My Take on Transfer ORM Event Model - BeforeCreate Example

Posted At : May 21, 2008 8:25 AM | Posted By : Bob Silverberg
Related Categories: ColdFusion, Coldspring, Transfer

Paul Marcotte wrote a blog entry describing how to automatically set a CreatedDate and ModifiedDate in your Transfer Objects when they are saved to the database. It is very well written and provides a great description of the problem, how Transfer observers work, and a solution.

I have used Transfer observers in a different manner, so I added a comment with a brief description to the blog entry. Dan Wilson suggested that it was too bad that such useful information was buried in a blog comment, so I decided to post it here as well. For the background, please check out Paul's entry.

The difference in my approach is that I create an observer which then watches for changes to all Transfer Objects, rather than adding the observer to a particular Transfer Object via its decorator. The code inside the BeforeCreateObserver looks like this:

```
<cffunction name="actionBeforeCreateTransferEvent" access="public" returntype="void" output="false" hint="I do stuff before an object is persisted for the first time.">
 <cfargument name="event" hint="The event object" type="transfer.com.events.TransferEvent" required="Yes" />

 <!--- Get a reference to the Object about to be persisted --->
 <cfset var theTO = arguments.event.getTransferObject() />

 <!--- Set the CreatedDate, if possible --->
 <cfif StructKeyExists(theTO,"setCreatedDate")>
  <cfset theTO.setCreatedDate(now()) />
 </cfif>

</cffunction>
```

Note that because this method will be called for all Transfer Objects just before they are created, you can also add code that only deals with specific classes as well. An updated example would be:

```
<cffunction name="actionBeforeCreateTransferEvent" access="public" returntype="void" output="false" hint="I do stuff before an object is persisted for the first time.">
 <cfargument name="event" hint="The event object" type="transfer.com.events.TransferEvent" required="Yes" />

 <!--- Get a reference to the Object about to be persisted --->
 <cfset var theTO = arguments.event.getTransferObject() />

 <!--- Set the CreatedDate, if possible --->
 <cfif StructKeyExists(theTO,"setCreatedDate")>
  <cfset theTO.setCreatedDate(now()) />
 </cfif>

 <!--- If it is a User TO, set the RegisteredDate --->
 <cfif theTO.getClassName() eq "user.user">
  <cfset theTO.setRegisteredDate(now()) />
 </cfif
</cffunction>
```

If you have a number of different classes for which you need to do different things, you could replace the second cfif with a cfswitch.

Now, the question is, how do we tell Transfer to use this BeforeCreateObserver? I like to have Coldspring do it for me. Many thanks to Brian Kotek and his TDOBeanInjectorObserver for pointing me in the right direction regarding getting Coldspring to play nice.

First we need a few entires in our Coldspring.xml file. Note that your path names may be different:

```
<bean id="transferFactory" class="transfer.TransferFactory">
 <constructor-arg name="datasourcePath"><value>model/config/datasource.xml.cfm</value></constructor-arg>
 <constructor-arg name="configPath"><value>model/config/transfer.xml.cfm</value></constructor-arg>
 <constructor-arg name="definitionPath"><value>/TransferTemp</value></constructor-arg>
</bean>

<bean id="Transfer" factory-bean="transferFactory" factory-method="getTransfer" />

<bean id="TransferBeforeCreateObserver" class="model.util.TransferBeforeCreateObserver"></bean>

<bean id="ObserverInjector" class="model.util.ObserverInjector" lazy-init="false">
 <constructor-arg name="Transfer"><ref bean="Transfer" /></constructor-arg>
 <constructor-arg name="BeforeCreateObserver"><ref bean="TransferBeforeCreateObserver" /></constructor-arg>
```

```
</bean>
```

These beans define our TransferFactory, a Transfer object (which is created via the Factory), our BeforeCreateObserver, and finally an injector for our observer. This last piece is what allows Coldspring to automatically add the Observer to the Transfer object when the application is initialized.

We've already seen the code for the TransferBeforeCreateObserver, so here's the code for the init of the ObserverInjector, which is all we really need it to do:

```
<cffunction name="init" access="public" returntype="any" hint="I am used to add the observer to Transfer.">

 <cfargument name="Transfer" type="transfer.com.Transfer" required="true" />

 <cfargument name="BeforeCreateObserver" type="any" required="true" />

 <cfset arguments.Transfer.addBeforeCreateObserver(arguments.BeforeCreateObserver) />

 <cfreturn this />

</cffunction>
```

If you wanted to add multiple observers you could add more arguments and corresponding adders (e.g., BeforeUpdateObserver). Also, if you wanted to add multiple observers to the same event, you could pass in a Coldspring map, and loop through it.