

Top 5 Exciting Maven Multi-Module Project Ideas & Topics for Beginners [2022]



by **Rohan Vats**

Software Engineering Manager @ upGrad. Passionate about building large scale web apps with delightful experiences. In pursuit of transforming engineers into leaders.

MAY 27, 2021

[Home](#) > [Full Stack Development](#) > Top 5 Exciting Maven Multi-Module Project Ideas & Topics for Beginners [2022]

As any project gets more complex and layered, the correct way to manage it is by splitting it into different modules. In Maven, while it is possible to link these multiple projects as dependencies, it makes the build process a lot more complex. Therefore, the preferred approach is to structure the project as a multi-modular one and pass on the rest to Maven.

Table of Contents

Benefits of Using Multi-Modules in Maven

Key Terms to Know Regarding Multi-Module Projects in Maven

1. Parent POM (Project Object Module)
2. Submodules
3. The Simple Multi-Module Project
4. Structure of a Multi-Module Project
5. Top Level POM
5. Module POM

Executing the Multi-Module Project

Conclusion

What are multi-modules projects?

What is project object model file in maven?

How to create a maven multi-module project?

Engineering Cou

Executive PG
in Software D
from IIITB

Specialisation in
Development

Online

Apply Now

[Development Speciali](#)
[Stack Development fr](#)
[Duration 12 Months](#)

[Executive PG Progra](#)
[Development Speciali](#)
[Security IIIT-B - Dur](#)

The significant advantage of using multi-modules instead of dependency injection or any other procedure is that it reduces duplication. To understand this better, suppose you're working on an application that consists of two modules – let's say the backend module and the frontend module.

If you work on both of those modules and change certain functionalities that affect both the modules, you'll need to build both the modules or components isolatedly without a specialised build tool. Or, you must write a script to compile the code, run tests, and display the final results.

Now, if the complexity increases even further and your project has more than two modules, it'll become progressively harder to perform this task without a specialised build tool. Further, in real-world problems, your projects may require some Maven plugins to perform some crucial operations during the entire build lifecycle, share dependencies, or include other BOM projects.

As a result of these challenges, the multi-modules in Maven are extremely helpful and let you build the application's modules in a single command line. It also allows you to share a vast amount of data and configuration between multiple modules seamlessly.

Also Read: [Maven Interview Question Answer](#)



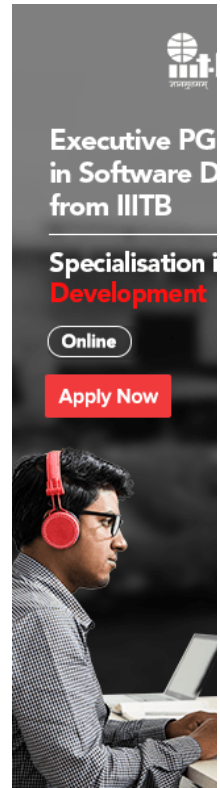
Key Terms to Know Regarding Multi-Module Projects in Maven

1. Parent POM (Project Object Module)

Each pom.xml file in [Maven](#) – for the support of inheritance – has a parent POM, known as the Super POM. This can be located in the Maven binaries, and you can also merge the two files to create the Effective POM.



Engineering Cou



the child modules, and enable inheritance. Except for inheritance, Maven also provides aggregation. The Parent POM that leverages the aggregation functionality is known as the aggregate POM – this kind of POM declares the child modules explicitly in the pom.xml file.

2. Submodules

Subprojects, also known as submodules, can be understood as standard Maven modules that inherit from the parent POM. Using inheritance, the modules share various dependencies and configurations. However, if you need to build the project quickly, you'll need to explicitly declare the submodules in the Parent POM, making it the aggregate POM.

3. The Simple Multi-Module Project

App and util are two modules that are contained in The Simple Multi-Module Project. The util module is required for providing a static method to join or append different strings using Apache Commons Lang library, and the app module is necessary to call the util module. Here's a snippet from the app.java file:

simple-multi/app/src/main/java/app/App.java

```
public class App {

    public static void main(String[] args) {

        System.out.println(new App().greet("World!"));

    }

    public String greet(String name) {

        return Util.join("Hello ", name);

    }

}
```

To build the project, run

```
$ cd simple-multi
```



```
$ mvn clean test
```

Engineering Cou

below:

```
[INFO] -----
[INFO] Reactor Summary:
[INFO]
[INFO] Maven Simple Multi Module Example ..... SUCCESS [ 0.005 s]
[INFO] Simple Util ..... SUCCESS [ 2.407 s]
[INFO] Simple App ..... SUCCESS [ 0.462 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
```

4. Structure of a Multi-Module Project

Here's the layout of a simple Multi-Module project:

```
simple-multi
├── pom.xml
├── app
│   ├── pom.xml
│   └── src
│       ├── main/java/app/App.java
│       └── test/java/app/AppTest.java
└── util
    ├── pom.xml
    └── src
        ├── main/java/util/Util.java
        └── test/java/util/UtilTest.java
```

The simple-multi directory is the root directory and is present at the very top of the entire project. It contains the top-level POM (Parent POM), but this Parent POM doesn't have any source folder.

