Moron Casting... I Mean "More on Casting"

Posted At : December 21, 2007 11:17 AM | Posted By : Mark Kruger
Related Categories: Coldfusion 8

In previous posts I have lauded the "new" ability of the Javacast function to handle arrays of primitive objects. David Stamm pointed out that javacast is capable of handling complex java types as well - *and you can cast them as arrays!* In his example he used the "java.net.URL" class and created an array of them, then cast the whole thing as an array of complex java objects. I'll show you his working example in a moment, but first let's talk about why would you ever need this?

If you are working with Java libraries there are many times that they require as an argument an array of objects of a certain type. Let's say you have a payroll system written in Java and one of the functions is "runPayroll()". This function takes an array of employee objects. To keep it simple let's just say that each employee object contains properties for pay amount and vacation accrual. Internally, runPayroll() loops through the employees and calls emp.getPayAmt() and emp.getAccrual() and populates some batch job or whatever. This is fairly standard stuff in Java. If you wanted to make use of the runPayroll() function how would you accomplish it?

You might try something like this:

```
<cfset emp = ArrayNew(1)/>

<!--- create an employee --->
<cfset Bob = CreateObject("java", "com.myclass.employee").init(1800.00, 6.24)/>
<!--- append the object --->
<cfset ArrayAppend(emp, bob)/>

<!--- create an employee --->
<cfset June = CreateObject("java", "com.myclass.employee").init(2100.00, 7.1)/>
<!--- append the object --->
<cfset ArrayAppend(emp, June)/>


<!--- create a payroll object--->
<cfset payroll = CreateObject("java", "com.myclass.payroll")/>
<!--- pass in the array --->
<cfset payroll.runPayroll(emp)/>
```

You might expect this to succeed because after all you *are* giving payroll an array of employee objects. But without casting it as such Coldfusion does not setup the necessary type information - so it will error out. But amazingly you can do the following with those last few lines of code.

```
<cfset emp = javacast("com.myclass.employee[]",emp)/>
<!--- create a payroll object--->
<cfset payroll = CreateObject("java", "com.myclass.payroll")/>
<!--- pass in the array --->
<cfset payroll.runPayroll(emp)/>
```

When Run payroll receives the argument it will be an array with the "type of" employee objects.

Working Example

David provided me with a working example as well. The following code builds an array

of java.net.URL objects and passes it to the URLClassLoader.

```
<!--- put the URL object in a CF array --->
<cfset myArray = ArrayNew(1)>

<!--- create a Java URL object --->
<cfset urlObj = CreateObject("java", "java.net.URL").init("http://www.cfwebtools.com")>
<!--- append the object --->
<cfset ArrayAppend(myArray, urlObj)>
<!--- create a Java URL object --->
<cfset urlObj = CreateObject("java", "java.net.URL").init("http://www.coldfusionmuse.com")>
<!--- append the object --->
<cfset ArrayAppend(myArray, urlObj)>


<!--- cast the CF array to a Java array (of URLs) --->
<cfset myArray = JavaCast("java.net.URL[]", myArray)>

<!--- pass the Java array to the URLClassLoader constructor --->
<cfset myLoader = CreateObject("java", "java.net.URLClassLoader").init(myArray)>
<h4>Here is my Array</h4>
<cfdump var="#myArray#"/>
<h4>Here is my Loader</h4>
<cfdump var="#myLoader#"/>
<h4>Here is the same array</h4>
<cfdump var="#myLoader.getURLs()#"/>
```

You can see the result **here**. In the example the first dump you see is the array after being cast. The second dump is the class loader and the third dump is the same array returned from the class loaders "getURLs() function.

I'm still searching for a real world example that actually does something useful. In the past we have written Java wrapper classes or CFX tags to get around some of these types of limitations when working directly with Java, so I know this will definitely come in handy at some point.