A better Java/Maven project

Table of Contents

Version 1.18-SNAPSHOT 2018-08-16 :toc: :toclevels: 3

Table 1. History

Date	Author	Detail Unresolved directive in subdocs/_init.adoc - include::D:\workspaceJava\cg-wm\target/generated-docs/history/better-java-project.adoc.psv[]

1. Eclipse plugins

TODO : put it inline here

Developer Guide

2. Useful Java libraries

2.1. Mockito / PowerMockito

Usage for static classes

```
@RunWith(PowerMockRunner.class)
@PrepareForTest({ TypeUtils.class })
@PowerMockIgnore("javax.management.*")
public class OpenPojoWebTest {

    @Before
    public void before() throws Exception {
        PowerMockito.mockStatic(TypeUtils.class);
        PowerMockito.when(TypeUtils.setterDate((Date) Mockito.any(), (Date) Mockito.any()))
                .thenAnswer(invocation -> invocation.getArgumentAt(1, Date.class));
    }

}
```

2.2. OpenPojo : Auto test Pojo classes for coverage

https://github.com/OpenPojo/openpojo

OpenPojo au tests Pojo classes, especially getters and setters. Very handy for large beans / auto generated classes for whom testing is boring.

Usage

```
import com.openpojo.reflection.filters.FilterNonConcrete;
import com.openpojo.validation.Validator;
import com.openpojo.validation.ValidatorBuilder;
import com.openpojo.validation.test.impl.GetterTester;
import com.openpojo.validation.test.impl.SetterTester;

public class OpenPojoTest {

    public static void validateBeans(String javaPackage) {
        Validator validator = ValidatorBuilder.create().with(new SetterTester()).with(new GetterTester()).build();
        //exclude enums, abstracts, interfaces
        validator.validateRecursively(javaPackage, new FilterNonConcrete());
    }

    @Test ①
    public void testPojoRecursiv() {
        // recursive
        validateBeans("my.full.java.package.with.sub.packages");
    }

    @Test ②
    public void testExludingSomeClasses() {
        List<PojoClass> listOfPojoClassInDto = PojoClassFactory.getPojoClasses("my.full.java.package.with.sub.packages",
null);
        listOfPojoClassInDto.remove(PojoClassFactory.getPojoClass(SomeSpecialClassNotToTest.class));
        validator.validate(listOfPojoClassInDto);
    }

}
```

① Fully recursive example

② Excluding some classes

Maven dependency

```
    <dependency>
        <groupId>com.openpojo</groupId>
        <artifactId>openpojo</artifactId>
        <version>0.8.6</version>
        <scope>test</scope>
    </dependency>
```

2.3. SLF4J : Abstract logging

Maven dependencies

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.21</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>jcl-over-slf4j</artifactId>
    <version>1.7.21</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.7</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.7</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-slf4j-impl</artifactId>
    <version>2.7</version>
</dependency>
```

2.4. Aspect4log : Logging functions starts/stops with inputs/outputs

http://aspect4log.sourceforge.net

Use Aspect4Log, which logs functions start/stop with inputs/outputs using AOP.

Result log example