The top-level directory also contains app and util, which are regular Maven projects with directories and pom.xml file.

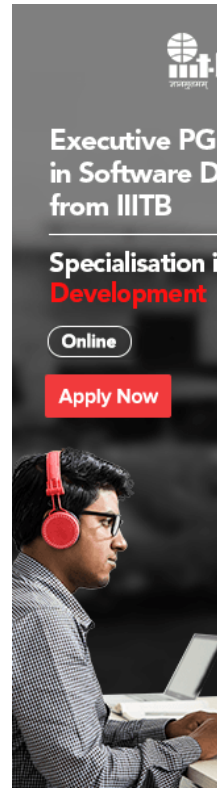
The top-level POM and the module POM differ slightly from the regular POM. Let's look in detail as to what they contain:

5. Top Level POM

A Parent POM or a top-level POM is required by the Maven Multi-Module Project at the root directory.

The top-level POM defines important aspects and project coordinates like artifactID, version, and groupId, as in any normal project. However, the packaging type for this is not in the war or jar format but as a pom. This is because the top-level module doesn't

Engineering Cou



The content of this file include:

simple-multi/pom.xml

```
<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>org.codetab</groupId>
```

```
  <artifactId>simple-multi</artifactId>
```

```
  <version>1.0</version>
```

```
  <packaging>pom</packaging>
```

```
  <modules>
```

```
    <module>app</module>
```

```
    <module>util</module>
```


```
  </modules>
```

```
</project>
```

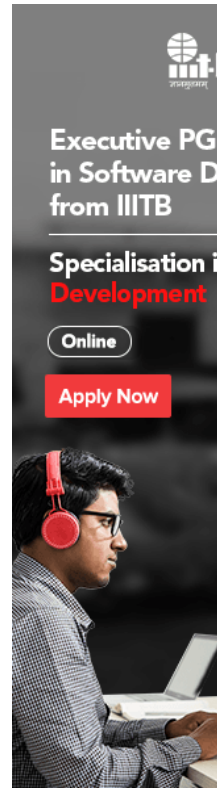
To sum it up, the top-level POM specifies the pom packaging type and explicitly lists all the submodules.

Keep in mind that you don't need to worry about the ordering while declaring submodules in the Parent POM as Maven uses a Reactor to order the listed modules correctly. In the simple-multi, maven builds util module and then app since the app depends on util.

5. Module POM

The  POM folder contains a regular source folder and its own pom.xml file. The content of this util/pom.xml are:

Engineering Cou



Executive PG
in Software D
from IIITB

Specialisation i
Development

Online

Apply Now

```

<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>

  <parent>

    <groupId>org.codetab</groupId>

    <artifactId>simple-multi</artifactId>

    <version>1.0</version>

  </parent>

  <artifactId>util</artifactId>

  <properties>

    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <maven.compiler.source>1.8</maven.compiler.source>

    <maven.compiler.target>1.8</maven.compiler.target>

  </properties>

  <dependencies>

    <dependency>

      <groupId>org.apache.commons</groupId>

      <artifactId>commons-lang3</artifactId>


      <version>3.6</version>

    </dependency>

```



Engineering Cou




Executive PG
in Software D
from IIITB

Specialisation i
Development

Online

Apply Now



```
<groupId>junit</groupId>
```

```
<artifactId>junit</artifactId>
```

```
<version>4.12</version>
```

```
<scope>test</scope>
```

```
</dependency>
```

```
</dependencies>
```

```
</project>
```

There is no need to specify the groupId and version for the module as they'll be inherited from the parent. The dependencies/dependency block defines the dependencies of util module – commons-lang3 and JUnit.

