

IT Problems

These are problems with Maven's own integration tests. The ones we use to make sure that changes to a new version of Maven don't break anything that previously worked.

Goals

- An IT should be completely self-contained so that the problem can be understood by looking in one place, in one Maven project.
- We should be able to create an Archetype so that users can easily create ITs for us
- The ITs should be in a project of their own so that we can reuse them across versions of Maven. We could actually run new versions of integration tests against old versions of Maven. Solution: the ITs are now in a separate build and it is possible to run them
- We should be able to easily integrate the IT into a larger run where we can use forked or embedded execution.
- We should create Archetypes for all categories of problems so that anyone can generate tests cases for us. Then there is so much that we can do in terms of automating this process of checking tests for quality along with the patches.
- automate the testing of ITs submitted by users
- Each IT should have its own repository if it needs resources from repository. We can't mess with a users repository when testing.
- We need to have a file system based remote repository for testing
- We need to standardize on integration testing in general. We have people going all over the place and it's a disaster.
 - We have too many IT plugins (3)
 - We have too many invokers (5)
 - We have too many verifiers (3)
- The ITs should run nicely from an IDE. Solution: this does work but requires that you run `mvn clean resources:testResources` first as the IDE doesn't know how to set that up. Needs to be fully fixed. But it is much nicer running this stuff in your IDE.

Problems with ITs

Problem	Status
Verifier jar required by the bootstrap requires a special verifier.jar there is no released version of this tool.	The bootstrap now uses Ant and we've gotten rid of a lot of the complexity.
The maven-core-it plugin needs to be decoupled into its separate purposes because there are currently 12 different things going on in the plugin and it would be really confusing for a new user to figure out what's going on in the plugin and how it applies to the integration testing	These have now been broken down into plugins that correspond to their function in the ITs.
it0006 is an integration test for the verifier plugin.	The test has been migrated to the plugin project.

it0013 is an integration test for the Plugin Plugin.	The test has been migrated to the plugin project.
it0014 uses the compiler plugin to test plugin configuration.	The Compiler Plugin has been replaced with an IT plugin.
it0016 is an integration test for the WAR plugin.	The test has been migrated to the plugin project.
it0017 is an integration test for the EJB plugin.	The test has been migrated to the plugin project.
it0018 uses real dependencies to test artifact resolution.	The test has been changed to use dedicated IT artifacts.
it0020 tests beanshell mojo support.	The test has been deleted due to discontinued support for BeanShell-based plugins.
it0024 uses the compiler plugin to test mojo configuration.	The Compiler Plugin has been replaced with an IT plugin.
it0028 uses the compiler plugin to test mojo configuration.	The Compiler Plugin has been replaced with an IT plugin.
it0029 uses the compiler plugin to test for pluginManagement injection of plugin configuration.	The Compiler Plugin has been replaced with an IT plugin.
it0031 uses the Modello Plugin to test plugin prefix resolution.	The Modello Plugin has been replaced with an IT plugin.
it0033 is an integration test for the EAR plugin.	The test has been migrated to the plugin project.
it0043 uses the help plugin to look at the state of the final POM, we should have this built into Maven for testing.	The Help Plugin has been replaced with an IT plugin.
it0047 uses plexus-container-default for testing.	The test has been changed to use dedicated IT artifacts.
it0048 uses the surefire plugin to test default value population for mojo parameters.	The Surefire Plugin has been replaced with an IT plugin.
it0050 is an integration test for the surefire plugin.	The test has been migrated to the plugin project.
it0051 is an integration test for the release plugin.	The test only checks the release profile from the super POM. It has nevertheless been decoupled from production versions of the Source Plugin and Javadoc Plugin.

it0052 is an integration test for the release plugin.	The test only checks the release profile from the super POM. It has nevertheless been decoupled from production versions of the Source Plugin and Javadoc Plugin.
it0054 is an integration test for the resources plugin.	The test has been migrated to the plugin project.
it0055 is an integration test for the compiler plugin.	The test has been migrated to the plugin project.
it0056 uses the compiler plugin to test multiple executions.	The Compiler Plugin has been replaced with an IT plugin.
it0060 uses the compiler plugin to test aggregation of list configuration items when using 'combine.children=append' attribute.	The Compiler Plugin has been replaced with an IT plugin.
it0061 uses the Deploy Plugin to test legacy repo layout.	The test has been changed to an IT plugin.
it0063 tests JDK 1.4.2 specifics.	The test has been changed to use a platform-independent fake artifact.
it0068 uses the modello plugin to test repository accumulation (c.f. MNG-757 , MNG-836)	The test has been changed to use a dedicated IT plugin.
it0069 uses classworlds dependency to test offline mode.	The test has been changed to use dedicated IT artifacts.
it0070 is an integration test for the RAR plugin.	The test has been migrated to the plugin project.
it0074 uses the compiler plugin to test mojo configuration.	The Compiler Plugin has been replaced with an IT plugin.
it0075 utilize the Eclipse Plugin and Help Plugin.	The test has been changed to use an IT plugin.
it0076 uses the WAR plugin to test pluginManagement.	The WAR Plugin has been replaced with an IT plugin.
it0078 uses the compiler plugin to test pluginManagement.	The Compiler Plugin has been replaced with an IT plugin.
it0079 uses the source plugin to test that attached artifacts have the same build number as the primary artifact.	The Source Plugin has been replaced with an IT plugin.
it0080 uses the WAR plugin to test an artifact handler that specifies no contribution to transitive dependencies. If the behavior of this handler changed the test would break.	The test has been rewritten to use IT plugins. However, the artifact handler under test is indeed part of the core, so the test continues to verify its behavior as per the test's original design.