```
07-31_14:13:48.491 DEBUG org.a.utils.ConfigUtils        - > getParameter(test)
07-31_14:13:48.491 DEBUG org.a.utils.wmcall.WmHelper    - >  getPackageName(true)
07-31_14:13:48.492 DEBUG g.a.utils.wmcall.WmCallEclipse - >      getPackageName(true)
07-31_14:13:48.492 DEBUG g.a.utils.wmcall.WmCallEclipse - .         getPackageName(true) -> DEFAULT
07-31_14:13:48.492 DEBUG org.a.utils.wmcall.WmHelper    - .     getPackageName(true) -> DEFAULT
07-31_14:13:48.492 DEBUG org.a.utils.ConfigUtils        - >  getParameter(DEFAULT, test)
07-31_14:13:48.494 DEBUG persistence.PersistenceManager - >      findParameterValue(test, MONO_IS)
07-31_14:13:48.500 DEBUG persistence.PersistenceManager - .         findParameterValue(test, MONO_IS) -> (null)
07-31_14:13:48.501 DEBUG org.a.utils.file.ConfigReader  - >      getValueFromConfigFile(DEFAULT, test)
07-31_14:13:48.501 DEBUG org.a.utils.file.ConfigReader  - >          getValueFromConfigFile(DEFAULT, config.properties,
test)
07-31_14:13:48.501 DEBUG org.a.utils.file.ConfigReader  - >              getConfigFileKeyValues(DEFAULT,
config.properties)
07-31_14:13:48.501 DEBUG org.a.utils.file.ConfigReader  - >                  getConfigPath(DEFAULT)
07-31_14:13:48.502 DEBUG org.a.utils.wmcall.WmHelper    - >                      getServerConfigFolder()
07-31_14:13:48.502 DEBUG g.a.utils.wmcall.WmCallEclipse - >                          getServerConfigFolder()
07-31_14:13:48.502 DEBUG g.a.utils.wmcall.WmCallEclipse - .                             getServerConfigFolder() ->
src/test/resources/config
07-31_14:13:48.502 DEBUG org.a.utils.wmcall.WmHelper    - .                         getServerConfigFolder() ->
src/test/resources/config
07-31_14:13:48.503 DEBUG org.a.utils.file.ConfigReader  - .                     getConfigPath(DEFAULT) ->
src/test/resources/config/packages/DEFAULT
07-31_14:13:48.503 DEBUG org.a.utils.file.ConfigReader  - .                 getConfigFileKeyValues(DEFAULT,
config.properties) -> {unitTest=OK, MaxAnomaliesSelectedForResubmission=100, useDbParameters=false}
07-31_14:13:48.504 DEBUG org.a.utils.file.ConfigReader  - .             getValueFromConfigFile(DEFAULT,
config.properties, test) -> (null)
07-31_14:13:48.504 DEBUG org.a.utils.file.ConfigReader  - .         getValueFromConfigFile(DEFAULT, test) -> (null)
07-31_14:13:48.505 DEBUG org.a.utils.ConfigUtils        - .     getParameter(DEFAULT, test) -> (null)
07-31_14:13:48.506 DEBUG org.a.utils.ConfigUtils        - . getParameter(test) -> (null)
```

LOGGER declaration

```java
import net.sf.aspect4log.Log;
import static net.sf.aspect4log.Log.Level.TRACE;

@Log ①
public class FooDao {

    public void tooLowLevelFunction(){ ②
        //[...]
    }

    @Log(enterLevel = Level.TRACE, exitLevel = Level.TRACE) ③
    public void delete(String foo) {
        //[...]
    }

    @Log(argumentsTemplate = "[...skipped...]", resultTemplate = "[...skipped...]") ④
    public void find(String bigXML) {
        //[...]
    }

    @Log(on = { @Exceptions(exceptions = { CgException.class }, level = Level.INFO) }) ⑤
    public void saveOrUpdate(String foo) {
        //[...]
    }
}
```

① @Log on a class will affect every methods not annotated

② So this method will be logged, in DEBUG by default

③ Lower the level to TRACE if some methods pollute the logs

④ You can skip only the arguments/results if they are too verbose

⑤ Some advanced functionnality are available, see the website

For runtime, have log4j & aspect4log configuration files in the classpath, examples : link:log4j2.xml & link:aspect4log.xml.

Dependencies

```xml
<dependencies>
    <!-- for @Log -->
    <dependency>
        <groupId>net.sf.aspect4log</groupId>
        <artifactId>aspect4log</artifactId>
        <version>1.0.7</version>
    </dependency>
    <!-- AspectJ for instrumentation -->
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjrt</artifactId>
        <version>1.8.9</version>
    </dependency>
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjtools</artifactId>
        <version>1.8.9</version>
    </dependency>
</dependencies>

