FusionLink Labs

Setting up CFEclipse with Subversion (SVN)
including Continuous Testing with CFCUnit and Selenium
by John Mason, mason@fusionlink.com

This is a quick and dirty run through in how to setup CFEclipse, Subversion, CFCUnit testing and Selenium testing into your development environment. It also explains how to automate your testing with ANT. We're not going to explain why with this article. If you are reading it, you probably have some interest in these tools already. We are simply showing how it can be setup.

Some basic assumptions about your current environment:

- Firefox or some other browser is installed
- Coldfusion is installed and running
- IIS, Apache or the ColdFusion web server is installed and running

You will need these items running before proceeding with this article.

Setting up CFEclipse

Download the latest Eclipse from
http://www.eclipse.org/downloads/

Eclipse Classic is fine. The other options are specialize for Java, C++ developers, etc. We just need to plain vanilla version.

Extract the Zip file to your Program Files directory.So for example, C:\Program Files\Eclipse

Make any shortcuts you need from your Desktop to point C:\Program Files\Eclipse\eclipse.exe

Go ahead and open up Eclipse for the first time. It comes with its own Java JVM so it should fire up without any further setup or configuration.

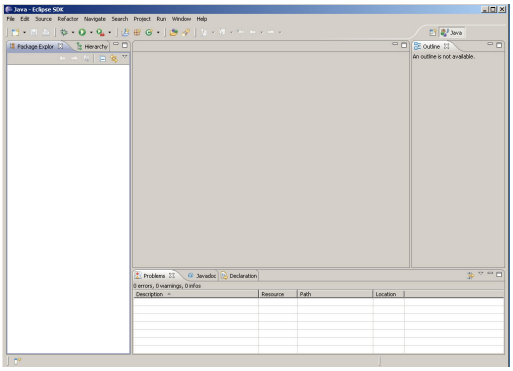
It will ask you for your workspace location, typically on a Windows box that has IIS, this would be the same location as your default web directory.

- C:\Inetpub\wwwroot

Go ahead and check the box that asks to use this as the default and click Ok.

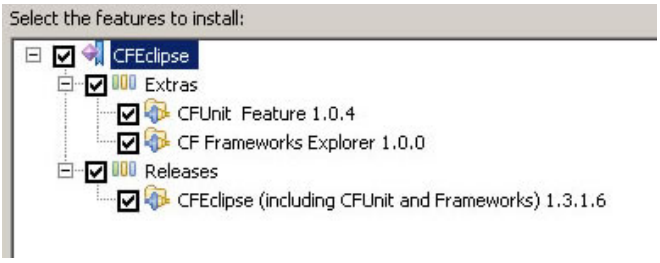
You should next see the default welcome screen for Eclipse. It may take a second for this to load up for the first time, be patient.

Go ahead and close out the welcome page. This will bring you to the default Eclipse perspective which happens to be Java.



Eclipse is organized in workspace perspectives. Each lanaguage has its unique workspace perspective with the tools and features that you will need. Eclipse does not provide, by default, a perspective for ColdFusion. So we will need to download and install the CFEclipse plug-in.

We can easily install CFElcipse from within Eclipse. The following instructions come straight from CFEclipse.

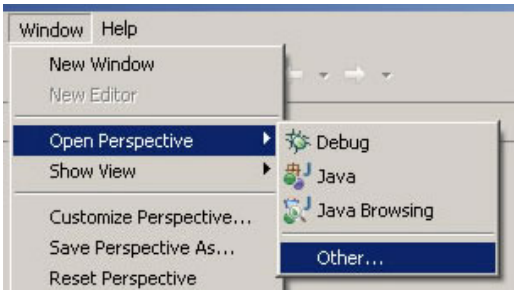


Direct quote from "http://www.cfeclipse.org"The best way to get CFEclipse is to use the built-in installation and update management tools that come with Eclipse. Here's how:

- 1. Select the "Help->Software Updates->Find and install" menu option
- 2. On the screen that is displayed, select 'Search for new features to install' and click the 'Next' button
- 3. Now click the 'New Remote Site' button
- 4. Enter a name for the update site, for example "CFEclipse". In the URL box, enter "http://www.cfeclipse.org/update" and click the OK button
- 5. You should now have a 'CFEclipse' node in the 'Sites to include in search' box.
- 6. Tick the 'Stable CFEclipse' box and click 'Next'
- 7. Eclipse will contact the CFEclipse site and retrieve the list of available plug-ins. Tick the plug-in with the highest version number and click 'Next'
- 8. The next instructions are self-explanatory, agree to the license, install the software and restart Eclipse

Once Eclipse has restarted CFEclipse should have been successfully installed

When you first bring Eclispe back up, it will default to the Java workspace perspective. Simply click Window -> Open Perspective -> Other



Choose CFEclipse and click Ok, this should bring you into the CFEclipse perspective.
Welcome to the CFEclipse!

