How I Use Transfer Today - A Gateway MapFactoryBean

Posted At : November 24, 2008 4:33 PM | Posted By : Bob Silverberg
Related Categories: OO Design, How I Use Transfer, ColdFusion, Coldspring, Transfer

I left something critical out of my last post about how I'm using Transfer (an ORM for ColdFusion) these days.

If you've been following along you'll know that I'm using an Abstract Transfer Decorator, which all of my concrete decorators extend, and that I'm using Brian Kotek's most excellent Bean Injector to load singletons into my Transfer objects. This raises an interesting issue: How to inject the appropriate Gateway Object into my Transfer Object.

Let's start with an overview of how the Bean Injector works. I won't get into the nitty gritty details, but basically it does this:

1. Whenever you create a new instance of a Transfer Object, it will check the Transfer Object (i.e., decorator) for any setter method names that correspond to beans in your Coldspring config file.
2. If any are found it will invoke the setter, passing the Coldspring bean into it, thereby injecting the bean into your object.
3. Optionally, when all of that is done, it will run a user-specified method name. I always use the name setUp(). So, after the bean is injected into my Transfer Object, the setup() method will be invoked, if one exists.

The third item is necessary if you need to do some initial set up of your Transfer Object and that set up requires access to one of the beans that was injected via the Bean Injector. If this isn't entirely clear, I'll be showing a code example of this below, when I show how I implemented my solution.

So, the problem is that you need to have a setter name in your Transfer Object that corresponds to the bean name in your Coldspring config. Therefore, in my User decorator I'd need a setUserGateway() method, whereas in my Product decorator I'd need a setProductGateway() method (because my beans are called UserGateway and ProductGateway, repectively). That isn't such a big deal, but as I'm using an Abstract Decorator I'd like to be able to write one method in the abstract object which could then be inherited by all of my decorators, rather than having to hardcode the setXXXGateway() method into each decorator.

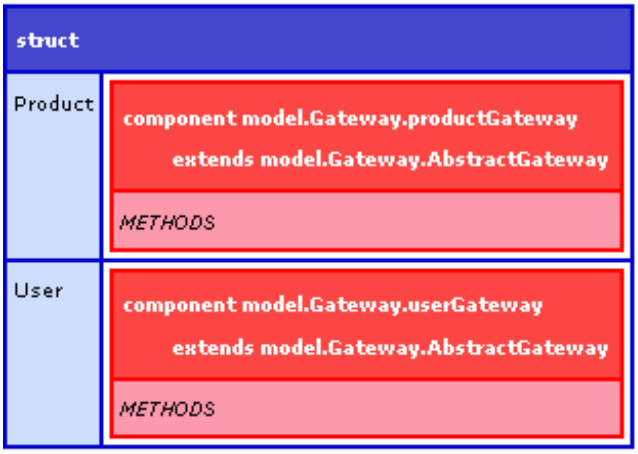
Enter Coldspring's MapFactoryBean! A MapFactoryBean allows you to specify a map in your Coldspring config, which translates into a Struct in your ColdFusion code, and then treat that map/struct as a bean, meaning that you can inject it into other components just as you would a cfc. Here's an example:

```
<bean id="GatewayMap"
 class="coldspring.beans.factory.config.MapFactoryBean">
 <property name="sourceMap">
  <map>
   <entry key="User">
    <ref bean="UserGateway" />
   </entry>
   <entry key="Product">
    <ref bean="ProductGateway" />
   </entry>
  </map>
 </property>
</bean>
```

If I then put a method in my Abstract Transfer Decorator called setGatewayMap():

```
<cffunction name="setGatewayMap" returntype="void" access="public">
 <cfargument name="GatewayMap" type="any" required="true" />
 <cfset variables.GatewayMap = arguments.GatewayMap />
</cffunction>
```

The Bean Injector will do its magic and I'll end up with a struct in variables.GatewayMap that looks like this:



So now I can create a method called setTheGateway() like this:

```
<cffunction name="setTheGateway" returntype="void" access="public">
 <cfargument name="GatewayName" type="any" required="true" />
 <cfset variables.TheGateway =
```

```
  variables.GatewayMap[arguments.GatewayName] />

</cffunction>
```

What I'm doing here is passing in a name, and getting back the corresponding Gateway Object from the map. The only issue left is how to specify the name to pass into the setTheGateway() method. There are a number of options for that. For example, you could use a private variable that you define in the configure method of your concrete decorators. Unfortunately that would somewhat defeat the purpose of having these methods in the Abstract Decorator in the first place. What I've done is followed a convention where each package in my transfer.xml file has a corresponding Gateway Object. So, to setTheGateway in my Abstract Decorator I use the setUp() method that was mentioned above:

```
<cffunction name="setUp" access="public" returntype="void">

 <cfset setTheGateway(ListFirst(getClassName(),".")) />

</cffunction>
```

You'll recall that setUp() is run automatically by the Bean Injector after all of the dependencies have been injected into the Transfer Object. I simply call setTheGateway() and pass it the package name of the Transfer Object (using ListFirst()).

And that's it - I now have the appropriate Gateway injected into each of my Transfer Objects without having to add any code into each of my concrete decorators.

I'd like to thank **Matt Quackenbush** for suggesting the use of the MapFactoryBean for this issue. It's a perfect fit.