Now, let's look at the app/pom.xml, which is quite similar to the util/pom.xml, except for the dependencies block, which is shown below:

app/pom.xml

....

```
<dependencies>
```

```
<dependency>
```

```
<groupId>org.codetab</groupId>
```

```
<artifactId>util</artifactId>
```

```
<version>1.0</version>
```

```
</dependency>
```

....

```
</dependencies>
```



Engineering Cou



Executive PG
in Software D
from IIITB

Specialisation i
Development

Online

Apply Now



To summarise all the things explained above:

The top-level project (parent project),

- Defines top-level pom.xml, which specifies all the coordinates and submodules, and sets the packaging type as pom.
- Contains the further modules folders.
- Contains no source folders.

Each of the module folders is nothing but regular Maven project directories. However, the pom.xml file contains:

- The parent element specifies the module's parent.
- Module coordinates are specified in the artifactID element. GroupID and version are not mentioned as they are inherited from parent coordinates.
- The dependency, if one module depends on another, is mentioned in the dependencies/dependency element.

Executing the Multi-Module Project

\$ mvn clean package command can be used to compile, test, and package the multi-module. However, to run it with the maven-exec-plugin requires the following steps:

Run the following commands:

```
$ cd simple-multi
```

```
$ mvn clean install
```

```
$ mvn exec:java -pl app -Dexec.mainClass=app.App
```

Engineering Cou

Executive PG
in Software Development
from IIITB

Specialisation in
Development

Online

Apply Now

Become a **Full Stack Development Specialist**

Executive PG Program in
Software Development from IIITB

Online Sessions + Live Lectures

Specialize in Java or Python

class app.App. Without install, the build process fails, as Maven cannot download and resolve the util module dependency.

However, it is quite cumbersome to install the project in a local repository each time before each run, especially when in the development phase. To ease that, you can directly execute the multi-module in Eclipse IDE without performing the install step.

Learn [Software development Courses online](#) from the World's top Universities. Earn Executive PG Programs, Advanced Certificate Programs, or Masters Programs to fast-track your career.

Conclusion

We hope this article helped you in getting started with the Maven Multi-Module process. As is with anything new, it's just about getting the initial hang of it. Once you've understood the basic structure and functioning, you'll be on your way to seamlessly manage multiple modules in Maven.

If you're interested to learn more about full-stack software development, check out upGrad & IIIT-B's [Executive PG Programme in Software Development – Specialisation in Full Stack Development](#) which is designed for working professionals and offers 500+ hours of rigorous training, 9+ projects, and assignments, IIIT-B Alumni status, practical hands-on capstone projects & job assistance with top firms.

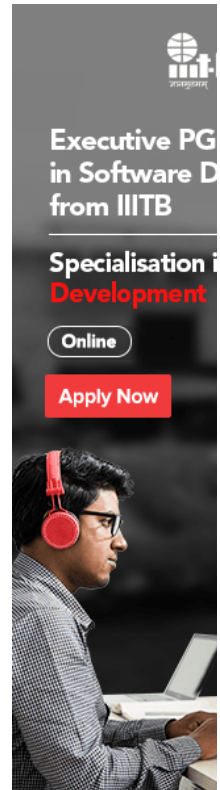
What are multi-modules projects?

Multi-modules enable you to group common artifacts into a single module in order to reduce the number of modules that are needed to build and deploy a project. Multi-module projects can be constructed from two or more modules by defining a parent-child relationship between these modules. The parent module of a multi-module project is called the root module. The child module of a multi-module project is called a submodule. The root module is the cornerstone of a multi-module build. The child modules cannot be built on their own, they must be built along with the root module.

What is a project object model file in maven?

A project object model (POM) file is a file describing a project. It is the fundamental unit of work by which developers publish artifacts to a repository. A project object model is a simple text file that defines a project, lists its components, and defines what they do. The project object model is a key component of the Maven software project management framework. It is a special type of XML document that a Maven project is composed of. A pom file can be thought of as a description of a project. It tells Maven what the project is about as well as a description of each of the modules in the project, their dependencies, and versions. Modules are the building blocks of a Maven project. The modules, and their dependencies, are used to build the final project.

Engineering Cou





Executive PG
in Software Development
from IIITB

Specialisation in
Full Stack Development

Online

Apply Now

For making a maven multi-module project, you need to define a parent pom and make sure all the sub-modules inherit it. It is also important to note that you can also put all the sub-modules in a single repository and point the parent pom to that. A maven multi-module project is comprised of a parent pom and one or more sub-modules. The parent pom can define the multi-module maven project metadata, such as artifact Id, version, packaging, and dependencies. Sub-modules can inherit the parent's metadata, and the parent pom can define high-level project configuration, such as the project's organization and description.



