**Trabajo investigativo laboratorio 2**

**Ciclos de vida y desarrollo de software**

**Juan Daniel Murcia y Mateo Forero Fuentes**

**Maven**

Cuál es su mayor utilidad:

Maven es una herramienta de gestión de proyectos que se utiliza para simplificar y gestionar el ciclo de vida del desarrollo de software. Su principal utilidad es automatizar el proceso de construcción y gestión de dependencias, lo que facilita el desarrollo, la compilación, las pruebas y la distribución de proyectos de software.

Fases de Maven:

Maven organiza el proceso de construcción en fases específicas conocidas como "fases del ciclo de vida". Algunas de las fases más comunes incluyen:

1. validate: Validar el proyecto.
2. compile: Compilar el código fuente del proyecto.
3. test: Ejecutar pruebas unitarias.
4. package: Empaquetar el código compilado en un formato distribuible.
5. install: Instalar el paquete en el repositorio local.
6. deploy: Copiar el paquete final en el repositorio remoto.

Ciclo de vida de la construcción:

El ciclo de vida de construcción de Maven está compuesto por tres ciclos de vida principales:

default: Se encarga de la construcción del proyecto desde la compilación hasta la instalación.

clean: Elimina los archivos generados en la fase default.

site: Genera documentación y sitios web del proyecto.

Para qué sirven los plugins:

Los plugins en Maven son extensiones de funcionalidad que proporcionan tareas específicas durante el ciclo de vida del proyecto. Los plugins pueden ejecutar acciones como compilar código, ejecutar pruebas, empaquetar artefactos, desplegar en servidores, y más. Los plugins permiten a los desarrolladores extender y personalizar el comportamiento de Maven según las necesidades del proyecto.

Qué es y para qué sirve el repositorio central de Maven:

El Repositorio Central de Maven es un repositorio público gestionado por la comunidad de Maven. Es un almacén centralizado de artefactos (como bibliotecas, frameworks y plugins) que pueden ser utilizados por proyectos Maven en todo el mundo. Permite a los desarrolladores compartir y reutilizar dependencias en sus proyectos, evitando la necesidad de descargar y gestionar bibliotecas manualmente.

**Crear un proyecto con Maven:**

Para crear un proyecto Maven básico de manera genérica usando arquetipos podemos contemplar los siguientes pasos:

1. Instalación de Maven: Asegurarnos de tener Maven instalado en tu dispositivo. Podemos descargarlo desde el sitio oficial de Apache Maven y seguir las instrucciones de instalación.
2. Creación del Proyecto Maven: Utilizaremos el siguiente comando en la línea de comandos para generar un nuevo proyecto Maven:

mvn archetype:generate

1. Selección de Arquetipo: Seleccionaremos un arquetipo específico según nuestras necesidades. Los arquetipos son plantillas que definen la estructura básica del proyecto. Puedes elegir desde proyectos básicos hasta configuraciones avanzadas.
2. Proporcionar Detalles del Proyecto: Durante el proceso, nos solicitarán detalles del proyecto, como groupId, artifactId, version y package. Estos detalles son esenciales para configurar la estructura del proyecto y organizar tu código de manera coherente.

Siguiendo estos pasos, Maven generará automáticamente la estructura inicial de tu proyecto.

Si deseamos ejecutar el objetivo generate del plugin archetype desde la línea de comandos de Maven con parámetros específicos, podemos utilizar el siguiente comando:

mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.0 -DgroupId=edu.eci.cvds -DartifactId=Patterns -Dpackage=edu.eci.cvds.patterns.archetype

**Compilar y ejecutar en Maven:**

El parámetro package en el comando mvn package se refiere a una de las fases del ciclo de vida de construcción de Maven. En este contexto, el objetivo del parámetro package es empaquetar el proyecto compilado (generalmente en forma de archivo JAR) para su distribución y uso.

Cuando ejecutas mvn package, Maven realizará varias tareas, incluida la compilación del código fuente, la ejecución de pruebas unitarias y la creación de un paquete del proyecto. El tipo de paquete generado dependerá de la configuración del proyecto y del empaquetado definido en el POM (Project Object Model) del proyecto.

