Normativa de Desarrollo de Maven y Nexus 3

Fecha: 27/09/2021

Referencia:

EJIE S.A.

Mediterráneo, 3

01010 Vitoria-Gasteiz

Posta-kutxatila / Apartado: 809

01080 Vitoria-Gasteiz

Tel. 945 01 73 00*

Fax. 945 01 73 01

www.EJIE.es

Control de documentación

Título de documento: **Normativa de Desarrollo Maven en WLS11**

Histórico de versiones

Código:	Versión:	Fecha:	Resumen de cambios:
	01	27.09.2021	Primera versión
	02	19.10.2021	Revisado por el grupo
	03	17.11.2021	Explicados los 2 grupos de usuarios para acceso a la interfaz web de nexus
	04	15.12.2021	Creado Anexo II como copia del Anexo que existía y añadido Anexo I, buenas prácticas en maven.
	05	13.01.2022	Añadido Anexo III
	06	09.02.2022	Añadido en settings-nexus3.xml (ver 3.1) mirror “internal-repository” para evitar que si los pom tienen URLs de otros repositorios se fuerce el uso de Nexus3.
	07	10.02.2022	Añadida la posibilidad de solicitar la credencial de nexus3 en PC local.
	08	11.02.2022	Añadido anexo 6.5 con detalle de instalación de certificado de CA subordinada de Nexus3 en keystore de la jdk local. Añadido anexo 6.6 con detalle de subida de artefactos “no conformados”.
	09	13.05.2022	Se administra nexus3 para que los repositorios de nexus2 (proxy) sean observables por los usuarios estándar. Hasta ahora sólo se veían en el grupo y sólo aquellos artefactos que hubieran sido descargados por una petición de dependencia a nexus3. Así al menos pueden ver que artefactos se han descargado de nexus2 desde nexus3. Se añade un comentario en el punto 2.1
	10	07.09.2022	Se añade el repositorio “maven-public-releases” y se añade una nota sobre la indexación de los repositorios remotos (ver maven_central) en el punto 2.1. También se revisan los repositorios que

			se incluyen en el grupo de repositorios “in-house-app-group”.
	11	22.09.2022	Añadido Anexo IV, que facilita la migración de ivy a maven.
	12	30.11.2023	Se sustituye bin1 por bin
	12	24.05.2023	Añadido apartado 1 en el Anexo IV indicando buenas prácticas para evitar la subida de los JAR de tipo “sources” a Nexus.

Cambios producidos desde la última versión

Control de difusión

Responsable:	
Aprobado por:	
Firma:	Fecha:
Distribución: Interna	

Referencias de archivo

Autor: Innovación y Consultoría
Nombre archivo: Normativa de Desarrollo Maven y Nexus3.docx
Localización:

Contenido

1. Introducción

Este documento presenta las herramientas disponibles para el desarrollador que quiera hacer uso de las herramientas de Ejie para gestión de dependencias basadas en Maven y Nexus 3.

Maven es una herramienta de software para la gestión y construcción de proyectos.
Se basa en la utilización de ficheros POM (Project Object Model) que describen el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Realiza tareas de compilación del código y empaquetado, gestiona dependencias, etcétera.

En EJIE las tareas de empaquetado y compilación se gestionan vía tareas ant. Maven se utilizará solamente para la gestión de dependencias y publicación de artefactos en los repositorios Maven 3.

EJIE pone a disposición del desarrollador repositorios Maven en el entorno de desarrollo y tareas ant que permitirán gestionar dependencias, así como publicar artefactos. Además, expone dichos repositorios vía https para obtención de artefactos mediante tareas en el PC de desarrollo y en el servidor de compilación o mediante una aplicación web (Nexus).

Esta normativa surge inicialmente como evolución de la normativa maven y Archiva.

2. Herramientas en Servidor de Desarrollo

En el desarrollo de una aplicación se podrá hacer uso de las tareas ant de maven en el servidor de desarrollo (o de compilación) para bajarse dependencias existentes en los repositorios Maven de EJIE, así como publicar artefactos en ellos.

2.1. Repositorios Maven en Nexus 3

Dentro del gestor de repositorios de Nexus se han creado los siguientes repositorios de uso general:

in-house-app-group: Grupo de Repositorios que contendrá aquellas librerías que las aplicaciones pueden y deben usar.

URL: https://bin.alm02.itbatera.euskadi.eus/repository/in-house-app-group/

Contiene estos repositorios con este orden de preferencia (el artefacto se descarga desde el primer repositorio en el que se encuentre):

- ***in-house-app-releases***: Repositorio local que contendrá aquellas librerías "release" que puedan generar las aplicaciones. Este repositorio no permite sobrescribir archivos, y no permite SNAPSHOTS.

  URL: https://bin.alm02.itbatera.euskadi.eus/repository/in-house-app-releases/

- ***in-house-app-snaphosts***: Repositorio local que contendrá aquellas librerías "snapshots" para poder hacer integración continua.

  URL: https://bin.alm02.itbatera.euskadi.eus/repository/in-house-app-snapshots/

- ***in-house-app-nc-releases***: Repositorio para artefactos que sean necesarios para la aplicación que no estén en los repositorios anteriores (no conformes) (ej. librerías privadas de terceros).

  URL: https://bin.alm02.itbatera.euskadi.eus/repository/in-house-app-nc-releases/

  Se podrán subir los artefactos a través de la interfaz web. Pero será necesario pertenecer a un grupo de usuarios con los permisos habilitados (ver apartado 2.3.2)

- ***in-house-app-nc-snapshots***: Repositorio para artefactos que sean necesarios para la aplicación que no estén en los repositorios anteriores (no conformes) (ej. librerías privadas de terceros) de tipo snapshot

  URL: https://bin.alm02.itbatera.euskadi.eus/repository/in-house-app-nc-snapshots/

  Se podrán subir los artefactos a través de la interfaz web. Pero será necesario pertenecer a un grupo de usuarios con los permisos habilitados (ver apartado 2.3.2)

- ***ejie-maven-legacy-**** Repositorios remotos de nexus2. Estos repositorios se sirven en modo "proxy" *mientras siga habiendo aplicaciones que suban artefactos con ivy a ese gestor de repositorios*. (carácter temporal).
  Se incluyen los repositorios de *prod*, *pru* y *desa*, así como el de homologadas de ese gestor.

  A TENER EN CUENTA que estos repositorios "proxy" no mostrarán los artefactos hasta que hayamos hecho alguna descarga de alguno y además no se encuentran en las búsquedas, por lo que se recomienda usar el "browse" para ver que artefactos ya ha descargado y, en caso de no encontrar alguno, consultar la fuente original del repositorio (Nexus2, Maven Central, etc.) y hacer una primera descarga para que sea observable desde Nexus3.

- ***maven-public-releases***: Repositorio interno, parecido a app_releases.

  URL: https://bin.alm02.itbatera.euskadi.eus/service/rest/repository/browse/maven-public-releases/

  Se podrán subir los artefactos a través de la interfaz web. Pero será necesario pertenecer a un grupo de usuarios con los permisos habilitados (ver apartado 2.3.2)

- ***hdivsecurity-releases***: Repositorio remoto del producto hdiv.
  Incluye los artefactos de https://artifacts.hdivsecurity.com/nexus/content/repositories/releases que requiere credencial de acceso. Nos permite tener sincronizadas las librerías de HDIV para UDA.

- ***maven-central***: Repositorio remoto de uso genérico en internet.
  No es observable desde la consola de Nexus, pero incluye los artefactos de https://repo.maven.apache.org/maven2/

- ***maven-apache***: Repositorio remoto de uso genérico en internet.
  Incluye los artefactos de https://repository.apache.org/content/repositories/releases/

- ***maven-oracle***: Repositorio remoto de productos Oracle.
  Incluye los artefactos de https://maven.oracle.com/ , que no es observable sin credenciales.

- ***maven-jenkins-public***: Repositorio remoto de uso genérico de producto Jenkins.
  Incluye los artefactos del repositorio público de Jenkins: https://repo.jenkins-ci.org/public/

Algunas de las ventajas de usar Nexus en Ejie son:

- Los servidores de compilación no requieren salir a internet para resolver las dependencias.
- No es necesario publicar en un servidor de internet las librerías que desarrollemos para aplicaciones internas.
- La disponibilidad de un servicio on-premise.

IMPORTANTE: Es posible que hagamos búsquedas de determinados artefactos en nexus3 que sean de repositorios remotos, pero no los encontremos en Nexus3. El motivo es que hasta que no se haga una petición al artefacto/versión del repositorio remoto, el buscador de Nexus3 no lo tendrá indexado. Es por tanto necesario hacer una primera descarga de esos artefactos de repositorios remotos para que Nexus3 lo encuentre cuando hagamos búsquedas.

2.2. Tareas Ant en servidor de Desarrollo

En el apartado anterior se citan los diferentes repositorios que estarán albergados en EJIE. Sobre ellos se podrá realizar diferentes actuaciones. Las tareas ant de servidor nos permitirán publicar artefactos en ellos y bajar dependencias de ellos.

Se han creado en el servidor 3 nuevas tareas ant:

- **mavenGetDependenciesApp**. Obtiene las dependencias de un proyecto de aplicación empresarial (Ear y sus módulos War).
- **mavenGetDependenciesLib**. Descarga las dependencias de un proyecto de librería (Jar).
- **mavenPublishLib**. Publica un artefacto generado (*.jar) en el repositorio de publicaciones compartidas (verifica que el groupid sea del tipo com.ejie.*). Selecciona el repositorio de snapshots o releases si la versión del proyecto de maven termina o no *-SNAPSHOT.