Land on Your Dream Job

APPLY NOW FOR EXECUTIVE PG PROGRAM IN FULL STACK DEVELOPMENT

Leave a comment

Your email address will not be published.

Comment


Name

Email

Website

Post Comment

Engineering Cou




Executive PG
in Software D
from IIITB


Specialisation in
Development

Online

Apply Now




Engineering Cou




upGrad's Knowledge Base

[Node JS Tutorial: Learn Node JS from Scratch](#)


 by Rohan Vats

Feb 17, 2022




upGrad's Career Advice

[Django Tutorial: Learn DJango from Scratch](#)


 by Rohit Sharma

Feb 17, 2022



upGrad's Knowledge Bi

[Power Function In Java: Special Advantage & Applications](#)

 by upGrad

De

Building Careers of Tomorrow

DATA SCIENCE

- Data Science | All Courses
- Data Science | Executive PG
- Data Science | Advanced Certificate
- Data Science | Master
- Machine Learning & AI | Executive PG
- Machine Learning & AI | Masters
- Machine Learning & NLP | Advanced Certificate
- Machine Learning and Cloud | Advanced Certificat...
- Executive PGP in Data Science – IIIT Bangalore
- M.Sc in Data Science – LJMU & IIIT Bangalore
- PCP in Data Science – IIM Kozhikode
- ACP in Data Science – IIIT Bangalore
- Executive Programme in Data Science – IIITB

SOFTWARE & TECHNOLOGY

- Software Engineering | All Courses
- Full Stack Development | Executive PG
- Computer Science | Masters
- Full Stack Development | Placement Track
- Full Stack Development | PG Certification
- Blockchain Technology | Executive PG
- Blockchain Technology | Executive Program
- Blockchain Technology | Advanced Certificate
- Big Data | Executive PG
- M.Sc in CS – LJMU & IIIT Bangalore
- PGC in Software Engineering – upGrad
- Full Stack Development – IIIT Bangalore
- Executive PGP Cyber Security – IIITB
- Executive PGP Cloud Computing – IIITB
- Executive PGP Big Data – IIITB
- Executive PGP DevOps – IIITB
- ACP Cyber Security – IIIT Bangalore
- ACP Cloud Computing – IIIT Bangalore
- ACP Big Data – IIIT Bangalore
- ACP DevOps – IIIT Bangalore



- MBA (Execu
- MBA (Global
- MBA (Global
- Digital Mark
- Management
- Product Man
- Sales & Digi
- Life Insuran
- Business Ana
- Life Insuran
- Management
- MBA (Global
- MBA (DFB) |
- Global MBA
- Master of Bu
- PG Diploma |
- MBA Executi
- MBA from O
- MBA (Global
- Global MBA
- MBA in Strat
- MBA in Adv
- Strategic Inn
- Product Man
- Operations M
- Design Think
- Executive PG
- Masters Qual
- PCP in HRM
- PGP in Mana
- Management
- Integrated St



BUSINESS ANALYTICS

- Executive PGP in Business Analytics – LIBA
- Certificate in Business Analytics – MSU
- Business Analytics Certification – upGrad

STUDY ABROAD

- Master in International Management – IMT & IU ...
- Master Degree in Data Science – IIITB & IU Germ...
- Master in Cyber Security – IIITB & IU Germany

BBA, MBA & MANAGEMENT

- Global Doctor of Business Administration

LAW

- LL.M. in Corporate & Financial Law – Jindal Global

HEALTH

- Executive PGP Healthcare Management – LIBA

MACHINE LEARNING

- Machine Learning | All Courses
- Executive PGP in Machine Learning & AI – IIITB
- M.Sc in Machine Learning & AI – LJMU & IIITB
- ACP in Machine Learning – IIT Delhi
- M.Sc in Machine Learning & AI – LJMU & IIT M...
- Certificate in ML and Cloud – IIT Madras
- ACP in ML & Deep Learning – IIIT Bangalore
- ACP in Machine Learning & NLP – IIIT Bangalore

BACHELOR DEGREE

- Business Analytics | All Courses
- Professional Certificate Program in Data Science a...
- Global Master Certificate in Business Analytics – ...

DIGITAL MARKETING

- Digital Marketing | All Courses
- Digital Marketing & Communications – IITB
- Digital Branding & Advertising – IIITB

Engineering Cou

MA in Commu

MA in Jour

MA in Jour

Executive PG

in Software D

from IIITB

Specialisation i

Development

Online

Apply Now

Blockchain

ACP Blo

Executive

