Testing Rules in a Neptune program.

Posted At : January 24, 2012 10:45 AM | Posted By : Steve
Related Categories: Neptune, Testing

I realize I haven't blogged about **Neptune** for a while, but I actually have been making progress on it. I have just been finding a hard time making time to blog. Hopefully I will get better about that.

What I want to cover today is the process of writing automated tests in Neptune to which you can then write your code.

As a caveat, I don't typically write "unit tests", preferring instead to write "business rule tests". The distinction between the two and my reasoning for my preference will hopefully be the topic of a future blog post.

For this example, I have a "Domains" program with the following XML:

```
<tables prefix="main">
   <table entity="Domain" />
</tables>
```

The **documentation** should explain what this XML does. In short, however, it creates a table called "mainDomains" with an incrementing integer primary key of "DomainID" and a string field named "DomainName".

It has the following rules:

- Any saved DomainName should consist of just the domain name and not any protocol or folder information.
- Any URL passed to convertURL should be converted to a relative URL (starting with '/') if it contains a domain name in the domains table (with or without 'www').

So, I will create a "tests" folder in my Domains program and put a "TestDomains.cfc" file in it:

```
<cfcomponent displayname="Domains" extends="com.sebtools.RecordsTester">

<cffunction name="setUp" access="public" returntype="void" output="no">

   <cfset Super.setUp()>
   <cfset loadExternalVars("Domains")>

</cffunction>

<cffunction name="shouldDomainNameByDomainOnly" access="public" output="no"
   hint="Any saved DomainName should consist of just the domain name and not any protocol or folder information.">

   <cfset fail("This test has not been implemented yet.")>

</cffunction>

<cffunction name="shouldConvertURLMakeURLsForSavedDomainsRelative" access="public" output="no"
   hint="Any URL passed to convertURL should be converted to a relative URL (starting with '/') if it contains a domain
name in the domains table (with or without 'www').">

   <cfset fail("This test has not been implemented yet.")>

</cffunction>
```

```
</cfcomponent>
```

Let's take a look.

```
<cfcomponent displayname="Domains" extends="com.sebtools.RecordsTester">
```

This code defines the displayname for the test component. This will be important if we decide to use the **Neptune RulesMgr** as a UI for our tests (which certainly isn't a requirement). We also set the component to extend "com.sebtools.RecordsTester" which itself extends an included build of **MXUnit**, but adds some helpful functionality that we might want to use.

```
<cffunction name="setUp" access="public" returntype="void" output="no">

   <cfset Super.setUp()>
   <cfset loadExternalVars("Domains")>

</cffunction>
```

The set up code simply makes the "Domains" program available in "Variables.Domains".

```
<cffunction name="shouldDomainNameByDomainOnly" access="public" output="no"
   hint="Any saved DomainName should consist of just the domain name and not any protocol or folder information.">

   <cfset fail("This test has not been implemented yet.")>

</cffunction>

<cffunction name="shouldConvertURLMakeURLsForSavedDomainsRelative" access="public" output="no"
   hint="Any URL passed to convertURL should be converted to a relative URL (starting with '/') if it contains a domain
name in the domains table (with or without 'www').">

   <cfset fail("This test has not been implemented yet.")>

</cffunction>
```

These are just test stubs. I could have these **automatically generated** or I could manually write them and then have the actual test code written by someone else. Regardless, I always write the stubs and then write the tests. This is true even if I write both the stubs and the tests. It just helps me to make sure I don't forget any tests.

Now I need to write the tests themselves.

