Maven – Parte 2

Conceptos generales

People matter, results count.

En la primera parte, veíamos lo que eran los grupos y artefactos, el alcance (*scope*) de las dependencias, lo que era un *Goal*, lo que era un arquetipo (*archetype*), como crear, compilar e instalar un proyecto y finalmente, como integrarlo con el IDE. En este capítulo veremos como configurar nuestro proyecto mediante el pom.xml.

Todo proyecto Maven tiene al menos un *pom.xml* que describe el proyecto, sus dependencias, los plugins que utiliza, y otros datos, como la conexión con el sistema de control de versiones, o definición de otros repositorios Maven que usemos en nuestro proyecto para descargar dependencias entre otras cosas.

Espacio de Nombres (NameSpace)

Lo primero que necesitamos, como es obvio, es el espacio de nombres, como en todo xml:

<?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/maven-v4_0_0.xsd"> </project>

Propiedades del Artefacto

```
<modelversion>4.0.0</modelversion>
<groupid>com.capgemini.escuelita</groupid>
<artifactid>maven-sample</artifactid>
<name>maven-sample</name>
<version>1.0.0-SNAPSHOT</version>
<packaging>war</packaging>
<description>maven-sample></description>
```

Los campos más importantes aquí son groupId, artifactId, version y packaging. Los tres primeros son imprescindibles, mientras que sino especificamos packaging, por defecto será un jar. *Packaging* tiene que ver con la fase de empaquetado del artefacto. En este caso, si ejecutamos *"mvn package"*, se creará un war.

Sección de Properties

Sección de properties

La sección properties, que es opcional, es muy útil para parametrizar las versiones de las dependencias. Por ejemplo, si queremos usar spring, tiene muchos artefactos. Lo normal es usar la misma versión de Spring en todos sus componentes:

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <java.version>1.6</java.version>
    <org.springframework.version>3.0.6.RELEASE</org.springframework.versio
    n>
</properties>
```

Sección de properties

```
<dependency>
    <groupid>org.springframework</groupid>
    <artifactid>spring-webmvc</artifactid> <version>${org.springframework.version}</version>
</dependency>
<dependency>
    <groupid>org.springframework</groupid>
    <artifactid>spring-web</artifactid> <version>${org.springframework.version}</version>
    <type>jar</type>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupid>org.springframework</groupid>
    <artifactid>spring-jms</artifactid>
    <version>${org.springframework.version}</version>
    <type>jar</type>
 </dependency>
```

Dependencias

Dependencias

Está es la sección más importante, ya que la misión principal por la que se concibió Maven, es para gestionar las dependencias. Los únicos campos obligatorios son el id del grupo y el del artefacto. Si no especificamos la versión, descargará la última. El tipo por defecto será jar y el alcance (scope) será "compile":

```
<dependencies>
        <dependency>
                <groupid>org.springframework</groupid>
                                <artifactid>spring-webmvc</artifactid>

        <version>${org.springframework.version}</version>
        </dependency>
</dependencies>
```

Repositorios

Repositorios

Se pueden definir repositorios a nivel de proyecto, es decir, en el pom de nuestro proyecto en vez de en el setting.xml de Maven:

```
<repositories>
        <repository>
                <id>org.springframework.maven.snapshot</id>
                 <name>Spring Maven Snapshot Repository</name>
                <url>http://maven.springframework.org/snapshot</url>
                 <releases>
                                <enabled>false</enabled>
                </releases>
                <snapshots>
                                <enabled>true</enabled>
                 </snapshots>
        </repository>
        <repository>
                 <id>org.springframework.maven.milestone</id>
                 <name>Spring Maven Milestone Repository</name>
                <url>http://maven.springframework.org/milestone</url>
                <snapshots>
                                 <enabled>false</enabled>
                 </snapshots>
         </repository>
 </repositories>
```

Repositorios

En este caso, he definido los repositorios de snapshots y milestones de Spring, para poder usar las últimas build de Spring en mi proyecto. Hay muchos repositorios públicos de Maven, como este de Spring. Pero muchos otros proyectos opensource tienen su propio repositorio público que normalmente da accesso a las últimas build del proyecto, ya que las últimas *releases* están en el repositorio central.

Build

Build

Esta sección es también bastante importante, pues es donde se configura el build del proyecto mediante plugins. Maven tiene una arquitectura muy modular, está construido mediante plugins. Así, la misma compilación de un proyecto, es un plugin:

```
<build>
<plugins>
<plugin>
		<groupId>org.apache.maven.plugins</groupId>
		<artifactId>maven-compiler-plugin</artifactId>
		<configuration>
			<source>${java-version}</source>
			<target>${java-version}</target>
		</configuration>
</plugin>
<plugin>
		<groupId>org.apache.maven.plugins</groupId>
		<artifactId>maven-war-plugin</artifactId>
		<configuration>
			<webappDirectory>src/main/webapp</webappDirectory>
			<warName>maven-sample</warName>
		</configuration>
</plugin>
</plugins>
</build>
```

Build

El primer plugin, es el maven-compiler-plugin, que es el que se encarga de compilar. Aquí le decimos que versión de Java estamos usando, en este caso, la 1.6. Es muy recomendable ponerlo, sino compilará para la versión 1.3.

El segundo plugin, maven-war-plugin, usado para proyectos web, que como saben, se empaquetan en un war. Esta configuración es muy simple, se indica la carpeta de los fuentes de la aplicación web, el nombre del war, pero se pueden configurar muchas otras cosas.