Filtering Undeclared Arguments

Posted At : July 21, 2009 10:00 AM | Posted By : Steve
Related Categories: ColdFusion

One thing I have wanted to do in ColdFusion for some time is to limit the arguments scope in a function to arguments I have declared with cfargument. That feature doesn't exist, but I made the next best thing.

Here is the syntax that I want:

```
<cffunction name="testMethod" limitargs="true">
   <cfargument name="a">
   <cfargument name="b">
   <cfargument name="c">

   <cfdump var="#arguments#">

</cffunction>
```

Then I could call the function like this:

```
<cfset testMethod(a=1,b=2,c=3,d=4,e=5,f=6,g=7)>
```

And get the following dump:

A	1
B	2
C	3

In reality, I get this:

A	1
B	2
C	3
d	4
e	5
f	6
g	7

So, I made a UDF to do the next best thing:

```
<cffunction name="testMethod" limitargs="true">
   <cfargument name="a">
   <cfargument name="b">
   <cfargument name="c">

   <cfset filterArguments(testMethod,arguments)>

   <cfdump var="#arguments#">

</cffunction>

<cfset testMethod(a=1,b=2,c=3,d=4,e=5,f=6,g=7)>
```

A 1

B 2

C 3

Without the limitargs attribute:

```
<cffunction name="testMethod">
   <cfargument name="a">
   <cfargument name="b">
   <cfargument name="c">

   <cfset filterArguments(testMethod,arguments)>

   <cfdump var="#arguments#">

</cffunction>

<cfset testMethod(a=1,b=2,c=3,d=4,e=5,f=6,g=7)>
```

A 1

B 2

C 3

d 4

e 5

f 6

g 7

Note that the code takes the function itself (not the name of the function) as the first argument. If you didn't know, you can actually pass a function around in ColdFusion (and even execute the passed function).

Here is the filterArguments UDF:

```
<cffunction name="filterArguments" access="public" returntype="any" output="false">
   <cfargument name="method" type="any">
   <cfargument name="args" type="any">

   <cfset var sMethod = getMetaData(arguments.method)>
   <cfset var arglist = "">
   <cfset var ii = "">

   <!--- Only take action if the function has limitargs attribute set to true --->
   <cfif StructKeyExists(sMethod,"limitargs") AND sMethod.limitargs IS true>

      <!--- Get a list of arguments --->
      <cfif ArrayLen(sMethod.Parameters)>
         <cfloop index="ii" from="1" to="#ArrayLen(sMethod.Parameters)#">
            <cfset arglist = ListAppend(arglist,sMethod.Parameters[ii].name)>
         </cfloop>
      </cfif>

      <!--- Ditch any argument not in the list --->
      <cfloop collection="#arguments.args#" item="ii">
```

```
        <cfif NOT ListFindNoCase(arglist,ii)>
           <cfset StructDelete(arguments.args,ii)>
        </cfif>
      </cfloop>

   </cfif>

   <!--- Just in case it is called as arguments=filterArguments() --->
   <cfreturn arguments>
</cffunction>
```

The first thing this function does is get the meta data of the function passed to it:

```
<cfset var sMethod = getMetaData(arguments.method)>
```

Then it checks the value of the limitargs attribute:

```
<cfif StructKeyExists(sMethod,"limitargs") AND sMethod.limitargs IS true>
```

I used "IS true" so that it will use an empty string as false, but accept true or 1 or yes as true.

After that, I loop through the arguments in the meta data to build a list of arguments and remove any arguments that aren't in the list.

As a safety measure, I return the arguments. This is just in case the function is called as arguments=filterArguments(method,arguments). I didn't want it to mess up the arguments scope in that case.

I would still like the feature to be native, but I think this meets my needs nicely until then.