Feel free to play ahead a bit here; you can create new projects on the left panel by right clicking the panel and choosing New then Project. Once a project has been created, you can start creating CFML and CFC pages.

Subversion

You now have the CFEclipse up and running, let us bring source control into the picture. We are not going to review what Subversion is here. The assumption is that you know about it and what it does. Just as a further note, Subversion and SVN are one in the same here.

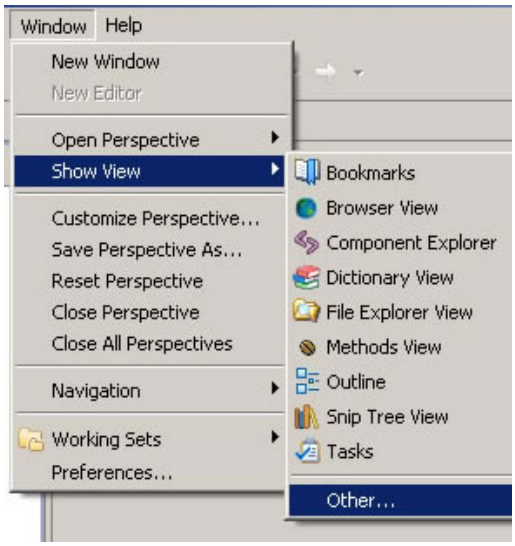
We will be logging into an Subversion service, if you do not have one. There are several companies providing them now including Google. FusionLink also offers a free 500MB Subversion repository for all of their clients. If you prefer to have your own, check out the VMware appliance center for a prebuilt Subversion servers. Many come bundled with bug trackers and Cruise Control.

There are several great Subversion plug-ins for Eclipse, we will be using Subclipse (http://subclipse.tigris.org/) in this example.

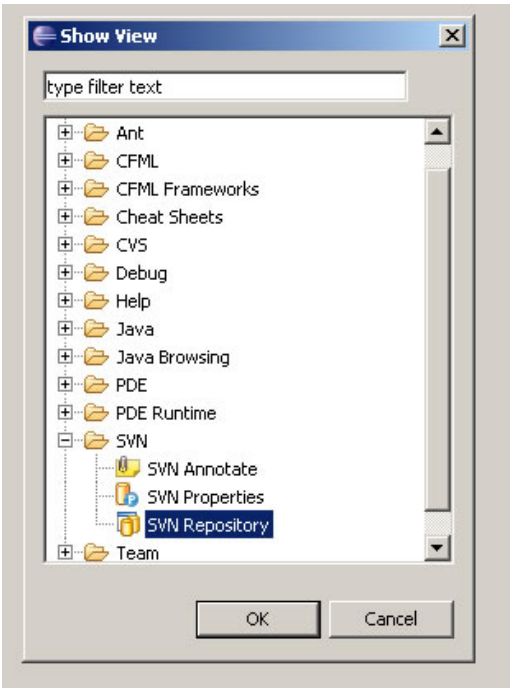
Subclipse install instructions: http://subclipse.tigris.org/install.html. Naturally being another Eclipse plug-in the installation is very similar to CFEclipse.

After installing Subclipse, restart your Eclipse IDE. It should bring you back into the workspace perspective that you were last in, which in our case should be CFEclipse.

We now need to add the SVN Repository panel to our workspace. Simply go to Window -> Show View -> Other

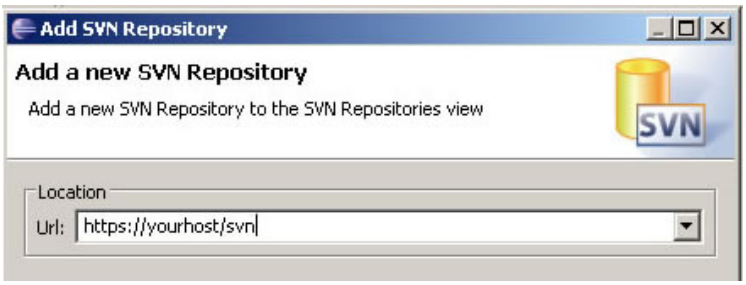


Open up the SVN Folder and pick the SVN Repository.



