Creating a Maven Integration Test

Naming your Test

In the past, tests were numbered sequentially (it001,it002, etc). This becomes problematic because it's not always obvious what the issue is related to the test. More importantly, it's hard to create and submit sequential tests via patches because non-committers have no way to reserve the next number. This leads to extra work to apply the patch.

New tests should be named using the JIRA issue id along with a meaningful name. This name should typically be the subject of the issue or a subset. It should be informative of the issue but not excessively long. Therefore a good name would be: `MavenITmng1234groupIdOrderIsBackwardsTest`. The corresponding resource directory for this test in `core-it-suite/src/test/resources/` would simply be named `mng-1234`, i.e. without the description to keep the path short.

Sample IT Project

A sample project has been created to aid in IT test creation. It can be retrieved using Subversion from our source repository or directly viewed with your browser via our SVN view.

Sample IT Archetype

This project has also been converted into an archetype to make it even easier to use.

> *It would be helpful to review the archetype plugin documentation here: http://maven.apache.org/plugins/maven-archetype-plugin/*

The Archetype Plugin doesn't currently know about this archetype, so we created a catalog file to be used.

```
mvn archetype:generate
-DarchetypeCatalog=http://docs.codehaus.org/
download/attachments/1835439/archetype-catal
og.xml
```

```
$ mvn archetype:generate
-DarchetypeCatalog=http://docs.codehaus.org/
download/attachments/1835439/archetype-catal
og.xml
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with
prefix: 'archetype'.
```

```
[INFO]
------------------------------------------
-----------------------------
[INFO] Building Maven Default Project
[INFO]    task-segment: [archetype:generate]
(aggregator-style)
[INFO]
------------------------------------------
-----------------------------
[INFO] Preparing archetype:generate
[INFO] No goals needed for project -
skipping
[INFO] Setting property:
classpath.resource.loader.class =>
'org.codehaus.plexus.velocity.ContextClassLo
aderResourceLoader'.
[INFO] Setting property:
velocimacro.messages.on => 'false'.
[INFO] Setting property: resource.loader =>
'classpath'.
[INFO] Setting property:
resource.manager.logwhenfound => 'false'.
[INFO] [archetype:generate]
Choose archetype:
1: remote -> maven-it-sample-archetype
(Archetype for creating a Maven 2
Integration Test)
Choose a number:  (1): 1
[INFO] snapshot
org.apache.maven.its:maven-it-sample-archety
pe:1.0-SNAPSHOT: checking for updates from
```

```
maven-it-sample-archetype-repo
Downloading:
http://people.apache.org/repo/m2-snapshot-re
pository/org/apache/maven/its/maven-it-sampl
e-archetype/1.0-SNAPSHOT/maven-it-sample-arc
hetype-1.0-20080303.190914-2.jar
9K downloaded
Define value for groupId: :
org.apache.maven.its
Define value for artifactId: : my-it
Define value for version: : 1
Define value for package: :
org.apache.maven.it
Confirm properties configuration:
groupId: org.apache.maven.its
artifactId: my-it
version: 1
package: org.apache.maven.it
 Y: : Y
[WARNING]
org.apache.velocity.runtime.exception.Refere
nceException: reference : template =
archetype-resources/src/test/resources/mng-x
xxx-descriptionOfProblem/checkstyle-test/pom
.xml [line 42,column 26] :
${project.build.directory} is not a valid
reference.
[INFO]
------------------------------------------
----------------------------
[INFO] BUILD SUCCESSFUL
```

```
[INFO]
------------------------------------------------
-----------------------------
[INFO] Total time: 44 seconds
[INFO] Finished at: Mon Mar 03 19:13:41 GMT
2008
[INFO] Final Memory: 10M/28M
[INFO]
------------------------------------------------
```

```
-----------------------------
```

The project will be created in a subfolder of your current directory with the name you chose for the artifact. (my-it in the above example).

