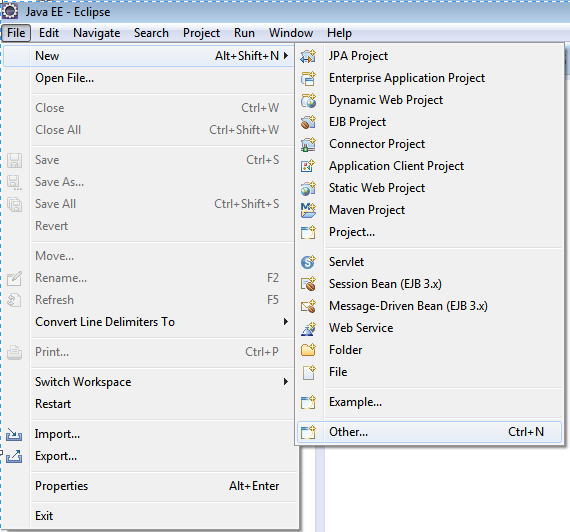
**Creating a Simple Webbased Maven Project in Eclipse**

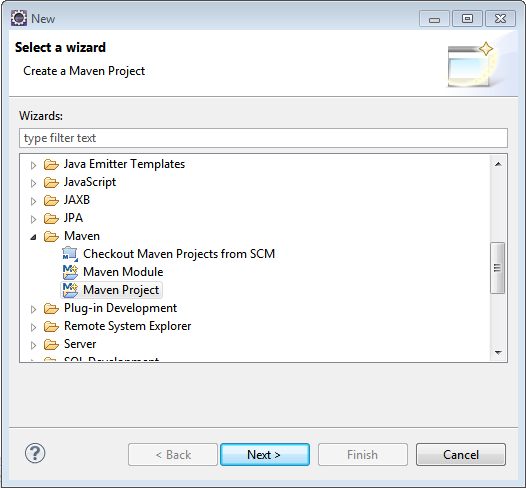
This note demonstrates creating a [Maven](http://maven.apache.org/) web project with eclipse. This is fairly simple.

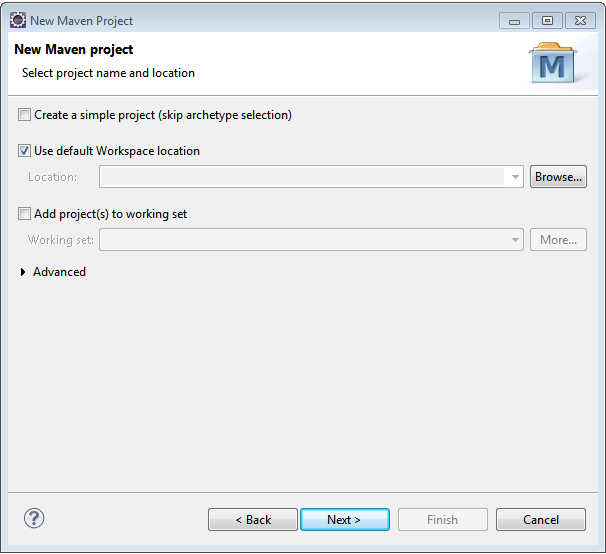
Let’s begin.