it0081 uses production dependency of wagon as a dependency.	The test has been updated to use dedicated IT artifacts.
it0083 uses the WAR plugin to test an artifact handler that the WAR artifact handler currently behaves like.	The test (which rather verifies the scope update of transitive dependencies by local override) has been changed to use IT plugins.
it0086 uses production dependency of bsh to test classloading.	The test has been changed to use dedicated IT artifacts.
it0087 uses production dependency of commons-logging to test classloading.	The test has been changed to use dedicated IT artifacts.
it0089 uses production dependency on checkstyle to test class loading.	The test has been identified as a duplicate of it0086 and as such has simply been deleted.
it0094 has a test that attempts to access a protected field in a ClassRealm.	The test has simply been deleted as it is now superseded by strengthened editions of it0086 and it0087.
it0095 uses the help and verifier plugins to test URL calculation.	The production plugins have been replaced with an IT plugin.
it0100 uses the antrun plugin to make sure <code>\${parent.artifactId}</code> resolves correctly.	The production plugins have been replaced with an IT plugin.
it0102 uses the help and antrun plugins to test profile activation.	The production plugins have been replaced with an IT plugin.
it0104 uses the surefire plugin to test interpolation.	The test uses an IT plugin now.
it0105 is an integration test for the resources plugin.	The test has been migrated to the plugin project.
it0111 uses the Checkstyle Plugin to test resources provided by extensions. Maybe just delete since it0114 seems to test the same issue?	The test has been deleted as it was an inferior duplicate of it0114.
it0112 uses the PIR Plugin to test extension dependencies.	The test has been rewritten to use IT artifacts/plugins.
it0119 places artifacts in the o.a.m.plugins and o.c.mojo group IDs to test plugin prefix order instead of using a replacement settings file and subgroups of o.a.m.its as it should.	Part of the IT is testing the default plugin groups which are burried as constants in maven-core. There is no other way of testing these defaults as by installing test plugins into these default groups. The only means to avoid messing with the user's local repo is to use an isolated IT repo but that is a more general issue.

it0127 uses the AntRun Plugin to check plugin class realms for multiple instances of the same plugin in the reactor.	The test has been rewritten to use IT plugins/dependencies.
it-mng-3426 uses the Castor Plugin to test plugin classpath overriding via plugin-level deps. Maybe just delete since it-mng-2972 seems to check the same issue?	The test has been deleted after it-mng-2972 has been improved to capture the aspects of this test as well.
it-mng-3372 uses the Dependency Plugin to test direct goal invocation.	
it-mng-3473 uses the Help Plugin and Plugin Plugin.	The test has been deleted from the core IT suite as it didn't actually test the core but the Plugin Tools where a copy of this IT continues to live.
artifactIds should be aligned with directories.	

The original document

*** Instructions**

Running the following from the top-level directory builds a distributable:

```
----  
mvn install  
----
```

If you wish to also run the integration tests themselves:

```
----  
mvn -Prun-its install  
----
```

This command can also be run from the

core-integration-tests directory if the prerequisite artifacts have already been installed or deployed.

* Reusing the suite

The suite is bundled up as a JAR containing all the tests and resources. All artifacts are now versioned according to the version of Maven they are testing (currently, 2.1-SNAPSHOT), though it can be run against older versions as the verifier will detect and omit tests that require the newer version of Maven to pass.

* Outstanding Issues

The top things that could be done:

1) The issues that would be most helpful that could be tackled on a piecewise basis by many would be to take plugin

specific ITs out of the ITs. There are many in there for the surefire plugin so while you're doing that you can look

at it. Piecewise but probably totally simple because you have to replace it with an IT that actually tests what it

was testing. A lot of time I have had to make a new IT plugin flavour.

2) The next issue of importance would be to collect all the in IT plugin plugins, invokers and verifiers and align all these.

3) Once 2) is done then we wire the embedder option into the resulting invoker.

4) Proper isolation. The tests currently pick up your Maven settings file and current local repository. A clean local

repo can be used but it would result in a lot of unnecessary artifact retrieval. This fails in a locked down

environment (as additional repositories declared in IT 92, 94 and 120 are ignored). It also causes other risks.

A possible solution is to construct a repository artifact (using a POM with the dependency plugin to create an

assembly) that represents everything used by the ITs, then to have the setUp lay that out as a remote repo and use a

controlled settings.xml that uses a clean local repository, and mirrors central to that remote repository.

