



# Gestión de Maven

Escuela de arquitectura y programación

Especialidad Java/XML/JEE

[Indra](#) & [Icono Training Consulting](#)

TRAINING CONSULTING



# Definiciones

- [Maven](#) (experto) es una herramienta para la gestión técnica de proyectos
- Proporciona soporte para:
  - Construcción
  - Documentación
  - Generación de informes
  - Ejecución automática de baterías de pruebas
  - Gestión de dependencias
  - Interacción con repositorios remotos
  - Generación de versiones (*releases*) y distribución de las mismas
  - Personalización del comportamiento estándar mediante *plug-ins*

# Preparación del entorno de trabajo

- [Web de Java](#)
  - [Descarga de Java](#)
- [Web de Apache Maven](#)
  - [Descarga de Apache Maven](#)
- [Instalación de Maven](#)
  - set java\_home=C:\Program Files\Java\jdk1.8.0\_45
  - set m2\_home=c:\apache-maven-3.3.3
  - set path=%m2\_home%\bin;%path%
- [Ejecutar Maven](#)
  - mvn -v ó mvn -h

# Integración con herramientas visuales

- [Web de Eclipse](#)
  - [Descarga de Eclipse](#)
- [Web de NetBeans](#)
  - [Descarga de NetBeans](#)
- [Web de IntelliJ Idea](#)
  - [Descarga de IntelliJ](#)

# Creación, construcción y ejecución de un [proyecto](#)

- Creacion usando un arquetipo
  - `mvn archetype:generate -DgroupId=com.curso.app -DartifactId=primero -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false`
- Se creará una nueva carpeta llamada “primero”
  - Examinar la estructura de directorios
  - Examinar el [pom.xml](#) (Modelo de Objetos de Proyecto)
- Construcción
  - `mvn package`
- Volver a examinar la estructura de directorios
- Ejecutar el proyecto
  - `java -cp target/primero-1.0-SNAPSHOT.jar com.curso.app.App`

# Algunas órdenes fundamentales

- Compilar el proyecto: `mvn compile`
- Ejecutar los test del proyecto: `mvn test`
- Construir el proyecto: `mvn package`
- Instalar el proyecto construido en el repositorio local: `mvn install`
- Limpiar los artefactos generados: `mvn clean`
- Generar automáticamente documentación: `mvn site`
- Publicar el componente en un repositorio remoto: `mvn deploy`
- Generar un proyecto para Eclipse: `mvn eclipse:eclipse`
- Generar un proyecto para IntelliJ IDEA: `mvn idea:idea`
- Trabajo con recursos: `mvn process-resources`
  - Carpeta `src\main\resources`
  - Carpeta `src\test\resources`

# Publicar en un repositorio remoto I

- Web de Apache Archiva
  - Descargar el zip de Apache Archiva
- Arrancar el repositorio
  - C:\apache-archiva-2.2.0\bin\archiva.bat console
  - Abrir la interfaz web en http://localhost:8080
- Crear un usuario de administración
  - Usuario: admin
  - Clave: admin1

# Publicar en un repositorio remoto II

- Crear un usuario
  - Usuario: user1
  - Clave: user1
- Editar sus roles
  - activar “snapshot” e “internal” bajo las categorías “Repository Manager” y “Repository Observer”



# Publicar en un repositorio remoto III

- Buscar o crear la carpeta USER\_HOME/.m2
- Incluir el archivo settings.xml si no existe, o añadirle si existe:

```
<settings>
  <servers>
    <server>
      <id>archiva.internal</id>
      <username>user1</username>
      <password>user1</password>
    </server>
    <server>
      <id>archiva.snapshots</id>
      <username>user1</username>
      <password>user1</password>
    </server>
  </servers>
</settings>
```

# Publicar en un repositorio remoto IV

- Añadir al pom del proyecto “primero”

```
<distributionManagement>
  <repository>
    <id>archiva.internal</id>
    <name>Internal Release Repository</name>
    <url>http://localhost:8080/repository/internal/</url>
  </repository>
  <snapshotRepository>
    <id>archiva.snapshots</id>
    <name>Internal Snapshot Repository</name>
    <url>http://localhost:8080/repository/snapshots/</url>
  </snapshotRepository>
</distributionManagement>
```

