Axis Test Rearchitecture

<!-- -->

1.

Axis Test Rearchitecture

Quick Start

To run the full build system, as the old was run, do the following:

ant clean functional-tests

To see how the componentized build structure is affected, as well as cross-dependancies, do the following:

ant clean compile cd test/encoding ant clean compile run

(The encoding test has a dependancy on the echo sample)
Back to the Top

Common Design Principles

There were 6 design goals that were laid out for me before undertaking this project:

1. Determine the needs and dependancies of each test and sample
2. Reduce the complexity of the xml that compiles, deploys, execs, and undeploys samples and tests
3. Enable a new test or sample to be "plugged in" to the suites, without disturbing currently working things
4. Enable the ability to instantiate any single sample or any single test
5. Identify tests and samples that need "network" or "internet" resources
6. Enable the ability to group tests and samples together in "bundles"

To this end, the general theory was applied to everything:

- Remove the neccessary steps that are in build.xml into build*thing*.xml (where *thing* is either "Samples" or "Test")

- Create a build.xml file in each samples/** and test/** directory
- Stub each build.xml with the following: <?xml version="1.0" ?> <!--
  ==
  --> <!-- Every project should begin with this next block --> <!--
  ==
  --> <!DOCTYPE project [<!ENTITY properties SYSTEM
  "file:../../xmls/properties.xml"> <!ENTITY paths SYSTEM
  "file:../../xmls/path_refs.xml"> <!ENTITY taskdefs SYSTEM
  "file:../../xmls/taskdefs.xml"> <!ENTITY taskdefs_post_compile SYSTEM
  "file:../../xmls/taskdefs_post_compile.xml"> <!ENTITY targets SYSTEM
  "file:../../../xmls/targets.xml">]> < project default="compile" > <!--
  ==
  --> <!-- This tells how "deep" you are (how many levels down from xml-axis/java -->
  <!--
  ==
  --> &ltproperty name="axis.home" location="../../.." /> <!--
  ==
  --> <!-- This is the path description of where you are right now, relative to xmls-axis/java
  --> <!--
  ==
  --> &ltproperty name="componentName" value="test/wsdl/marrays" /> &properties;
  &paths; &taskdefs; &taskdefs_post_compile; &targets; <!--
  ==
  --> <!-- This target should remove anything created by this component --> <!--
  ==
  --> < target name="clean"/> <!--
  ==
  --> <!-- This target should copy in anything that this component needs, or copy out to
  staging --> <!-- This target should also do any neccessary "setup" tasks (manage pre-reqs,
  etc) --> <!--
  ==
  --> < target name="copy"/> <!--
  ==
  --> <!-- This target actually "makes" things --> <!--
  ==
  --> < target name="compile"/> <!--
  ==
  --> <!-- This target "runs" things in/for this component, if neccessary --> <!--
  ==
  --> &lttarget name="run" > &ltantcall target="execute-Component" /> </target>

</project>

- If you want to instantiate this component directly, then add the following type stub into buildSamples.xml or buildTest.xml as appropriate: < target name="echo"> < ant inheritAll="true" antfile="samples/echo/build.xml"/> </target> This should be done if this sample/test is to be a dependancy of another sample or test.
- If you don't want to instantiate it directly, the "compile" target looping in the buildSamples.xml or buildTest.xml file will pick up any build.xml files in the appropriate path.
- In order to reference a dependancy in a component build.xml file, do the following (taken from test/soap/build.xml): < target name="copy"> < ant inheritAll="true" antfile="build.xml" target="utils"/> < ant inheritAll="true" antfile="build.xml" target="RFCDispatch"/> </target>

Back to the Top

Generic Changes

A new top-level path was created, named "xmls". This path houses the common xml configuration file fragments. Roughly, in order of their inclusion, they are:
properties.xml contains common properties, eliminating the large property setting blocks at the top of each build*.xml file.
path_refs.xml contains the common CLASSPATH setup, to eliminate needing to track, or multiply define this large structure in all files separately.
taskdefs.xml contains common ant tasks that are set up, the are inherent in the configuration.
taskdefs_post_compile.xml contains the definitions of tasks that are built during the compilation of tasks, such as forEach, java2wsdl, and wsdl2java.
targets.xml contains common configuration level targets that required hundreds of lines of repeated code in the build*.xml files.

Back to the Top

Samples

The old $(TOP)/build.xml had a target called "samples" which did a very simple full-compile on everything referenced by samples/**/*.java. Although this was very simple, it was not very "strong" in that everything in the samples tree needed to be able to be compiled by this rule. Whenever anything needed to be altered in the samples, this main file needed to be modified. It was very possible for someone to add a sample that needed a change, and by changing the master file, could break an large number of things.

In order to alleviate this risk, and to better define the actual samples compilation and use, the samples building has been moved to a new xml file **buildSamples.xml** but the original target is still stubbed into build.xml for backwards compatibility and use does not change.

Then, I extracted the actual compilation logic for each sample, and componentized it into a

build.xml file, located in the actual sample sub-directory. For example, for the echo sample is now run by the file samples/echo/build.xml. It can be singularly instantiated by invoking:

ant -buildfile buildSamples.xml echo

or as part of the batch (as the old function) by invoking:

ant samples

which is what build.xml does in the "samples" target.
Back to the Top

Tests

The old $(TOP)/build.xml had a target called "buildTest" which did a very simple full-compile on every thing referenced by test/**/*.java. Although this was very simple, it was not very "strong" in that everything in the test tree needed to be able to be compiled by this rule. Whenever anything needed to be altered in the test, this main file needed to be modified. It was very possible for someone to add a test that needed a change, and by changing the master file, could break an large number of things.

In order to alleviate this risk, and to better define the actual test compilation and use, the test building has been moved to a new xml file **buildTest.xml** but the original target is still stubbed into build.xml for backwards compatibility and use does not change.

Then, I extracted the actual compilation logic for each test, and componentized it into a build.xml file, located in the actual test sub-directory. For example, for the session test is now compiled by the file test/session/build.xml. It can be singularly instantiated by invoking:

ant -buildfile buildTest.xml session

or as part of the batch (as the old function) by invoking:

ant functional-tests

which is what build.xml does in the "buildTest" target.