The SVN Repository folder should appear somewhere in your workspace area. You can drag it (or any panel for that matter) to the location of your choosing. I like to place it on the left hand side. To drag a panel, click and hold the panel tab then drag it where you like.

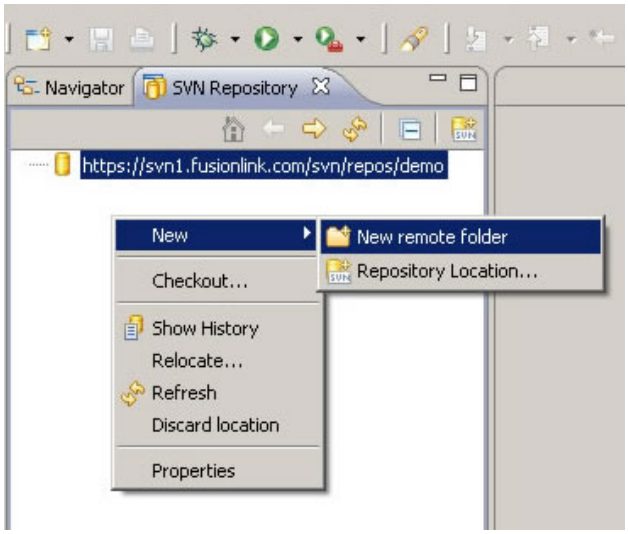
Now, click on the "Add SVN Respository" button and enter in the Url location of your SVN server.



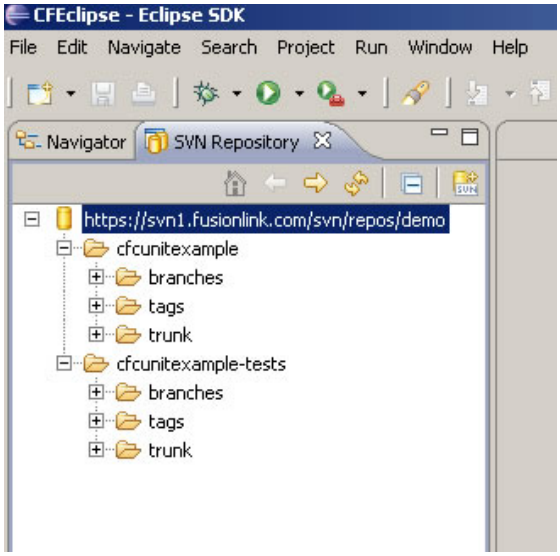
You will be next asked for your username and password. Also choose to save your password and hit Ok.

You should now be logged into your SVN respository but unless you have created anything before it should be empty. Don't worry; we just got to setup some folders to organize our projects.

Right click anywhere in your SVN respository panel and choose New, then New remote folder. Create a new folder called "cfcunitexample"



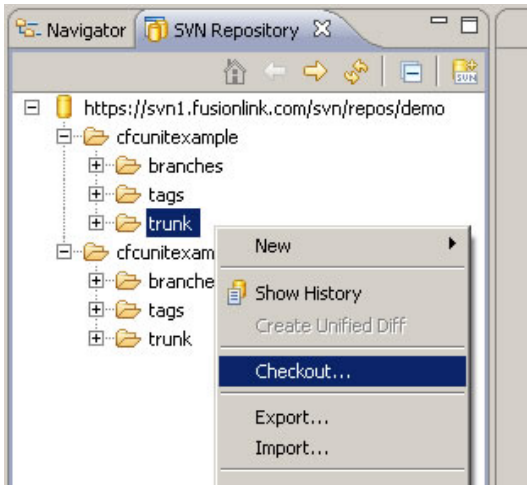
We need to create several folders to build out the structure we need for our first project and its Unit Tests. Just build out your folders like so..



