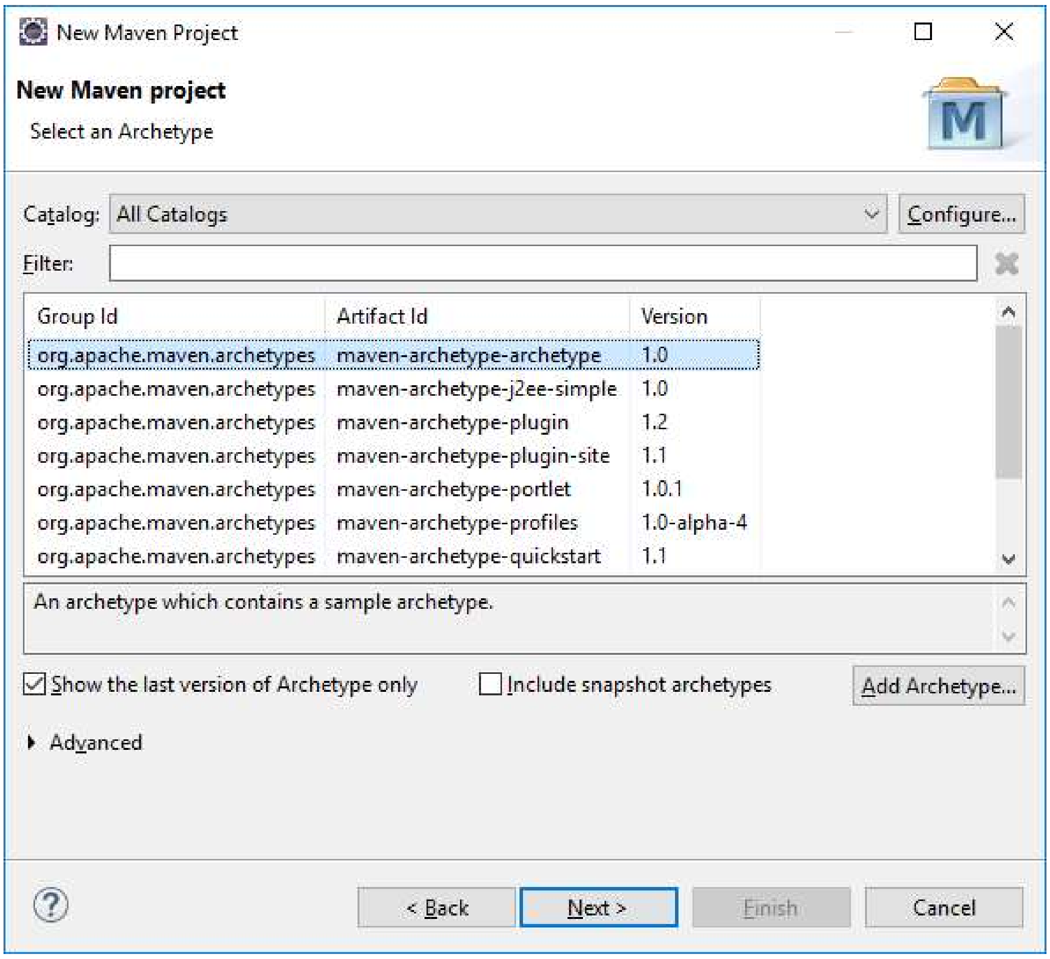
How to create Java web project with Maven in Eclipse

Written by Nam Ha Minh
Last Updated on 07 August 2019 | Print Email

Creating a Java web project in Eclipse with Maven support sounds simple as Eclipse has great support for Maven, but actually it doesn't. You can create a Maven project for Java webapp by clicking menu **File > New > Maven Project** (you need to switch to the Java EE perspective to see this menu).

In the *New Maven Project* dialog appears, click Next. Then you see a list of built-in archetype (type of Maven project) to choose from, as shown below:



New Maven Project

New Maven project

Select an Archetype

Catalog: All Catalogs

Filter:

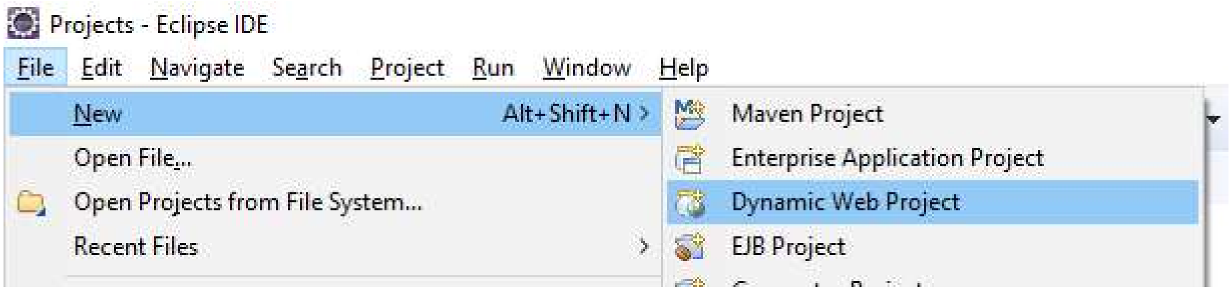
Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-archetype	1.0
org.apache.maven.archetypes	maven-archetype-j2ee-simple	1.0
org.apache.maven.archetypes	maven-archetype-plugin	1.2
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.1