2.2.1. mavenGetDependenciesApp

Esta tarea baja las dependencias especificadas en los ficheros pom ubicados en el directorio /aplic/bbb/tmp/deps: *pom_app.xml* y los diversos *pom_*<nombreWar>*.xml*.
El fichero pom_app.xml contendrá las dependencias de la aplicación EAR y las descargará en la ruta */aplic/bbb/src/bbbEAR/APP-INF/lib*.
Si existen módulos War, por cada uno de ellos existirá un *pom_*<nombreWar>*.xml*. Sus dependencias las bajará a la ruta */aplic/bbb/src/bbbEAR/wars/<nombreWar>/WEB-INF/lib* de cada módulo web.

Permite un parámetro opcional *DconfEnv* que podrá tener los valores "/" si el desarrollo está en el entorno de desarrollo productivo y "/softbase_ejie/" si se desarrolla en el entorno de softbase. Este parámetro se ha de pasar obligatoriamente en los traspasos con Anthill Pro.

Ej: ant mavenGetDependenciesApp –DconfEnv=/

2.2.2. mavenGetDependenciesLib

Esta tarea baja las dependencias especificadas en el fichero pom /aplic/bbb/tmp/deps/*pom_lib.xml*. Este fichero contiene las dependencias de la librería. La tarea ant descargará las dependencias en la ruta */aplic/bbb/src/bbbShLibClasses/lib*.

Permite un parámetro opcional *DconfEnv* que podrá tener los valores "/" si el desarrollo está en el entorno de desarrollo productivo y "/softbase_ejie" si se desarrolla en el entorno de softbase. Este parámetro se ha de pasar obligatoriamente en los traspasos con Anthill Pro.

Ej.: ant mavenGetDependenciesLib –DconfEnv=/

2.2.3. mavenPublishLib

Publica una librería en el repositorio de librerías compartidas de EJIE:
- app_releases, caso de ser una librería de tipo release.
- app_snapshots case de ser una librería de tipo snapshot.

La tarea parsea el pom, en concreto busca el tag <version> definido dentro del <project>. Si el valor termina en -SNAPSHOT (Ej: 1.0-SNAPSHOT) se sube al repositorio de snapshots.

Si no contiene al final de la versión la cadena -SNAPSHOT (Ej: 1.0) se sube al repositorio de releases.

App_releases	App_snapshosts
No se permiten snapshots	No se permiten releases
Mejor custodia de los artefactos a largo plazo	Puede tener menor retención de snapshots a largo plazo
En ese repositorio no se permite la sobreescritura de un artefacto.	

Esta tarea si no se le pasa parámetros publicará la librería generada por la tarea *createShLibJar*.y necesitará tener configurado correctamente el fichero pom_lib.xml en /aplic/bbb/tmp/deps.

Ej: ant mavenPublishLib

En el caso de que se pretenda subir otro artefacto generado, como pudieran ser clientes generados de EJBs (Enterprise JavaBean), etc se suministrarán parámetros adicionales:

- pomfile: fichero pom del artefacto
- jarfile: artefacto jar a publicar.

Ej: ant mavenPublishLib -Dpomfile=/aplic/tmp/deps/pom_stubs.xml -Djarfile=../bbbStubsEJBs.jar

2.2.4. Relación con antiguas tareas de Archiva

En la normativa de maven y Archiva se permitía el uso de repositorios privados. En maven con Nexus no vamos a crear repositorios privados, pues ahora los repositorios no son públicos, y la relación beneficio/coste de administración entre tener o no repositorios privados no parece ser mayor a uno.
Sin embargo, las tareas de maven para nexus se han hecho lo más compatibles hacia atrás, manteniendo para la mayoría de las aplicaciones que no usaban repositorios privados la migración a estas nuevas.
En esta tabla se muestra las tareas equivalentes.

Tareas maven archiva	Tareas maven nexus	Observaciones
getDependenciesAppArchiva	mavenGetDependenciesApp	
getDependenciesLibArchiva	mavenGetDependenciesLib	
publishPublicLib	mavenPublishLib	
publishPrivateLib		En nexus no hay repositorios privados.

2.3. Aplicación web Nexus

Los desarrolladores podrán acceder a la aplicación web para realizar consultas sobre las librerías soportadas en EJIE, o incluso y si fuera necesario, poder subir un artefacto que no se pueda crear desde tareas de compilación en servidor y no esté en los repositorios de maven central (librería no conformada).

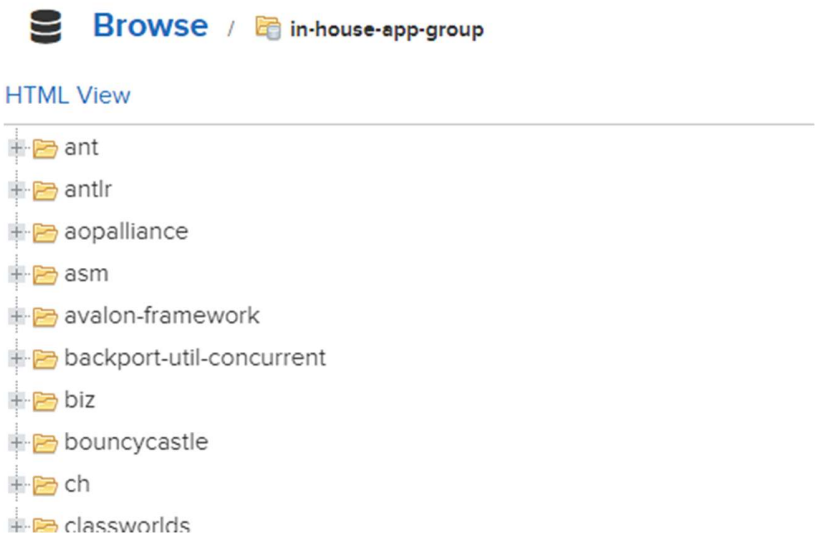
La URL es https://bin.alm02.itbatera.euskadi.eus/

Al ser accesible desde internet, se requiere usuario y contraseña para acceder a los repositorios.

El acceso como solo lectura lo da este grupo
CN=GGALMNexusViewers,OU=Productos,OU=Taldeak,DC=ejsarea,DC=net
Las credenciales de acceso son de tipo usuario y contraseña, las mismas que usamos en red corporativa.
Se debe solicitar a SASU a través de la petición: 30.3 Gestión Accesos / Zonas de Red

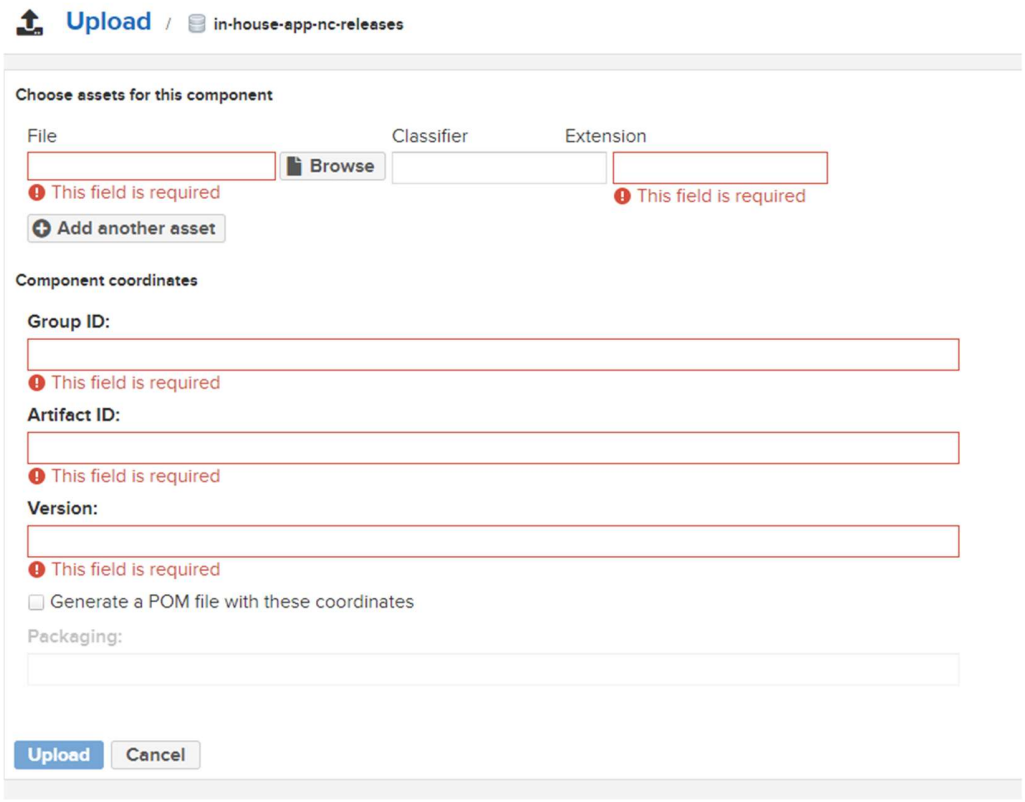
2.3.1. Explorar artefactos

Todos los usuarios autenticados podrán explorar el grupo de repositorios de nexus.