So, for this rule: "**Any saved DomainName should consist of just the domain name and not any protocol or folder information.**", how can I test that? My first thought is to pass in values like "http://www.example.com/folder/file.html" and "https://www.example.com/" and make sure I get back "www.example.com" for both. I should probably also test for a different protocol and no protocol and at (with and without folders).

```
<cffunction name="shouldDomainNameByDomainOnly" access="public" output="no"
   hint="Any saved DomainName should consist of just the domain name and not any protocol or folder information.">

   <cfset var sDomain1 = Variables.Domains.validateDomain(DomainName="http://www.example.com/folder/file.html")>
   <cfset var sDomain2 = Variables.Domains.validateDomain(DomainName="https://www.example.com/")>
   <cfset var sDomain3 = Variables.Domains.validateDomain(DomainName="ftp://www.example.com/")>
   <cfset var sDomain4 = Variables.Domains.validateDomain(DomainName="www.example.com")>
   <cfset var sDomain5 = Variables.Domains.validateDomain(DomainName="www.example.com/folder/file.html")>
```

```
   <cfset assertEquals("www.example.com",sDomain1.DomainName,"The domain name was not limited to just the domain name,
without protocol or folder information.")>
   <cfset assertEquals("www.example.com",sDomain2.DomainName,"The domain name was not limited to just the domain name,
without protocol or folder information.")>
   <cfset assertEquals("www.example.com",sDomain3.DomainName,"The domain name was not limited to just the domain name,
without protocol or folder information.")>
   <cfset assertEquals("www.example.com",sDomain4.DomainName,"The domain name was not limited to just the domain name,
without protocol or folder information.")>
   <cfset assertEquals("www.example.com",sDomain5.DomainName,"The domain name was not limited to just the domain name,
without protocol or folder information.")>

</cffunction>
```

Here I am simply calling the "validateDomain" method which I know Records.cfc will create automatically and where this sort of work should be done (see **Neptune validation**).

For this rule: "**Any URL passed to convertURL should be converted to a relative URL (starting with '/') if it contains a domain name in the domains table (with or without 'www').**"

I know from this rule that I need a "convertURL" that will take a string. I am going to write my test assuming that this method exists. If it doesn't exist when I run the test then I will get an exception, but that is OK. After all, my tests should fail to start.

This test requires data in the table, so I may need to actually save data to the database. As such, I will take advantage of functionality built into the build of MXUnit that ships with Neptune and add a mxunit:transaction="rollback" attribute on the test to make sure that any data changes get rolled back after the test completes.

```
<cffunction name="shouldConvertURLMakeURLsForSavedDomainsRelative" access="public" output="no"
   mxunit:transaction="rollback"
   hint="Any URL passed to convertURL should be converted to a relative URL (starting with '/') if it contains a domain
name in the domains table (with or without 'www').">

   <cfset var qDomain1 = getTestRecord(Variables.Domains,StructFromArgs(DomainName="www.example.com"))>
   <cfset var qDomain1 = getTestRecord(Variables.Domains,StructFromArgs(DomainName="example.org"))>
   <cfset var uid = CreateUUID()>

   <cfset assertEquals("/folder/file.html",Variables.Domains.convertURL("http://www.example.com/folder/file.html"),"A
relative URL was not returned from 'convertURL' for a domain saved in the system.")>
   <cfset assertEquals("/folder/file.html",Variables.Domains.convertURL("http://example.com/folder/file.html"),"A relative URL
was not returned from 'convertURL' for a domain saved in the system.")>
   <cfset assertEquals("/folder/file.html",Variables.Domains.convertURL("http://www.example.org/folder/file.html"),"A relative
URL was not returned from 'convertURL' for a domain saved in the system.")>
   <cfset assertEquals("/folder/file.html",Variables.Domains.convertURL("http://example.org/folder/file.html"),"A relative URL
was not returned from 'convertURL' for a domain saved in the system.")>
   <cfset
assertEquals("http://www.#uid#.net/folder/file.html",Variables.Domains.convertURL("http://www.#uid#.net/folder/file.html"),"An
absolute URL was not returned from 'convertURL' for a domain not saved in the system.")>

</cffunction>
```

The gist of this code is that it tests relative domain name creation with and without the "www" in the domain name and ensures an absolute URL for a domain not in the system.