Además de package existen múltiples parámetros que pueden acompañar al comando mvn, algunos de ellos son:

1. mvn clean : Limpia el proyecto, eliminando archivos generados en fases anteriores de construcción.
2. mvn compile : Compila el código fuente del proyecto, verificando sintaxis e integridad del código.
3. mvn test : Ejecuta las pruebas unitarias del proyecto para verificar el cumplimiento con requisitos específicos.
4. mvn package : Empaqueta el proyecto compilado en un formato distribuible, como JAR o WAR.
5. mvn install : Instala el paquete en el repositorio local de Maven para su uso en otros proyectos.
6. mvn deploy : Copia el paquete al repositorio remoto para compartirlo con otros desarrolladores o equipos.
7. mvn site : Genera informes y documentación, creando un sitio web detallado sobre el proyecto.
8. mvn clean install : Realiza una limpieza completa del proyecto y luego instala el paquete en el repositorio local.

Para ejecutar el proyecto sin argumentos desde consola de comandos se puede hacer uso del siguiente comando:

mvn exec:java

Este comando nos ayudara a ejecutar nuestro proyecto Maven apartir de las especificaciones que le demos a nuestro archivo POM, donde es importante que en la sección build exista un plugin que referencie de manera correcta a nuestro archivo java que vamos a ejecutar, el plugin debe tener este estilo:

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.0.0</version>

<configuration>

<mainClass>edu.eci.cvds.patterns.archetype.App</mainClass>

</configuration>

</plugin>

Para enviar parámetros a la función main en la ejecución del proyecto debemos hacer uso del comando anterior de ejecución con la siguiente modificación:

mvn exec:java -Dexec.args = "arg1 arg2"

En este caso arg1 arg2 serian los argumentos pasados a nuestro main en caso de querer pasar mas argumentos se agregan de manera secuencial, el archivo App ya considera esto y a la hora de ejecutar el saludo lo realizara con los argumentos dados en caso de que no sean vacíos, en caso de serlo hace una ejecución por defecto.

**Esqueleto del proyecto:**

Para el desarrollo de este proyecto se nos pedía el uso del patrón fábrica, en este caso dada la complejidad del ejercicio desarrollado se hizo uso de simple factory, decido a que este nos cumplía con los requisitos de funcionalidad del programa y nos permitía el desarrollo completo de las necesidades que teníamos, además su implementación fue sencilla lo cual es una característica especial del simple factory con respecto a los otros patrones de fábrica.

En nuestra elección entre la Fábrica Simple y la Fábrica Abstracta para implementar el patrón de diseño de fábrica, sentimos que la mejor es la Fábrica Abstracta. Reconocemos la simplicidad y facilidad de implementación de la Fábrica Simple, pero consideramos que la Fábrica Abstracta proporciona una flexibilidad crucial para nuestro sistema. Esta elección se basa en la necesidad de manejar posibles evoluciones en la familia de productos y mantener una arquitectura adaptable. Creemos que la Fábrica Abstracta nos brindará la estructura necesaria para afrontar cambios y expansiones futuras, permitiéndonos mantener un código sólido y escalable.

¿Cuál(es) de las siguientes instrucciones se ejecutan y funcionan correctamente y por qué?

* Sin parámetros
* Parámetro: qwerty
* Parámetro: pentagon
* Parámetro: Hexagon

Después de la implementación completa del proyecta se ejecuto de manera correcta solo el ultimo comando donde efectuábamos el proyecto con parámetro Hexagon en los casos fallidos el error era adecuado y coherente.

Bibliografía:

<https://maven.apache.org/ref/3.6.1/maven-embedder/cli.html>

<https://www.campusmvp.es/recursos/post/java-que-es-maven-que-es-el-archivo-pom-xml.aspx>

<https://www.mojohaus.org/exec-maven-plugin/usage.html>

<https://refactoring.guru/design-patterns/catalog>