<plugins>
    <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>aspectj-maven-plugin</artifactId>
        <version>1.7</version>
        <executions>
            <execution>
                <goals>
                    <goal>compile</goal>
                </goals>
            </execution>
        </executions>
        <configuration>
            <showWeaveInfo>false</showWeaveInfo>
            <Xlint>adviceDidNotMatch=ignore,noGuardForLazyTjp=ignore</Xlint>
            <aspectLibraries>
                <aspectLibrary>
                    <groupId>net.sf.aspect4log</groupId>
                    <artifactId>aspect4log</artifactId>
                </aspectLibrary>
            </aspectLibraries>
        </configuration>
        <dependencies>
            <dependency>
                <groupId>org.aspectj</groupId>
                <artifactId>aspectjtools</artifactId>
                <version>1.8.9</version>
            </dependency>
        </dependencies>
    </plugin>
</plugins>
```

2.5. Log methods duration

2.5.1. using JCabi @Loggable

https://aspects.jcabi.com/annotation-loggable.html

With AOP, get selected methods duration :

```
2016-10-11 14:22:52.716 [main] INFO  PERFORMANCES - #setTestMode(...): in 30,51ms
2016-10-11 14:22:52.857 [main] INFO  PERFORMANCES - #setTestMode(...): in 1,20ms
```

Loggable example

```
@Loggable(skipArgs = true, skipResult = true, name = "PERFORMANCES")
public static void topLevelJarFunction(IData pipeline) throws ServiceException {
    //[...]
}
```

3. Best practices

3.1. Java

3.1.1. Java packages & classes naming

- Best package organization is by fonctionnality first, and then technically when many classes of the same type
- Always put classes in subpackage of the project
  - If a java project is **bar-a-b**, all packages are **mycorp.bar.a.b.***
- Don't use different packages for a few classes, regroup them (if below or equal 3 classes by package)
- Don't put in the class name what is already in the package name, except for too generic file name

Some naming conventions

http://stackoverflow.com/questions/3226282/are-there-best-practices-for-java-package-organisation
http://www.javapractices.com/topic/TopicAction.do?Id=205

Some widely used examples

http://commons.apache.org/proper/commons-lang/javadocs/api-2.6/overview-tree.html
https://commons.apache.org/proper/commons-lang/apidocs/overview-tree.html

3.1.2. Java 7 try with closable objects

Before Java 7, you had to close() streams and other closable objects in a try/catch/finally. Now Java handles everything if you use the right pattern :

try-with-resource

```
try (
    ZipOutputStream zos = new ZipOutputStream(new FileOutputStream(dstDirectory + "/" + fileName + ".zip"));
    FileInputStream in = new FileInputStream(foundFile.getAbsolutePath())
    ) {
    ZipEntry ze = new ZipEntry(fileName);
    zos.putNextEntry(ze);

    int len;
    while ((len = in.read(buffer)) > 0) {
        zos.write(buffer, 0, len);
    }

    if (delete)
        foundFile.delete();
} catch (IOException e) {
    LOGGER.error("Unable to zip or delete the file=" + srcDirectory + "/" + fileName + ", dest=" + dstDirectory, e);
    throw e;
}
```

3.1.3. Static Java Maps

When a **Map** is static (and then accessed by multiple threads), declare it Map and instantiate it **ConcurrentHashMap** :

Thread-safe Map

```
Map<a,b> myMap == new ConcurrentHashMap<>();
```

Idem for a **Set** but this is a bit tricky :

Thread-safe Set

```
Set<String>
mySet = Collections.newSetFromMap(new ConcurrentHashMap<String,Boolean>());
```

3.1.4. Init on demand

For objects used by static functions, try to initialize them only once and do it in thread safe mode.

Init on demand pattern

```
public class Something {
    private Something() {}

    private static class LazyHolder {
        private static final Something INSTANCE = new Something();
    }

    public static Something getInstance() {
        return LazyHolder.INSTANCE;
    }
}
```

3.1.5. Enum and String

A String from an Enum must be used with a custom **toString()**, never with **getName()** or default

toString().

Enum.toString() pattern