2.3.2. Subir artefactos no conformados

Si tenemos la credencial adecuada (sólo los usuarios del grupo CN=GGALMNexusNCUsers,OU=Productos,OU=Taldeak,DC=ejsarea,DC=net) pueden subir a artefactos al repositorio de artefactos “no conformados”:



Se debe tener en cuenta:

- No se recomienda la subida de artefactos no conformados, pues lo que estamos haciendo es que nuestra aplicación dependa de un artefacto que no podemos reconstruir en servidor. Este es el motivo por el que no dejamos ni siquiera subir artefactos desde entorno de PC. **Si queremos subir jar, tendremos que lanzar las tareas en el servidor ALM (Jenkins, …) que lanzará la tarea mavenPublishLib.**
- No es posible subir artefactos ”snapshot”, deben tener propósito “release”.
- Se recomienda marcar “Generate a POM file with these coordinates” para que maven pueda procesar luego esa dependencia atendiendo a su POM.

3. Herramientas de desarrollo en equipo local

Los equipos de desarrollo se podrán conectar a los repositorios de EJIE vía tareas maven para descargar las librerías en su local. Además, siempre se podrán conectar a la aplicación de nexus para navegar en los repositorios y descargar los artefactos manualmente.

No se contempla la subida de artefactos ni al repositorio de releases ni al de snapshosts desde entorno PC.

3.1. Tareas maven y normativa ficheros POM

Se descomprime apache-maven-3.2.5 en c:\

Se crea en C:\apache-maven-3.2.5\conf\ el fichero: settings-nexus3.xml con este contenido:

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
https://maven.apache.org/xsd/settings-1.0.0.xsd">
  <localRepository>C:\apache-maven-3.2.5\repoLocal</localRepository>
  <interactiveMode>false</interactiveMode>
  <offline>false</offline>

  <!--Activar en caso necesario -->
  <!--
  <proxies>
      <proxy>
            <active>true</active>
            <protocol>http</protocol>
            <host>proxyejgv.ejgvdns</host>
            <port>8080</port>
            <username>usuario</username>
            <password>contraseña</password>
            <nonProxyHosts>localserver</nonProxyHosts>
      </proxy>
    </proxies>
      -->

  <activeProfiles>
      <!--make the profile active all the time -->
      <activeProfile>nexus</activeProfile>
  </activeProfiles>

    <profiles>
    <profile>
      <id>nexus</id>
      <repositories>
        <repository>
          <id>nexus3</id>
          <url>https://bin.alm02.itbatera.euskadi.eus/repository/in-
house-app-group/</url>
          <releases>
            <enabled>true</enabled>
          </releases>
```

```
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>nexus3</id>
          <url>https://bin.alm02.itbatera.euskadi.eus/repository/in-
house-app-group/</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>

  <mirrors>
    <mirror>
      <id>internal-repository</id>
      <name>Maven Repository Manager running on nexus3</name>
      <url>https://bin.alm02.itbatera.euskadi.eus/repository/in-house-
app-group/</url>
      <mirrorOf>*,!nexus3</mirrorOf>
    </mirror>
  </mirrors>

  <servers>
    <server>
      <id>nexus3</id>
      <username>USUARIO</username>
      <password>PASSWORD</password>
    </server>
    <server>
      <id>internal-repository</id>
      <username>USUARIO</username>
      <password>PASSWORD</password>
    </server>
<!-- Alternativamente, se pueden usar variables que pasen desde la linea
de comandos con -Dnexus3.username=USUARIO y -Dnexus3.password=PASSWORD.
Ver más adelante “dependencias.cmd”
  <servers>
    <server>
      <id>nexus3</id>
      <username>${nexus3.username}</username>
      <password>${nexus3.password}</password>
    </server>
    <server>
      <id>internal-repository</id>
      <username>${nexus3.username}</username>
      <password>${nexus3.password}</password>
    </server>
  </servers>
```

```
  -->

  </servers>
</settings>
```

Una vez instalado maven se creará en cada módulo (proyecto de Eclipse):

- Módulo EAR: pom.xml con las dependencias.
- Módulo War: un pom.xml con las dependencias
- Módulo ClassesShlib: un pom.xml con las dependencias

La tarea de dependencias se lanza creando un dependencias.cmd con este contenido, ajustando la ruta a su JAVA_HOME. Se requiere al menos JDK 1.6 con esta versión de maven, pero como se hace una conexión https, es necesario que la JDK tenga la CA que ha emitido el certificado del servidor de Nexus como confiable y jdk1.8 o superior por usar TLS 1.2 el servidor de Neuxs.
Para ello, añadir la CA intermedia que ha generado el certificado de https://bin.alm02.itbatera.euskadi.eus/ como confiable en su JAVA_HOME\jre\lib\security\cacerts (la contraseña por defecto es changeit). Se puede usar la herramienta "KeyStoreExplorer" (ver https://keystore-explorer.org/) para facilitar esta tarea. Ver anexo 6.5 para detalles.

Detalle del fichero "dependencias.cmd" para lanzar maven en PC local:

```
set M2_HOME=C:\apache-maven-3.2.5
set PATH=%M2_HOME%\bin;%PATH%
set JAVA_HOME=C:\programas\jdk1.8\

@mvn -s %M2_HOME%\conf\settings-nexus3.xml -f pom.xml dependency:copy-
dependencies -DoutputDirectory=./EarContent/APP-INF/lib/
```

Alternativamente, recoger las credenciales desde línea de comandos y pasarlas a mvn (usando un settings-nexus.xml que recoja estas variables):

```
@echo off

set M2_HOME=C:\apache-maven-3.2.5
set PATH=%M2_HOME%\bin;%PATH%
set JAVA_HOME=C:\programas\jdk1.8\

for /F %%a in ('echo prompt $E ^| cmd') do set "ESC=%%a"
set color_white=%ESC%[37m
set color_black=%ESC%[30m
echo(
set /p cmd.nexus3.username=Nexus3 Username:
set /p cmd.nexus3.password=Nexus3 password:%color_black%
echo %color_white%

@mvn -s %M2_HOME%\conf\settings-nexus3.xml -f pom.xml dependency:copy-
dependencies -DoutputDirectory=./EarContent/APP-INF/lib/ -
Dnexus3.username=%cmd.nexus3.username% -
Dnexus3.password=%cmd.nexus3.password%
```

3.1.1. Ejemplo pom módulo EAR

Crear el fichero pom.xml con las dependencias como el que sigue a modo de ejemplo:

```
<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.ejie.aaa</groupId>
        <artifactId>aaaEAR</artifactId>
        <packaging>pom</packaging>
        <version>1.0-SNAPSHOT</version>
        <name>aaaEAR</name>
        <url>http://www.ejie.eus</url>

        <properties>

<org.springframework.version>4.3.22.RELEASE</org.springframework.version>

<org.springframework.security.version>4.2.11.RELEASE</org.springframework
.security.version>
                <org.logback.version>1.2.3</org.logback.version>
                <org.slf4j.version>1.7.25</org.slf4j.version>
                <com.ejie.x38.version>5.0.0-
RELEASE</com.ejie.x38.version>

<org.apache.tiles.version>3.0.8</org.apache.tiles.version>
                <!-- <org.jackson.version>2.8.11.3</org.jackson.version>
-->
                <org.jackson.version>2.7.9.5</org.jackson.version>
        </properties>

        <dependencies>
                <!-- Spring -->
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                        <version>${org.springframework.version}</version>
                        <exclusions>
                                <!-- Exclude Commons Logging in favor of
logback -->
                                <exclusion>
                                        <groupId>commons-
logging</groupId>
                                        <artifactId>commons-
logging</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-webmvc</artifactId>
                        <version>${org.springframework.version}</version>
                </dependency>
        </dependencies>


        <repositories>
        </repositories>
```

```
        <build>
                <plugins>
                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-dependency-
plugin</artifactId>
                                <executions>
                                        <execution>
                                                <id>copy-
dependencies</id>
                                                <phase>package</phase>
                                                <goals>
                                                        <goal>copy-
dependencies</goal>
                                                </goals>
                                                <configuration>

<outputDirectory>./EarContent/APP-INF/lib</outputDirectory>

<overWriteReleases>false</overWriteReleases>

<overWriteSnapshots>true</overWriteSnapshots>

<excludeTransitive>false</excludeTransitive>

<excludeScope>provided</excludeScope>
                                                </configuration>
                                        </execution>
                                </executions>
                        </plugin>
                </plugins>
        </build>
</project>
```

Es importante que el elemento del group id sea un com.ejie.bbb en todos los pom. En el caso del EAR no se publicará dicho artefacto.

Como ejemplo usar dependencias.cmd anterior.

3.1.2. Ejemplo pom módulo WAR

Crear el fichero pom.xml con las dependencias como el que sigue a modo de ejemplo:

```
<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.ejie.aaa</groupId>
        <artifactId>aaaModuloWAR</artifactId>
        <packaging>pom</packaging>
        <version>1.0-SNAPSHOT</version>
        <name>aaaModuloWAR </name>
        <url>http://www.ejie.eus</url>
```

```
        <properties>

<org.springframework.version>4.3.22.RELEASE</org.springframework.version>

<org.springframework.security.version>4.2.11.RELEASE</org.springframework
.security.version>
                <org.logback.version>1.2.3</org.logback.version>
                <org.slf4j.version>1.7.25</org.slf4j.version>
                <com.ejie.x38.version>5.0.0-
SNAPSHOT</com.ejie.x38.version>

<org.apache.tiles.version>3.0.8</org.apache.tiles.version>
                <!-- <org.jackson.version>2.8.11.3</org.jackson.version>
-->
                <org.jackson.version>2.7.9.5</org.jackson.version>
        </properties>

