ANT to Maven2

One of the things we have been meaning to do is convert our build framework over from ANT to Maven2. The time has come, and the work is in-progress (jira:fcrepo-394).

The basic strategy was laid out in our last face-to-face committers' meeting in Ithaca. As you can see, there are a few things that need to be done. If you are interested in helping, I would be very happy to coordinate efforts. Send an email to the group: "`fedora-ant-to-maven@googlegroups.com`"

Approach

Since moving to Maven2 requires significant restructuring of the baseline directories, we want to make small, incremental changes back to svn/trunk so that the on-going Fedora development does not become too far separated. This means that the ANT builds must continue to work as the baseline becomes incrementally transformed.

Progress/Status

⚠ Not yet
ℹ Partial
✅ Complete

* ✅ **1a - Source conformant to m2 standards (files and packages) + tweak build.xml**

- 1. This was checked-into svn:fedora/trunk.rev.8076
  2. The basic structure of maven has been created (src/main/java, src/test/java, etc)
  3. The renaming of packages based on Maven-central's requirements will happen at the very end of this whole effort
- ✅ **1b - Get poms for all dependencies**
  1. Based on the jars in the /lib dir, a simple install script pushes them into the local repository
  2. Need to create proper *.pom and <dependency> for each artifact
- ✅ **2a - Ant-maven integration (eight subprojects) generating source and building**
  - This initial integration requires a minimum break-out of the following subprojects
  - This is just building, unit tests turned off
  - This was checked-into svn:fedora/trunk.rev.8083
    1. ✅ generate
       - This is a small project that creates the executable stubwrapper generator
    2. ✅ common
       - This contains 'fedora.common' package
       - includes wsdl2java generation and stubwrappers
       - has dependencies on '/generate'
    3. ✅ server
       - This is the 'fedora.server' package
       - has dependencies on '/common'
    4. ✅ client
       - This is the 'fedora.client' package
       - has dependencies on both '/common' and '/server'
    5. ✅ integrationtest
       - This is the 'fedora.test' package
       - has dependencies on both '/client' and '/server'
    6. ✅ localservices
       - This contains three sub-projects
         - fop
         - imagemanip
         - saxon
       - War files are created for all contained sub-projects when running `mvn clean install`
    7. ✅ democontent
       - This creates a war file (fedora-demo.war) containing only demo content (no application code)
    8. ✅ installer
       - This project has no source java files of its own
       - It uses the 'assembly' plugin to aggregate other artifacts into `fedora-installer.jar`
       - To create `fedora-installer.jar`, run: `mvn clean install -P fedora-installer`
- ✅ **2b - All maven unit tests** (test inventory)
  - This was checked-into svn:fedora/trunk.rev.8086
    1. ✅ generate
    2. ✅ common
    3. ✅ server
    4. ✅ client
    5. ✅ integrationtest
-
  -
    - No unit tests here by definition
- ✅ **2c - All maven integration tests** (test inventory)
  - Like the system tests previously provided in the ANT tests (ConfigA/B/Q), for now, the Maven2 integration tests will require a pre-started application server.
  - Because integration tests are run during the default build lifecycle of Maven2, you must pass the following flag at build-time to disable the running of integration tests ('integration.test.skip')
    - `mvn clean install -Dintegration.test.skip=true`

- *# ✅ generate
    1. ✅ common
    2. ✅ server
    3. ✅ client
    4. ✅ integrationtest
- ✅ **2d - All maven system tests** (test inventory)
    - These tests run within the 'integrationtest' project as separate profiles.
    - This was checked-into svn:fedora/trunk.rev.8088

- *# ✅ ConfigA
    1. `mvn clean install -P configA`
    2. ✅ ConfigB
        - `mvn clean install -P configB`
    3. ✅ ConfigQ
        - `mvn clean install -P configQ`
- ℹ️ **2e - installer.jar (and other artifacts verified)**
    1. ✅ client.jar
        - This is being created with the default build target: (`mvn package`)
        - Its artifact name contains the 'fedora' classifier (see note below for *todo*)
    2. ✅ installer.jar
    3. ✅ rmi-journal-receiver.jar
        - This was checked-into svn:fedora/trunk.rev.8114
    4. ✅ fedora-demo.war (democontent)
    5. ✅ local-services
        - This was checked-into svn:fedora/trunk.rev.8092

- *# *## ✅ fop.war
    1. a. ✅ imagemanip.war
       b. ✅ saxon.war
    2. ⚠️ api.jar
        - Is this still used?
    3. ✅ messaging-client.jar
    4. ✅ messaging-client.zip
        - This was checked-into svn:fedora/trunk.rev.8113
    5. ✅ fedorahome.zip
        - created by running: `mvn clean install -P fedora-installer`
    6. ✅ fedora.war
- ⚠️ **3 - m2-only (full split-out)**
    - Admin Client (Swing)
    - Java Client API (FedoraClient.jar)
    - Messaging Client
    - WebAdmin Client (Flex)
    - ✅ Server Webapp
    - ✅ Installer
- ⚠️ **4 - Split out server modules as projects**
- ⚠️ **5 - Rename packages based on Maven-central's requirements**

Commands

The following are some useful commands in working with the Fedora Maven2 builds

1. `mvn clean install`
    - builds all source code
    - runs all unit & integration tests
2. `mvn install -Dintegration.test.skip=true`
    - runs all unit tests
    - skips all integration tests
3. `mvn install -Dmaven.test.skip=true`
    - skips all unit tests
4. `mvn integration-test -P config[A|B|Q]`
    - runs system tests per given configuration
5. `mvn install -P fedora-installer`
    - generates fedora-installer.jar
    - found in /installer/target

Unresolved

> If you would like to help, these nuggets need a friend.