- Publicar el proyecto en el repo remoto
  - mvn deploy

# Descargar de un repositorio remoto I

- Crear un proyecto nuevo llamado “segundo”
- Añadir una dependencia al proyecto “primero” en el pom.xml

```
<dependencies>
  <dependency>
    <groupId>com.curso.app</groupId>
    <artifactId>primero</artifactId>
    <version>1.0-SNAPSHOT</version>
    <type>jar</type>
  </dependency>
</dependencies>
```

# Descargar de un repositorio remoto II

- Crear una clase java Principal que llame al main del proyecto primero
- Añadir al pom.xml del proyecto “segundo” el repositorio remoto

```
<repositories>
```

```
  <repository>
```

```
    <id>archiva.snapshots</id>
```

```
    <url>http://localhost:8080/repository/snapshots/</url>
```

```
  </repository>
```

```
</repositories>
```

# Descargar de un repositorio remoto III

- Si queremos publicar en remoto el proyecto “segundo”, añadir al pom.xml

```
<distributionManagement>
  <repository>
    <id>archiva.internal</id>
    <name>Internal Release Repository</name>
    <url>http://localhost:8080/repository/internal/</url>
  </repository>
  <snapshotRepository>
    <id>archiva.snapshots</id>
    <name>Internal Snapshot Repository</name>
    <url>http://localhost:8080/repository/snapshots/</url>
  </snapshotRepository>
</distributionManagement>
```

- Ejecutar el proyecto con `mvn exec:java -Dexec.mainClass="com.mycompany.segundo.Principal"`

# Configuración de Maven I

- Opcionalmente, la variable de entorno MAVEN\_OPTS
  - set MAVEN\_OPTS="-Xms1024m -Xmx4096m -XX:PermSize=1024m" (JSE 1.0-7.0)
  - set MAVEN\_OPTS="-Xms1024m -Xmx4096m" (JSE 8.0)
- El archivo [settings.xml](#), en USER\_HOME/.m2
- En D:\apache-maven-3.3.3\conf\settings.xml hay otro archivo de configuración que se mezcla con el nuestro, el cual tiene preferencia en caso de conflicto

# Configuración de Maven II

- Puede ser necesario [configurar un proxy](#)
- Es posible sustituir los repos estándar por los nuestros y añadir otros, empleando [mirrors](#)
- Las credenciales de acceso pueden [encriptarse](#)
- [Recomendaciones](#) para la configuración de Maven
- [Filtrado](#) de recursos y uso de propiedades

# Proyectos con más de un módulo I

- Crear un directorio llamado “modulos”
- En su interior, ejecutar:
  - `mvn archetype:generate -DgroupId=com.curso.app -DartifactId=tercero -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false`
  - `mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-webapp -DgroupId=com.curso.app -DartifactId=appweb`



# Proyectos con más de un módulo II

- Crear un pom.xml bajo el directorio “modulos”

```
<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.curso.app</groupId>
  <artifactId>app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>pom</packaging>
  <modules>
    <module>tercero</module>
    <module>appweb</module>
  </modules>
</project>
```

# Proyectos con más de un módulo III

- Actualizar el pom.xml de appweb

```
<dependency>
```

```
  <groupId>com.curso.app</groupId>
```

```
  <artifactId>tercero</artifactId>
```

```
  <version>1.0-SNAPSHOT</version>
```

```
</dependency>
```

# Proyectos con más de un módulo IV

Añadir la siguiente información a los proyectos “tercero” y “appweb”

```
<parent>
```

```
<groupId>com.curso.app</groupId>
```

```
<artifactId>app</artifactId>
```

```
<version>1.0-SNAPSHOT</version>
```

```
</parent>
```

Desde el directorio “modulos” ejecutar: mvn clean install