POM File

Introduction to POM File

Project Object Model or POM is the starting point for a maven project. It is an XML file that contains details about the project and configurations consumed by Maven for project building processes. It has the values for a standard project which can be changed according to user preferences. When executing a task or goal in Maven terminology, Maven `searches for the POM file in the current directory.` It consumes the POM, fetches the needed configuration detail, then executes the goals.

POM can define the **project dependencies, the plugins or goals to be executed, the build profiles, and more**. Other information like the project version, description, developers, mailing lists, and such can also be specified.

There has to be a POM file in every Maven project and all POM files need at least the project tag and four other inner tags named as modelVersion, groupId, artifactId, and version (Last three called as GAV in short).

```
<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/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>

   <groupId>com.companyname.project-group</groupId>
   <artifactId>project</artifactId>
   <version>1.0</version>
</project>
```

The project tag is the root of the file. In the project tag, you need to describe the basic schema settings such as apache schema and w3.org specification.

The model version describes the `version of the Maven.`

Group Id is the `id of the project's group` (Simply it shows the company or the organization or the owner of the project). This should be long enough to give **uniqueness to the project.**

Artifact id is the `id for specifying the project` among others belonging to the same group. It mainly shows the name of the project like pet-clinic-server.

Version defines the `version number of the project`. Together with the groupId, it is used within an artifact's repository to separate versions.

Java 9 or Later

By default, your version of Maven might use an old version of the maven-compiler-plugin that is not compatible with Java 9 or later versions. To target Java 9 or later, you should at least use version 3.6.0 of the maven-compiler-plugin and set the maven.compiler.release property to the Java release you are targetting (e.g. 9, 10, 11, 12, etc.).

In the following example, we have configured our Maven project to use version 3.8.1 of maven-compiler-plugin and target Java 11. The following configuration snippet should be added into `<project>` tag of the POM file.

```
<properties>
   <maven.compiler.release>11</maven.compiler.release>
</properties>

<build>
   <pluginManagement>
      <plugins>
            <plugin>
               <groupId>org.apache.maven.plugins</groupId>
               <artifactId>maven-compiler-plugin</artifactId>
               <version>3.8.1</version>
            </plugin>
      </plugins>
   </pluginManagement>
</build>
```

Super POM

The Super POM is Maven's default POM. `All POMs extend the Super POM unless explicitly set.` This means that the configuration of the Super POM is inherited by the pom.xml file that the developer creates for the project. Together with super POM and project POM creates `the Effective POM which is the overall configuration file`. Effective POM can be examined by easily running the following command in your project's root directory: `mvn help:effective-pom`. However, it's also possible to see the same file in one of the tabs of the pom.xml file if you're using Eclipse IDE.