```
// Natures d echange
public enum EsbNatureType {
     DIFFUSION_FICHIER("DiffusionFichier"), DIFFUSION_MESSAGES("DiffusionMessages");

     private String name = null;

     EsbNatureType(String nameString) {
            this.name = nameString;
     }

     @Override
     public String toString() {
            return this.name;
     }

};
```

If you don’t do this way, we loose the flexibility to rename either the Enum or the String.

3.1.6. MyEnum.toEnum(String)

Comment déclarer l’Enum :

toEnum pattern

```
public enum ServiceOption {
    COMPLEMENTS,
    RESTRICTIONS,
    RISQUES,
    SNGI_EM_DECEDE,
    SNGI_EM_NON_IDENT,
    SNGI_ID_OBLIGATOIRE,
    DCR,
    DCR_STATUT,
    DCR_DELAI,
    LISTE_PSORTANTS,
    ID_TIERS,
    NOM_FLUX_SORTIE,
    ABO_ACTIF,
    DENOM_METIER,
    REF_ABO,
    UNKNOWN;

    public static ServiceOption toEnum(String optionName) {
        switch (optionName) {
        case "priseEnCptLstRisque":
            return RISQUES;
        case "priseEnCptLstCompl":
            return COMPLEMENTS;
        default:
            return UNKNOWN;
        }
    }
}
```

Puis ton Builder tu fais un **ServiceOption.toEnum(tonOption)** et le tour est joué.

Never write files outside of target/

3.2. Maven

3.2.1. Config files location

Config files have to be put in the right folder in Eclipse.

- **src/main/resources/**
  - Only files that will is not likely to be modified, because it will be in the jar
- **config/**
  - Files that is likely to be modified on IS
  - Don’t forget to put them manually on IS
- **src/test/resources/**
  - File used in JUnit tests only for this sub-module
- **../src/test/shared-resources**
  - Files used in JUnit tests accross multiple modules
  - requires some maven configuration

⚠ TODO give Maven details for shared-resources

3.2.2. Checkstyle : check javadoc

With Checkstyle, you can enforce continuous javadoc check

pom.xml plugin

```xml
<!-- checkstyle to fail the build on javadoc warnings -->
<!-- to skip : mvn install -Dcheckstyle.skip=true -->
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-checkstyle-plugin</artifactId>
    <version>2.17</version>
    <executions>
        <execution>
            <id>validate</id>
            <phase>validate</phase>
            <configuration>
                <configLocation>checkstyle-javadoc.xml</configLocation>
                <encoding>UTF-8</encoding>
                <consoleOutput>true</consoleOutput>
                <failsOnError>true</failsOnError>
                <linkXRef>false</linkXRef>
            </configuration>
            <goals>
                <goal>check</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

checkstyle-javadoc.xml to be created in the root project

```xml
<?xml version="1.0"?>
<!DOCTYPE module PUBLIC
    "-//Puppy Crawl//DTD Check Configuration 1.2//EN"
    "http://www.puppycrawl.com/dtds/configuration_1_2.dtd">
<module name="Checker">
    <module name="TreeWalker">
        <module name="JavadocMethod"/>
        <module name="JavadocType"/>
        <module name="JavadocVariable"/>
        <module name="JavadocStyle"/>
    </module>
</module>
```

3.2.3. Add version and date to Asciidoc PDFs

pom.xml

```xml
<plugins>

    <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>buildnumber-maven-plugin</artifactId>
        <version>1.2</version>
        <executions>
            <execution>
                <phase>validate</phase>
                <goals>
                    <goal>create-timestamp</goal>
                </goals>
            </execution>
        </executions>
        <configuration>
            <timestampFormat>yyyy-MM-dd</timestampFormat>
            <timestampPropertyName>build.date</timestampPropertyName>
        </configuration>
    </plugin>

    <!-- Ant tasks plugin -->
    <!-- single usage : mvn antrun:run -->
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-antrun-plugin</artifactId>
        <version>1.7</version>
        <inherited>true</inherited>
        <executions>
            <execution>
                <!-- add version to generated pdf filenames -->
                <id>pdfsAddVersion</id>
                <configuration>
                    <failOnError>false</failOnError>
                    <target name="add version and date to all generated pdf filenames">
                        <move todir="${project.build.directory}/generated-docs" includeemptydirs="false">
                            <fileset dir="${project.build.directory}/generated-docs" />
                            <mapper type="glob" from="*.pdf" to="*_V${project.version}_${build.date}.pdf" />
                        </move>
                    </target>
                </configuration>
                <goals>
                    <goal>run</goal>
                </goals>
            </execution>
        </executions>
    </plugin>

</plugins>
```

