

Guión Práctica 4. Sesión S.4.2

Gestión de Pruebas

<u>1</u>	<u>INTRODUCCIÓN-----</u>	<u>2</u>
<u>2</u>	<u>CONFIGURACIÓN DEL ENTORNO DE PRUEBAS-----</u>	<u>2</u>
<u>3</u>	<u>DEFINIR UN PLAN DE PRUEBAS E IMPLEMENTARLO PARA VUESTRO PROYECTO. -----</u>	<u>10</u>
3.1	EVALUACIÓN-----	10

1 Introducción

En esta sesión de laboratorio el objetivo es instalar el entorno de pruebas basado en JUnit gestionando las dependencias con Maven, para así poder definir e implementar el plan de pruebas para vuestro proyecto. La sesión se estructura en los siguientes puntos:

- Parte 1 enseña como añadir las dependencias y plugins necesarios al ***pom.xml***
- Parte 2 muestra brevemente como codificar tests en Junit.

2 Configuración del entorno de pruebas¹

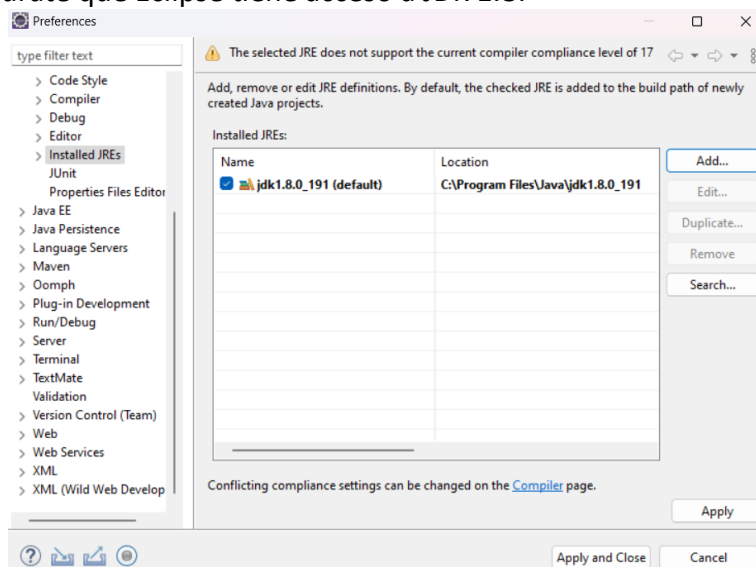
Para ilustrar como añadir todas las dependencias y plugins necesarios usaremos un pequeño proyecto en github que se puede encontrar aquí: <https://github.com/pmisarwala/JunitExample>

1. Clonar el repositorio git en Eclipse
2. Importar el proyecto como un proyecto maven
3. Actualizar dependencias del proyecto (Alt+F5)
4. Ejecutar *mvn clean install*
5. Comprobar que el proyecto está bien construido y que se han generado los informes de prueba.

Ahora se debería poder usar el proyecto de ejemplo en tu workspace, así que ahora añadamos los plugins necesarios. Añade las siguientes propiedades de compilación al *pom.xml* para asegurarte que se compila con la versión 8.

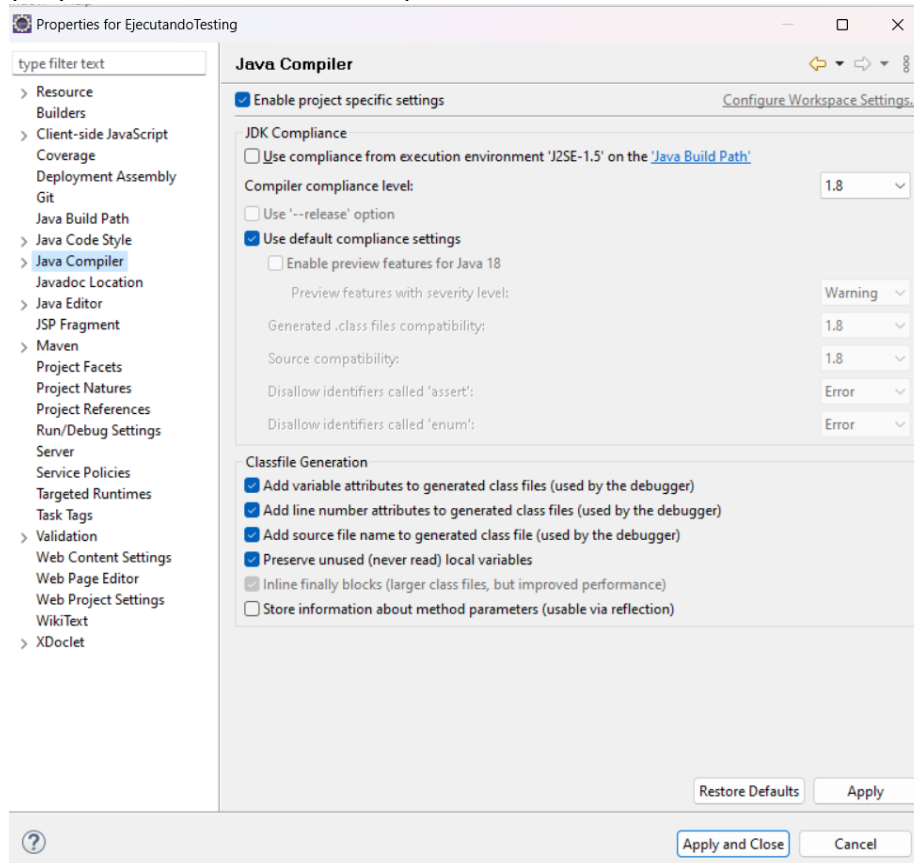
```
<properties>
  <maven.compiler.source>8</maven.compiler.source>
  <maven.compiler.target>8</maven.compiler.target>
</properties>
```