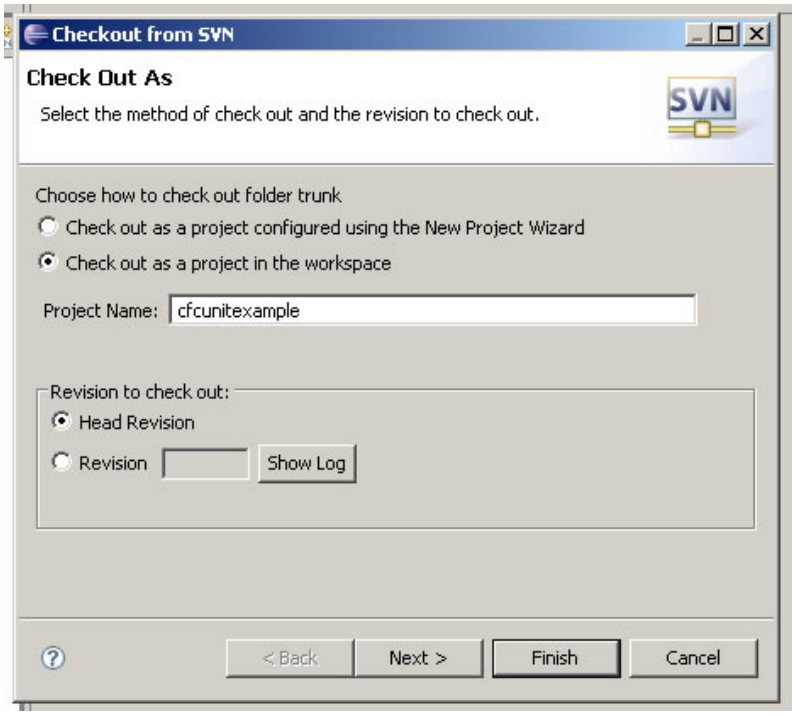
We are not going to review the purpose of the branches,tags, and trunk folders here. There are several resources that explain the purpose of these. In this example, we will be developing directly off the trunk.

> Note:
> Now, for some of the more advance developers will be asking why create a separate SVN area for the unit tests. It is just a pattern to organize this. The unit tests also need source control but may follow a different pattern as far as tagging and branching is concern. You will want to do tagging for your unit tests in order to do regression testing of your project. So to keep things completely flexible we put them in their own area. Special thanks to Scott Talsma of Echo11 for suggesting this to me.

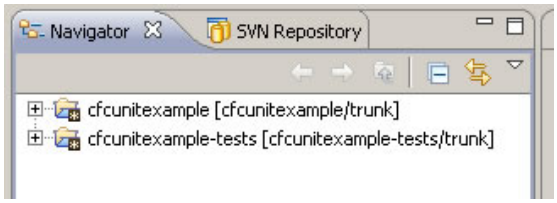
Next, checkout the trunks for both the cfcunitexample and the cfcunitexample-tests projects. Just right click on the trunk and choose Checkout.



The next panel will ask a couple of basic questions. I prefer to dictate the exact Project Name and the use the Head Revision. In this case, I said cfcunitexample and cfcunitexample-tests. Click next, and choose keep your default workspace location. Finally, click Finish.

