## 1. Introduction

Maven is an universal software project management, in order to get maven users familiar with maven projects, maven defines some conventions or directory layouts.

Through those directory layouts maven achieves an uniform way to organize projects and files inside of it. This a very good approach because you can work on several projects and you always will have the same project structure, so you will switch between projects and you don´t have to expend time in order to learn how the project is organized.

## 2. Directory layout. Files

Maven defines some conventions in order to organize the normal files inside a project. In this directories you can put all application sources files.

The directories are the following

* src/main/java
* src/test/java
* src/it

### **2.1 src/main/java**

Inside this folder you can put all the application source files. Classes and packages for the main (real) artifact should be put in this folder.

ADVERTISEMENT

All the content inside of this directory will be put in the classpath of the generated artifact. If the artifact is a jar file, all the classes and packages will be in the root folder of the generated jar, so it will be available by default on the runtime classpath.

If the artifact is a war, all the classes and packages will be put inside the WEB-INF/classes directory, so it will be available on the runtime classpath by default.

When the project is build or packaged all those classes and packages will be put in the target folder.

If you use eclipse as your IDE, this directory will be put inside the java build path automatically when you give the maven nature to the project.

### **2.2 src/test/java**

Inside this folder you can put all the application test source files. Classes and packages for the test artifact should be put in this folder.

All the content inside of this directory will NOT be put in the classpath of the generated artifact.

When the project is build or packaged all those classes and packages will be put in the target folder.

When you run your test you muyst be awared that maven surefire plugin will run the classes from the target directory.

If you use eclipse as your IDE, this directory will be put inside the java build path automatically when you give the maven nature to the project.

### **2.3 src/it**

Inside this folder you can put all the application integration test source files. Classes and packages for the integration test artifact should be put in this folder.

All the content inside of this directory will NOT be put in the classpath of the generated artifact.

When the project is build or packaged all those classes and packages will be put in the target folder.

When you run your integration test you must be awared that the implicated plugin will run the classes from the target directory.

If you use eclipse as your IDE, this directory will be put inside the java build path automatically when you give the maven nature to the project.

## 3. Directory layout. Resources

Maven defines some conventions in order to organize the normal files inside a project. In this diectories you can put all application sources files.

The directories are the following

* src/main/resources
* src/test/resources
* src/main/filters
* src/test/filters

### **3.1 src/main/resources**

Inside this folder you can put all the application resource files. Resources for the main (real) artifact should be put in this folder.

All the content inside of this directory will be put in the classpath of the generated artifact. If the artifact is a jar file, all the resources will be in the root folder of the generated jar, so it will be available by default on the runtime classpath.

If the artifact is a war, all resources will be put inside the WEB-INF/classes directory, so it will be available on the runtime classpath by default.

When the project is build or packaged all those resources will be put in the target folder.

If you use eclipse as your IDE, this directory will be put inside the java build path automatically when you give the maven nature to the project.

### **3.2 src/test/resources**

Inside this folder you can put all the application test resource files. Resources for the test artifact should be put in this folder.

All the content inside of this directory will NOT be put in the classpath of the generated artifact.

When the project is build or packaged all those test resources will be put in the target folder.

When you run your test you must be awared that maven surefire plugin will use resources from the target directory.

If you use eclipse as your IDE, this directory will be put inside the java build path automatically when you give the maven nature to the project.

### **3.3 src/main/filters**

Inside this folder you can put all the application filters files. Filters for the artifact should be put in this folder.

* You can see more details [here](https://maven.apache.org/shared/maven-filtering/).

### **3.4 src/test/filters**

Inside this folder you can put all the application test filters files. Filters for the test artifact should be put in this folder.

* You can see more details [here](https://maven.apache.org/shared/maven-filtering/).

## 4. Directory layout. Misc

Maven defines some conventions for several purposes, like

* src/assembly
* LICENSE.txt: This file represents the project license file.
* NOTICE.txt: This file are notes, notices and attributions for the project, as third party libraries mentions, licenses, etc…
* README.txt: Project readme file.

### **4.1 src/assembly**

Inside this folder you can put all the maven assembly plugin file. This files will be used by the maven assembly plugin.

* You can see an example of the maven assembly plugin [here](https://examples.javacodegeeks.com/enterprise-java/maven/maven-assembly-plugin-example/).

## 5. Directory layout. Webapp

Maven project structure defines a folder in order to store all resources and files needed by a web application.

* src/main/webapp

Inside this folder you can put all the required files for a web application like jsp files, js files, html files, css files, template files, reports files, WEB-INF files (like web.xml), META-INF files, etc…

All the content inside of this directory will be put in the classpath of the generated war artifact, all resources will be put inside the WEB-INF directory, so it will be available on the runtime classpath by default.

When the project is build or packaged all those resources will be put in the target/WEB-INF folder.

## 6. Directory layout. Target

The target folder is the maven default output folder. When a project is build or packaged, all the content of the sources, resources and web files will be put inside of it, it will be used for construct the artifacts and for run tests.

You can delete all the target folder content with mvn clean command.

## 7. Directory layout. Pom file

The pom (Project Object Model) file is a maven special file that describe how the project have to be build and from when maven should download artifacts/dependencies, what are those dependecies and so many more things.

This file is placed on the root project folder.

## 8. Directory layout. Personal files

You can put more folders inside of the project structure, but in those cases you have to instruct maven on the build tag of the pom file in order to manage those folder.