Además, asegúrate que Eclipse tiene acceso a JDK 1.8:



¹ **Importante:** Este tutorial funciona con versiones Java 1.8 o anteriores. Asegúrate de tener instalado el JDK 1.8 y tenerlo seleccionado en tu IDE.

Y que el proyecto usa ese nivel de compilación:



Lo siguiente es añadir información de los plugins y de las dependencias al **pom.xml**. La estructura del pom tiene una sección *build* donde podemos añadir ciertos plugins que se ejecutaran con ciertos *goals* como *package* o *install*. Además, podemos definir una sección *reporting* fuera de la etiqueta *build* donde añadir otros plugins que se ejecutaran bajo el *goal site:site*. Este *goal* de Maven está pensado para crear un sitio html de desarrollo donde incluir diferentes informes sobre el sistema.

Para la gestión de pruebas nosotros usaremos una dependencia y 3 plugins. Primero la dependencia JUnit, el cual es un framework de pruebas que usaremos para codificar las pruebas (<http://junit.org/junit4/>).

Ejemplo:

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

Luego tenemos que configurar los informes de prueba *Surefire* y *JaCoCo* en el fichero pom. Primero de todo tenemos que *añadir maven-project-info-reports-plugin* (<https://maven.apache.org/plugins/maven-project-info-reports-plugin/>) que es un plugin que habilita en sí la generación de informes.

1. Añade en el fichero pom, fuera de la etiqueta *build* un elemento *reporting*, y luego *plugins* y el *plugin* para la generación de informes:

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-project-info-reports-plugin</artifactId>
      <version>2.7</version>
    </plugin>
  </plugins>
</reporting>
```

2. Luego añadimos el plugin *Surefire* que será usado durante la fase de *test* del ciclo de vida de generación en Maven para crear un informe de la ejecución de los test. (<http://maven.apache.org/surefire/maven-surefire-plugin/>). Básicamente muestra los resultados de los test JUnit.

```
<reporting>
  <plugins>
    ...
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-report-plugin</artifactId>
      <configuration>
        <skipEmptyReport>>false</skipEmptyReport>
      </configuration>
    </plugin>
  </plugins>
</reporting>
```

3. *JaCoCo* es una librería Java creada por el equipo de EclEmma para la evaluación de cobertura de sentencia y se puede integrar con varias librerías de pruebas como JUnit. (<http://www.eclemma.org/jacoco/>). EclEmma era la versión del plugin para Eclipse que ahora está disponible como una herramienta de generación de informes html.

```
<reporting>
  <plugins>
    ...
    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <reportSets>
        <reportSet>
          <reports>
            <report>report</report>
          </reports>
        </reportSet>
      </reportSets>
    </plugin>
  </plugins>
</reporting>
```

4. Además, la ejecución de JaCoCo tiene que configurarse en la etiqueta *build* del fichero pom. El siguiente fragmento está extraído directamente de la página de uso del plugin. Entre otras cosas que podéis definir se encuentra el límite de cobertura deseada, que en el ejemplo se pone como 60%.

