Legitimate use for Evaluate()? Do Not Want!

Posted At : December 16, 2010 3:00 PM | Posted By : Jon Hartmann
Related Categories: ColdFusion

I hate `Evaluate()`. I hate it like I hate mornings without coffee. I hate it like getting Visual Studio's Debugger to work. I hate `Evaluate()`, and thats why I'm sad to say that I think I've found a case where I can't use any of my standard tricks to avoid using `Evaluate()`, and have to give the ugly beast its moment of glory.

The Problem

I'm working on a project where I'm trying to streamline lits of little operations into single code blocks of code to make sure that I don't mess things up later. For example, I'd normally not worry about a block of code like this:

```
<cfparam name="Url.ID" default="" />


<cfif NOT IsNumeric(Url.ID)>
    <cfset Url.ID = 0 />
</cfif>
```

Its simple, straight forward, and that exact block isn't going to be used all over the place is it? Well, no, but the concept involved in those 5 lines is something I'm going to need to use a lot, and if I used `IsValid()` instead of `IsNumeric()` it means I'm going to be doing this everywhere: define a parameter, see if its value is acceptable, and reset it to the default if not. How can I break this out into something reusable?

Attempt One: Expand

If you've been paying attention to your CF documentation, you'd know that has an attribute called "type" which handles type checking for you already. Normally I use whats already available to solve something like this, but the problem is that if the param'd variable already exists, and its type doesn't match it throws an error rather that use the default value like I want it to do. In order to do what I want with the type attribute, I'd have to build it something like this:

```
<cftry>
    <cfparam name="Url.ID" default="" type="numeric" />


    <cfcatch>
        <cfset Url.ID = 0 />
    </cfcatch>
</cftry>
```

As you can see, thats no better; its more likes of code and introduces an error stack if the value's don't match. This probably isn't a significant performance hit, but I take it as a rule of thumb to never use try/catch blocks for anything other than catching unexpected or unmanageable errors. In this case, there has to be a better way.

Attempt Two: ParamDefault()

For my second attempt, I wanted to extract this to a UDF so I could use it all over the place. I built out my method stub with some comments to guide me, and then hit a snag.

```
<cffunction name="ParamDefault" access="public" output="false" returntype="void">
    <cfargument name="VariableName" type="string" required="true" />
    <cfargument name="DefaultValue" type="string" required="false" default="" />
    <cfargument name="Type" type="string" required="false" />


    <!--- Define the variable and set the default value if it doesn't exist --->
    <cfparam name="#Arguments.VariableName#" default="#Arguments.DefaultValue#" />
```

```
    <!--- See if a type is set for this default --->
    <cfif StructKeyExists(Arguments, "Type")>
        <!--- See if the value isn't of the type --->
        <cfif NOT IsValid(Arguments.Type, ?)>
            <!--- Set the value to the type --->
            <cfset "#Arguments.VariableName#" = Arguments.DefaultValue />
        </cfif>
    </cfif>

    <cfreturn />
</cffunction>
```

I'd solved how to set the value back using the "left side quotation" feature of ColdFusion, but I was stumped as to how to get the value of the named variable: it didn't exist in any scopes I could get at to use structure notation... and thats when I remembered `Evaluate()`.

I couldn't bring myself to do it for about 15 minutes; I kept looking for any other way to make this UDF work withing using `Evaluate()`.

The Solution

My final version looks like this:

```
<cffunction name="ParamDefault" access="public" output="false" returntype="void">
    <cfargument name="VariableName" type="string" required="true" />
    <cfargument name="DefaultValue" type="string" required="false" default="" />
    <cfargument name="Type" type="string" required="false" />

    <!--- Define the variable and set the default value if it doesn't exist --->
    <cfparam name="#Arguments.VariableName#" default="#Arguments.DefaultValue#" />

    <!--- See if a type is set for this default --->
    <cfif StructKeyExists(Arguments, "Type")>
        <!--- See if the value isn't of the type --->
        <cfif NOT IsValid(Arguments.Type, Evaluate(Arguments.VariableName))>
            <!--- Set the value to the type --->
            <cfset "#Arguments.VariableName#" = Arguments.DefaultValue />
        </cfif>
    </cfif>

    <cfreturn />
</cffunction>
```

I feel unclean. I post this in hopes that one of you out there has an idea of how to do this without `Evaluate()`... is there a scope I can check or something? Is there anything that can get rid of that `Evaluate()` statement?