ℹ️ Currently, all of the maven dependencies are in the top-level pom.xml

1. a. ✅ They need to be surgically pushed down to their appropriate subproject pom.xml
        i. An initial push down is complete though
   b. ✅ Their dependency declarations need to point to the proper version on maven-central, not the locally created artifact
        i. All of the artifacts that can be resolved to maven-central without selected new versions is complete
        ii. Additional validation is needed especially to ensure that conflicts do not cause problems

    c. ✅ Libraries which cannot be resolved to maven-central need to added to the Duraspace Mave repository (under thirdparty)
    d. ✅ Libraries which are supplied by Duraspace projects need to be added to the Duraspace Maven repository (under releases)
    e. ℹ️ The Duraspace Maven repository public groups need to be revised to NOT mirror maven-central or other public repositories
2. The continued need for each junit suite aggregator class needs to be re-evaluated
3. Unit test naming conventions need to be standardized (since maven invokes them based on a regex at different build phases)
    a. unit-test: '**/*Test.class'
    b. integration-test: '**/Test*.class'
4. ✅ Unit/system/integration tests used to fall under 'fedora.test'
    a. now that they have been split across subprojects ('server', 'client', 'integrationtest'), they are not aggregated with a single call (i.e. fedora.test.AllUnitTests)
    b. a fix to this issue would only be needed as long as we continue to use ANT, Maven2 has its own test aggregation
5. ✅ Build number & timestamp needed for artifact names and manifest files
    - `client.jar`
        - needs 'fedora.version' in filename and manifest
        - needs 'build.tstamp' in manifest
    - The following is a plugin that should do the trick
    - It should be configured in the top-level /pom.xml and /client/pom.xml
    - http://mojo.codehaus.org/buildnumber-maven-plugin/index.html
    - The build number plug-in fails if the SCM is unavailable or is not installed on the host platform. A <revisionOnScmFailure> parameter has been added to set a default build number. A warning is still shown during build but the build runs properly to completion. The default build number has been set to the "version" of the fedora-repository pom assuming that pom's version will eventually match a current or planned release.
6. Refactor fedora.test.FedoraTestCase.java into two classes (one dependent on '/server' and one on '/client') so tests the inherit from it can be pushed back down from '/integrationtest' to their respective projects.
7. fedora.test.integration.cma.SimpleDeploymentTests has a bug. See source file for details.
8. In creating fedorahome.zip, deploy and undeploy *.wsdd files are token-swapped (e.g. 'Fedora-API-M-Port-SOAPHTTP' to 'management').
    - This mangles token instances such as 'Fedora-API-M-Port-SOAPHTTP **S**'. This is a pre-existing bug.
9. ✅ Script needs to be written to support Maven Bamboo CI build
10. ✅ The file `server/src/main/resources/properties/lib.properties` may be a candidate for removal as it was tied to the Ant build
11. Libraries listed in `server/src/resource/properties/install.properties` do not automatically update with library updates and duplicate information in the pom which does not benefit from pom dependency checking
    a. ℹ️ Updated to be consistent with Maven dependencies but still requires manual editing
12. Split client into four modules in a branch (See Item 3 Above). Keep them under a client subdirectory.
    a. Swing Client
    b. FedoraClient Jar
    c. Messaging Client
    d. WebAdmin Client
13. Experiment with saxon, fop and imagemanip stored under a localservices directory but being dependencies of fedora-repository. This may help Eclipse since each local service is a standalone, deployable war.

Rationale

Subproject break-out

generate

Finding the appropriate lines on which to draw subproject (maven module) boundaries is not an exact science.

During this initial conversion to Maven2, we want to minimize the degree of baseline upheaval while splitting out subprojects enough to allow maven to properly function.

A logical starting point is:

- client
- server
- common

Where 'client' has a dependency on 'server', and they both depend on 'common'.
This becomes a little problematic with the introduction of wsdl stubs/skeletons and in the case of Fedora, second-level generated source code wrapping the generated wsdl stubs.

The generated artifacts are as follows:

- wsdl-stubs
- wsdl-skeletons
- wsdl-stub wrappers

The sequence of events to produce these artifacts is as follows:

1. compile fedora class (BuildAxisStubWrapper.java)
2. run axis-wsdl2java on fedora wsdl's
3. run executable BuildAxisStubWrapper on wsdl2java outputs (wsdl-stubs)

The binary output of step-2 above is needed by 'server' during compilation.
The source output of step-2 above is needed as input to step-3.
The source output of step-3 above is needed by 'client' during compilation.

- Note: the source output is needed here instead of the binary because the generated source contains import statements with dependencies on packages within 'client'.
  Therefore, the source output of step-2 is needed by 'client' before the compilation phase.

This being the case, the three-step sequence above can all be pushed down to 'common'.
The only problem here is that an executable BuildAxisStubWrapper is needed before the source-generation phase of running axis-wsdl2java.
Therefore, a fourth subproject is needed to create the BuildAxisStubWrapper executable: '**generate**'.

integrationtest

There is a combination of unit, integration, and system tests within the baseline.
The '**integrationtest**' subproject grew out of the need to have tests with dependencies on all of the other functional subprojects.

As it so happens, their is a test base class (FedoraTestCase.java) that is extended by several other tests from both 'server' and 'client'.
FedoraTestCase itself depends on both 'server' and 'client'.
Therefore, all tests that extend from FedoraTestCase are implicitly *integration tests* and have been migrated to the 'integrationtest' subproject.