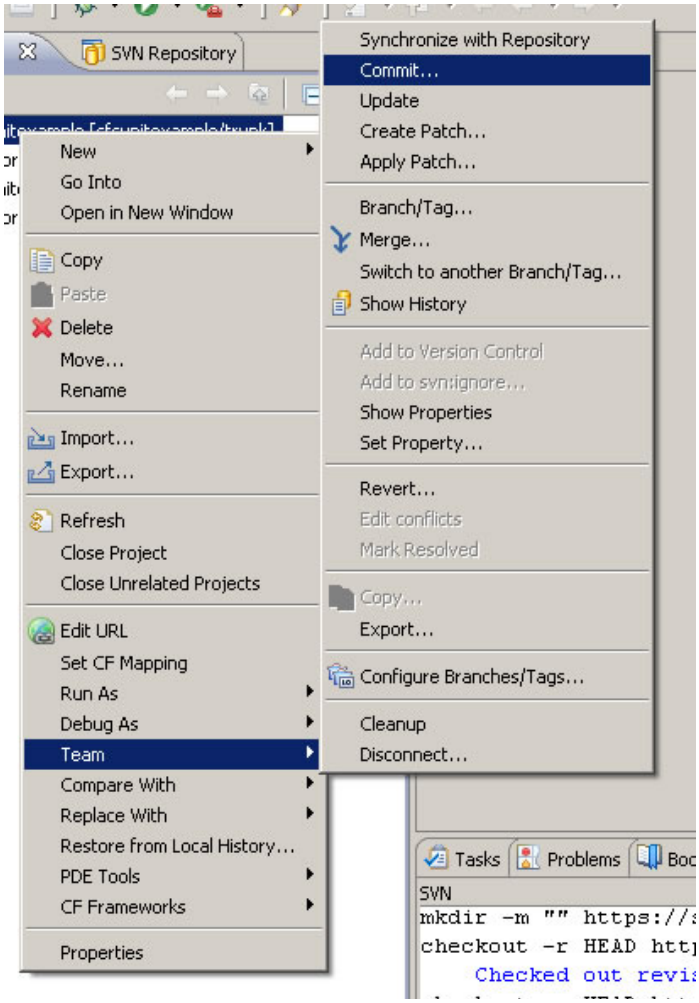


Click the Navigator Panel and you see two new projects listed.



We have checkout of Subversion these two projects to our local workspace. We can now develop locally just like we normally would. When we want to save our work to the repository, simply right click on the project and go to Team and then Commit. This will save changes to the repository and provide us with version control. Even as a single developer you will like Subversion because it will provide organization, backup and undo features for your work. If you are working with a development team, the merge, switch and branch commands tend to be used more and will help integrate your work as a group. There are several other Subversion commands

available but that is a quick 5 minute intro into it.

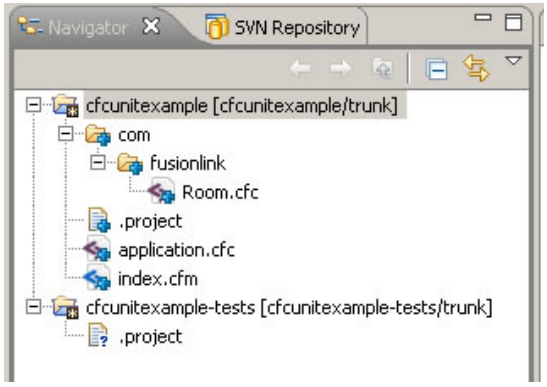


We are about half way through our setup. You have CFEclipse installed and you have two projects checkout into your local workspace and you can start developing locally from CFEclipse.

Download our cfcunitexample.zip file and unzip the files to C:\Inetpub\wwwroot\cfcunitexample

You may need to tell subversion to include some folders/files when it runs commits. Any folders/files that have a "?" mark, simply right click on them and choose Team, then Add to Version Control.

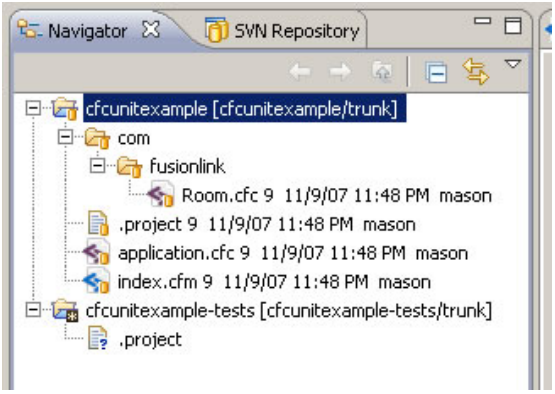
When everything looks right, your files should look something like this with the plus signs.



Time to do our first Commit to Subversion. Right click on the project and choose Team, then Commit. Make sure everything is

selected and hit Ok. The console area should show the files being uploading to your repository and report back your revision number.

Your navigator will note the change, files and folder icons will change. You will also see a revision number, date/time stamp and who committed them to SVN next to each file and folder.



Great, you have your CFEclipse installed. You are doing local development and committing your code to Subversion. Feel free to jump up and down in joy :)

Most developers stop here and are happy with just the IDE and source code. There is nothing wrong with that, but we naturally want to go to the next level and start implementing unit testing, usability testing and then continuous testing in our development. Its not hard and we are almost there.

Unit Testing

ColdFusion has two commonly used unit tests: CFUnit and CFCUnit. They are very similar and work great. CFEclipse actually comes with CFUnit. We are going to run through the process of installing CFCUnit since I prefer it.

First, make certain the cfcunitexample application works correctly. Go to..