```
<build>
...
<plugins>
  <plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.7.9</version>
    <executions>
      <execution>
        <id>default-prepare-agent</id>
        <goals>
          <goal>prepare-agent</goal>
        </goals>
      </execution>
      <execution>
        <id>default-prepare-agent-integration</id>
        <goals>
          <goal>prepare-agent-integration</goal>
        </goals>
      </execution>
      <execution>
        <id>default-report</id>
        <goals>
          <goal>report</goal>
        </goals>
      </execution>
      <execution>
        <id>default-report-integration</id>
        <goals>
          <goal>report-integration</goal>
        </goals>
      </execution>
      <execution>
        <id>default-check</id>
        <goals>
          <goal>check</goal>
        </goals>
        <configuration>
          <rules>
            <!-- implementation is needed only for Maven 2 -->
            <rule implementation="org.jacoco.maven.RuleConfiguration">
              <element>BUNDLE</element>
              <limits>
                <!-- implementation is needed only for Maven 2 -->
                <limit implementation="org.jacoco.report.check.Limit">
                  <counter>COMPLEXITY</counter>
                  <value>COVEREDRATIO</value>
                  <minimum>0.60</minimum>
                </limit>
              </limits>
            </rule>
          </rules>
        </configuration>
      </execution>
    </executions>
  </plugin>
</plugins>
```

En resumen, el fichero pom para el proyecto de ejemplo quedaría como el siguiente:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mslc</groupId>
  <artifactId>JUnitExample</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>JUnitExample Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <!-- https://mvnrepository.com/artifact/junit/junit -->
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
    </dependency>
  </dependencies>
  <build>
    <finalName>JUnitExample</finalName>
    <plugins>
      <plugin>
        <groupId>org.jacoco</groupId>
        <artifactId>jacoco-maven-plugin</artifactId>
        <version>0.7.9</version>
        <executions>
          <execution>
            <id>default-prepare-agent</id>
            <goals>
              <goal>prepare-agent</goal>
            </goals>
          </execution>
          <execution>
            <id>default-prepare-agent-integration</id>
            <goals>
              <goal>prepare-agent-integration</goal>
            </goals>
          </execution>
          <execution>
            <id>default-report</id>
            <goals>
              <goal>report</goal>
            </goals>
          </execution>
          <execution>
            <id>default-report-integration</id>
            <goals>
              <goal>report-integration</goal>
            </goals>
          </execution>
          <execution>
            <id>default-check</id>
            <goals>
              <goal>check</goal>
            </goals>
            <configuration>
              <rules>
                <!-- implementation is needed only for Maven 2 -->
                <rule implementation="org.jacoco.maven.RuleConfiguration">
                  <element>BUNDLE</element>
                  <limits>
                    <!-- implementation is needed only for Maven 2 -->
                    <limit implementation="org.jacoco.report.check.Limit">
                      <counter>COMPLEXITY</counter>
                      <value>COVEREDRATIO</value>
                      <minimum>0.60</minimum>
                    </limit>
                  </limits>
                </rule>
              </rules>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
```

```
<artifactId>maven-project-info-reports-plugin</artifactId>
<version>2.7</version>
</plugin>

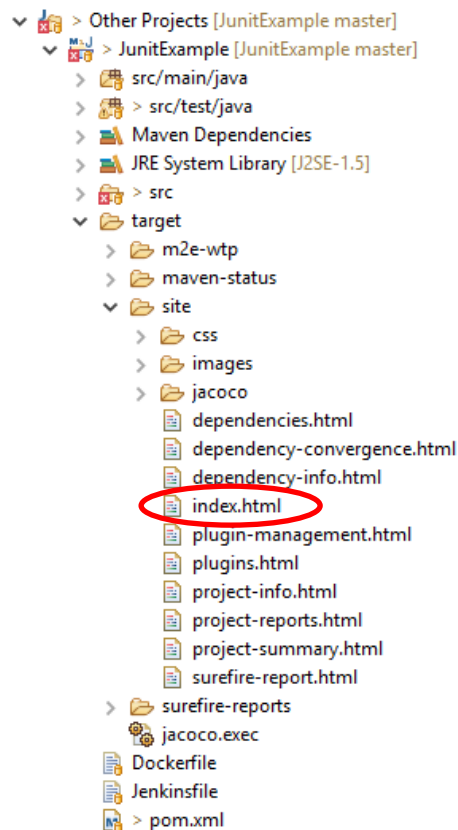
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-surefire-report-plugin</artifactId>
<configuration>
<skipEmptyReport>>false</skipEmptyReport>
</configuration>
</plugin>
<plugin>
<groupId>org.jacoco</groupId>
<artifactId>jacoco-maven-plugin</artifactId>
<reportSets>
<reportSet>
<reports>
<report>report</report>
</reports>
</reportSet>
</reportSets>
</plugin>

<!-- commented for lab session concerning maintenance -->
<!--
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-pmd-plugin</artifactId>
<configuration>
<skipEmptyReport>>false</skipEmptyReport>
</configuration>
</plugin>
<plugin>
<groupId>org.codehaus.mojo</groupId>
<artifactId>findbugs-maven-plugin</artifactId>
<configuration>
<skipEmptyReport>>false</skipEmptyReport>
</configuration>
</plugin>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-checkstyle-plugin</artifactId>
<configuration>
<skipEmptyReport>>false</skipEmptyReport>
</configuration>
</plugin>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-jxr-plugin</artifactId>
</plugin>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-javadoc-plugin</artifactId>
</plugin>
-->

</plugins>
</reporting>

</project>
```