        <dependencies>
                <!-- Spring -->
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                        <version>${org.springframework.version}</version>
                        <exclusions>
                                <!-- Exclude Commons Logging in favor of
logback -->
                                <exclusion>
                                        <groupId>commons-
logging</groupId>
                                        <artifactId>commons-
logging</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-webmvc</artifactId>
                        <version>${org.springframework.version}</version>
                </dependency>
        </dependencies>


        <repositories>
        </repositories>


        <build>
                <plugins>
                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-dependency-
plugin</artifactId>
                                <executions>
                                        <execution>
```

```
                                                    <id>copy-
dependencies</id>
                                                    <phase>package</phase>
                                                    <goals>
                                                            <goal>copy-
dependencies</goal>
                                                    </goals>
                                                    <configuration>

<outputDirectory>./WebContent/WEB-INF/lib</outputDirectory>

<overWriteReleases>false</overWriteReleases>

<overWriteSnapshots>true</overWriteSnapshots>

<excludeTransitive>false</excludeTransitive>

<excludeScope>provided</excludeScope>
                                                    </configuration>
                                            </execution>
                                    </executions>
                            </plugin>
                    </plugins>
            </build>
</project>
```

Es importante que el elemento del group id sea un com.ejie.bbb en todos los pom. En el caso del WAR no se publicará dicho artefacto.

Ejemplo fichero “dependencias.cmd” para lanzar maven:

```
set M2_HOME=C:\apache-maven-3.2.5
set PATH=%M2_HOME%\bin;%PATH%
set JAVA_HOME=C:\programas\jdk1.8\

@mvn -s %M2_HOME%\conf\settings-nexus3.xml -f pom.xml dependency:copy-
dependencies -DoutputDirectory=./WebContent/WEB-INF/lib/
```

3.1.3. Ejemplo pom módulo Classes Shlib

En el caso de que se pretenda crear una librería para ser utilizada por otras aplicaciones se generará un artefacto jar en un proyecto Java de Eclipse de tipo ShLib.

En el fichero pom de la librería se describe cómo resolver las dependencias que tuviera de otras librerías. Este mismo pom se usará en el servidor en el uso de la tarea de publicación.

Se creará un pom.xml con el siguiente contenido:

```
<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.ejie.aaa</groupId>
        <artifactId>aaaShLibClasses</artifactId>
        <packaging>jar</packaging>
        <version>1.0-SNAPSHOT</version>
        <name>aaaShLibClasses</name>
        <url>http://www.ejie.eus</url>

        <properties>

<org.springframework.version>4.3.22.RELEASE</org.springframework.version>

<org.springframework.security.version>4.2.11.RELEASE</org.springframework
.security.version>
                <org.logback.version>1.2.3</org.logback.version>
                <org.slf4j.version>1.7.25</org.slf4j.version>
                <com.ejie.x38.version>5.0.0-
SNAPSHOT</com.ejie.x38.version>

<org.apache.tiles.version>3.0.8</org.apache.tiles.version>
                <!-- <org.jackson.version>2.8.11.3</org.jackson.version>
-->
                <org.jackson.version>2.7.9.5</org.jackson.version>
        </properties>

        <dependencies>
                <!-- Spring -->
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                        <version>${org.springframework.version}</version>
                        <exclusions>
                                <!-- Exclude Commons Logging in favor of
logback -->
                                <exclusion>
                                        <groupId>commons-
logging</groupId>
                                        <artifactId>commons-
logging</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-webmvc</artifactId>
                        <version>${org.springframework.version}</version>
                </dependency>
        </dependencies>


        <repositories>
        </repositories>


        <build>
                <plugins>
                        <plugin>
```

```
<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-dependency-
plugin</artifactId>
                                <executions>
                                        <execution>
                                                <id>copy-
dependencies</id>
                                                <phase>package</phase>
                                                <goals>
                                                        <goal>copy-
dependencies</goal>
                                                </goals>
                                                <configuration>

<outputDirectory>./lib</outputDirectory>

<overWriteReleases>false</overWriteReleases>

<overWriteSnapshots>true</overWriteSnapshots>

<excludeTransitive>false</excludeTransitive>

<excludeScope>provided</excludeScope>
                                                </configuration>
                                        </execution>
                                </executions>
                        </plugin>
                </plugins>
        </build>
</project>
```

Es importante que el elemento del group id sea un com.ejie.bbb en todos los pom. En este caso se usará cuando se publique desde servidor el artefacto.

Ejemplo “dependencias.cmd” para lanzar maven:

```
set M2_HOME=C:\apache-maven-3.2.5
set PATH=%M2_HOME%\bin;%PATH%
set JAVA_HOME=C:\programas\jdk1.8\

@mvn -s %M2_HOME%\conf\settings-nexus3.xml -f pom.xml dependency:copy-
dependencies -DoutputDirectory=./lib/
```

4. Anexo I. Buenas prácticas en maven

1. No incluir url de repositorios en vuestros pom, de manera que las url de los repositorios se resuelvan en el fichero de “maven-settings”. De esta manera, si cambiamos de servidor de dependencias el cambio está controlado en un fichero, y no en múltiples pom.
2. Usar propiedades para definir las versiones de objetos de dependencias, de esta manera se puede ver más rápidamente que versiones usamos sin tener que recorrer todo el pom.
3. Intentar evitar propiedades inherentes, sobre todo si trabajamos con multiproyectos, donde una propiedad puede afectar a un proyecto dependiente, y por otro lado, es difícil saber dónde se ha definido una de esas propiedades si vemos el pom “hijo”.
4. Usa SNAPSHOT si construyes librerías y quieres que alguien comience a utilizarla, pero aún no son definitivas. Cuando crees una “release” ten en cuenta que no se debería sobrescribir bajo ningún supuesto, y el servidor de artefactos tendrá definida una política de gestión de SNAPSHOT con auto purgado, pero no de releases. Tener muchas releases cuando realmente sean SNAPSHOT no es adecuado.

4.1. Evitar subida de sources en formato JAR a Nexus

1. Los ficheros “sources” en formato JAR no han de ser subidos a Nexus, con el objetivo de que el código fuente de las aplicaciones no pueda ser descargado por usuarios que tengan acceso y permisos de descarga en Nexus pero no en la aplicación. El objetivo final es minimizar los riesgos de seguridad que puedan producirse en dichas aplicaciones.
2. Si se requiere para el proyecto el uso del plugin “maven-source-plugin” y se encuentra así definido en el pom.xml, del proyecto o del pom.xml padre en su caso, se ha de incluir la propiedad “attach” con el valor “false”. Con ello Maven puede continuar creando el fichero JAR de los sources pero no se realiza su subida a Nexus al ejecutar un comando “mvn deploy”. Se incluye un ejemplo de esta configuración en negrita:

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-source-plugin</artifactId>
        <version>${version.source.plugin}</version>
        <executions>
                <execution>
                        <id>attach-sources</id>
                        <goals>
                                <goal>jar-no-fork</goal>
                        </goals>
                        <configuration>
                                <attach>false</attach>
                        </configuration>
                </execution>
        </executions>
</plugin>
```

5. Anexo II. Detalle de tareas ant en servidor

En servidor se usa jdk 1.6 y maven 3.2.5, que es la versión más alta que permite usar jdk 1.6. Además, en servidor se accede a los repositorios de nexus3 por http, debido a que jdk1.6 no es capaz de soportar el *handshake* de seguridad que establece el servidor por https.

Es por tanto ***necesario el acceso desde los servidores de compilación de desarrollo al repositorio nexus3 por http.***
http://bin.alm02.batera.euskadi.eus

Desde los PCs es preferible el acceso por https tal y como está indicado anteriormente.

5.1. Variables necesarias para las tareas de maven

```
#Nexus3
MAVEN_HOME=${dir.j2se}/apache-maven-3.2.5
maven.settings-security.file=${MAVEN_HOME}/conf/settings-security.xml
maven.settings-nexus3.file=${MAVEN_HOME}/conf/settings-nexus3.xml
jelly.dependencies_war_nexus3=${dir.jelly.src}/build-ejie-dependencies-war-nexus3.jelly
#Estas urls son accesibles desde Servidor, no desde PC de desarrollo. En entorno PC usar https en vez de http.
maven.nexus3.reporeleases=http://bin.alm02.batera.euskadi.eus/repository/in-house-app-releases/
maven.nexus3.reposnapshots=http://bin.alm02.batera.euskadi.eus/repository/in-house-app-snapshots/
```