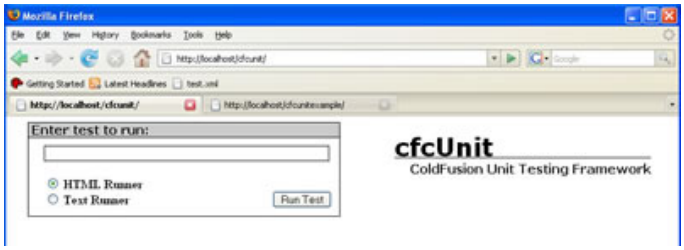
> http://localhost/cfcunitexample/

You should see a simple room registration system where you can add or remove students from a room. Very simple.

We now need to install CFCUnit. Go to cfcunit.org and download the zip file. Extract the contents into your C:\Inetpub\wwwroot like so..

> /org/cfcunit/ - core cfcUnit library components
> /cfcunit/ - Application files

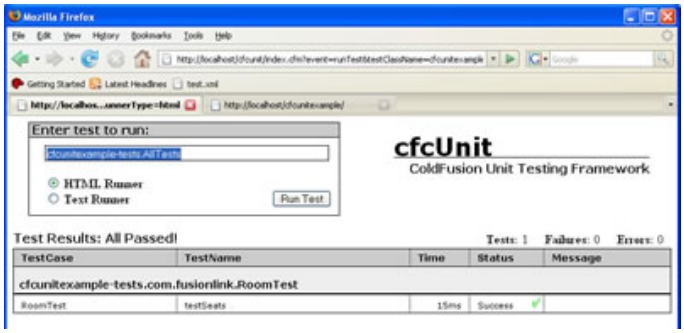
That is it. Now, in your browser, go to http://localhost/cfcunit



Now we do not have our unit tests in place yet, but to give CFCUnit a test run point it to its own unit tests. Type in "cfcunit.tests.AllTests" and hit Run Test. It will take several seconds to run and show report back to you with 144 tests that all passed.

Now, you can unzip the cfcunitexample-tests.zip folder into C:\Inetpub\wwwroot\cfcunitexample-tests folder. Do the same procedure to add these files to Subversion.

Next, try to run the unit tests. Go back to http://localhost/cfcunit and type in "cfcunitexample-tests.AllTests" and you should see one test pass.



You are now able to manually run unit tests against you code, but trust me you will want to automate this.

ANT

The best way to automate things like this is with ANT. Luckily, Eclipse has ANT already bundled with it. So with a Windows machine, we will want to add the ANT directory to our computer's PATH variable to make it easier to use. Go to your desktop, right click on "My Computer", go to properties, then click the Advanced tab and then the Environment Variables button. Under system variable, double click Path and add a semicolon ";" then add your ANT directory path. In my case this was, "C:\Program Files\eclipse \plug-ins\org.apache.ant_1.6.5\bin". Your ANT path may be slightly different just search through your Eclipse plug-ins directory. When you have added that in, reboot your computer.

ANT is really simple. We will first give it a test run via command line. Then we will implement it with our CFEclipse. First, in CFEclipse go to the cfcunitexample project and open up the build.properties file and see if any of those settings need to be modified. When you first open it, you may see a Build Configurations page. Just ignore that, and click on "build properties" on the lower left hand side to directly view the page. Remember to save any changes you make.

Now, go to command line and go to the C:\Inetpub\wwwroot\cfcunitexample directory and simply type in "ant" then hit enter.

Warning:
Did you get a SAXParseException error? If so, this is due to a small bug in the coding for the XMLService.cfm page that CFCUnit gives you. If you have ColdFusion debugging on, it will confuse ANT. Two ways to fix this: turn debugging off; or the better option,modify some of that CFCUnit code.

Go to your /cfcunit directory and open up the XmlService.cfm page. Right below the commented section. Type in the following..

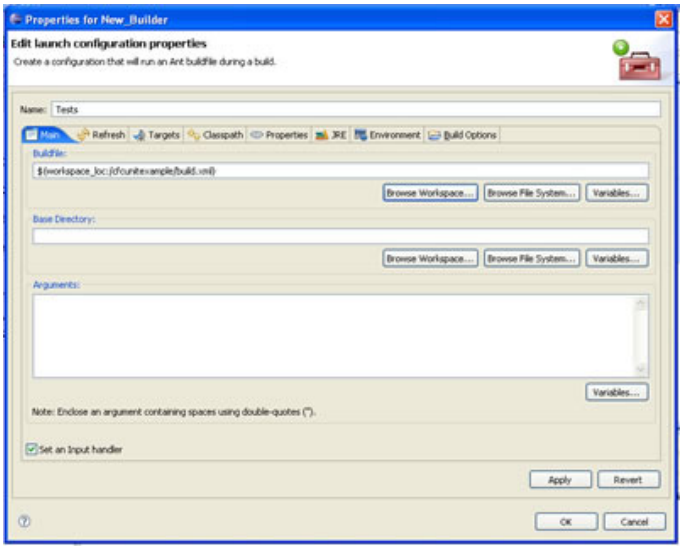
<cfsetting enablecfoutputonly="true" showdebugoutput="false">

