Summary

This document explains how to auto-perform TestSuite class or TestCases using build tools such as Ant and Maven.
Test result report will be created, this will be covered in Test Reporting.

Description

Once the developer writes TestCase class for unit test, they are tested individually.
However, code based TestCases can be tested automatically via build tools.
And and Maven both supports automatic testing, and Eclipse provides GUI environment once Plug-in for either Ant or Maven is installed. If registered to CI server, the same process is run on the server, then CI server provides XML files to merge the report
Ant and Maven both follows the same process.

1. Write script to test (Ant - build.xml, Maven - pom.xml)
2. Run test under developer’s environment using Eclipse plug-in
3. Check Test result XML Report

1)

4. Register to CI Server, set it to run the test, check merged report in the server.

✔ How to register and check in the CI server is out of scope of this documentation.

Environmental settings

- Install Ant and Maven Eclipse Plugins
- Install JUnit Eclipse Plugin
- Add library to write TestSuite (pom.xml) : Refer to pom.xml dependency setup for Unit Test

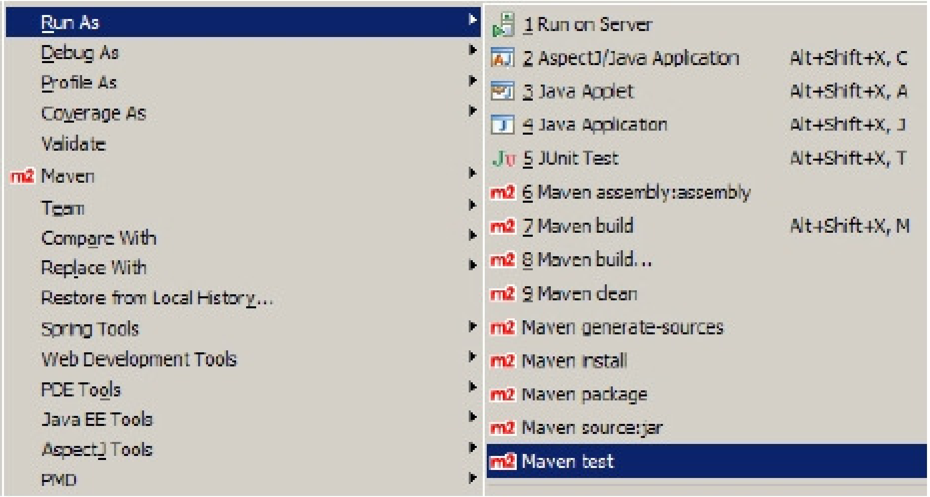
Manual

Use Maven to automate test

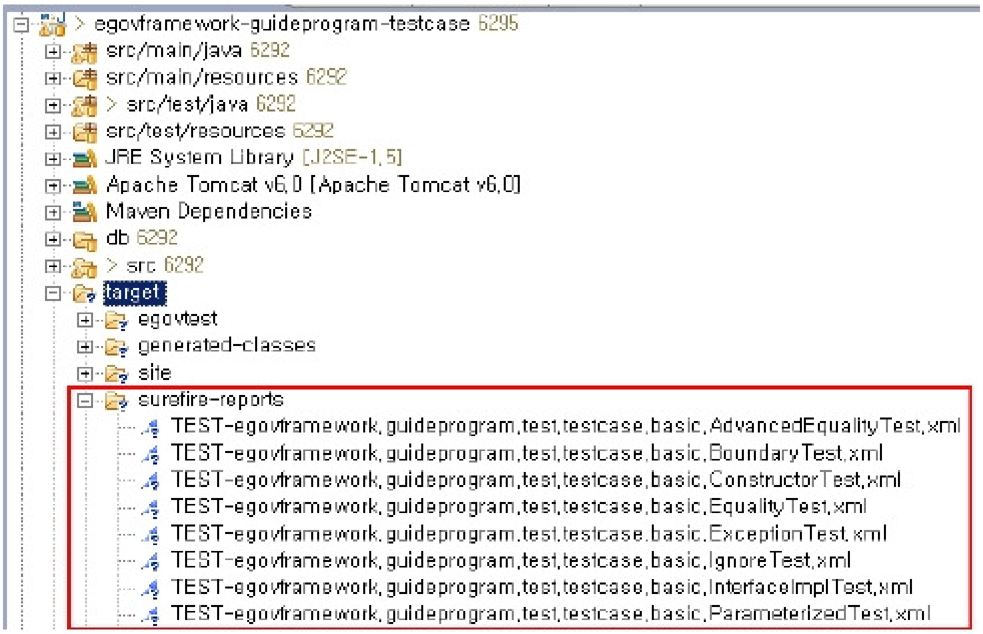
Follow below to automate the test using Maven when building.

1. write pom.xml : pom.xml configuration sample for Test Automation.: write as test.