5.2. Tareas ant

```
      <target name="mavenGetDependenciesEar_">
          <delete>
              <fileset dir="${dir.app.fuentes.EAR}/APP-INF/lib/"
includes="**/*.jar"/>
          </delete>
            <echo message="DEBUG:  MAVEN_HOME = ${MAVEN_HOME}"/>
          <exec dir="${dir.app.ant}" executable="cmd.exe" osfamily="windows">
            <!--env key="JAVA_HOME"
value="C:\Users\xxx\Documents\programas\jdk1.8\java_home"/-->
            <env key="JAVA_HOME" value="C:\Program Files\Java\jrockit-
jdk1.6.0_45-R28.2.7-4.1.0"/>
            <env key="repolocal.app" value="${repolocal.app}"/>
              <arg value="/C"/>
              <arg value="${MAVEN_HOME}\bin\mvn.bat"/>
              <arg line="-Dsettings.security=${maven.settings-security.file} -s
${maven.settings-nexus3.file} -f ${pom.file.EAR} dependency:copy-dependencies -
DoutputDirectory=${dir.app.fuentes.EAR}/APP-INF/lib/" />
            </exec>
          <exec dir="${dir.app.ant}" executable="/bin/ksh" osfamily="unix">
            <env key="JAVA_HOME" value="${java.home}"/>
            <env key="repolocal.app" value="${repolocal.app}"/>
```

```
              <arg value="${MAVEN_HOME}/bin/mvn"/>
              <arg line="-Dsettings.security=${maven.settings-security.file} -s
${maven.settings-nexus3.file} -f ${pom.file.EAR} dependency:copy-dependencies -
DoutputDirectory=${dir.app.fuentes.EAR}/APP-INF/lib/" />
            </exec>
      </target>

<target name="mavenGetDependenciesApp_">
          <antcall target="mavenGetDependenciesEar_" />
          <antcall target="mavenGetDependenciesWar_" />

          <delete includeEmptyDirs="true" failonerror="true">
              <fileset dir="${repolocal.app}"/>
          </delete>

      </target>
      <target name="mavenGetDependenciesLib_">
          <delete>
            <fileset dir="${dir.app.fuentes.JAR}/lib/" includes="**/*.jar"/>
          </delete>
            <echo message="DEBUG:  MAVEN_HOME = ${MAVEN_HOME}"/>
          <exec dir="${dir.app.ant}" executable="cmd.exe" osfamily="windows">
            <env key="JAVA_HOME"
value="C:\Users\xxx\Documents\programas\jdk1.8\java_home"/>
            <env key="repolocal.app" value="${repolocal.app}"/>
              <arg value="/C"/>
              <arg value="${MAVEN_HOME}\bin\mvn.bat"/>
              <arg line="-Dsettings.security=${maven.settings-security.file} -s
${maven.settings-nexus3.file} -f ${pom.file.JAR} dependency:copy-dependencies -
DoutputDirectory=${dir.app.fuentes.JAR}/lib/" />
            </exec>
          <exec dir="${dir.app.ant}" executable="/bin/ksh" osfamily="unix">
            <env key="JAVA_HOME" value="${java.home}"/>
            <env key="repolocal.app" value="${repolocal.app}"/>
              <arg value="${MAVEN_HOME}/bin/mvn"/>
              <arg line="-Dsettings.security=${maven.settings-security.file} -s
${maven.settings-nexus3.file} -f ${pom.file.JAR} dependency:copy-dependencies -
DoutputDirectory=${dir.app.fuentes.JAR}/lib/" />
            </exec>

          <delete includeEmptyDirs="true" failonerror="true">
              <fileset dir="${repolocal.app}"/>
          </delete>

      </target>

          <target name="mavenPublishLibSnapshot_" if="isSnapshot"
depends="mavenParseaPomVersion">
        <echo message="&gt;&gt;&gt; Publicando dependencia snapshot." />
        <property name="jarfile"
value="${dir.app.distribuibles}/${app}ShLibClasses.jar"/>
        <property name="pomfile" value="${pom.file.JAR}"/>
        <echo message="DEBUG:  MAVEN_HOME = ${MAVEN_HOME}"/>
        <echo message="DEBUG:  jarfile = ${jarfile}"/>
        <echo message="DEBUG:  pomfile = ${pomfile}"/>
```

```
        <exec dir="${dir.app.ant}" executable="cmd.exe" osfamily="windows">
            <env key="JAVA_HOME"
value="C:\Users\jriobell\Documents\programas\jdk1.8\java_home"/>
            <env key="repolocal.app" value="${repolocal.app}"/>
            <env key="jarfile" value="${jarfile}"/>
            <env key="pomfile" value="${pomfile}"/>
            <arg value="/C"/>
            <arg value="${MAVEN_HOME}\bin\mvn.bat"/>
            <arg line="-Dsettings.security=${maven.settings-security.file} -s
${maven.settings-nexus3.file} -Durl=${maven.nexus3.reposnapshots} -
DrepositoryId=nexus3_app-snapshots -Dfile=${jarfile} -DpomFile=${pomfile}
deploy:deploy-file" />
        </exec>
        <exec dir="${dir.app.ant}" executable="/bin/ksh" osfamily="unix">
            <env key="JAVA_HOME" value="${java.home}"/>
            <env key="repolocal.app" value="${repolocal.app}"/>
            <env key="jarfile" value="${jarfile}"/>
            <env key="pomfile" value="${pomfile}"/>
            <arg value="${MAVEN_HOME}/bin/mvn"/>
            <arg line="-Dsettings.security=${maven.settings-security.file} -s
${maven.settings-nexus3.file} -Durl=${maven.nexus3.reposnapshots} -
DrepositoryId=nexus3_app-snapshots  -Dfile=${jarfile} -DpomFile=${pomfile}
deploy:deploy-file" />
        </exec>
    </target>

    <target name="mavenPublishLibRelease_" if="isRelease"
depends="mavenParseaPomVersion">
        <echo message="&gt;&gt;&gt; Publicando dependencia release." />
        <property name="jarfile"
value="${dir.app.distribuibles}/${app}ShLibClasses.jar"/>
        <property name="pomfile" value="${pom.file.JAR}"/>
        <echo message="DEBUG:  MAVEN_HOME = ${MAVEN_HOME}"/>
        <echo message="DEBUG:  jarfile = ${jarfile}"/>
        <echo message="DEBUG:  pomfile = ${pomfile}"/>
        <exec dir="${dir.app.ant}" executable="cmd.exe" osfamily="windows">
            <env key="JAVA_HOME"
value="C:\Users\jriobell\Documents\programas\jdk1.8\java_home"/>
            <env key="repolocal.app" value="${repolocal.app}"/>
            <env key="jarfile" value="${jarfile}"/>
            <env key="pomfile" value="${pomfile}"/>
            <arg value="/C"/>
            <arg value="${MAVEN_HOME}\bin\mvn.bat"/>
            <arg line="-Dsettings.security=${maven.settings-security.file} -s
${maven.settings-nexus3.file} -Durl=${maven.nexus3.reporeleases} -
DrepositoryId=nexus3_app-releases -Dfile=${jarfile} -DpomFile=${pomfile}
deploy:deploy-file" />
        </exec>
        <exec dir="${dir.app.ant}" executable="/bin/ksh" osfamily="unix">
            <env key="JAVA_HOME" value="${java.home}"/>
            <env key="repolocal.app" value="${repolocal.app}"/>
            <env key="jarfile" value="${jarfile}"/>
            <env key="pomfile" value="${pomfile}"/>
            <arg value="${MAVEN_HOME}/bin/mvn"/>
```

```
            <arg line="-Dsettings.security=${maven.settings-security.file} -s
${maven.settings-nexus3.file} -Durl=${maven.nexus3.reporeleases} -
DrepositoryId=nexus3_app-releases -Dfile=${jarfile} -DpomFile=${pomfile}
deploy:deploy-file" />
        </exec>
    </target>



Jelly:
<?xml version="1.0"?>
<!-- Esta tarea realiza la obtencion de dependencias de los modulos WEB.-->
<j:jelly trim="false" xmlns:j="jelly:core" xmlns:log="jelly:log"
xmlns:ant="jelly:ant" xmlns:jxml="jelly:xml" xmlns:util="jelly:util">


      <!-- +=======================================================+ -->
      <!-- |                     IMPORTACIONES                     | -->
      <!-- +=======================================================+ -->
      <ant:property name="dir.j2se" value="/j2se"/>
      <ant:property name="dir.scripts.ant" value="${dir.j2se}/apache-ant-
1.7.1/ejie_wls11"/>
      <ant:property file="${dir.scripts.ant}/build-ejie-variables.properties"/>
        <ant:property name="confEnv" value="${entorno}"/>
      <ant:property file="${dir.app}/.m2/maven.properties"/>

      <ant:property name="MAVEN_HOME" value="${dir.j2se}/apache-maven-3.2.5"/>
      <ant:property name="maven.settings-security.file"
value="${dir.j2se}/apache-maven-3.2.5/conf/settings-security.xml"/>
      <ant:property name="maven.settings-nexus3.file"
value="${dir.j2se}/apache-maven-3.2.5/conf/settings-nexus3.xml" />
      <ant:property name="java.home"
value="/bea/jvm/jrockit_160_33_R28.2.4_4.1.0_x64"/>
      <!-- +=======================================================+ -->
      <!-- |                      OPERACIONES                      | -->
      <!-- +=======================================================+ -->
      <j:new className="java.io.File" var="fileReader">
            <j:arg value="${dir.app.fuentes.EAR}/wars"/>
      </j:new>

      <j:forEach var="elemento" items="${fileReader.listFiles()}">
            <util:file var="fpom"
name="${dir.pom.file}/pom_${elemento.getName()}.xml"/>
            <!--ant:echo>pasa ${fpom.exists()}</ant:echo-->
            <j:if test="${elemento.isDirectory() &amp;&amp; fpom.exists()
&amp;&amp; (elemento.getName().endsWith('WAR') ||
elemento.getName().endsWith('War'))}">
                        <ant:delete>
                           <ant:fileset
dir="${dir.app.fuentes.EAR}/wars/${elemento.getName()}/WEB-INF/lib/"
includes="**/*.jar"/>
                        </ant:delete>

                  <ant:property name="pom.file.WAR"
value="${dir.pom.file}/pom_${elemento.getName()}.xml"/>
```

```
                    <!-- TAREA MVN de dependencias -->
                    <ant:echo>pom ${pom.file.WAR}</ant:echo>
                    <ant:echo>MAVEN_HOME ${MAVEN_HOME}</ant:echo>
                    <ant:echo>repolocal.app ${repolocal.app}</ant:echo>
                    <!-- ant:echo>-Dsettings.security=${dir.j2se}/apache-maven-
3.2.5/conf/settings-security.xml -s ${dir.j2se}/apache-maven-
3.2.5/conf/settings-nexus3.xml -f ${pom.file.WAR} dependency:copy-dependencies -
DoutputDirectory=${dir.app.fuentes.EAR}/wars/${elemento.getName()}/WEB-
INF/lib/</ant:echo-->
                <ant:exec dir="${dir.app.ant}" executable="cmd.exe"
osfamily="windows">
            <!--  ant:env key="JAVA_HOME"
value="C:\Users\xxx\Documents\programas\jdk1.8\java_home"/-->
            <ant:env key="JAVA_HOME" value="C:\Program Files\Java\jrockit-
jdk1.6.0_45-R28.2.7-4.1.0"/>

            <ant:env key="repolocal.app" value="${repolocal.app}"/>
              <ant:arg value="/C"/>
              <ant:arg value="${MAVEN_HOME}\bin\mvn.bat"/>
              <ant:arg line="-Dsettings.security=${dir.j2se}/apache-maven-
3.2.5/conf/settings-security.xml -s ${dir.j2se}/apache-maven-
3.2.5/conf/settings-nexus3.xml -f ${pom.file.WAR} dependency:copy-dependencies -
DoutputDirectory=${dir.app.fuentes.EAR}/wars/${elemento.getName()}/WEB-INF/lib/"
/>
                    </ant:exec>

                <ant:exec dir="${dir.app.ant}" executable="/bin/ksh"
osfamily="unix">
            <ant:env key="JAVA_HOME" value="${java.home}"/>
            <ant:env key="repolocal.app" value="${repolocal.app}"/>
              <ant:arg value="${MAVEN_HOME}/bin/mvn"/>
              <ant:arg line="-Dsettings.security=${dir.j2se}/apache-maven-
3.2.5/conf/settings-security.xml -s ${dir.j2se}/apache-maven-
3.2.5/conf/settings-nexus3.xml -f ${pom.file.WAR} dependency:copy-dependencies -
DoutputDirectory=${dir.app.fuentes.EAR}/wars/${elemento.getName()}/WEB-INF/lib/"
/>
                    </ant:exec>


            </j:if>

      </j:forEach>
</j:jelly>
```