Save and try running ANT again. You should see a successful run of the unit test.

```
C:\Inetpub\wwwroot\cfcunitexample>ant
Buildfile: build.xml

unittest:
  [cfcunit] URL: http://localhost/cfcunit/XmlService.cf
ample-tests.AllTests
  [cfcunit] Running   :  cfcunitexample-tests.AllTests
  [cfcunit] Tests Run :  1
  [cfcunit] Failures  :  0
  [cfcunit] Errors    :  0

main:

BUILD SUCCESSFUL
Total time: 1 second
C:\Inetpub\wwwroot\cfcunitexample>_
```

ANT looks for the build.xml file and then runs through the commands inside that build file. If you saw the test run successfully, you can now run your unit tests by command line which opens up all sorts of possibilities. Now, what we really want is to have ANT fire off the unit tests each time we save a ColdFusion file. That is actually very easy to do.

In CFEclipse, right click the cfcunitexample project and go to Properties. Select Builders and click New. Select ANT Build and hit Ok. Name it "Tests" and browse to select the build.xml file as your buildfile. Then click on your Targets tab, and make certain that for Auto Build that default target is selected. Hit Ok, then Ok again.



Now, when you changed all of the files in the cfcunitexample project and hit save. That ANT script should fire off and show the results of the unit testing in your Console window in CFEclipse.

You can now continuously test your ColdFusion coding against CFCUnit tests with ANT. Congratulations, you can now crack open a cold one :)

Note:
You should also see one of the other reasons why the unit tests are separated from the actual project. Imagine if they weren't, each time you try to modify a CFCUnit test and it fired off that ANT script.

Selenium

Now we want to incorporate our usability testing to this environment. Once again, not a problem with ANT on our side. We are again assuming that you have probably played with the Selenium IDE before. If you have not, go ahead and install it into your browser and play around with it.

We first need Selenium Core installed in our web directory.Download Selenium Core from http://www.openqa.org/selenium/ and extract the files to C:\Inetpub\wwwroot\selenium

Next we need Selenium-RC (aka Remote Control). Download it from http://www.openqa.org/selenium-rc/The zip file they provide will several Selenium clients. We what just one file, the selenium-server.jar file found in the selenium-server-x.x.x folder. Extract this jar file to: C:\Inetpub\wwwroot\selenium\lib (you will need to create that lib folder)

Next, we need to modify our build.xml file that is located in our project's folder.

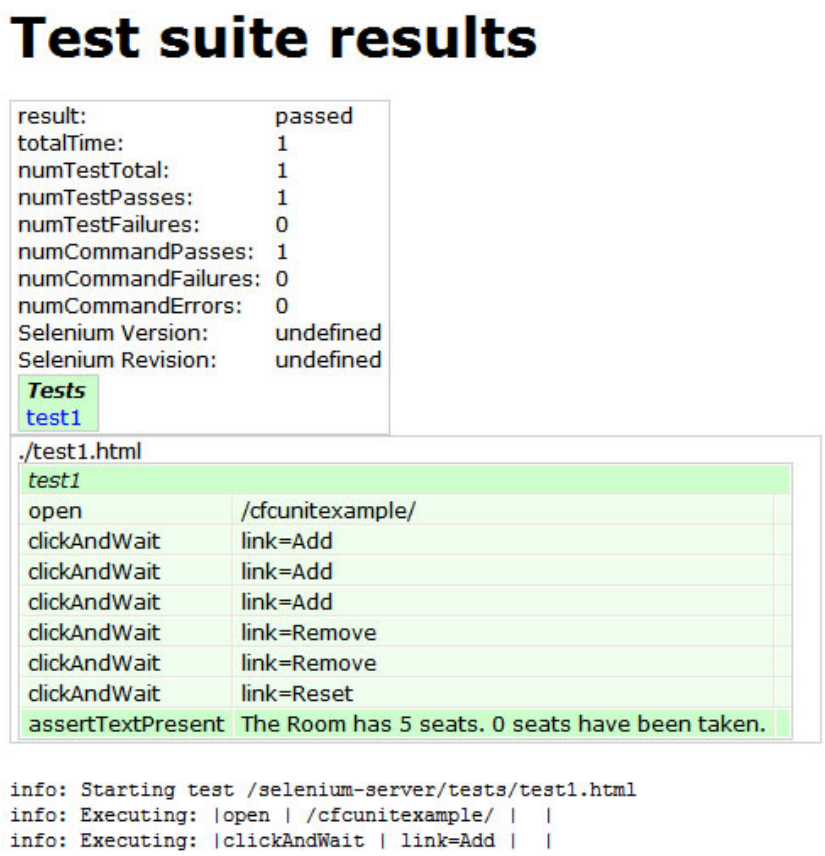
Uncomment the following..

```
<taskdef resource="selenium-ant.properties">
<classpath>
<pathelement location="${seleniumLib}/selenium-server.jar"/>
</classpath>
</taskdef>
```