2.run test as m2eclipse: Select Run As > Maven test, check BUILD SUCCESSFUL in Console View.



3. Check XML Report : <reportFormat>xml</reportFormat> item specifies either xml, or txt. File is created as TEST-{TestcaseFullName}.xml.

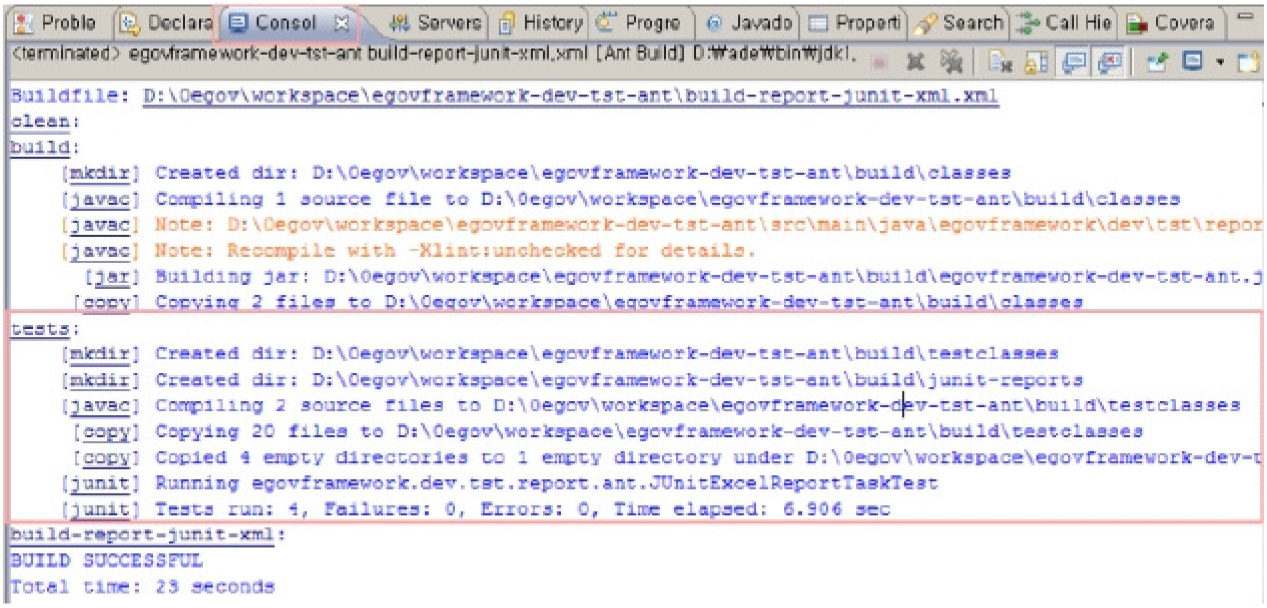


```
> egovframework-guideprogram-testcase 6295
  src/main/java 6292
  src/main/resources 6292
  > src/test/java 6292
  src/test/resources 6292
  JRE System Library [J2SE-1.5]
  Apache Tomcat v6.0 [Apache Tomcat v6.0]
  Maven Dependencies
  db 6292
  > src 6292
  target
    egovtest
    generated-classes
    site
    surefire-reports
      TEST-egovframework.guideprogram.test.testcase.basic.AdvancedEqualityTest.xml
      TEST-egovframework.guideprogram.test.testcase.basic.BoundaryTest.xml
      TEST-egovframework.guideprogram.test.testcase.basic.ConstructorTest.xml
      TEST-egovframework.guideprogram.test.testcase.basic.EqualityTest.xml
      TEST-egovframework.guideprogram.test.testcase.basic.ExceptionTest.xml
      TEST-egovframework.guideprogram.test.testcase.basic.IgnoreTest.xml
      TEST-egovframework.guideprogram.test.testcase.basic.InterfaceImplTest.xml
      TEST-egovframework.guideprogram.test.testcase.basic.ParameterizedTest.xml
```

Automated test using Ant

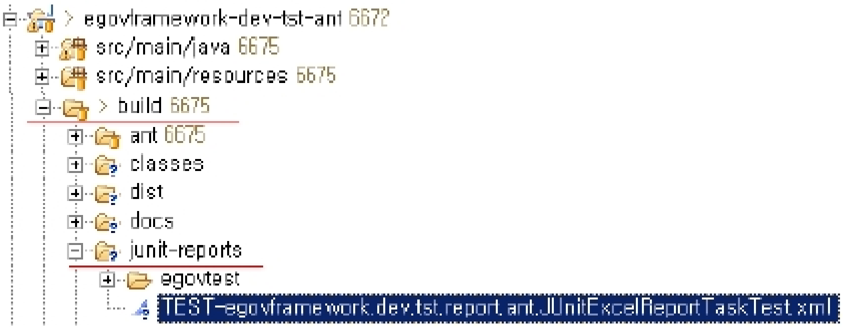
If using Ant to do personal builds, you can automate your tests as below.