Tour of the Sample Project

The sample IT project contains the following files:

```
|   pom.xml
|
\---src
    \---test
        +---java
        |   \---org
        |       \---apache
        |           \---maven
        |               \---it
 |                     |
MavenITmngXXXXDescriptionOfProblemTest.java
        |
        \---resources
            \---mng-xxxx
                |   pom.xml
                |   readme.txt
                |
                +---checkstyle-assembly
                |   |   pom.xml
                |   |
                |   +---src
                |      \---main
                |          \---resources
```

```
                  |
rule_set.xml
                  |
stc_checks.xml
                  |
                  |
                  |
                  \---checkstyle-test
                      |   pom.xml
                      |
                      \---src
                          \---main
                              \---java
```

```
Class.java
```

The directory `src/test/java/` contains a junit file used to execute the sample project. It is heavily commented to guide you along. Stop here and Take a look before continuing.

The directory `src/test/resources` contains a sample project tree that will be invoked to test our condition. The sample consists of a few modules, first an extension (checkstyle-assembly) that is jarred up and installed into the repository. Then a second invocation is made on a project that uses the extension (checkstyle-test)

The sample IT project uses the simplest method of using the verifier, that is the sample build fails if the test should fail. More complicated ITs can be created with beanshell scripts to check complex conditions. (Example Needed) (Link to verifier docs)

Once your IT is complete, you should be able to execute it:

```
>mvn test
[INFO] Scanning for projects...
[INFO]
----------------------------------------------
--------------------------------
[INFO] Building Maven Integration Tests
[INFO]    task-segment: [test]
[INFO]
----------------------------------------------
--------------------------------
[INFO] [resources:resources]
[INFO] Using default encoding to copy
filtered resources.
[INFO] [compiler:compile]
[INFO] No sources to compile
[INFO] [resources:testResources]
[INFO] Using default encoding to copy
filtered resources.
[INFO] [compiler:testCompile]
```

```
[INFO] Compiling 1 source file to
E:\svn\Maven\maven-it-tests\test\artifact\ta
rget\test-classes
[INFO] [surefire:test]
[INFO] Surefire report directory:
E:\svn\Maven\maven-it-tests\test\artifact\ta
rget\surefire-reports

-------------------------------------------
-----------
 T E S T S
-------------------------------------------
-----------
Running
org.apache.maven.it.MavenITmngXXXXDescriptio
nOfProblemTest
mngXXXXDescriptionOfProblem(testitMNGxxxx)..
OK
Tests run: 1, Failures: 0, Errors: 0,
Skipped: 0, Time elapsed: 16.802 sec

Results :

Tests run: 1, Failures: 0, Errors: 0,
Skipped: 0
[INFO]
-------------------------------------------
----------------------------
[INFO] BUILD SUCCESSFUL
[INFO]
-------------------------------------------
```

```
------------------------------
[INFO] Total time: 25 seconds
[INFO] Finished at: Wed Aug 08 22:47:44 EDT
2007
[INFO] Final Memory: 4M/10M
[INFO]
```

```
-----------------------------------------------
----------------------------
```

Submitting your IT Test

Now that your IT is complete, create a zip, tgz or some other *normal* archive of your test and attach it to your JIRA issue.

If you are a Maven committer, read the next section to see how to integrate your test into the core, or to apply someone else's test.

