Testing and Analysis with Ant

Ant must perform the following steps in the specified order to analyze your project with Jtest:

1. Register the Parasoft build listener.
2. Compile the project.
3. Run the Jtest task.

Before you run analysis or collect coverage information with Maven, ensure that the Jtest license is properly configured (see Setting the License).

- Running Static Analysis
- Executing Unit Tests
- Collecting Coverage for Unit Tests
  - About the agent Task
  - Collecting Application Coverage
  - Disabling Coverage Data Collection

Running Static Analysis

To perform static analysis on your code:

1. Ensure that the Jtest Plugin for Maven is set up (see Configuring the Jtest Plugin for Ant).
2. Ensure that the target you execute has the `jtest` task is configured in it.
3. Add the Parasoft listener to your command line with the `-listener` switch or configure the `jtest:listener` task in your target.
4. Build your project with Ant. Your command line may resemble the following:

   ```
   ant -lib PATH/TO/jtest-ant-plugin.jar -listener com.parasoft.Listener myTarget
   ```

   The above example the Jtest Plugin for Ant is added to the command line, but you can deploy the JAR file in the Ant's lib directory (see Initial Setup).

   By default, test sources are excluded from analysis. To analyze test code, disable the `excludeTestSources` option; see Jtest Goals Reference for Ant.

Executing Unit Tests

You can include unit test results in the Jtest report by running your tests with the `jtest` task and the dedicated `Unit Tests` built-in test configuration:

1. Ensure that the Jtest Plugin for Maven is set up (see Configuring the Jtest Plugin for Ant).
2. Ensure that the target you execute to run your tests has the following configured in it:
   - the `jtest` task
   - the dedicated `Unit Tests` built-in test configuration
3. Add the Parasoft listener to your command line with the `-listener` switch or configure the `jtest:listener` task in your target.
4. Build your project with Ant. Your command line may resemble the following (the `jtest.coverage.skip` option allows you to skip collecting coverage):

   ```
   ant -lib PATH/TO/jtest-ant-plugin.jar -listener com.parasoft.Listener buildTestAnalyze -Djtest.coverage.skip=true
   ```

Collecting Coverage for Unit Tests

You can collect coverage information during execution of unit tests by running your tests with the `jtest` and `agent` tasks and the dedicated `Unit Tests` built-in test configuration:

1. Ensure that the Jtest Plugin for Maven is set up (see Configuring the Jtest Plugin for Ant).
2. Ensure that the target you execute to run your tests has the following configured in it:
   - the `jtest` task
   - the `junit` task wrapped with the Jtest `agent` task
   - the dedicated `Unit Tests` built-in test configuration
3. Add the Parasoft listener to your command line with the `-listener` switch or configure the `jtest:listener` task in your target.
4. Build your project with Ant. Your command line may resemble the following:
About the agent Task

Collecting coverage with the Java agent is the recommended way of collecting coverage data with Jtest. The agent task, which allows runtime bytecode instrumentation (without modification of .class files) must be configured in the appropriate target. The following example shows the build script with the required tasks configured in one target:

```xml
<project name="agent-sample" default="buildTestAnalyze" xmlns:jtest="antlib:com.parasoft.jtest.plugin.ant">
  <property name="sources" value="src"/>
  <property name="classes" value="bin"/>
  <target name="buildTestAnalyze">
    <!-- compiling sources -->
    <javac srcdir="${sources}" destdir="${classes}"/>
    <!-- instrumenting bytecode -->
    <jtest:agent>
      <!-- Following attributes will be set: haltonerror="no" haltonfailure="no" fork="yes"
          If agent task skipped nested task will be executed without any config changes. -->
      <junit>
        <!-- Formatter element will be set as following: <formatter type="xml"/>
          If different formatter type has been set it will get overridden.
          If formatter has been configured by classname it will be left as it is. -->
        <classpath>
          <pathelement location="${classes}"/>
        </classpath>
      </junit>
    </jtest:agent>
    <coverage>
      <includes>
        <fileset dir="${classes}" />
      </includes>
    </coverage>
    <jtest:jtest/>
  </target>
</project>
```

Collecting Application Coverage

Jtest's coverage agent allows you to collect coverage data during manual or automated tests performed on a running application. See Application Coverage for information about collecting application coverage with Jtest.

Disabling Coverage Data Collection

You can disable collecting coverage in one of the following ways:

- Set the `skip` parameter to skip collecting coverage (this will not disable Jtest execution).
- Run Ant with the following `-D` option: `:-Djtest.skip=true`
- Configure the following project property: `<property name="jtest.skip" value="true"/>` (make sure the subprojects inherit this property if the Jtest tasks were configured there).