6. Anexo III. Errores conocidos

6.1. Falla la dependencia

Si Nexus3 no encuentra un artefacto, por ejemplo, vemos este error al lanzar la tarea de maven de nexus3:

[exec] [ERROR] Failed to execute goal on project xxxEAR: Could not resolve dependencies for project com.ejie.xxx:xxxEAR:pom:1.0-SNAPSHOT: Failed to collect dependencies at jackson:jackson-annotations:jar:2.1.4: Failed to read artifact descriptor for jackson:jackson-annotations:jar:2.1.4: Could not transfer artifact jackson:jackson-annotations:pom:2.1.4 from/to central (https://repo.maven.apache.org/maven2): repo.maven.apache.org: Unknown host repo.maven.apache.org -> [Help 1] [exec] [ERROR] [exec] [ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch. [exec] [ERROR] Re-run Maven using the -X switch to enable full debug logging. [exec] [ERROR] [exec] [ERROR] For more information about the errors and possible solutions, please read the following articles: [exec] [ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/DependencyResolutionException [exec] Result: 1

Tener en cuenta que la tarea de maven si ve que el repo de Nexus3 no resuelve la dependencia, lo intenta con el repo de macen central, pero el servidor de compilación no tendrá acceso a internet (que no debería, porque Nexus3 lo hace en su lugar).

Revisar el pom de xxx, pues es posible que la dependencia no sea correcta. En este caso, el pom tenía:

```
<dependency>
<groupId>jackson</groupId>
<artifactId>jackson-annotations</artifactId>
<version>2.1.4</version>
</dependency>
```

Pero vimos que la dependencia debería ser:

```
<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations -->
<dependency>
   <groupId>com.fasterxml.jackson.core</groupId>
   <artifactId>jackson-annotations</artifactId>
   <version>2.1.4</version>
</dependency>
```

(tal y como indica https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations/2.1.4)

Las tareas de Archiva, con el objetivo de minimizar los tiempos de resolución de dependencias, copiaban al repo local ciertos artefactos, por lo que podemos tener pom incorrectos que funcionaban con Archiva, pero al pasar a las nuevas tareas descubramos estos errores.

6.2. No puedo subir artefactos al repositorio de no conformadas

Se ha establecido que el grupo de los artefactos del repositorio de artefactos “no conformados” (in-house-app-nc-releases) tenga que ser obligatoriamente: com.ejie.nc

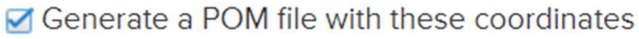
Se recomienda usar código de aplicación como sufijo del grupo. Ejemplo: com.ejie.nc.aaa.

Se recomienda usar código de aplicación como parte del nombre del artefacto. Ejemplo: aaaCoreLib. Para más detalles de cómo realizar la subida, puede consultar más adelante el punto 6.6

Tener en cuenta que también que desde la interfaz HTML de Nexus no se permite la subida a cualquier repositorio de tipo “snapshots”.

6.3. Fallan las dependencias con artefactos subidos a no conformados

Es posible que no haya marcado que se genere un POM por defecto, y maven intentará su descarga, pero al no encontrarlo, no procesa esa dependencia.
Se recomienda subir marcando esa generación de POM:



6.4. Se descargan varias versiones de un mismo artefacto

Es posible que nuestro pom especifique una versión específica de determinado artefacto, pero alguna otra dependencia que tengamos dependa de otra versión del mismo.

Miproyecto depende de

- Artefacto 1, v1.0.0
- Artefacto 2, v1.2.0
- …

Artefacto2, v1.2.0 depende de:

- Artefacto 1, v1.2.0

¿qué versión de artefacto1 resolverá maven?

Parece que por defecto prevalece el nivel de dependencia, de manera que posiblemente use Artefacto 1, v1.0.0, porque es una dependencia de un nivel superior a la del Artefacto2.

Pero maven plantea otras posibilidades que podemos considerar cuando la colisión sea en el mismo nivel de dependencias. En ese caso, maven suele usar la primera versión dependiente que encuentre en el pom en conflicto, pero podemos usar <exclude> para evitar ese orden por defecto.

- Excluir las dependencias de Artefacto 1 en Artefacto2 usando <exclude>

```
<groupId>xxx2</groupId>
<artifactId>Artefacto 2</artifactId>
<version>1.2.0</version>
<exclusions>
<exclusion>
<groupId>xxx1</groupId>
<artifactId>Artefacto1</artifactId>
</exclusion>
</exclusions>
```

```
</dependency>
```

- Cambiar nuestra aplicación para que a un nivel superior dependa de Artefacto1, v1.2.0 (esto fuerza las dependencias de los “hijos”);

```
<dependency>
        <groupId>xxx1</groupId>
        <artifactId>Artefacto 1</artifactId>
        <version>1.2.0</version>
<dependency>
        <groupId>xxx2</groupId>
        <artifactId>Artefacto 2</artifactId>
        <version>1.2.0</version>
</dependency>
```

Se puede consultar más información en esta URL:
https://maven.apache.org/guides/introduction/introduction-to-optional-and-excludes-dependencies.html

Es interesante lanzar en estos casos, en el entorno de PC: *dependency:tree -Dverbose*
Ejemplo:

mvn -s %M2_HOME%\conf\settings-nexus3_PC.xml -f C:\aplic\aaa\tmp\deps\pom_aaaEAR.xml dependency:tree -Dverbose -DoutputDirectory=./EarContent/APP-INF/lib/

Maven detecta posibles colisiones de versiones. Si la colisión es en el mismo nivel, es posible que seleccione la última versión que encuentre en el pom, y como hemos dicho, prevalece el nivel de dependencias superior frente al inferior. O fijas la versión en un nivel superior o usas <exclude> para resolver los conflictos.

Lo adecuado en general sería usar las versiones más actualizadas en nuestros proyectos.

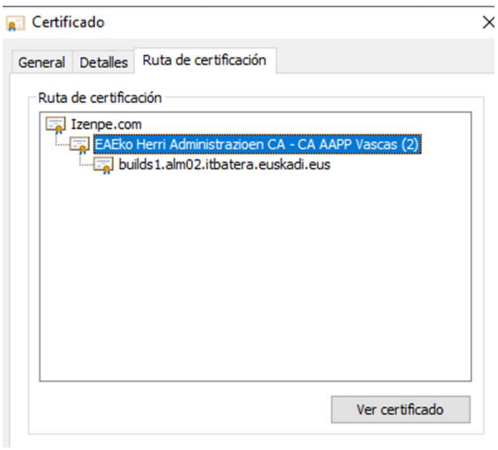
Alternativamente a esas soluciones, puede fijarse qué versión usar en todas las dependencias “hijas” usando <dependencyManagement>, al menos para multiproyectos, que no suele ser el caso de uso de aplicaciones jee en Ejie, por lo que se recomienda o usar <exclude> o definir en un nivel superior de dependencias la versión que queremos fijar...
https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html#Dependency_Management

6.5. Añadir certificado subordinado de Nexus3 en JDK local

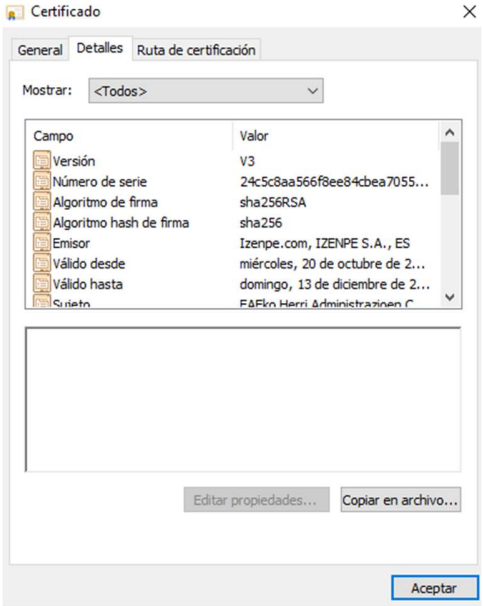
Para que *maven* en PC local pueda comunicarse con Nexus3 con https, puede ser necesario añadir en la JDK 1.8 o superior el certificado que ha emitido el certificado del servidor. No es buena

práctica instalar el certificado del servidor como confiable porque suele renovarse cada pocos años, pero el certificado de la CA subordinada que lo ha emitido es mucho más longevo.