Next add the selenium task to our main target..

```
<target name="main" depends="unittest, selenium" />
```

Save your changes and you should see the ANT script run through your unit test and then your usability test. You can view the report that Selenium generates at: http://localhost/cfcunitexample-tests/selenium/report/index.html



Test suite results

result:	passed
totalTime:	1
numTestTotal:	1
numTestPasses:	1
numTestFailures:	0
numCommandPasses:	1
numCommandFailures:	0
numCommandErrors:	0
Selenium Version:	undefined
Selenium Revision:	undefined

Tests
test1

./test1.html

test1	
open	/cfcunitexample/
clickAndWait	link=Add
clickAndWait	link=Add
clickAndWait	link=Add
clickAndWait	link=Remove
clickAndWait	link=Remove
clickAndWait	link=Reset
assertTextPresent	The Room has 5 seats. 0 seats have been taken.

```
info: Starting test /selenium-server/tests/test1.html
info: Executing: |open | /cfcunitexample/ |  |
info: Executing: |clickAndWait | link=Add |  |
```

You can modify your ANT script again and have it zip up your projects files after a successful run of your testing.

```
<target name="main" depends="unittest, selenium, build" />
```

It will dump a zip file under your C:\Inetpub\wwwroot directory.

Concluding Thoughts:

You can now modify your ANT script to do just unit testing or combine it with usability testing with Selenium, and also zip up your projects files for deployment, etc.

I did not put Subversion into the ANT script because in most cases you will want to run Subversion commits completely separately from this process. The same holds true for FTP uploads. But, if you feel that you would like that functionality, it is fairly easy to setup with ANT.

By the way, you are going to really learn to love ANT. Read up on it. There are a lot of features and plug-ins for it. You can also run ANT scripts in ColdFusion with the <CFANT> tag. It is undocumented and not directly supported by Adobe, but it is actually used by ColdFusion when it is installing itself on your machine. So I doubt it will disappear anytime soon.

Here is the information on the tag:

```
<cfant buildFile="completePathToBuildFile"
defaultDirectory="seemsTobeIgnoredButMustBePresent"
anthome="alsoSeemsToBeIgnoredButMayNeedToHaveAnEnvironmentVariableSetup"
messages="coldFusionVariableToHoldOutPutMessages"
target="defaultTarget"/>
```

For example, to run our build.xml script

```
<cfant buildFile="C:\Inetpub\wwwroot\cfcunitexample\build.xml"
defaultDirectory=""
anthome=""
messages="antResult"
target="main"/>

<cfdump var="#antResult#">
```

Take the Selenium part off your main target before running this. Your cfdump should report back a successful ANT build.

What's next?

The next step is called Continuous Integration. You will want to use Cruise Control http://cruisecontrol.sourceforge.net or a similar tool for that. CI is a pretty in-depth topic and is beyond the scope of this article. We may in the future write up a similar article for it. So please subscribe to the RSS feed. In the meantime, enjoy doing your continuous testing :)

Feel free to send us comments or recommendations, you can email me directly at mason@fusionlink.com

Also, if you are ever in need of ColdFusion or Flex hosting, FusionLink is more than happy to assist.

Page modified on 11/11/2007 3:10 PM