A Generic Decorator for Transfer - A Populate Method

Posted At : September 26, 2007 7:00 PM | Posted By : Bob Silverberg
Related Categories: ColdFusion, Transfer

Update: I've written a new blog post in my "How I Use Transfer" series which contains a much updated version of this method. So if you're interested in a Generic Decorator for Transfer, I suggest you check out this post, and the related posts.

I've been working with Transfer, on and off, for a few months now, and have come up with some methods that I've put in a generic decorator that I've found very useful. I thought I'd share them via my blog and get some feedback in the process.

The first one I want to discuss is a populate() method. This method is used to take data provided by a user (generally via a form post) and load that information into a Transfer Object, validating the data types along the way. This lets me avoid having to manually code a whole bunch of setXXX() statements, and also avoids the situation of trying to pass bad data into a setXXX() method. Note that this generic decorator is very much a work in progress. There are no promises that it will work perfectly as is, and I'm sure there's plenty of room for improvement. Oh, and this contains pieces of code that I've borrowed from a number of examples that I've seen on this list and elsewhere, but I didn't keep track of what came from whom, so please don't be offended if you see your code in here.

I am using a generic formbean to display data on any html form and also to accept data posted from any html form. I will go into the details of what my formbean looks like and how it works in a future blog post, but I mention it because it is material to this discussion.

So for now, just understand that when a form is posted I will have a formbean to work with, which contains the contents of the form scope, plus some other data that I automatically add to it on all requests (such as information about the current user).

Here's the complete code for the function. After that I'll walk through most of it to provide explanations:

```
<cffunction name="populate" access="public" output="false" returntype="any" hint="Populates the TO with values from a formbean">
 <cfargument name="FormBean" type="any" required="yes" />
 <cfargument name="Errors" type="any" required="yes" />
 <cfset var fn = 0 />
 <cfset var varName = 0 />
 <cfset var varValue = 0 />
 <cfset var argName = 0 />
 <cfset var argType = 0 />
 <cfset var transferClassName = 0 />
 <cfset var IsValid = true />

 <cfloop collection="#this#" item="fn">
  <cfset IsValid = true />
  <cfif Left(fn,3) EQ "set" AND ArrayLen(getMetadata(this[fn]).parameters) EQ 1>
   <cfset varName = Right(fn,Len(fn)-3) />
   <!--- get the argument name and type --->
   <cfset argName = getMetadata(this[fn]).parameters[1].name />
   <cfset argType = getMetadata(this[fn]).parameters[1].type />
   <!--- Update the LastUpdateTimestamp if found --->
   <cfif varName EQ "LastUpdateTimestamp" AND argType EQ "Date">
    <cfset setLastUpdateTimestamp(Now()) />
   <!--- Code to deal with child transfer objects --->
   <cfelseif argType EQ "transfer.com.TransferObject" AND arguments.FormBean.varExists(Right(fn,Len(fn)-3) & "Id")>
    <cfset transferClassName = GetTransferClassName(Right(fn,Len(fn)-3)) />
    <cftry>
     <cfset varValue = GetTransfer().get(transferClassName,arguments.FormBean.getVar(Right(fn,Len(fn)-3) & "Id")) />
     <cfcatch type="any">
      <cfset IsValid = false />
     </cfcatch>
    </cftry>
    <cfif NOT varValue.getIsPersisted()>
     <cfset IsValid = false />
    </cfif>
    <cfif IsValid>
     <!--- Load the transfer object into the current TO --->
     <cfinvoke component="#this#" method="#fn#">
      <cfinvokeargument name="#argName#" value="#varValue#" />
     </cfinvoke>
    <cfelse>
     <cfset ArrayAppend(arguments.Errors,"Configuration error: Trying to get a transfer object of class #transferClassName# with an Id of #arguments.FormBean.getVar(Right(fn,Len(fn)-3) & 'Id')# failed.") />
    </cfif>
   <!--- Add the value from the FormBean, if it exists --->
   <cfelseif arguments.FormBean.varExists(varName)>
    <!--- get the value from the FormBean --->
    <cfset varValue = arguments.FormBean.getVar(varName) />
```

```
   <!--- validate the datatype --->
   <cfswitch expression="#argType#">
    <cfcase value="numeric">
     <cfif NOT isNumeric(varValue)>
      <cfset ArrayAppend(arguments.Errors,"The contents of the " & varName & " field must be numeric.") />
      <cfset IsValid = false />
     </cfif>
    </cfcase>
    <cfcase value="date">
     <cfif NOT isDate(varValue)>
      <cfset ArrayAppend(arguments.Errors,"The contents of the " & varName & " field must be a valid date.") />
      <cfset IsValid = false />
     </cfif>
    </cfcase>
    <cfcase value="boolean">
     <cfif NOT isBoolean(varValue)>
      <cfset ArrayAppend(arguments.Errors,"The contents of the " & varName & " field must be a boolean value.") />
      <cfset IsValid = false />
     </cfif>
    </cfcase>
   </cfswitch>
   <cfif IsValid>
    <cfinvoke component="#this#" method="#fn#">
     <cfinvokeargument name="#argName#" value="#varValue#" />
    </cfinvoke>
   </cfif>
  </cfif>
 </cfif>
</cfloop>
 <cfreturn arguments.Errors>
</cffunction>
```

Whew, that's a bit of code. Let me explain a few parts.