- Descargar el certificado subordinado con el navegador. Es el certificado que emite el certificado del servidor. “EAEko Herri Administrazioen CA - CA AAPP Vascas (2)” Ejemplo en navegadores que usen los visores de certificados del propio Windows:



Pulsamos en “Ver Certificado”

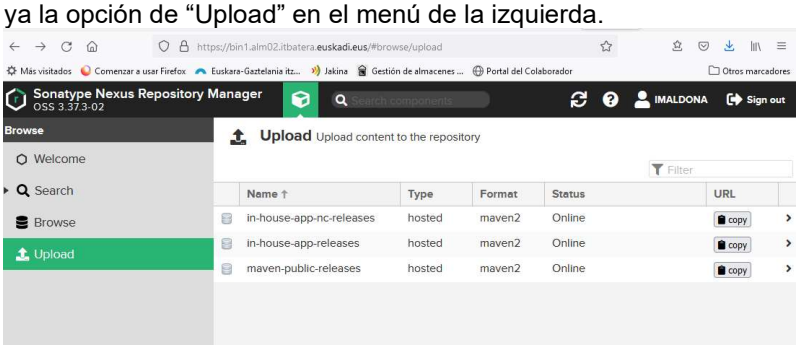


Pulsamos en “Copiar en archivo…”. Seleccionar “DER” y guardar en una ruta local temporal.

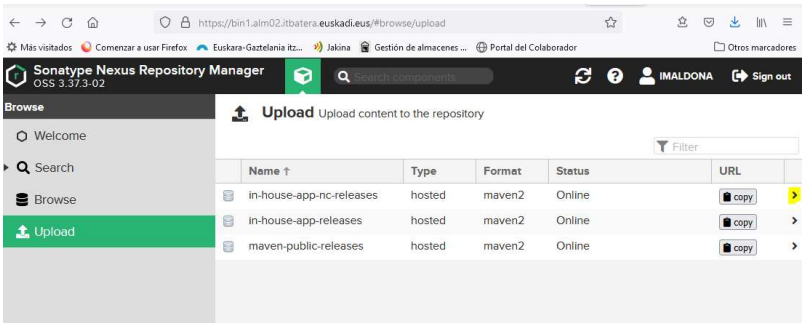
- Instalar la herramienta KeyTool.
- Lanzar la herramienta KeyTool y abrir el keystore “%JAVA_HOME%\lib\security\cacerts. La password por defecto es “changeit”.
- Añadir como “trusted” el certificado subordinado del paso 1.
- Guardar el kesytore “cacerts” en la propia herramienta KeyTool. Si da error, pude ser que tengas permisos para modificar el fichero; prueba ejecutando KeyTool con usuario administrador local.

6.6. Detalle de formulario de subida de artefactos “no conformados”

Entramos en el nexus https://bin.alm02.itbatera.euskadi.eus/ y, si tenemos permiso, debemos ver ya la opción de “Upload” en el menú de la izquierda.



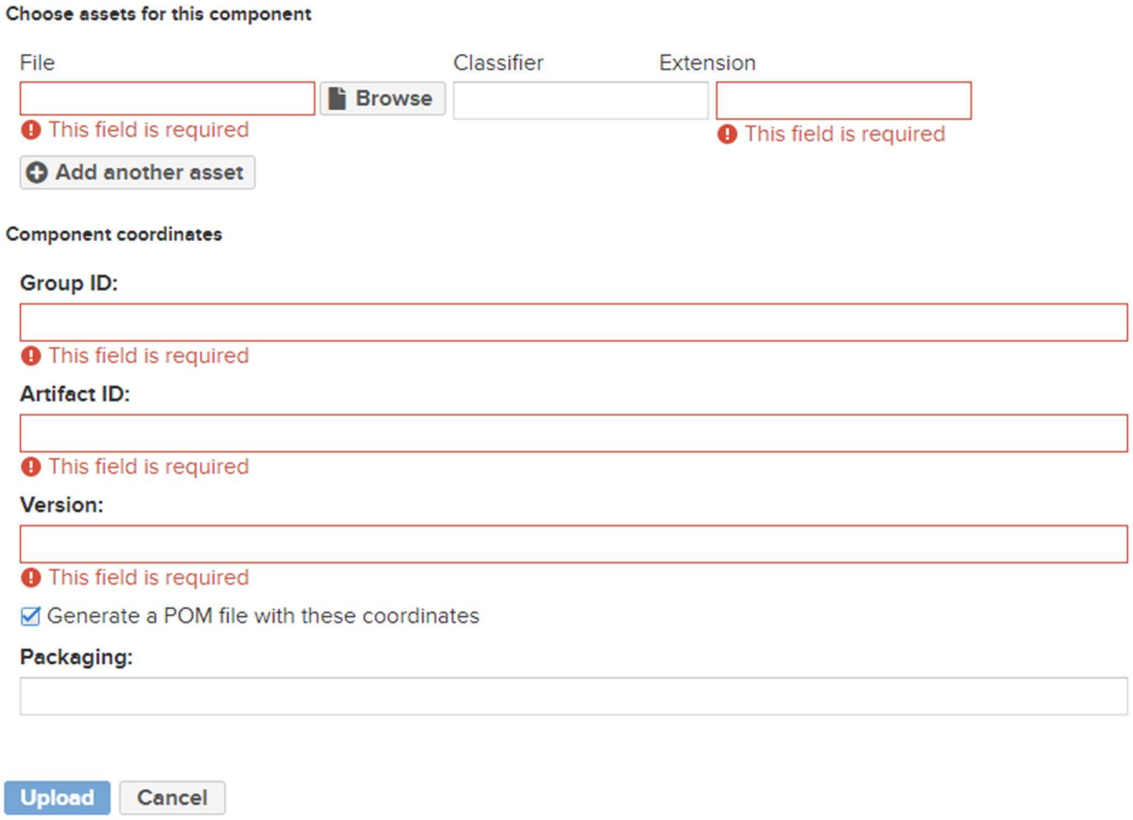
Nuestros ficheros .jar hay que subirlos a in-house-app-nc-releases. Pinchamos en la flechita de la derecha



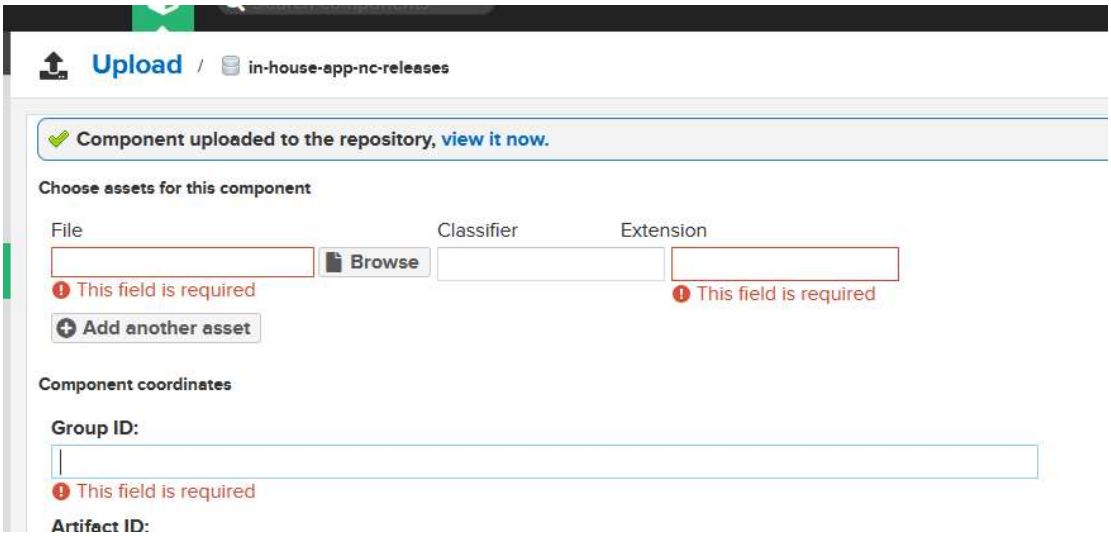
Nos lleva a la pantalla siguiente, que es donde subiremos nuestro fichero.

Por defecto se informará únicamente los campos requeridos, marcando “Generate a POM file with these coordinates” si no se dispone del mismo. Si dispones de un pom para el artefacto, debes usar “Add another asset” para subir el pom. No suele ser lo habitual.