3.2.4. Javadoc generation with UML diagrams

pom.xml

```
    <!-- javadoc html, fix or generate -->
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-javadoc-plugin</artifactId>
        <version>2.10.4</version>
        <configuration>
            <!-- usage : javadoc:javadoc or javadoc:jar -->
            <show>public</show>
            <reportOutputDirectory>${project.reporting.outputDirectory}</reportOutputDirectory>
            <destDir>javadoc</destDir>
            <!-- for UML diagram in javadoc:javadoc -->
            <!-- Locally : need http://www.graphviz.org/Download_windows.php to work -->
            <!-- and add "C:\Program Files (x86)\Graphviz\bin" to windows path -->
            <doclet>org.umlgraph.doclet.UmlGraphDoc</doclet>
            <docletArtifact>
                <groupId>org.umlgraph</groupId>
                <artifactId>umlgraph</artifactId>
                <version>5.6.6</version>
            </docletArtifact>
            <additionalparam>-views -attributes -visibility -types -enumerations -enumconstants</additionalparam>
            <useStandardDocletOptions>true</useStandardDocletOptions>
        </configuration>
    </plugin>
```

3.2.5. Install provided dependencies in local repository

pom.xml

```
<plugins>
    <!-- install WM jars in local repository -->
    <!-- part of mvn clean because maven check them early in the process -->
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-install-plugin</artifactId>
        <version>2.5.2</version>
        <!-- We do not want children attempting to install these jars to the repository -->
        <inherited>false</inherited>
        <executions>
            <execution>
                <id>wm-isclient95</id>
                <phase>clean</phase>
                <goals>
                    <goal>install-file</goal>
                </goals>
                <configuration>
                    <file>lib/wm9.5/wm-isclient-9.5.jar</file>
                    <groupId>webmethods</groupId>
                    <artifactId>wm-isclient</artifactId>
                    <version>9.5</version>
                    <packaging>jar</packaging>
                </configuration>
            </execution>
    </plugin>
</plugins>
```

3.2.6. To generate AsciiDoc PDF files

pom.xml

```xml
<plugins>
    <!-- to generate asciidoc pdf documents -->
    <!-- part of mvn install -->
    <!-- single usage : mvn asciidoctor:process-asciidoc -->
    <!-- We don't bind it to the official phase to choose the moment in Jenkins pipeline -->
    <plugin>
        <groupId>org.asciidoctor</groupId>
        <artifactId>asciidoctor-maven-plugin</artifactId>
        <version>1.5.5</version>
        <dependencies>
            <dependency>
                <groupId>org.asciidoctor</groupId>
                <artifactId>asciidoctorj-pdf</artifactId>
                <version>1.5.0-alpha.14</version>
            </dependency>
            <dependency>
                <groupId>org.asciidoctor</groupId>
                <artifactId>asciidoctorj-diagram</artifactId>
                <version>1.5.4</version>
            </dependency>
        </dependencies>
        <configuration>
            <backend>pdf</backend>
            <sourceDirectory>src/docs/asciidoc</sourceDirectory>
            <sourceHighlighter>rouge</sourceHighlighter>
            <requires>
                <require>asciidoctor-diagram</require>
            </requires>
            <!-- Attributes common to all output formats -->
            <attributes>
                <imagesdir>${project.build.directory}/generated-docs/images</imagesdir>
                <pdf-style>${user.dir}/src/docs/asciidoc/themes/cg-theme.yml</pdf-style>
                <icons>font</icons>
                <pagenums />
                <toc />
                <idprefix />
                <idseparator>-</idseparator>
                <!-- custom -->
                <source-dir>../../main/java</source-dir>
                <test-dir>../../test/java</test-dir>
                <project-version>${project.version}</project-version>
                <root-project-dir>${user.dir}</root-project-dir>
                <history-dir>${project.build.directory}/generated-docs/history</history-dir>
                <project-images-dir>${project.basedir}/src/main/resources/images</project-images-dir>
            </attributes>
        </configuration>
    </plugin>
</plugins>
```

