CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA

UD 1. Introducción

Módulo: Programación de servicios y procesos

Víctor J. Vergel Rodríguez

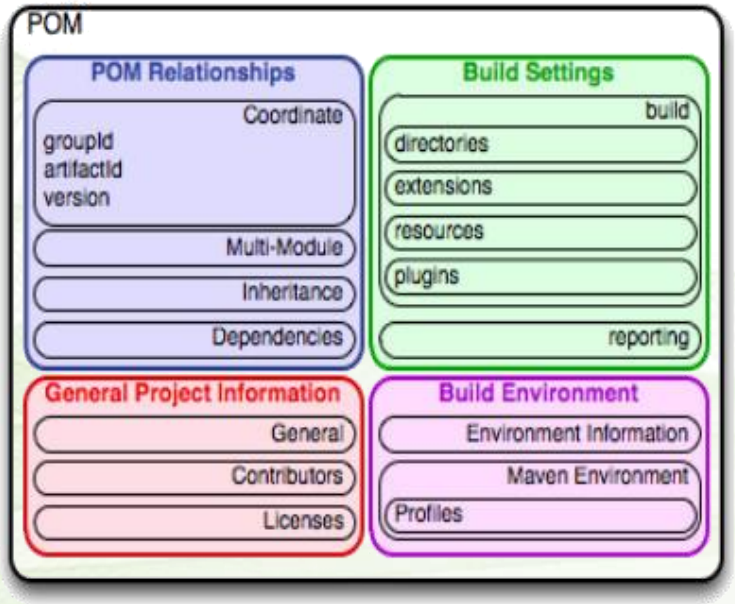
Centro de Enseñanza
Gregorio Fernández

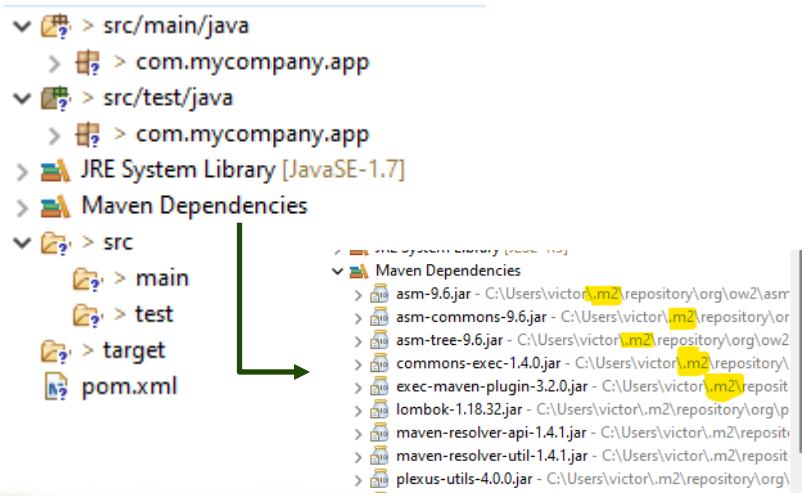
Maven

- Qué es?
  - Nos soluciona la vida en cuanto a escalabilidad, portabilidad, migración, generación de ejecutables. Podemos abrir un proyecto Maven con cualquier IDE (artefactos)
  - **MVN Repository** con arquetipos.
  - Otras herramientas gestión proyectos:
    - Apache Ant ("Another Neat Tool") y Gradle (no XML, ej: Android.)
- Conceptos:
  - **Artefacto**. Proyecto Maven en sí
  - **Arquetipo**. Especie de plantilla. Ej.: maven-archetype-quickstart ,
  - Plugin eclipse:

Maven

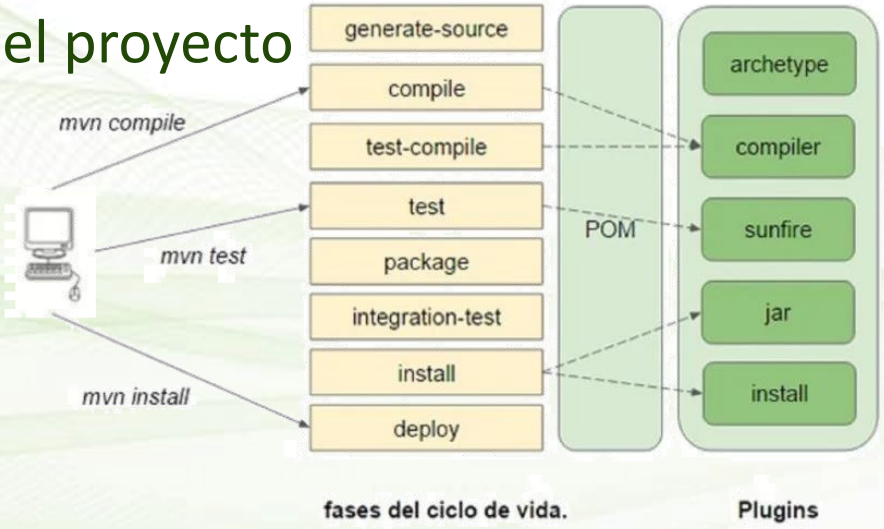
- pom.xml (Project Object Model). Link Apache
  - Identificación del mismo y características
  - properties
  - dependencies
  - profiles
  - build
    - Filters
    - Resources
    - Plugins
  - reports





Maven

- Ciclo de vida
  - **clean**. Borra el contenido del directorio target
  - **compile**. Compila los fuentes en el directorio target
  - **package**. compile y los empaqueta (jar)
  - **install**. package y además instala en el repositorio local.
  - **deploy**. install y además instala en un repositorio remoto
  - **site**. Desarrolla el site del proyecto



Maven - ejecución

- **Para ejecutar el código del proyecto Maven indicando una clase inicial:**

```
<plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>exec-maven-plugin</artifactId>
        <version>3.3.0</version>
        <configuration>
                <mainClass>paquete_inicial.Aplicacion</mainClass>
        </configuration>
</plugin>
```

Maven - compilación

- **Para ejecutar el código del proyecto Maven indicando una clase inicial:**

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.13.0</version>
        <configuration>
                <source>17</source>
                <target>17</target>
        </configuration>
</plugin>
```

Maven - jar

- **Cuando generas un JAR en Maven no incluye automáticamente dependencias. Usaremos el maven-assembly-plugin o el complemento maven-shade-plugin.**

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-assembly-plugin</artifactId>
        <version>3.3.0</version>
        <configuration>
                <descriptorRefs>  <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
                <archive>
                        <manifest>          <mainClass>ejemplo.Aplicacion</mainClass>
        </manifest>
                </archive>
        </configuration>
        <executions>
                <execution>
                        <phase>package</phase>
                        <goals>  <goal>single</goal>          </goals>
                </execution>
        </executions>
</plugin>
```

Maven - Informes

- **Generar el site:**

```
<reporting>
        <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-site-plugin</artifactId>
                <version>4.0.0-M4</version>
        </plugin>
</reporting>
```

- **Generar el javadoc:**

```
<reporting>
        <plugins>
                <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-javadoc-plugin</artifactId>
                        <version>3.6.3</version>
                </plugin>
        </plugins>
</reporting>
```

Thread

Atributos	
Métodos	
run()	Método que se ejecuta al iniciar un proceso a través del método start().
start()	Una vez invocado este método, se encargará de llamar al método run().
Thread()	Constructor de la clase
Thread(String)	Constructor de la clase que recibe el nombre del proceso como parámetro
Thread(Runnable,String)	Creamos un objeto Thread a partir de un Runnable y del nombre de proceso que queremos darle.
Thread(ThreadGroup,String)	Crea el proceso asociándolo a un grupo y le da un nombre al proceso.
String getName()	Recupera el nombre del proceso .
setName(String)	Asigna el nombre del proceso .
Static sleep(long millis)	Duerme el proceso x milisegundos. Implica obligatoriamente pérdida de CPU.
join()	Permite que un proceso continúe una vez que finalice otro proceso sobre el que se ha llamado al join.
setPriority()	De 1 a 10 asigna prioridad.
Static Thread currentThread()	Método estático que devuelve el Thread que se está ejecutando en ese momento. Por regla general, coincide con el objeto this dentro de un Thread.
ThreadGroup getThreadGroup()	Recupera el grupo de procesos al que pertenece el proceso actual que ejecuta este método concretamente.
setDaemon(boolean estado)	Indicamos al proceso que será un demonio metiendo como parámetro el valor true. Siempre se ejecuta antes del start, puesto que no se puede cambiar de estado una vez iniciado.