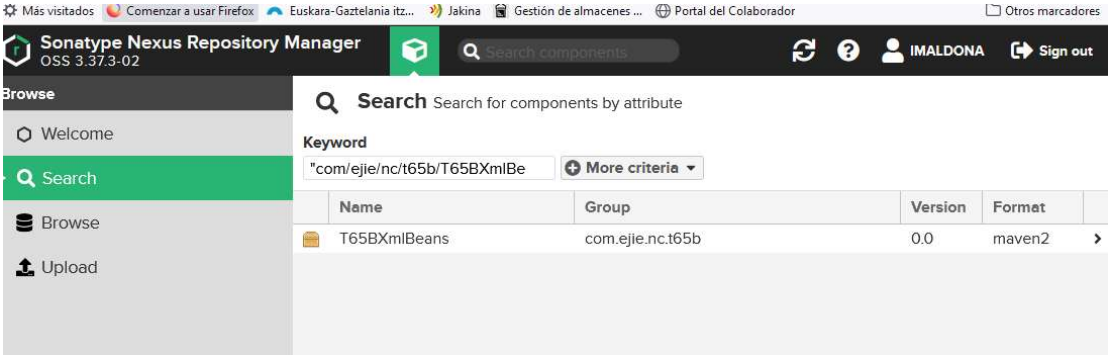
- **File**: Seleccionar el “jar” o el “pom” si lo tienes, del PC local a subir a Nexus 3.
- **Extensión**: jar,pom (depende de si subes o no un pom con ese jar)
- **Group ID**: com.ejie.nc.xxx siendo xxx el código de aplicación del jar. Es importante com.ejie.nc, si no lo pones no se subirá.
- **Artifact ID**: nombre del artefacto. Suele ser xxx*DescripciónCorta*. Ejemplo: T65*XMLBeans*
- **Version**: suele ser del estilo x.y.z (Ejemplo: 1.0.0)



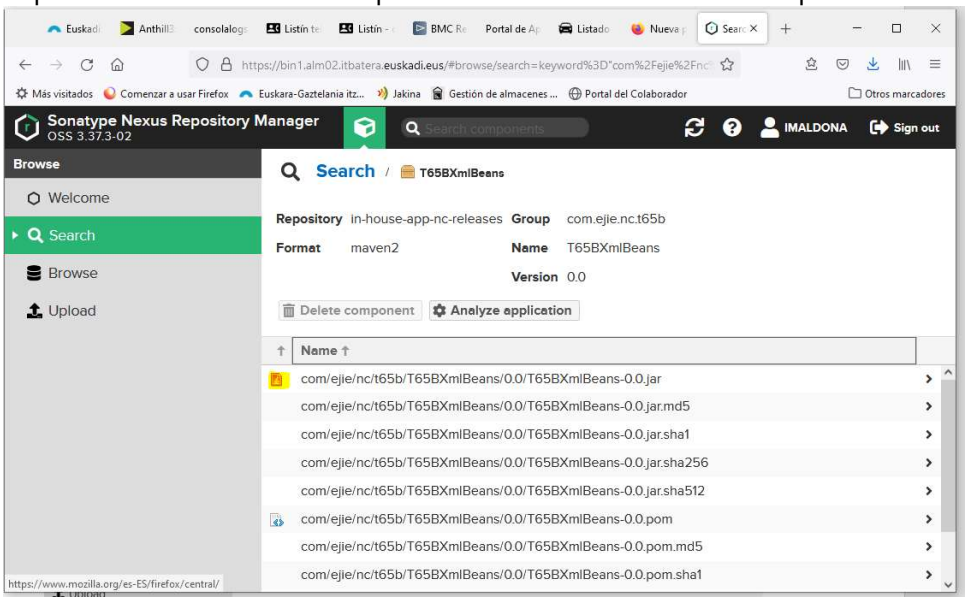
La subida la hace muy rápida y, si ha ido bien, veremos que en la parte superior de la pantalla nos la ha confirmado.



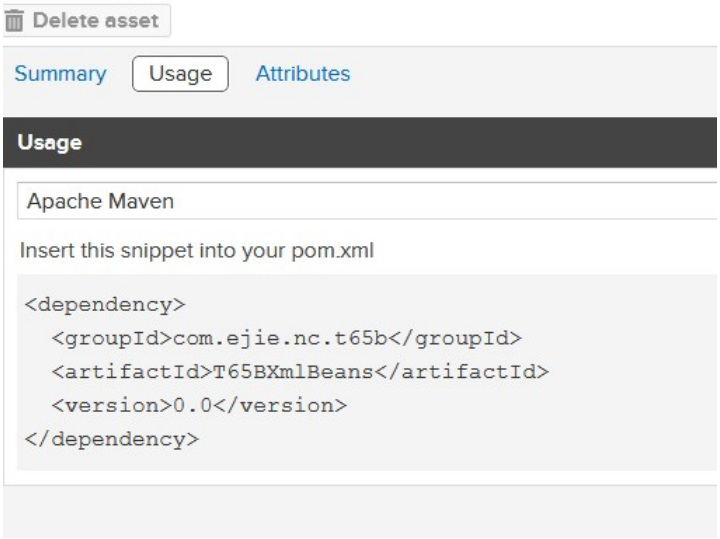
Damos “Upload”, y pinchando en “view it now”, vemos ya nuestra librería en Nexus.



Si pinchamos en el icono a la izquierda de la librería nos lleva a esta pantalla:



Pinchando en el icono a la izquierda del .jar, en la pestaña “usage”, vemos qué es lo que tenemos que poner en nuestro pom para que nos descargue esta librería desde Nexus.



Recuerda que siempre puedes usar la ayuda de Nexus.
https://help.sonatype.com/repomanager3/using-nexus-repository/uploading-components

“*Packaging*” se usaría si maven fuera la herramienta de empaquetamiento de artefactos, en base a lo que se indique en su pom con el tag: <packaging></packaging>. Actualmente lo usamos sólo para dependencias de artefactos tipo “jar”, así que pudiera ser una buena práctica a futuro, si se usara maven, indicar “jar” en la paquetería.

Los valores posibles son:

- jar (por defecto)
- war
- ear
- pom
- maven-plugin
- ejb
- rar
- …

“Classifier” permite añadir en el artefacto una clasificación. Por ejemplo, si añadimos “debug”, se añadirá después de la versión como “-debug”, y permitiría en los pom indicar ese clasificador para resolver las dependencias. Normalmente no se usan clasificadores, y por eso proponemos dejarlo en blanco.
https://maven.apache.org/plugins/maven-deploy-plugin/examples/deploying-with-classifiers.html

7. Anexo IV. Ayuda xslt para pasar de ivy a maven

La mayoría de los proyectos son consumidores de artefactos. Algunos proyectos han podido usar ivy como tecnología para resolver sus dependencias. Si estos proyectos quieren pasar a maven deberán transformar su ivy.xml a un pom.xml

Para facilitar esta acción, planteamos usar esta transformación xslt.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
File: ivy2pom.xsl
Author:     Innovación y Consultoría, Ejie.S.A
-->
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:m="http://ant.apache.org/ivy/maven">

<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

<xsl:template match="/">
<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><xsl:value-of select="ivy-
module/info/@organisation"/></groupId>
        <version><xsl:value-of select="ivy-
module/info/@revision"/></version>
        <name><xsl:value-of select="ivy-module/info/@module"/></name>
            <xsl:choose>
                  <xsl:when test="ivy-module/publications/artifact">
            <artifactId><xsl:value-of select="ivy-
module/publications/artifact/@name"/></artifactId>
        <packaging><xsl:value-of select="ivy-
module/publications/artifact/@type"/></packaging>
                  </xsl:when>
            <xsl:otherwise>
            <artifactId><xsl:value-of select="ivy-
module/info/@module"/></artifactId>
            </xsl:otherwise>
            </xsl:choose>
        <url>http://maven.apache.org</url>

            <dependencies>
            <xsl:for-each select="ivy-module/dependencies/dependency">
                  <dependency>
                        <groupId><xsl:value-of select="@org"/></groupId>
                        <artifactId><xsl:value-of
select="@name"/></artifactId>
                        <version><xsl:value-of select="@rev"/></version>
                        <!--exclusions>
                              <exclusion>
                                    <groupId>*</groupId>
                                    <artifactId>*</artifactId>
                              </exclusion>
```

```
                        </exclusions-->
                  <xsl:choose>
                        <xsl:when test="./@m:classifier">
                        <classifier><xsl:value-of select="./@m:classifier"
/></classifier>
                        </xsl:when>
                  <xsl:otherwise>
                        <xsl:choose>
                              <xsl:when test="artifact">
                              <classifier><xsl:value-of
select="./artifact/@m:classifier" /></classifier>
                              </xsl:when>
                        <!--xsl:otherwise></xsl:otherwise-->
                        </xsl:choose>
                  </xsl:otherwise>
                  </xsl:choose>
            </dependency>
            </xsl:for-each>
            </dependencies>
            <build>
             <plugins>
                        <plugin>
                              <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-dependency-plugin</artifactId>
                    <configuration>
                        <!-- en ivy no se usaba transitividad -->
                         <excludeTransitive>true</excludeTransitive>
                    </configuration>
                </plugin>
             </plugins>
        </build>
</project>
</xsl:template>

</xsl:stylesheet>
```

Tener en cuenta que el pom.xml generado con este xslt tiene configurado “excludeTransitive” a “true”, que evita que se descarguen las dependencias transitivas de los artefactos, como normalmente se usaba ivy.
Si el uso de ivy hubiera sido usando la transitividad, poner a false (valor por defecto) “excludeTransitive” y gestionar las posibles exclusiones con lo indicado en el apartado 6.4.