Integrating an IT into the core-integration-tests

First, check out the core-it-suite

The sample project maps directly to the core-it-suite structure. Assuming the test has been named uniquely, you can copy the contents of src/test directly over to core-it-suite (don't copy from the root or you'll overwrite the POM).

```
/cygdrive/e/svn/Maven/maven-it-tests/core-it
-suite
$ cp ../test/artifact/src . -R -v
`../test/artifact/src/test/java/org/apache/m
aven/it/MavenITmngXXXXDescriptionOfProblemTe
st.java' ->
`./src/test/java/org/apache/maven/it/MavenIT
mngXXXXDescriptionOfProblemTest.java'
`../test/artifact/src/test/resources/mng-xxx
x/checkstyle-assembly/pom.xml' ->
`./src/test/resources/mng-xxxx/checkstyle-as
sembly/pom.xml'
`../test/artifact/src/test/resources/mng-xxx
x/checkstyle-assembly/src/main/resources/rul
e_set.xml' ->
`./src/test/resources/mng-xxxx/checkstyle-as
sembly/src/main/resources/rule_set.xml'
`../test/artifact/src/test/resources/mng-xxx
```

```
x/checkstyle-assembly/src/main/resources/stc
_checks.xml' ->
`./src/test/resources/mng-xxxx/checkstyle-as
sembly/src/main/resources/stc_checks.xml'
`../test/artifact/src/test/resources/mng-xxx
x/checkstyle-assembly/target/classes/rule_se
t.xml' ->
`./src/test/resources/mng-xxxx/checkstyle-as
sembly/target/classes/rule_set.xml'
`../test/artifact/src/test/resources/mng-xxx
x/checkstyle-assembly/target/classes/stc_che
cks.xml' ->
`./src/test/resources/mng-xxxx/checkstyle-as
sembly/target/classes/stc_checks.xml'
`../test/artifact/src/test/resources/mng-xxx
x/checkstyle-test/pom.xml' ->
`./src/test/resources/mng-xxxx/checkstyle-te
st/pom.xml'
`../test/artifact/src/test/resources/mng-xxx
x/checkstyle-test/src/main/java/Class.java'
->
`./src/test/resources/mng-xxxx/checkstyle-te
st/src/main/java/Class.java'
`../test/artifact/src/test/resources/mng-xxx
x/pom.xml' ->
`./src/test/resources/mng-xxxx/pom.xml'
`../test/artifact/src/test/resources/mng-xxx
x/readme.txt' ->
```

```
`./src/test/resources/mng-xxxx/readme.txt'
```

Now add your test to the test suite by adding a new line to `src\test\java\org\apache\maven\it\IntegrationTestSuite.java`

```
suite.addTestSuite(
MavenITmngXXXXDescriptionOfProblemTest.class
);
```

Now run the suite:

```
brianf@sonoma
/cygdrive/e/svn/Maven/maven-it-tests/core-it
-suite
$ mvn test
[INFO] Scanning for projects...
[INFO]
----------------------------------------------
---------------------------------
[INFO] Building Maven Integration Tests
[INFO]    task-segment: [test]
[INFO]
----------------------------------------------
---------------------------------
[INFO] [resources:resources]
[INFO] Using default encoding to copy
filtered resources.
[INFO] [compiler:compile]
[INFO] No sources to compile
[INFO] [resources:testResources]
```

```
[INFO] Using default encoding to copy
filtered resources.
[INFO] [compiler:testCompile]
[INFO] Compiling 124 source files to
e:\svn\Maven\maven-it-tests\core-it-suite\ta
rget\test-classes
[INFO] [surefire:test]
[INFO] Surefire report directory:
e:\svn\Maven\maven-it-tests\core-it-suite\ta
rget\surefire-reports
Running integration tests for Maven 2.0.7

-------------------------------------------
-----------
 T E S T S
-------------------------------------------
-----------
Running
org.apache.maven.it.IntegrationTestSuite
Running integration tests for Maven 2.0.7
0000(testit0000).. Ok
....
got get some coffee, it's gonna be a while
....
MavenITmngXXXXDescriptionOfProblemTest...OK
<-- our test...yeah!

Tests run: 115, Failures: 22, Errors: 3,
Skipped: 0, Time elapsed: 534.241 sec <<<
FAILURE!
```

```
Results :

Failed tests:

testit0002(org.apache.maven.it.MavenIT0002Te
st)

Tests in error:

testit0086(org.apache.maven.it.MavenIT0086Te
st)

testit0087(org.apache.maven.it.MavenIT0087Te
st)

testit0104(org.apache.maven.it.MavenIT0104Te
st)

Tests run: 115, Failures: 22, Errors: 3,
Skipped: 0

[INFO]
------------------------------------------
----------------------------
[ERROR] BUILD FAILURE
[INFO]
------------------------------------------
----------------------------
[INFO] There are test failures.
[INFO]
------------------------------------------
```

```
------------------------------
[INFO] For more information, run Maven with
the -e switch
[INFO]
------------------------------------------------
------------------------------
[INFO] Total time: 9 minutes 12 seconds
[INFO] Finished at: Wed Aug 08 23:39:48 EDT
2007
[INFO] Final Memory: 4M/12M
[INFO]
```

```
----------------------------------------------------------------
----------------------------------------
```

Don't panic if some tests fail. That's probably normal. Only panic if your test doesn't fail and it was supposed to (especially if you didn't write it).

Now add all those new files to svn (you can figure this one out on your own by now...you're an expert...and its late here) and commit.