Para ejecutar Maven y generar los informes hay que ejecutar **mvn site:site** y como resultado se obtiene cierto código html para el sitio del proyecto en la ruta `target/site/index.html`



Después podéis comprobar el informe *surefire* donde tenéis los tests que han pasado satisfactoriamente y los erróneos. Además, se puede ver el informe de cobertura *JaCoCo* donde se pueden ir desagregando los resultados hasta nivel de clase y finalmente ver las sentencias cubiertas en verde.

JUnitExample Maven Webapp

Project Documentation

- Project Information
- Project Reports
- Surefire Report
- Jacoco

Surefire Report

Summary

[Summary] [Package List] [Test Cases]

Tests	Errors	Failures	Skipped	Success Rate	Time
2	0	0	0	100%	0

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Package List

[Summary] [Package List] [Test Cases]

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
com.msic	2	0	0	0	100%	0




Note: package statistics are not computed recursively, they only sum up all of its test suites numbers.

com.msic





Class	Tests	Errors	Failures	Skipped	Success Rate	Time
TestHelloWorld	2	0	0	0	100%	0




JUnitExample Maven Webapp

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
com.msic	100%	100%	100%	0	5	0	8	0
Total	0 of 31	100%	0 of 2	100%	0	5	0	4

 JUnitExample Maven Webapp >  com.mslc >  HelloWorld

HelloWorld

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
 getMessage()	<div><div></div></div>	100%	<div><div></div></div>	100%	0	2	0	3	0	1
 HelloWorld()	<div><div></div></div>	100%	n/a	n/a	0	1	0	2	0	1
 setName(String)	<div><div></div></div>	100%	n/a	n/a	0	1	0	2	0	1
 getName()	<div><div></div></div>	100%	n/a	n/a	0	1	0	1	0	1
Total	0 of 31	100%	0 of 2	100%	0	5	0	8	0	4

 JUnitExample Maven Webapp >  com.mslc >  HelloWorld.java

HelloWorld.java

```
1. package com.mslc;
2.
3. public class HelloWorld {
4.     private String name = "";
5.
6.     public String getName()
7.     {
8.         return name;
9.     }
10.
11.     public String getMessage()
12.     {
13.         if (name == "")
14.         {
15.             return "Hello!";
16.         }
17.         else
18.         {
19.             return "Hello " + name + "!";
20.         }
21.     }
22.
23.     public void setName(String name)
24.     {
25.         this.name = name;
26.     }
27.
28. }
```

3 Definir un plan de pruebas e implementarlo para vuestro proyecto.

Se debe definir un plan de pruebas para vuestro proyecto e implementarlo usando *JUnit*. Recuerda que lo más importante y esencial es el diseño apropiado de las pruebas en base a las estrategias que se vieron en las sesiones teóricas más que la codificación en *Junit* en sí mismo.

Nota: os dejamos algunos ejemplos (algunos un poco más complejos):

- <https://github.com/ricpdc/gestormesas>
- <https://bitbucket.org/macariopolo/banco20193capas/src/master/>
- Este es un ejemplo muy sencillo: [git@github.com:IsmaelCaballero/EjecutandoTesting.git](https://github.com:IsmaelCaballero/EjecutandoTesting.git)

3.1 Evaluación

- **Objetivo:** Definir un plan de pruebas e implementarlo, asegurando como mínimo un 75% de cobertura
- **Qué se va a evaluar:**
 - Que se hayan definido test de acuerdo Junit
 - La calidad de los tests (consultar la sesión teórica)
 - Definición de casos de prueba.
 - Definición de valores de prueba
 - Cobertura alcanzada (nivel de sentencia).
 - Que se hayan incluido los plugins de Surefire y Jacoco para generar los informes.

Nota: Cuanta más cobertura se logre mejor nota se obtendrá, pero sin descuidar la calidad del diseño de las pruebas.

