Multi-Module-Project-with-Maven

Last updated 05.2025

A) Multi modules project

// creating a parent project

```
mvn archetype:generate '-DgroupId=com.baeldung' '-DartifactId=parent-project'
```

// creating submodules

cd parent-project

```
mvn archetype:generate '-DgroupId=com.baeldung' '-DartifactId=core'
mvn archetype:generate '-DgroupId=com.baeldung' '-DartifactId=service'
mvn archetype:generate '-DgroupId=com.baeldung' '-DartifactId=webapp'
```

// building the Project

In the parent's project directory, we'll run:

```
mvn package
```

Let's see example pom files for a microservice project.

// C:\HONG\JAVATest202504\TESTh2\test3\parent-project

In the parent-project's pom.xml it will add all the submodules inside the modules section:

```
// pom.xml  // parent
...
 <groupId>com.baeldung</groupId>
 <artifactId>parent-project</artifactId>
 <version>1.0-SNAPSHOT</version>
 <packaging>pom</packaging>      // the project will serve as a parent, it won't produce artifacts.
 <name>parent-project</name>
…
 <modules>
  <module>core</module>
  <module>service</module>
  <module>webapp</module>
 </modules>
```

and in the individual submodules' pom.xml, it will add the parent-project in the parent section:

```
// pom.xml   // child
...
<parent>
 <groupId>com.baeldung</groupId>
 <artifactId>parent-project</artifactId>
 <version>1.0-SNAPSHOT</version>
</parent>
```

B) Why you should use parent pom for your multi-module Java projects

A microservice project that will be composed of several services each of which will be different module in our project.

How we can we can organize our project and advantages ---

1- Unified build management and versioning

With parent module, you can define them under <modules> tag and manage build lifecycles from one place as you can see below.

Another advantage is that you can version all your modules directly from your parent pom and keep them consistent.

2- Uniform /dependency/ and /plugin/ versioning

2a) <dependencies> ; <dependencyManagement>

Are this dependency going to be definetely in all sub modules => 2a-1) ; or only in some of them => 2a-2) ?

2a-1)
“yes, it will be in all modules”,
then you need to define the dependency in your parent pom under <dependencies> tag as follows and do not have to define dependency in your child pom.

```
// parent pom
<dependencies>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.20</version>
  </dependency>
</dependencies>
```

2a-2)
“no, just some modules will use it”
which is the most common one, then define dependencies in pom under <dependencyManagement> tag

By defining dependency versions in the parent pom, you will just need to add the dependency without version to your sub module

```
// parent pom
<dependencyManagement>

<dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.20</version>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>

// child pom

<dependencies>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>
  </dependencies>
</project>
```

2b) <plugins> ; <pluginManagement>
Again, same applies for plugins with <plugins> and <pluginManagement> tags.

We can change the packaging type of each submodule. For example, let’s change the packaging of the webapp module to WAR by updating the pom.xml file:

```
<packaging>war</packaging>
```

and adding maven-war-plugin in the plugins list:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-war-plugin</artifactId>
      <version>3.4.0</version>
      <configuration>
        <failOnMissingWebXml>false</failOnMissingWebXml>
      </configuration>
    </plugin>
  </plugins>
</build>
```

3- Declaring commons in one place and using everywhere

In your pom file, you can define several different configurations like /distribution management/.
When you define the common configurations in the parent pom, it will be inherited to sub modules.
<distributionManagement>

```
// parent pom
<distributionManagement>
    <repository>
      <id>github</id>
      <url>url</url>
    </repository>
  </distributionManagement>
</project>
```

C) Spring-Boot-Multi-Module-Project-with-Maven (example template)

1) microservices-demo

```
// pom.xml
<project ...>
  ...
  <groupId>com.hong.msv</groupId>
  <artifactId>Microservices</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>Ecom</name>
  <modules>
      <module>ProductServer</module>
      <module>ProductWeb</module>
  </modules>
</project>
```

<packaging>pom</packaging>
declaring that the project will serve as a parent, it won't produce further artifacts.

all the submodules inside the modules section:

1.1) submodule "ProductServer"

// cd microservices-demo/ProductServer

```
// pom.xml
<project ...>
  ...
  <groupId>com.hong.msv</groupId>

  <artifactId>Product-Server-Microservice</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>Product-Server-Microservice</name>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
   <version>3.2.0</version>
  </parent>
```

1.2) submodule "ProductWeb"

// cd microservices-demo/ProductWeb

```
// pom.xml
<project ...>

  <groupId>com.hong.msv</groupId>

  <artifactId>Product-Web-Microservice</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>Product-Web-Microservice</name>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.0</version>
  </parent>
```

--- see too ---
https://www.baeldung.com/maven-multi-module
https://kurular4.medium.com/why-you-should-use-parent-pom-for-your-multi-module-java-projects-b575017fab2e

C:\HONG\JAVATest202504\SB3-Beginn\beginning-spring-boot-3 // beginning-spring-boot-3
C:\HONG\DEVOPSTest\Java-Microservices-and-Containers-in-the-Cloud-main\89288133-Binildas-ch04-Source\ch04\ch04-01
C:\HONG\JAVATest202504\Java23BeginnCos\Java-23-for-Absolute-Beginners

https://github.com/hong1234/microservices-demo
https://github.com/hong1234/Multi-Module-Project
book: Introducing Maven (2019)
book: Mastering Apache Maven 3 (2014)