5) The support artifacts are currently deployed to the central repository, meaning they can get out of sync with those in SVN (unless they are assumed to have

been 'released'). After solving (4), these could be injected into the constructed repository instead.

6) It's not obvious how to run them from an IDE and that's where I've found it to be most convenient to run them.

In particular, the resources directory is treated as test sources, which is not the case as each are projects

7) it0119 places artifacts in the o.a.m.plugins and o.c.mojo group IDs to test plugin prefix order instead of using a replacement settings file and subgroups of o.a.m.its as it should

Other issues:

- [] An IT should be completely self-contained so that the problem can be understood by looking in one place, in one Maven project.

- [] We should be able to create an Archetype so that users can

easily create ITs for us. The is not completed but we do have the

core-integration-test-sample
directory with has a sample I
offered to users.

- [] We should be able to easily
integrate the IT into a larger
run where we can use forked or
embedded execution.

- [] We should create Archetypes for
all categories of problems so
that anyone can generate tests
cases for us. Then there is so
much that we can do in terms of
automating this process of
checking tests for quality along
with the patches.

- [] automate the testing of ITs
submitted by users

- [] Each IT should have its own
repository if it needs resources
from repository. We can't mess
with a users repository when
testing.

- [] We need to have a file system
based remote repository for
testing

- [] We need a primary run that can
done entirely offline to simply test
the guts of Maven from a baseline,
then a secondary run possibly using
the exact same repository except
served via different means like http,

ftp, scp which would allow us to find all the holes in the transport mechanisms.

- [] We need to standardize on integration testing in general. We have people going all over the place and it's a disaster.

- [] We have too many IT plugins
(3)

- [] We have too many invokers (5)

- [] We have too many verifiers (3)

- [] The ITs should run nicely from an IDE. Solution: this does

work but requires that you run mvn clean

resources:testResources first as the IDE doesn't know how to

set that up. Needs to be fully fixed. But it is much nicer

running this stuff in your IDE.

- [-] Problems with ITs

- [] it0006 is an integration test for the verifier plugin.

- [] it0014 uses the compiler plugin to test plugin configuration.

- [] it0016 is an integration test for the WAR plugin.

- [] it0017 is an integration test for the EJB plugin.

- [] it0018 uses real dependencies to

test artifact resolution.

- [] it0020 tests beanshell mojo support.

- [] it0024 uses the compiler plugin to test mojo configuration.

- [] it0028 uses the compiler plugin to test mojo configuration.

- [] it0029 uses the compiler plugin to test for pluginManagement injection of plugin configuration.

- [] it0033 is an integration test for the EAR plugin.

- [] it0043 uses the help plugin to look at the state of the final

- [] it0043 uses the help plugin to look at the state of the final POM, we should have this built into Maven for testing.

- [] it0047 uses plexus-container-default for testing.

- [] it0048 uses the surefire plugin to test default value

- [] it0048 uses the surefire plugin to test default value population for mojo parameters.

- [] it0050 is an integration test for the surefire plugin.

- [] it0051 is an integration test for the release plugin.

- [] it0052 is an integration test for the release plugin.

- [] it0054 is an integration test for the resources plugin.

- [] it0055 is an integration test for the compiler plugin.

- [] it0056 uses the compiler plugin to test multiple executions.

- [] it0060 uses the compiler plugin to test aggregation of list configuration items when using 'combine.children=append' attribute.

- [] it0063 tests JDK 1.4.2 specifics.

- [] it0068 uses the modello plugin to test repository accumulation (i'm not sure what that means, John?)

- [] it0069 uses classworlds dependency to test offline mode.

- [] it0070 is an integration test for the RAR plugin.

- [] it0074 uses the compiler plugin to test mojo configuration.

- [] it0076 uses the WAR plugin to test pluginManagement.

- [] it0078 uses the compiler plugin to test pluginManagement.

- [] it0079 uses the source plugin to test that attached artifacts have the same build number as the primary artifact.

- [] it0080 uses the WAR plugin to test an artifact handler that specifies no contribution to transitive dependencies. if the behavior of this handler changed

the test would break.

- [] it0081 uses production dependency of wagon as a dependency

- [] it0083 uses the WAR plugin to test an artifact handler that

 - the WAR artifact handler currently behaves like

 - [] it0086 uses production dependency of bsh to test classloading

 - [] it0087 uses production dependency of commons-logging to test classloading

 - [] it0089 is an integration test for the checkstyle plugin,

 - and places an artifact in the wrong group ID

 - [x] it0094 has a test that attempts to access a protected field

 - in a ClassRealm

 - [] it0095 uses the help and verifier plugins to test URL

 - calculation

 - [] it0100 uses the antrun plugin to make sure

 - `${parent.artifactId}` resolves correctly

 - [] it0102 uses the help and antrun plugins to test profile

 - activation

 - [] it0104 uses the surefire plugin to test interpolation

- [] it0105 is an integration test for the resources plugin
- [] artifactIds should be aligned with

directories