Thread

MIN_PRIORITY, NORM_PRIORITY, MAX_PRIORITY	**Constantes estáticas, definidas a nivel de la clase Thread, que tienen asociado un entero de 1 al 10, según la prioridad que queramos dar.**
wait() (heredado de object)	Espera un tiempo no determinado, hasta que recibe un notify y despierta de ese estado. Se ejecutan sobre un objeto sincronizado.
wait(milisegundos) (heredado de object)	Si no es depertado antes de x milisegundos .
notify()	Despierta a un proceso de un wait. Se ejecutan sobre un objeto sincronizado.
notifyAll()	Despierta a todos los procesos que estaban en un wait. Se ejecutan sobre un objeto sincronizado.
interrupt()	Método que hace que el proceso sobre el que se le aplica finalice cuando libere recursos y estime necesario. Ojo, si el interrupt llega cuando el proceso está en un sleep / wait generará un mensaje de error y no se cambia la variable isInterrupted() a true.
Boolean isInterrupted()	Devuelve true si el proceso ha sido interrumpido a través del método interrupt. Éste valor no vuelve a false una vez interrumpido.

Boolean Thread.interrupted() ()	Devuelve true solo el instante de tiempo en el que se ha interrumpido el proceso.
	//for (; i < 100000 && !Thread.interrupted(); i++) {
	for (; i < 100000; i++) {
	System.out.println(Thread.currentThread().isInterrupted()+"-"+i);
	}

Static yield()	Obliga al proceso actual a liberar la CPU (sin necesidad de perder 1 milisegundo como haría el sleep(1)). Se accede desde la clase, al ser un método estático.
setDaemon(boolean)	Hace que el proceso sobre el que se le aplica pase a ser un demonio y por tanto muera automáticamente cuando muere el padre.

ThreadGroup

Atributos	
Métodos	
getName()	Recupera el nombre del grupo de Threads. Por defecto, el grupo que se crea siempre se llama main.
int activeCount()	Recupera el número de procesos vivos dentro de ese grupo de procesos. Realmente el valor estimado de los procesos vivos.
Int enumerate(Thread[] procesos)	Rellena un array de Threads (procesos) con el conjunto de procesos que pertencen al ThreadGroup y devuelve el número de procesos introducido en el array.
ThreadGroup(String nombre)	Constructor de un grupo de procesos con el nombre que se le asigne por parámetro.
setMaxPriority(int pri)	Asigna la prioridad de un grupo, después de aplicar este método al ThreadGroup los procesos que se agrupen en él no podrán superar esa prioridad. Si asignamos a un proceso un valor superior a esta máxima prioridad de grupo, se le cambiará automáticamente a esa prioridad máxima del grupo.
interrupt	Mata, o mejor dicho cambia la bandera a interrupted=true, a todos los procesos del grupo.

ThreadGroup

Atributos	
Métodos	
getName()	Recupera el nombre del grupo de Threads. Por defecto, el grupo que se crea siempre se llama main.
int activeCount()	Recupera el número de procesos vivos dentro de ese grupo de procesos. Realmente el valor estimado de los procesos vivos.
Int enumerate(Thread[] procesos)	Rellena un array de Threads (procesos) con el conjunto de procesos que pertencen al ThreadGroup y devuelve el número de procesos introducido en el array.
ThreadGroup(String nombre)	Constructor de un grupo de procesos con el nombre que se le asigne por parámetro.
setMaxPriority(int pri)	Asigna la prioridad de un grupo, después de aplicar este método al ThreadGroup los procesos que se agrupen en él no podrán superar esa prioridad. Si asignamos a un proceso un valor superior a esta máxima prioridad de grupo, se le cambiará automáticamente a esa prioridad máxima del grupo.
interrupt	Mata, o mejor dicho cambia la bandera a interrupted=true, a todos los procesos del grupo.

Ciclo de vida de procesos