A few potentially unfamiliar pieces deserve explanation.

```
StructFromArgs(DomainName="www.example.com")
```

This is equivalent to the following code, but for versions of ColdFusion that do not support that syntax:

```
{DomainName="www.example.com"}
```

The "getTestRecord" method uses the "saveTestRecord" method internally. The "saveTestRecord" calls the appropriate save method on the component passed in and populates all of the fields with appropriate dummy data except those fields specified in the structure. In this case, we didn't need any fields populated other than "DomainName", so this is a wasted effort. I include it just in case anyone needs it later. The difference between "saveTestRecord" and "getTestRecord" is that "saveTestRecord" returns the primary key value of the created record and "getTestRecord" returns a query holding the created record.

Here is the completed TestDomains.cfc:

```
<cfcomponent displayname="Domains" extends="com.sebtools.RecordsTester">

<cffunction name="setUp" access="public" returntype="void" output="no">

	<cfset Super.setUp()>
	<cfset loadExternalVars("Domains")>

</cffunction>

<cffunction name="shouldDomainNameByDomainOnly" access="public" output="no"
	hint="Any saved DomainName should consist of just the domain name and not any protocol or folder information.">

	<cfset var sDomain1 = Variables.Domains.validateDomain(DomainName="http://www.example.com/folder/file.html")>
	<cfset var sDomain2 = Variables.Domains.validateDomain(DomainName="https://www.example.com/")>
	<cfset var sDomain3 = Variables.Domains.validateDomain(DomainName="ftp://www.example.com/")>
	<cfset var sDomain4 = Variables.Domains.validateDomain(DomainName="www.example.com")>
	<cfset var sDomain5 = Variables.Domains.validateDomain(DomainName="www.example.com/folder/file.html")>

	<cfset assertEquals("www.example.com",sDomain1.DomainName,"The domain name was not limited to just the domain name,
without protocol or folder information.")>
	<cfset assertEquals("www.example.com",sDomain2.DomainName,"The domain name was not limited to just the domain name,
without protocol or folder information.")>
	<cfset assertEquals("www.example.com",sDomain3.DomainName,"The domain name was not limited to just the domain name,
without protocol or folder information.")>
	<cfset assertEquals("www.example.com",sDomain4.DomainName,"The domain name was not limited to just the domain name,
without protocol or folder information.")>
	<cfset assertEquals("www.example.com",sDomain5.DomainName,"The domain name was not limited to just the domain name,
without protocol or folder information.")>

</cffunction>

<cffunction name="shouldConvertURLMakeURLsForSavedDomainsRelative" access="public" output="no"
	mxunit:transaction="rollback"
	hint="Any URL passed to convertURL should be converted to a relative URL (starting with '/') if it contains a domain
name in the domains table (with or without 'www').">

	<cfset var qDomain1 = getTestRecord(Variables.Domains,StructFromArgs(DomainName="www.example.com"))>
	<cfset var qDomain1 = getTestRecord(Variables.Domains,StructFromArgs(DomainName="example.org"))>
	<cfset var uid = CreateUUID()>

	<cfset assertEquals("/folder/file.html",Variables.Domains.convertURL("http://www.example.com/folder/file.html"),"A
relative URL was not returned from 'convertURL' for a domain saved in the system.")>
	<cfset assertEquals("/folder/file.html",Variables.Domains.convertURL("http://example.com/folder/file.html"),"A relative URL
was not returned from 'convertURL' for a domain saved in the system.")>
	<cfset assertEquals("/folder/file.html",Variables.Domains.convertURL("http://www.example.org/folder/file.html"),"A relative
```

```
URL was not returned from 'convertURL' for a domain saved in the system.")>
   <cfset assertEquals("/folder/file.html",Variables.Domains.convertURL("http://example.org/folder/file.html"),"A relative URL
was not returned from 'convertURL' for a domain saved in the system.")>
   <cfset
assertEquals("http://www.#uid#.net/folder/file.html",Variables.Domains.convertURL("http://www.#uid#.net/folder/file.html"),"An
absolute URL was not returned from 'convertURL' for a domain not saved in the system.")>

</cffunction>

</cfcomponent>
```

So, now all of our tests are written and we can proceed to write our code agains these tests.

Neptune is open source and free for any use.