COURSEWARE

Adding Dependencies

Contents

Overview

Maven allows us to easily manage dependencies our code has, which are additional libraries that we use within our code.
For example, for testing in Java we will want to use JUnit.
Usually we would have to add the JUnit JAR to our classpath, but with Maven we can instead add a dependency on that library and it will be dynamically downloaded when it is needed.

Tutorial

In order to add dependencies to our Maven project we will need to add the `dependencies` tag to our `pom.xml`.
Once we have the `dependencies` tag we can then define each single `dependency` that we want to use within our code.
Inside each individual dependency we need to specify three things; the `groupId` of the library, the `artifactId`, and the `version`.
We can also specify the `scope` of the dependency if we only want to use that library in a specific area of our code base.
For example if we want to add a dependency on JUnit to our code, we only need this for our test classes, so we will give the dependency a *test* scope.
To add the JUnit dependency to our code we would need to add the following to our `pom.xml`.

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

If we wanted to add another dependency we would not need to add new `dependencies` tags, instead we would add new `dependency` tags within the `dependencies` tags we already have.

Our `pom.xml` files can eventually grow quite large with the more dependencies we add into it, making maintaining version numbers difficult.
We can solve this by adding the version number in as a property, then referring to the property within the dependency instead of hard coding the version number.
This allows us to house all of our version numbers at the top of the `pom.xml` file, making it easier and quicker to upgrade to a later version.
To do this we need to use the `properties` tag, then within that tag we will need a new tag that can have a user defined name, in this case we will use `junit.version`.
So we will have something like the following at the top of our `pom.xml`.

```
<properties>
    <junit.version>4.12</junit.version>
</properties>
```

To then reference this in our dependency we can change the version number in our dependency to the following.

```
<version>${junit.version}</version>
```

This is especially helpful when we want to use multiple libraries from the same framework as it makes keeping a standard version number for that framework much easier.
Our `pom.xml` file should now look like the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<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>com.qa</groupId>
    <artifactId>maven-example</artifactId>
    <version>1.0.0-SNAPSHOT</version>

    <properties>
        <junit.version>4.12</junit.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

Exercises

There are no exercises for this module.