**1:**Start Eclipse, click on File->New->Other

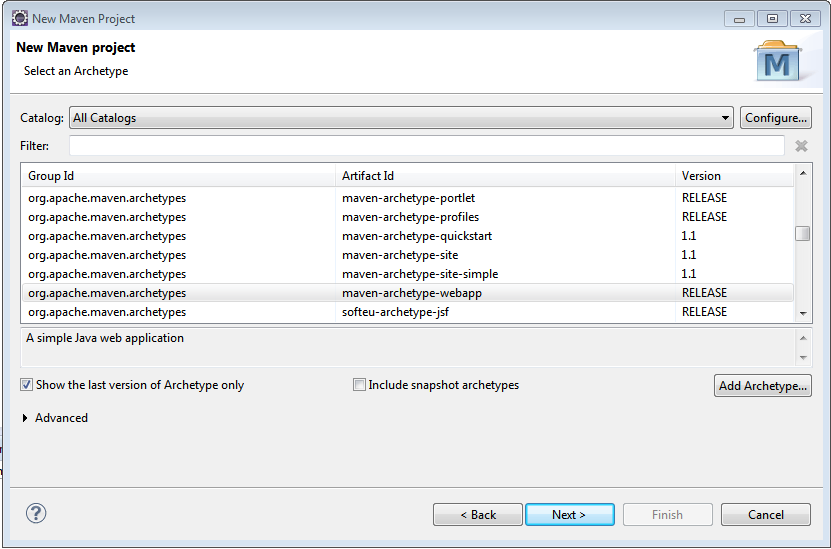


**2:**In the popped New wizard, scroll down and select Maven->Maven Project, click Next

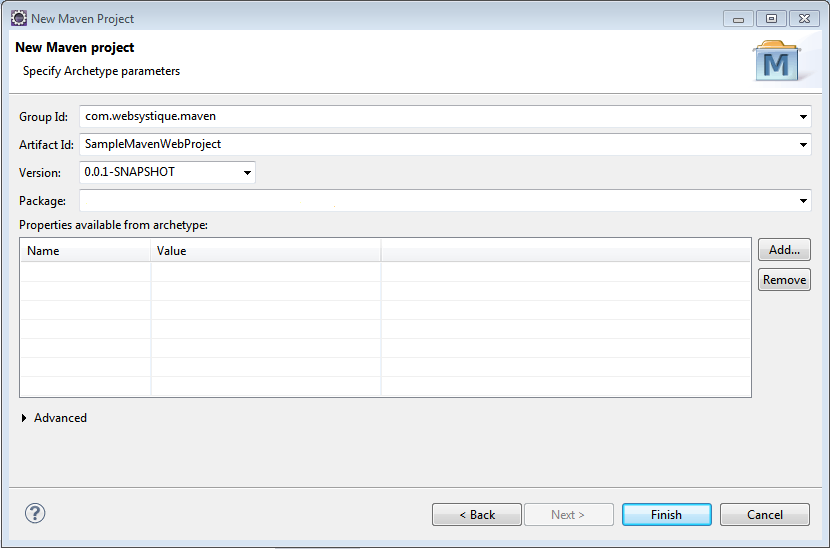


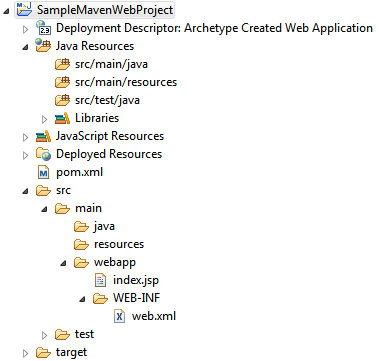


**3:**In above screen, leave the ‘Use default workspace location’ selected to create this project in the current workspace.Click Next. Scroll down and select option with Artifact Id as maven-archetype-webapp.



**4:**Click Next. Provide Group Id, Artifact Id & version. These parameters gives your project deliverable(jar/war/ear …) a unique identity in repository. Click Finish. Refresh your project.

  
**5:**Your project structure should like below one.



**Note:** In case you do not see the src/main/java and src/test/java folder in your project structure, go to Project>Properties>Java BuildPath>Libraries, select/change-to appropriate Java version, click OK, you should see those folders now. You may also perform a maven-update on your project.

**Note:** With recent Eclipse versions you don’t need to use mvn eclise:eclipse anymore, this functionality is built-in with recent m2e eclipse plugin.

**6:** Finally, update your pom.xml based on your project requirement (adding dependencies, profile, plugins, repositories, output format etc…)

Below is the default pom.xml generated for our project.

|  |
| --- |
| <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/maven-v4_0_0.xsd>">      <modelVersion>4.0.0</modelVersion>      <groupId>com.javalive.maven</groupId>      <artifactId>SampleMavenWebProject</artifactId>      <packaging>war</packaging>      <version>0.0.1-SNAPSHOT</version>      <name>SampleMavenWebProject Maven Webapp</name>      <url>[http://maven.apache.org](http://maven.apache.org/)</url>      <dependencies>          <dependency>              <groupId>junit</groupId>              <artifactId>junit</artifactId>              <version>3.8.1</version>              <scope>test</scope>          </dependency>      </dependencies>      <build>          <finalName>SampleMavenWebProject</finalName>      </build>  </project> |

Dependency section refers to the library your project is replying upon. These libraries are found on different [maven repositories](http://mvnrepository.com/).

For instance, if your code will be using spring framework and your test cases relies upon TestNG instead of junit, your pom.xml will include the dependency for Spring and TestNG like below:

|  |
| --- |
| <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/maven-v4_0_0.xsd>">      <modelVersion>4.0.0</modelVersion>      <groupId>com.javalive.maven</groupId>      <artifactId>SampleMavenWebProject</artifactId>      <packaging>war</packaging>      <version>0.0.1-SNAPSHOT</version>      <name>SampleMavenWebProject Maven Webapp</name>      <url>[http://maven.apache.org](http://maven.apache.org/)</url>      <properties>          <springframework.version>4.0.5.RELEASE</springframework.version>          <testng.version>6.8.8</testng.version>      </properties>      <dependencies>          <dependency>              <groupId>org.springframework</groupId>              <artifactId>spring-core</artifactId>              <version>${springframework.version}</version>          </dependency>          <dependency>              <groupId>org.testng</groupId>              <artifactId>testng</artifactId>              <version>${testng.version}</version>          </dependency>      </dependencies>      <build>          <pluginManagement>              <plugins>                  <plugin>                      <groupId>org.apache.maven.plugins</groupId>                      <artifactId>maven-compiler-plugin</artifactId>                      <version>3.2</version>                      <configuration>                          <source>1.6</source>                          <target>1.6</target>                      </configuration>                  </plugin>              </plugins>          </pluginManagement>          <finalName>SampleMavenWebProject</finalName>      </build>  </project> |

Additionally, The Maven plugin maven-compiler-plugin has been added here to explicitly specify the jdk-version we are going to use. Do note that it also forces eclipse to respect the jdk-version being used for the project. if it is not present, and you perform mvn-update from within your eclipse, eclipse switches jdk-version back to default jdk-version [1.5] which is annoying. So do add it in your project pom as well.

That’s it.