First off, the function accepts two arguments, the formbean that contains the data and an array to which error messages will be added if any need to be reported back to the caller. At the end of the function that array is returned.

I'm looping through all of the functions that exist in the Transfer Object and if I find any that start with the letters "set" and accept one argument, then I treat that as a candidate for population and do some more processing:

```
<cfloop collection="#this#" item="fn">
 <cfset IsValid = true />
 <cfif Left(fn,3) EQ "set" AND ArrayLen(getMetadata(this[fn]).parameters) EQ 1>
```

I extract the name of the field from the name of the function (varName) and determine the name of the argument that the function expects as well as the type of the argument:

```
<cfset varName = Right(fn,Len(fn)-3) />
<cfset argName = getMetadata(this[fn]).parameters[1].name />
<cfset argType = getMetadata(this[fn]).parameters[1].type />
```

Next I do some processing that's specific to my application. You may find it useful to do some of your own application-specific processing in here instead of what I've coded. Most of my tables have a LastUpdateTimestamp field which I want to be updated whenever any changes are written to the database. So I'm looking for a method that matches that field, and if I find one I call it passing in Now():

```
<cfif varName EQ "LastUpdateTimestamp" AND argType EQ "Date">
 <cfset setLastUpdateTimestamp(Now()) />
```

Child Transfer Objects of the current object may have been selected via a <select> tag on the form, and therefore I may have the Id of the child Transfer Object in my form bean. If that's the case then I need to get the child Transfer Object in order to pass it into the method.

This next block of code looks for a function that expects a Transfer Object as an argument and also checks to see whether a corresponding Id field exists in the formbean. If so, it gets the corresponding transferClassName via a function that also exists within the generic decorator. That function (GetTransferClassName) maps field names to transfer classes (e.g. User=user.user, Payment=order.payment, etc.). It then attempts to get the child Transfer Object, and checks to see if a persisted Transfer Object is returned (because GetTransfer().get() may not find an existing object and therefore may just return a new object). If all is OK, it loads the child Transfer Object into the current Transfer Object. If there are any problems it reports an error back to the caller via the Errors array:

```
<cfelseif argType EQ "transfer.com.TransferObject" AND arguments.FormBean.varExists(Right(fn,Len(fn)-3) & "Id")>

 <cfset transferClassName = GetTransferClassName(Right(fn,Len(fn)-3)) />

 <cftry>

  <cfset varValue = GetTransfer().get(transferClassName,arguments.FormBean.getVar(Right(fn,Len(fn)-3) & "Id")) />

  <cfcatch type="any">

   <cfset IsValid = false />

  </cfcatch>

 </cftry>

 <cfif NOT varValue.getIsPersisted()>

  <cfset IsValid = false />

 </cfif>

 <cfif IsValid>

  <!--- Load the transfer object into the current TO --->

  <cfinvoke component="#this#" method="#fn#">

   <cfinvokeargument name="#argName#" value="#varValue#" />

  </cfinvoke>

 <cfelse>

  <cfset ArrayAppend(arguments.Errors,"Configuration error: Trying to get a transfer object of class #transferClassName# with an Id of #arguments.FormBean.getVar(Right(fn,Len(fn)-3) & 'Id')# failed.") />

 </cfif>
```

If neither of the two previous cases processed the given function, then I just need to check my formbean for a corresponding variable, and if one is found I validate the datatype. If the validation fails I send an error message back to the caller in the Errors array (yes, these messages could be friendlier), otherwise I call the method to populate the current Transfer Object with the value from the formbean:

```
<cfelseif arguments.FormBean.varExists(varName)>

 <!--- get the value from the FormBean --->

 <cfset varValue = arguments.FormBean.getVar(varName) />

 <!--- validate the datatype --->

 <cfswitch expression="#argType#">

  <cfcase value="numeric">

   <cfif NOT isNumeric(varValue)>

    <cfset ArrayAppend(arguments.Errors,"The contents of the " & varName & " field must be numeric.") />

    <cfset IsValid = false />

   </cfif>

  </cfcase>

  <cfcase value="date">

   <cfif NOT isDate(varValue)>

    <cfset ArrayAppend(arguments.Errors,"The contents of the " & varName & " field must be a valid date.") />

    <cfset IsValid = false />

   </cfif>

  </cfcase>

  <cfcase value="boolean">

   <cfif NOT isBoolean(varValue)>

    <cfset ArrayAppend(arguments.Errors,"The contents of the " & varName & " field must be a boolean value.") />

    <cfset IsValid = false />

   </cfif>

  </cfcase>

 </cfswitch>

 <cfif IsValid>

  <cfinvoke component="#this#" method="#fn#">

   <cfinvokeargument name="#argName#" value="#varValue#" />

  </cfinvoke>

 </cfif>

</cfif>
```

And that's it! So far, after quite a bit of fiddling, it seems to be working for me, which makes processing of html forms almost automatic. Plus I don't have to worry about the Transfer police busting down my door for using setMemento().

I'd be keen to hear what others think of this method, and any suggestions you may have for improving it.