3.2.7. SonarQube with Jacoco for coverage

https://www.sonarqube.org

SonarQube ensures code quality with static analysis and Jacoco checks code coverage.

pom.xml properties

```xml
<properties>
    <custom.ut.skip>${skipTests}</custom.ut.skip>
    <sonar.java.coveragePlugin>jacoco</sonar.java.coveragePlugin>
    <jacoco.reportPath>../target/jacoco.exec</jacoco.reportPath>
    <jacoco.itReportPath>../target/jacoco-it.exec</jacoco.itReportPath>
    <sonar.jacoco.reportPaths>${jacoco.reportPath}, ${jacoco.itReportPath}</sonar.jacoco.reportPaths>

<sonar.coverage.exclusions>**/WmCall.*,**/Broker*.*,**/UniversalMessaging*.*,**/MsgServerBroker.*,**/UmListener.*,**/Per
fLogger.*,**/elastic/*DataSender.*</sonar.coverage.exclusions>
    <sonar.host.url>http://localhost:9000</sonar.host.url>
    <sonar.scm.disabled>true</sonar.scm.disabled>
    <sonar.scm.provider>git</sonar.scm.provider>
</properties>
```

pom.xml without powermock static

```xml
<dependencies>

    <!-- For unit tests coverage in Sonar -->
    <dependency>
        <groupId>org.sonarsource.java</groupId>
        <artifactId>sonar-jacoco-listeners</artifactId>
        <version>4.9.0.9858</version>
        <scope>test</scope>
    </dependency>

</dependencies>

<plugins>

    <!-- SonarQube -->
    <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>sonar-maven-plugin</artifactId>
        <version>3.2</version>
    </plugin>

    <!-- handling unit tests coverage with Jacco -->
    <plugin>
        <groupId>org.jacoco</groupId>
        <artifactId>jacoco-maven-plugin</artifactId>
        <version>0.8.0</version>
        <executions>
            <execution>
                <id>pre-unit-test</id>
                <phase>test-compile</phase>
                <goals>
                    <goal>prepare-agent</goal>
                </goals>
                <configuration>
                    <destFile>${sonar.jacoco.reportPath}</destFile>
                    <dataFile>${sonar.jacoco.reportPath}</dataFile>
                    <append>true</append>
                </configuration>
            </execution>
            <execution>
                <id>prepare-jacoco-agent-it</id>
                <phase>pre-integration-test</phase>
                <goals>
                    <goal>prepare-agent-integration</goal>
                </goals>
                <configuration>
                    <destFile>${sonar.jacoco.itReportPath}</destFile>
                    <dataFile>${sonar.jacoco.itReportPath}</dataFile>
                    <append>true</append>
                </configuration>
            </execution>
        </executions>
    </plugin>

    <!-- Unit Tests -->
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <!-- version 2.19.1 is broken on jenkins -->
        <version>2.18.1</version>
        <configuration>
            <testFailureIgnore>false</testFailureIgnore>
            <runOrder>alphabetical</runOrder>
            <skipTests>${custom.ut.skip}</skipTests>
            <properties>
                <property>
                    <name>listener</name>
                    <value>org.sonar.java.jacoco.JUnitListener</value>
                </property>
            </properties>
        </configuration>
    </plugin>

    <!-- Integration Tests -->
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-failsafe-plugin</artifactId>
        <!-- version 2.19.1 is broken on jenkins -->
        <version>2.18.1</version>
        <configuration>
            <runOrder>alphabetical</runOrder>
            <properties>
                <property>
                    <name>listener</name>
                    <value>org.sonar.java.jacoco.JUnitListener</value>
                </property>
            </properties>
        </configuration>
        <executions>
            <execution>
                <id>integration-tests</id>
                <phase>integration-test</phase>
                <goals>
                    <goal>integration-test</goal>
                </goals>
            </execution>
            <!-- to exit in error on test fail -->
            <execution>
                <id>verify</id>
                <phase>verify</phase>
                <goals>
                    <goal>verify</goal>
                </goals>
            </execution>
        </executions>
    </plugin>

</plugins>
```