- **Cómo se evaluará:**
 - Tenéis que subir a *github* el proyecto con los test *JUnit* implementados.
 - Tenéis que integrar en la wiki del proyecto una página con los resultados de *Surefire* y *JaCoCo*.
Notad que los test se re-ejecutarán manualmente ya que es fácil modificar código html ;-).
 - Debéis escribir un informe breve con un resumen de la selección de casos de prueba y la definición de los valores de prueba y la estrategia que habéis seguido. El informe debería describir todos los tests que fallaron y la solución que hicisteis en vuestro código para acabar pasando dichos tests. Para facilitar esta comprobación podéis crear un Branch en *github* donde se puede ver el antes y el después.
 - Probablemente nos reuniremos para comprobar los resultados juntos.
 - **NOTA:** El código que se va a usar es el que implementéis a partir de vuestros trabajos de pruebas.

3.2 Entregando las prácticas

Para entregar las prácticas, os pedimos que hagáis las siguientes tareas:

- Cread, para cada proyecto del trabajo teórico 2 un nuevo repositorio en GitHub, incluyendo a todos los compañeros del grupo y a vuestros profesores de prácticas
- Enlaza los tres repositorios en vuestro repositorio principal

4 Anexo

Este es el fichero POM con las versiones actualizadas de los plugins para el ejemplo propuesto en [git@github.com:IsmaelCaballero/EjecutandoTesting.git](https://github.com/IsmaelCaballero/EjecutandoTesting.git)

```
project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.msic</groupId>
  <artifactId>EjecutandoTesting</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>Ejecutando Testing</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.resources.sourceEncoding>UTF-8</project.resources.sourceEncoding>
  </properties>
  <dependencies>
    <!-- https://mvnrepository.com/artifact/junit/junit -->
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.13.2</version>
    </dependency>

    <dependency>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.0.0-M7</version>
    </dependency>

    <dependency>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-report-plugin</artifactId>
      <version>3.0.0-M7</version>
    </dependency>

    <dependency>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-war-plugin</artifactId>
      <version>3.3.2</version>
      <type>maven-plugin</type>
    </dependency>

    <dependency>
      <groupId>org.apache.maven.doxia</groupId>
      <artifactId>doxia-site-renderer</artifactId>
      <version>1.9.2</version>
    </dependency>

    <dependency>
      <groupId>commons-logging</groupId>
      <artifactId>commons-logging</artifactId>
      <version>1.1.1</version>
    </dependency>

    <dependency>
      <groupId>commons-chain</groupId>
      <artifactId>commons-chain</artifactId>
      <version>1.2</version>
    </dependency>

    <dependency>
      <groupId>commons-digester</groupId>
      <artifactId>commons-digester</artifactId>
      <version>2.1</version>
    </dependency>
  </dependencies>
```

```

<build>
  <finalName>EjecutandoTesting</finalName>

  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-site-plugin</artifactId>
      <version>3.10.0</version>
    </plugin>

    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-project-info-reports-plugin</artifactId>
      <version>3.4.1</version>
    </plugin>

    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <version>0.8.7</version>
      <executions>
        <execution>
          <id>default-prepare-agent</id>
          <goals>
            <goal>prepare-agent</goal>
          </goals>
        </execution>

        <execution>
          <id>default-prepare-agent-integration</id>
          <goals>
            <goal>prepare-agent-integration</goal>
          </goals>
        </execution>
        <execution>
          <id>default-report</id>
          <goals>
            <goal>report</goal>
          </goals>
        </execution>
        <execution>
          <id>default-report-integration</id>
          <goals>
            <goal>report-integration</goal>
          </goals>
        </execution>
        <execution>
          <id>default-check</id>
          <goals>
            <goal>check</goal>
          </goals>
          <configuration>
            <rules>
              <rule implementation="org.jacoco.maven.RuleConfiguration">
                <element>BUNDLE</element>
                <limits>
                  <limit implementation="org.jacoco.report.check.Limit">
                    <counter>COMPLEXITY</counter>
                    <value>COVEREDRATIO</value>
                    <minimum>0.60</minimum>
                  </limit>
                </limits>
              </rule>
            </rules>
          </configuration>
        </execution>
      </executions>
    </plugin>

    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-javadoc-plugin</artifactId>
      <version>3.4.1</version>
    </plugin>
  </plugins>
</build>

<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-project-info-reports-plugin</artifactId>
      <version>3.4.1</version>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jxr-plugin</artifactId>
      <version>3.3.0</version>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-report-plugin</artifactId>
      <version>3.0.0-M7</version>
      <configuration>
        <skipEmptyReport>>false</skipEmptyReport>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <reportSets>

```

```
        <reportSet>
          <reports>
            <report>report</report>
          </reports>
        </reportSet>
      </reportSets>
    </plugin>
  </plugins>
</reporting>
</project>
```