1. build.xml : configuring build.xml for test automation: identical to JUnit.
2. Run tests in Eclipse
   I. Use Ant in Eclipse to run target.
   II. Check results in Console View.



```
<terminated> egovframework-dev-tst-ant build-report-junit-xml.xml [Ant Build] D:₩ade₩bin₩jdk1.
Buildfile: D:\0egov\workspace\egovframework-dev-tst-ant\build-report-junit-xml.xml
clean:
build:
    [mkdir] Created dir: D:\0egov\workspace\egovframework-dev-tst-ant\build\classes
    [javac] Compiling 1 source file to D:\0egov\workspace\egovframework-dev-tst-ant\build\classes
    [javac] Note: D:\0egov\workspace\egovframework-dev-tst-ant\src\main\java\egovframework\dev\tst\repor
    [javac] Note: Recompile with -Xlint:unchecked for details.
      [jar] Building jar: D:\0egov\workspace\egovframework-dev-tst-ant\build\egovframework-dev-tst-ant.j
     [copy] Copying 2 files to D:\0egov\workspace\egovframework-dev-tst-ant\build\classes
tests:
    [mkdir] Created dir: D:\0egov\workspace\egovframework-dev-tst-ant\build\testclasses
    [mkdir] Created dir: D:\0egov\workspace\egovframework-dev-tst-ant\build\junit-reports
    [javac] Compiling 2 source files to D:\0egov\workspace\egovframework-dev-tst-ant\build\testclasses
     [copy] Copying 20 files to D:\0egov\workspace\egovframework-dev-tst-ant\build\testclasses
     [copy] Copied 4 empty directories to 1 empty directory under D:\0egov\workspace\egovframework-dev-t
    [junit] Running egovframework.dev.tst.report.ant.JUnitExcelReportTaskTest
    [junit] Tests run: 4, Failures: 0, Errors: 0, Time elapsed: 6.906 sec
build-report-junit-xml:
BUILD SUCCESSFUL
Total time: 23 seconds
```

3. Check XML report: In junit task you can set xml, txt, or brief for <formatter type=“xml”/> item, and for above examples, TEST-{TestcaseFullName}.xml will be generated under build/junit-reports directory.



```
> egovframework-dev-tst-ant 6672
  src/main/java 6675
  src/main/resources 5675
  > build 6675
    ant 6675
    classes
    dist
    docs
    junit-reports
      egovtest
      TEST-egovframework.dev.tst.report.ant.JUnitExcelReportTaskTest.xml
```

Sample

Following is an example of how to create Maven and Ant scripts.

pom.xml configuration sample for Test Automation : test

Following example is pom.xml configuration for running test automatically.

```
<build>
        <!-- goal should be 'test'. -->
        <plugins>
                <!-- test -->
                <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-surefire-plugin</artifactId>
                        <configuration>
                                <skipTests>false</skipTests>
                                <forkMode>always</forkMode>
                                <reportFormat>xml</reportFormat>
                                <excludes>
                                        <exclude>**/Abstract*.java</exclude>
                                </excludes>
                                <includes>
                                        <include>**/*Test.java</include>
                                </includes>
                        </configuration>
                </plugin>
        </plugins>
</build>
```

build.xml configuration sample for Test Automation: junit

```
<property name="testreports.dir" value="build/junit-reports" />
. . . 중략 . . .
<junit forkmode="perBatch" printsummary="true" haltonfailure="yes" haltonerror="yes">
    <classpath refid="master-classpath" />
    <classpath refid="test-classpath" />
    <classpath path="${testbuild.dir}" />
    <formatter type="xml" />
    <batchtest fork="yes" todir="${testreports.dir}">
        <fileset dir="${testbuild.dir}">
            <include name="**/*Test.class" />
            <exclude name="**/Abstract*Test.class" />
        </fileset>
    </batchtest>
</junit>
```

References

- Maven Surefire Plugin Home : http://maven.apache.org/plugins/maven-surefire-plugin/ [http://maven.apache.org/plugins/maven-surefire-plugin/]
- Maven Surefire Report Plugin Home : http://maven.apache.org/plugins/maven-surefire-report-plugin/ [http://maven.apache.org/plugins/maven-surefire-report-plugin/]
- Ant JUnit Task : http://ant.apache.org/manual/OptionalTasks/junit.html [http://ant.apache.org/manual/OptionalTasks/junit.html]
- Ant JUnit Report Task : http://ant.apache.org/manual/OptionalTasks/junitreport.html [http://ant.apache.org/manual/OptionalTasks/junitreport.html]

[1] XML files are generated after using the build tools to run tests. Raw data gets stored as txt, xml files, and you can set the destination folder for them; however in case of Maven, the default target/surefire-report folder will be used.