Creando un proyecto

Necesitas un lugar donde residir tu proyecto. Cree un directorio en algún lugar e inicie un shell en ese directorio. En su línea de comando, ejecute el siguiente goal de Maven:

```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -Darch
etypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4 -Dinteracti
veMode=false
```

Si acaba de instalar Maven, la primera ejecución puede tardar un poco. Esto se debe a que Maven está descargando los artefactos más recientes (jars de complementos y otros archivos) en su repositorio local. Es posible que también deba ejecutar el comando un par de veces antes de que tenga éxito. Esto se debe a que es posible que el servidor remoto expire antes de que se completen las descargas

Notará que el goal *de generación* creó un directorio con el mismo nombre que el artefactoId. Cambie a ese directorio.

```
cd my-app
```

Bajo este directorio observará la siguiente estructura de proyecto estándar .

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

El `src/main/java` directorio contiene el código fuente del proyecto, el `src/test/java` directorio contiene el código fuente de prueba y el `pom.xml` archivo es el modelo de objetos del proyecto, o POM.

el pom

El `pom.xml` archivo es el núcleo de la configuración de un proyecto en Maven. Es un archivo de configuración único que contiene la mayor parte de la información necesaria para construir un proyecto de la manera que desee. El POM es enorme y puede resultar desalentador por su complejidad, pero no es necesario comprender todas las complejidades todavía para utilizarlo de forma eficaz. El POM de este proyecto es:

```
1. <project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   http://maven.apache.org/xsd/maven-4.0.0.xsd">
3. <modelVersion>4.0.0</modelVersion>
4.
5. <groupId>com.mycompany.app</groupId>
6. <artifactId>my-app</artifactId>
7. <version>1.0-SNAPSHOT</version>
8.
9. <properties>
10.   <maven.compiler.source>1.7</maven.compiler.source>
11.   <maven.compiler.target>1.7</maven.compiler.target>
12.   </properties>
13.
14.   <dependencies>
15.   <dependency>
16.   <groupId>junit</groupId>
17.   <artifactId>junit</artifactId>
18.   <version>4.12</version>
19.   <scope>test</scope>
20.   </dependency>
21.   </dependencies>
22.   </project>
```

¿Qué acabamos de hacer?

Ejecutó el *arquetipo de goal de Maven: generar* y pasó varios parámetros a ese goal. El *arquetipo* de prefijo es el complemento que proporciona el goal. Si está familiarizado con Ant , puede concebirlo como algo similar a una tarea. Este goal *archetype:generate* creó

un proyecto simple basado en un arquetipo maven-archetype-quickstart . Baste decir por ahora que un *complemento* es una colección de *goals* con un propósito general común. Por ejemplo, el complemento jboss-maven-plugin, cuyo propósito es "tratar con varios elementos de jboss".

Construya el proyecto

```
Mvn package
```

La línea de comando imprimirá varias acciones y terminará con lo siguiente:

```
...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.953 s
[INFO] Finished at: 2019-11-24T13:05:10+01:00
[INFO] ------------------------------------------------------------------------
```

A diferencia del primer comando ejecutado (*archetype:generate*), el segundo es simplemente un *paquete* de una sola palabra . Más que una *meta* , ésta es una *fase* . Una fase es un paso en el ciclo de vida de la construcción , que es una secuencia ordenada de fases. Cuando se proporciona una fase, Maven ejecuta cada fase de la secuencia hasta la definida inclusive. Por ejemplo, si ejecuta la fase *de compilación , las fases que realmente se ejecutan son:*

1. validar
2. generar-fuentes
3. fuentes de proceso
4. generar-recursos
5. recursos-de-proceso
6. compilar

Puede probar el JAR recién compilado y empaquetado con el siguiente comando:

```
java -cp target/my-app-1.0-SNAPSHOT.jar com.mycompany.app.App
```

Que imprimirá lo por excelencia:

```
Hello World!
```

Ejecución de herramientas Maven

Fases de Maven

Aunque no es una lista exhaustiva, estas son las fases del ciclo de vida *predeterminadas* más comunes que se ejecutan.

- **validar** : validar que el proyecto sea correcto y que toda la información necesaria esté disponible
- **compilar** : compila el código fuente del proyecto.
- **prueba** : prueba el código fuente compilado utilizando un marco de prueba unitario adecuado. Estas pruebas no deberían requerir que el código esté empaquetado o implementado.
- **paquete** : tome el código compilado y empaquetelo en su formato distribuible, como un JAR.
- **prueba de integración** : procesa e implementa el paquete si es necesario en un entorno donde se pueden ejecutar pruebas de integración
- **verificar** : ejecute cualquier verificación para verificar que el paquete sea válido y cumpla con los criterios de calidad.
- **instalar** : instala el paquete en el repositorio local, para usarlo como una dependencia en otros proyectos localmente
- **implementar** : realizado en un entorno de integración o lanzamiento, copia el paquete final al repositorio remoto para compartirlo con otros desarrolladores y proyectos.

Hay otros dos ciclos de vida de Maven dignos de mención además de la lista *predeterminada* anterior. Ellos son

- **clean** : limpia artefactos creados por compilaciones anteriores

- **site** : genera documentación del sitio para este proyecto

En realidad, las fases se asignan a goals subyacentes. Los goals específicos ejecutados por fase dependen del tipo de embalaje del proyecto. Por ejemplo, *el paquete* ejecuta *jar:jar* si el tipo de proyecto es JAR, y *war:war* si el tipo de proyecto es (lo has adivinado) WAR.

Algo interesante a tener en cuenta es que las fases y los goals se pueden ejecutar en secuencia.

```
mvn clean dependency:copy-dependencies package
```