An effective POM example can be seen below.

```
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'help'.
[INFO] ------------------------------------------------------------------------
[INFO] Building Unnamed - com.companyname.project-group:project-name:jar:1.0
[INFO]    task-segment: [help:effective-pom] (aggregator-style)
[INFO] ------------------------------------------------------------------------
[INFO] [help:effective-pom {execution: default-cli}]
[INFO]

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: < 1 second
[INFO] Finished at: Thu Jul 05 11:41:51 IST 2012
[INFO] Final Memory: 6M/15M
[INFO] ------------------------------------------------------------------------

<?xml version="1.0" encoding="UTF-8"?>
<!-- ============================================ -->
<!-- -->
<!-- Generated by Maven Help Plugin on 2015-04-09T11:41:51 -->
<!-- See: http://maven.apache.org/plugins/maven-help-plugin/ -->
<!-- -->
<!-- ============================================-->

<!-- ============================================-->
<!-- -->
<!-- Effective POM for project -->
<!-- 'com.companyname.project-group:project-name:jar:1.0' -->
<!-- -->
<!-- ============================================ -->

<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/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.companyname.project-group</groupId>
   <artifactId>project</artifactId>
   <version>1.0</version>
   <build>
      <sourceDirectory>C:\MVN\project\src\main\java</sourceDirectory>
      <scriptSourceDirectory>src/main/scripts</scriptSourceDirectory>

      <testSourceDirectory>C:\MVN\project\src\test\java</testSourceDirectory>
      <outputDirectory>C:\MVN\project\target\classes</outputDirectory>
      <testOutputDirectory>C:\MVN\project\target\test-classes
          </testOutputDirectory>
      <resources>
         <resource>
            <mergeId>resource-0</mergeId>
            <directory>C:\MVN\project\src\main\resources</directory>
         </resource>
      </resources>
      <testResources>
         <testResource>
            <mergeId>resource-1</mergeId>
            <directory>C:\MVN\project\src\test\resources</directory>
         </testResource>
      </testResources>
      <directory>C:\MVN\project\target</directory>
      <finalName>project-1.0</finalName>

      <pluginManagement>
         <plugins>
            <plugin>
               <artifactId>maven-antrun-plugin</artifactId>
               <version>1.3</version>
            </plugin>
            <plugin>
               <artifactId>maven-assembly-plugin<  /artifactId>
               <version>2.2-beta-2</version>
            </plugin>
            <plugin>
               <artifactId>maven-clean-plugin<  /artifactId>
               <version>2.2</version>
            </plugin>

            <plugin>
               <artifactId>maven-compiler-plugin</artifactId>
               <version>2.0.2</version>
            </plugin>
            <plugin>
               <artifactId>maven-dependency-plugin</artifactId>
               <version>2.0</version>
            </plugin>
            <plugin>
               <artifactId>maven-deploy-plugin</artifactId>
               <version>2.4</version>
            </plugin>

   <repositories>
      <repository>
         <snapshots>
            <enabled>false</enabled>
         </snapshots>
         <id>central</id>
         <name>Maven Repository Switchboard</name>
         <url>http://repo1.maven.org/maven2</url>
      </repository>
   </repositories>
   <pluginRepositories>
      <pluginRepository>
         <releases>
            <updatePolicy>never</updatePolicy>
         </releases>
         <snapshots>
            <enabled>false</enabled>
         </snapshots>
         <id>central</id>
         <name>Maven Plugin Repository</name>
         <url>http://repo1.maven.org/maven2</url>
      </pluginRepository>
   </pluginRepositories>
   <reporting>
      <outputDirectory>C:\MVN\project\target/site</outputDirectory>
   </reporting>
</project>
```

Project Inheritance

As in the object-oriented programming, `POM files can also be inherited by other POM files.` Child POM has the opportunity to **either inherit from a parent POM** or **override the features** coming from its parent. Specifically, a parent POM can be a company's default project structure or a general template for project building.

Not every item in the parent POM is inherited. There are some items that should be declared specifically in every POM file. Notable elements that are not inherited include `artifactId, name, and prerequisites`.

The Super POM is an example of project inheritance, however, you can also introduce your own parent POMs by specifying the parent element in the POM, as demonstrated in the following examples.

Parent POM

```
<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
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.clarusway.mojo</groupId>
  <artifactId>my-parent</artifactId>
  <version>2.0</version>
  <packaging>pom</packaging>     <!-- NOTICE THE PACKAGING TYPE -->
</project>

```

The `packaging type of the parent POM should be "pom" for parent` and aggregation (multi-module) projects (will be discussed soon). The packaging type defines the goal of the POM file. If it were jar (java archive), then the package phase would produce the jar:jar goal.

Child POM

```
<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
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>com.clarusway.mojo</groupId>
    <artifactId>my-parent</artifactId>
    <version>2.0</version>
    <relativePath>../my-parent/pom.xml</relativePath>
  </parent>

  <artifactId>my-project</artifactId>
</project>

```

Child POM builds its `relation to parent POM via its parent tag`. Parent tag has the necessary information about the parent's groupId, artifactId, and version. It's also possible to declare the relative path to parenting POM. With this setup, child POM can now inherit the properties of its parent.

For example, if you want your project's groupId and the version of your project to be the same as their parents, you need to remove the groupId and the version tag for the POM file.

Project Aggregation

A project with modules (children) is called a `multi-module, or aggregator project.` Modules are **projects that a parent POM file specifies**, and those all modules are `built together as a group.`

An aggregator POM should have the `package tag with "pom"` expression in it and `modules tag specifying the relative paths` to **the directories or the POM files** of those projects.

```
<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
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.clarusway.mojo</groupId>
  <artifactId>my-parent</artifactId>
  <version>2.0</version>
  <packaging>pom</packaging>

  <modules>
    <module>my-project</module>
    <module>another-project</module>
    <module>third-project/pom-example.xml</module>
  </modules>
</project>

```