Jacoco Plugin

What is Code Coverage

Code coverage is a quality measurement during development process to see how much of your code has been tested (or executed). Your unit test should cover most of business logic of your code, however, reaching 100% of code coverage does not mean that your code is bug-free. You can reach 100% code coverage while still having bugs.

What is JaCoCo

Jacoco maven plugin is a free code coverage library for Java projects. It is used to generate code coverage reports.

How it works is that JaCoCo plugin attaches a runtime agent to the JVM when it starts. The JaCoCo agent will then instrument the class so that it can see when the class is called what lines of codes are called during the testing process. It then build up the code coverage statistics during the testing phase.

How to setup JaCoCo

To set up Jacoco-maven-plugin you would need to include this following plugin in your pom.xml:

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.6</version>
</plugin>
```

The latest version you would need to determine based on what is released. Then you would also want to add the following executions tag in order to trigger Jacoco maven plugin whenever you run the unit test of your project.

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.6</version>
  <executions>
```

```
   <execution>
       <id>prepare-agent</id>
            <goals>
             <goal>prepare-agent</goal>
            </goals>
      </execution>
      <execution>
            <id>report</id>
            <phase>test</phase>
               <goals>
               <goal>report</goal>
               </goals>
        </execution>
    </executions>
</plugin>
```

- **Prepare-agent goal:** This goal prepares the Jacoco runtime agent to record the execution data, record the number of lines that's executed, backtraced, etc. This must be attached otherwise, Jacoco will not have data to generate the code coverage
- **Report goal**: This goal will finally create the code coverage reports from the execution data recorded by the runtime agent. Since this is attached to the test phase, whenever the test phase finishes it will generate the report automatically without having you to invoke the goal manually.

Add code coverage check

You can also enforce minimum code coverage check when you are executing the test

Toggle me for full code

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.10</version>
  <executions>
    <execution>
      <id>prepare-agent</id>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
```

```
    <execution>
      <id>jacoco-report uwu</id>
      <phase>test</phase>
      <goals>
        <goal>report</goal>
      </goals>
    </execution>
    <execution>
      <phase>test</phase>
      <id>coverage-check</id>
      <goals>
        <goal>check</goal>
      </goals>
      <configuration>
        <rules>
          <rule>
            <element>PACKAGE</element>
            <limits>
              <limit>
                <counter>LINE</counter>
                <value>COVEREDRATIO</value>
                <minimum>0.8</minimum>
              </limit>
            </limits>
          </rule>
        </rules>
      </configuration>
    </execution>
  </executions>
</plugin>
```

The goal will fail if your code coverage doesn't meet the specified percentage.

Ignore class files for coverage

You can add to the configuration the list of class files that you should ignore because it is difficult / unnecessary to test them for coverage.

```
<plugin>
  <groupId>org.jacoco</groupId>
```

```
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.8.10</version>
    <configuration>
      <excludes>
        <exclude>**/App.class</exclude>
      </excludes>
    </configuration>
  </plugin>
```

Revision #2
Created 30 September 2023 18:13:10 by Tamarine
Updated 30 September 2023 18:34:46 by Tamarine