Using MXUnit's injectMethod() to Reverse an injectMethod() call

Posted At : July 14, 2009 2:15 PM | Posted By : Bob Silverberg
Related Categories: MXUnit, ColdFusion

I encountered a situation today in which I wanted to reverse the effects of an injectMethod() call in an MXUnit ColdFusion unit test. Here's some code that resembles my test case:

```
<cfcomponent extends="mxunit.framework.TestCase">


<cffunction name="setUp" access="public" returntype="void">
 <cfscript>
  variables.CUT = CreateObject("component","myComponentUnderTest");
 </cfscript>
</cffunction>


<cffunction name="test1IsActuallyProperlyNamed" access="public" returntype="void">
 <cfscript>
  injectMethod(variables.CUT, this, "overrideVerifyToTrue", "verify");
  assertTrue(variables.CUT.aMethodThatWantsVerifyToReturnTrue());
 </cfscript>
</cffunction>


<cffunction name="test2IsActuallyProperlyNamed" access="public" returntype="void">
 <cfscript>
  injectMethod(variables.CUT, this, "overrideVerifyToTrue", "verify");
  assertTrue(variables.CUT.anotherMethodThatWantsVerifyToReturnTrue());
 </cfscript>
</cffunction>


<cffunction name="test3IsActuallyProperlyNamed" access="public" returntype="void">
 <cfscript>
  injectMethod(variables.CUT, this, "overrideVerifyToTrue", "verify");
  assertTrue(variables.CUT.aThirdMethodThatWantsVerifyToReturnTrue());
 </cfscript>
</cffunction>


<cffunction name="test4IsActuallyProperlyNamed" access="public" returntype="void">
 <cfscript>
  assertTrue(variables.CUT.aMethodThatWantsVerifyToBehaveAsCoded());
 </cfscript>
</cffunction>


<cffunction name="overrideVerifyToTrue" access="private" returntype="Any" hint="will be used as a test-time override with injectMethod()">
 <cfreturn true />
</cffunction>


</cfcomponent>
```

From the above we can see that the first three tests want the method called verify() in the Component Under Test (CUT) to return TRUE, so we use injectMethod() to take a fake method (overrideVerifyToTrue), and use it to replace the actual verify() method in the CUT. The last test wants the verify() method in the CUT to behave as it's coded, so it doesn't call injectMethod(). This all works, but it introduces some duplication that I'd rather not have; I have to issue the same injectMethod() call in most of the tests. To remove that duplication I'd like to be able to move the call to injectMethod() into the setup() function. But, if I move injectMethod() into the setup() function, I'd need a way to "undo" the injectMethod() call in the final test.

I took a peek at the souce code for TestCase.cfc and ComponentBlender.cfc, and saw that it would not be possible to simply "undo" an injectMethod() call, as the original method would be long gone. I thought about a patch that would allow injectMethod() to actually save a copy of the method, which could then be restored by calling a new restoreMethod() function, but that seemed like a silly thing to add for an edge case like this.

I considered other routes, such as moving all of the tests that did not want verify() overridden into a separate test case, but then realized that I could simply use injectMethod() again to inject the correct method back into the CUT by instantiating a new copy of the CUT and injecting the method from it. Like so:

```
<cfcomponent extends="mxunit.framework.TestCase">


<cffunction name="setUp" access="public" returntype="void">
 <cfscript>
  variables.CUT = CreateObject("component","myComponentUnderTest");
  injectMethod(variables.CUT, this, "overrideVerifyToTrue", "verify");

 </cfscript>

</cffunction>


<cffunction name="test1IsActuallyProperlyNamed" access="public" returntype="void">

 <cfscript>

  assertTrue(variables.CUT.aMethodThatWantsVerifyToReturnTrue());

 </cfscript>

</cffunction>


<cffunction name="test2IsActuallyProperlyNamed" access="public" returntype="void">

 <cfscript>

  assertTrue(variables.CUT.anotherMethodThatWantsVerifyToReturnTrue());

 </cfscript>

</cffunction>


<cffunction name="test3IsActuallyProperlyNamed" access="public" returntype="void">

 <cfscript>

  assertTrue(variables.CUT.aThirdMethodThatWantsVerifyToReturnTrue());

 </cfscript>

</cffunction>


<cffunction name="test4IsActuallyProperlyNamed" access="public" returntype="void">

 <cfscript>

  freshCUT = CreateObject("component","myComponentUnderTest");

  injectMethod(variables.CUT, freshCUT, "verify", "verify");

  assertTrue(variables.CUT.aMethodThatWantsVerifyToBehaveAsCoded());

 </cfscript>

</cffunction>


<cffunction name="overrideVerifyToTrue" access="private" returntype="Any" hint="will be used as a test-time override with injectMethod()">

 <cfreturn true />

</cffunction>


</cfcomponent>
```

In the above example it may look like I removed a few lines of code only to have them replaced by additional lines of code elsewhere, but in the actual test case the number of tests is far greater, so the benefit is more obvious. I also extracted the code that creates the fresh CUT and injects a method from it into the CUT into a separate method, so I don't have to repeat those lines of code for each test that wants the original verify() method.

So, nothing earth shattering here, but I thought it was interesting to find a way of using injectMethod() that I hadn't considered before.