Unexpected Structure Behavior with {}

Posted At : April 7, 2009 8:32 AM | Posted By : Jon Hartmann
Related Categories: ColdFusion

Yesterday I ran into an interesting issue with one of my applications that lead to an even more interesting discovery about how CF8 executes the {} notation for creating structures.

The Problem

The issue that I ran into is that in my application's startup and reinitialization function, I had a declaration like this:

```
<cfset application.config = {
    email = "myemail@mycompany.com",
    title = "My Application Title"
} />
```

I setup and got running with the standard error handlers to email me issues, and everything seemed fine. A few days later, I get an error email letting me know that "element email is not defined in application.config"... Well, ok, I forgot to use a lock around my read, so thats my fault, but the more I thought about it *why would email ever be undefined?* Wouldn't CF look at the {} and evaluate it as a block?

The Tests

In order to find out how this was happening, I needed to setup some tests cases. First thing was to "slow things down" so that I could see how ColdFusion was evaluating the structure declaration. To do this I setup a test without the new notation:

```
<!--- Application.cfc --->
<cfcomponent output="false">
    <cfset this.name = "AppLockTest" />

    <cffunction name="onApplicationStart">
        <cfset onReinitialization() />
    </cffunction>

    <cffunction name="onRequestStart">

        <cfparam name="url.reinit" default="0" />

        <cfif url.reinit>
            <cflock scope="application" timeout="5" type="exclusive">
                <cfset onReinitialization() />
            </cflock>
        </cfif>

    </cffunction>

    <cffunction name="onReinitialization">

        <cfset application.struct = StructNew() />
        <cfset sleep(2000) />
        <cfset application.struct.server = RandRange(10,100) />
        <cfset sleep(2000) />
        <cfset application.struct.datasource = RandRange(10,100)/>
        <cfset sleep(2000) />
```

```
        <cfset application.struct.users = RandRange(10,100) />

    </cffunction>

</cfcomponent>
```

And I setup a basic test page in the index.cfm:

```
<!--- index.cfm --->
<cfdump var="#application#">
```

If you open up a browser to this app, it will take about 6 seconds to initialize the first time, and if you open up another browser you can tell one two reinit and then refresh the other to watch as each part gets added to the structure. Thats neat, but we already expect this behavior. Lets update our application to something more like the original setup:

```
<!--- Application.cfc 2 --->
<cfcomponent output="false">
    <cfset this.name = "AppLockTest2" />

    <cffunction name="onApplicationStart">
        <cfset onReinitialization() />
    </cffunction>

    <cffunction name="onRequestStart">

        <cfparam name="url.reinit" default="0" />

        <cfif url.reinit>
            <cflock scope="application" timeout="5" type="exclusive">
                <cfset onReinitialization() />
            </cflock>
        </cfif>

    </cffunction>

    <cffunction name="onReinitialization">

        <cfset application.struct = {
            server         = setupValue(),
            datasource     = setupValue(),
            users         = setupValue()
        } />

    </cffunction>

    <cffunction name="setupValue">
        <cfset sleep(2000) />
        <cfreturn RandRange(10,100) />
    </cffunction>

</cfcomponent>
```

I couldn't in-line the sleep() calls with this example, so I pulled the value return into a separate function to net the same result. If you open to browsers and try this again, you'll see something interesting... it behaves exactly like the first example. I'd expect it to execute as a block level declaration, similar to if I had done the following:

```
<cfcomponent output="false">
    <cfset this.name = "AppLockTest4" />

    <cffunction name="onApplicationStart">
```

```
        <cfset onReinitialization() />
    </cffunction>

    <cffunction name="onRequestStart">

        <cfparam name="url.reinit" default="0" />

        <cfif url.reinit>
            <cflock scope="application" timeout="5" type="exclusive">
                <cfset onReinitialization() />
            </cflock>
        </cfif>

    </cffunction>

    <cffunction name="onReinitialization">

        <cfset application.struct = createStruct(
            server         = setupValue(),
            datasource     = setupValue(),
            users          = setupValue()
        ) />

    </cffunction>

    <cffunction name="setupValue">
        <cfset sleep(2000) />
        <cfreturn RandRange(10,100) />
    </cffunction>

    <cffunction name="createStruct">
        <cfset var x = StructNew() />
        <cfloop collection="#arguments#" item="arg">
            <cfset x[arg] = arguments[arg] />
        </cfloop>
        <cfreturn x />
    </cffunction>

</cfcomponent>
```

In this example I've faked a "block" level evaluation of a struct declaration with createStruct().

The Conclusion

The only conclusion I can draw from this is that ColdFusion does not execute {} declarations as block level elements (and probably doesn't do [] as blocks either). I would not have expected this behavior... is this something you would have expected?