An archetype which contains a sample archetype.

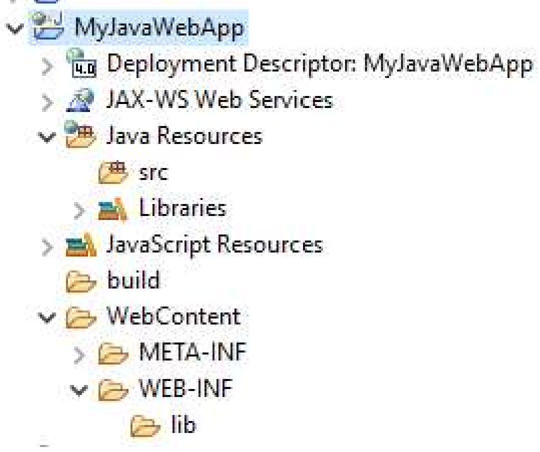
☑ Show the last version of Archetype only ☐ Include snapshot archetypes

I tested all of those archetypes but none can generate a properly configured basic Java web project. So here's the proper way to create a Java web project in Eclipse with Maven support:

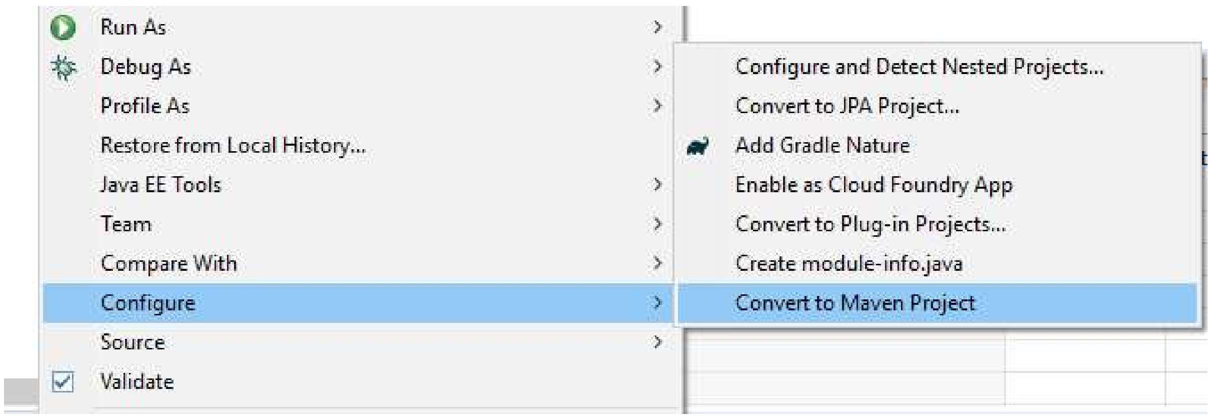
First, you create a new project as usual, click **File > New > Dynamic Web Project**:



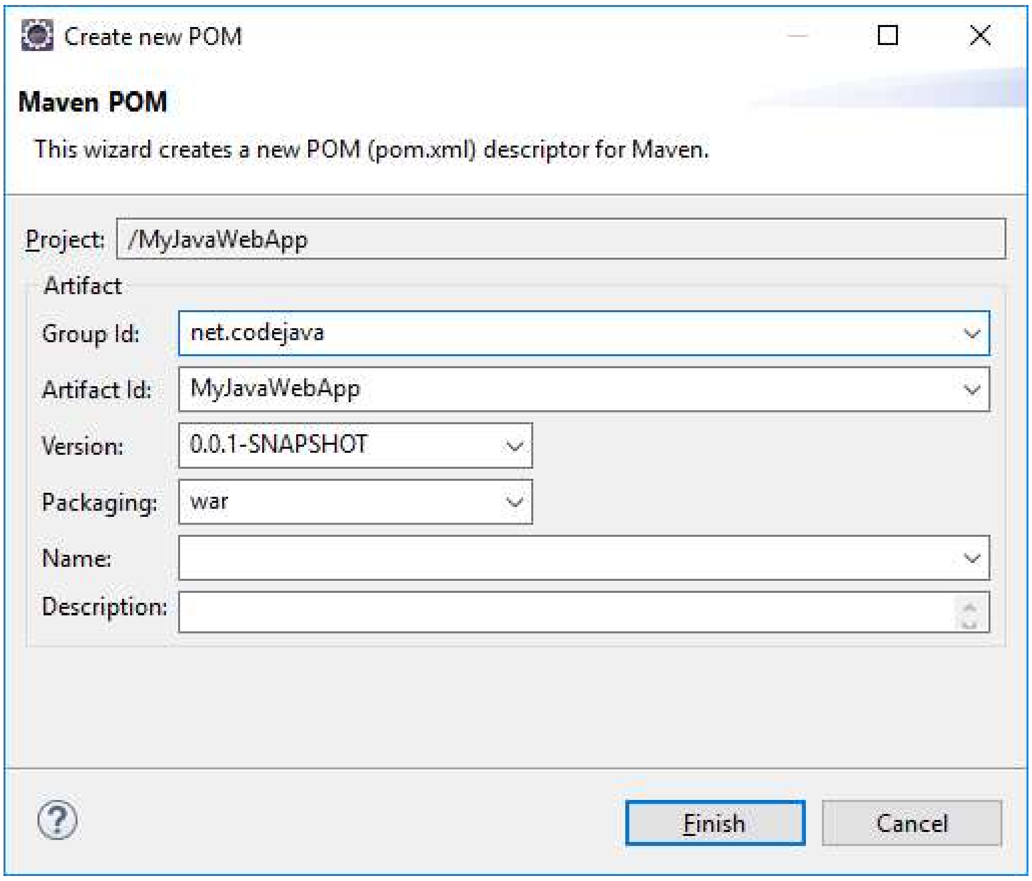
Then follow the wizards to create a Java dynamic web project normally. The newly created project would look like this:



Now, right click on the project name and click **Configure > Convert to Maven Project**:


Run As
Debug As
Profile As
Restore from Local History...
Java EE Tools
Team
Compare With
Configure
Source
Validate
Configure and Detect Nested Projects...
Convert to JPA Project...
Add Gradle Nature
Enable as Cloud Foundry App
Convert to Plug-in Projects...
Create module-info.java
Convert to Maven Project

Then in the *Create new POM* dialog, enter essential information for a Maven project like Group Id, artifact Id, version, name and description:


Create new POM

Maven POM

This wizard creates a new POM (pom.xml) descriptor for Maven.

Project: /MyJavaWebApp

Artifact

Group Id: net.codejava

Artifact Id: MyJavaWebApp

Version: 0.0.1-SNAPSHOT

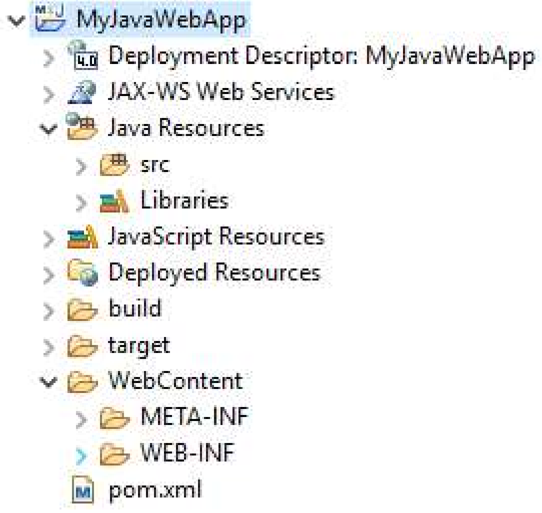
Packaging: war

Name:

Description:

Note that the Packaging type is war by default because this is a Java web project which will be packaged into a WAR file to deploy.

Then click Finish. You will the project's icon is updated with "M" letter – indicating it is a Maven project:



You also see the `pom.xml` file is generated in the project's root directory. It is the Project Object Model configuration file used by Maven.

Now edit the `pom.xml` file to specify the dependency for Java Servlet API:

```xml
<dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.0.1</version>
            <scope>provided</scope>
        </dependency>
</dependencies>
```

This dependency is needed to write code that uses Servlet API, e.g. servlet classes. The whole content of the `pom.xml` file would look like this:

```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>net.codejava</groupId>
    <artifactId>MyJavaWebApp</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <build>
        <sourceDirectory>src</sourceDirectory>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
            <plugin>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.2.1</version>
                <configuration>
                    <warSourceDirectory>WebContent</warSourceDirectory>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.0.1</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
</project>
```

Now you have a Java dynamic web project with Maven support. Happy coding!