remove Transfer completely, I ideally only need to modify one layer of the application (the Gateways).

I definitely see your point about the service being too dependent on Transfer. As I was writing this I actually thought to myself, "Should I move the TQL stuff to a gateway?". I guess ultimately that would mean moving the guts of the Get() method from the service to the gateway, which, come to think of it, is a very interesting idea :-)

As for the needsClone argument, I'm starting to rethink that one. I thought that I needed a way to tell the service, which is the API, whether or not I need a business object that is going to be manipulated. If I word it that way (rather than referring to a clone) it no longer seems to be so tied to Transfer ;-) But that stems from the way I've implemented the get() method, which is used, in turn, by other methods. It's because of that design that I needed this needsClone argument, so maybe that is not the best design. The wheels are turning...

> Lastly, as big a fan as I am of Transfer, I'm less a fan of TQL. It looks to me like the same thing could be done with normal SQL in about half the lines. Since it looks like you prefer the TQL approach, I'd be interested to hear your thoughts on its use compared to straight SQL.

Honestly, when I started using Transfer I just figured that I'd try to use all of its features to learn it well. The only advantage that I see currently is that it provides cross database support, but even that is questionable as a simple select statement can probably be written in a way that will run on most databases. I believe that in the future Mark is considering adding caching to these types of queries, which may yield another benefit. Other than that I don't really see any reason to use it, so I may give up on it at some point.

> On a parallel note, it appears that you're enforcing some sort of security here within the service by limiting the review by the userID. Security can always be a difficult problem to tackle, but I wondered if you had considered any alternate methods of enforcing your security rules, such as a ColdSpring AOP advice?

I agree that it's always difficult to figure out how and where to enforce security. I haven't really considered many other options as in this case it seems to make sense and work here. Again, as I was writing this I did think that that rule really belonged in my Business Object, but that didn't fit in with my current design. If the rule needs to be enforced in order to return the correct business object, how can the rule exist _inside_ the business object? I guess I could start with an empty business object and then ask it to get itself from the database. Hmm, that sounds like an interesting twist.

The other issue here is that I'm not just enforcing security, I'm using logic to determine which object to return. It's not just that a user should only be able to edit their own review, it's also the fact that I'm trying to get a user's review without having the primary key of that review - just the Product and User, so I don't really see this as strictly a security concern.

I have yet to use Coldspring AOP for anything, so perhaps it would be a good candidate. My understanding of AOP is quite limited, but I thought that because this logic is specific to this one object and this one method, it wouldn't make a lot of sense to use AOP. I'd be interested to hear how you'd address this requirement with AOP.

So to summarize the main ideas I've gleaned from your comments:

1. Think about moving logic from the service into the gateway.
2. Think about moving logic from the service into business objects.

Interesting that those are strategies that I discussed in the second article. I guess sometimes it's easier to see things from an outside perspective. Thanks again for the excellent comments. I look forward to more of them.