pom.xml with powermock : instrumentation in conflict, offline jacoco instrumentation is needed

```
<dependencies>

    <!-- For unit tests coverage in Sonar -->
    <dependency>
        <groupId>org.jacoco</groupId>
        <artifactId>org.jacoco.agent</artifactId>
        <classifier>runtime</classifier>
        <version>0.8.0</version>
```

```xml
            <scope>test</scope>
        </dependency>

    </dependencies>


    <plugins>

        <!-- SonarQube -->
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>sonar-maven-plugin</artifactId>
            <version>3.2</version>
        </plugin>

        <!-- handling unit tests coverage with Jacco -->
        <!-- offline instrumentation is mandatory when using other instrumentation framework such as PowerMock -->
        <!-- https://github.com/powermock/powermock/wiki/Code-coverage-with-JaCoCo -->
        <!-- to separate UT and IT : -->
        <!-- (1) mvn test jacoco:restore-instrumented-classes -->
        <!-- (2) mvn install -Dcustom.ut.skip=true -Dcheckstyle.skip=true -->
        <plugin>
            <groupId>org.jacoco</groupId>
            <artifactId>jacoco-maven-plugin</artifactId>
            <version>0.8.0</version>
            <executions>
                <execution>
                    <id>jacoco-instrument</id>
                    <phase>test-compile</phase>
                    <goals>
                        <goal>instrument</goal>
                    </goals>
                    <configuration>
                        <skip>${skipTests}</skip>
                    </configuration>
                </execution>
                <execution>
                    <id>jacoco-restore-instrumented-classes</id>
                    <phase>post-integration-test</phase>
                    <goals>
                        <goal>restore-instrumented-classes</goal>
                    </goals>
                    <configuration>
                        <skip>${skipTests}</skip>
                    </configuration>
                </execution>
            </executions>
        </plugin>

        <!-- Unit Tests -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <!-- version 2.19.1 is broken on jenkins -->
            <version>2.18.1</version>
            <configuration>
                <testFailureIgnore>false</testFailureIgnore>
                <runOrder>alphabetical</runOrder>
                <skipTests>${custom.ut.skip}</skipTests>
                <systemPropertyVariables>
                    <jacoco-agent.destfile>${jacoco.reportPath}</jacoco-agent.destfile>
                </systemPropertyVariables>
            </configuration>
        </plugin>

        <!-- Integration Tests -->
        <!-- usage full test : mvn integration-test -->
        <!-- usage only IT (but does not fill jacoco-it) : mvn test-compile failsafe:integration-test -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-failsafe-plugin</artifactId>
            <!-- version 2.19.1 is broken on jenkins -->
            <version>2.18.1</version>
            <configuration>
                <runOrder>alphabetical</runOrder>
                <systemPropertyVariables>
                    <jacoco-agent.destfile>${jacoco.itReportPath}</jacoco-agent.destfile>
                </systemPropertyVariables>
            </configuration>
            <executions>
                <execution>
                    <id>integration-tests</id>
                    <phase>integration-test</phase>
                    <goals>
                        <goal>integration-test</goal>
                    </goals>
                </execution>
                <execution>
                    <id>verify</id>
                    <phase>verify</phase>
                    <goals>
                        <goal>verify</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
```