13_herramientas_para_gestion_de_proyectos

October 15, 2020

1 Herramientas de gestión de proyectos.

1.1 Nota preliminar.

Antes de iniciar, cabe aclarar que este capítulo no trata sobre la gestión de proyectos (PM).

Para los alcances de este curso, la gestión de proyectos se refiere a la gestión de código, binarios y paquetes a lo largo de un proyecto de desarrollo de sofware en *Java*.

1.2 Proyectos de desarrollo de software.

En la actualidad, un proyecto de desarrollo de software consiste en la colaboración de grupos de trabajo que comparten diversas piezas de código y echan mano de múltiples bibliotecas a lo largo del proceso que da por resultado un paquete que contiene a todos los archivos necesarios para la correcta ejecución de una aplicación. En el caso de *Java*, dichos paquetes pueden ser archivos de tipo *JAR*, *WAR* o *EAR*.

La estructura de un proyecto de desarrollo de software puede comprender una gran cantidad de directorios y archivos de diversas índoles. Sin embargo, dichas estructuras son susceptibles de ser estandarizadas y homologadas con la finalidad de que un equipo de trabajo pueda colaborar con relativa facilidad.

La complejidad que implica el desarrollo de un proyecto de software hace muy complicado que una persona pueda llevar el control completo de cada componente y estructura del proyecto. Es por ello por lo que se ha popularizado el uso de *frameworks* que automatizan la construcción, prueba y gestión de proyectos de desarrollo de software.

1.2.1 Herramientas para gestión de proyectos en Java.

Los herramientas más populares para la gestión de proyectos en Java son:

- *Apache Ant*
- *Apache Maven*
- *Gradle*

Cada una de estas herramientas tiene características particulares y son utilizadas en diversos proyectos, por lo que es recomendable que un desarrollador esté familiarizado con cada una de ellas.

El siguiente hipervínculo apunta a una comparación entre *Ant*, *Maven* y *Gradle* en el sitio StackShare:

https://stackshare.io/stackups/ant-vs-gradle-vs-maven

Como se puede apreciar, la herramienta más simple pero con menos funcionalidades es *Ant*, mientras que *Maven* y *Gradle* son utilizados para proyectos de mayor complejidad.

Una de las grandes ventajas de *Gradle* es que este *framework* puede integrarse con *Ant* y *Maven*.

1.3 Funcionalidades básicas.

1.4 Construcción de binarios.

La funcionalidad mínima que se puede esperar de estas herramientas es la de construcción de binarios a partir del código fuente.

1.4.1 El caso de *Ant*.

Ant es una herramienta especializada en la construcción de binarios de Java mediante acciones específicas llamadas tareas, o "*tasks*", las cuales son ejecutadas mediante un script en formato *XML* cuyo nombre por convención es *build.xml*.

1.4.2 El caso de *Maven*.

Maven es una herramienta más compleja que *Ant*, la cual en vez de definir una serie de acciones a ejecutar una después de otra, define un ciclo de vida del proyecto, el cual incluye cálaculo de dependencias, pruebas y construcción de binarios en lo que se conoce como *Project Object Model* (*POM*), los cuales se denfinen en archivos *pom.xml*. En es sentido, la creación de binarios es sólo una parte del *POM*.

1.4.3 El caso de *Gradle*.

Gradle es un framework que puede soportar diversos lenguajes además de Java como lo son Groovy, Scala, Kotlin e incluso Python y Go; así como frameworks de programación de aplicaciones como Spring. Es posible acceder a estas funcionalidades mediante el uso de plug-ins, los cuales incluyen integración con *Ant* y *Maven*.

A diferencia de *Ant* y *Maven*, *Gradle* cuenta con *Lenguajes Especializados de Dominio* (*SDL*) basados en *Kotlin* y *Groovy*, los cuales definen el proceso del ciclo de vida del proyecto por medio de un archivo llamado *build.gradle*.

Gradle puede ser extendido mediante *plug-ins*.

1.5 Gestión de dependencias y repositorios.

Es común en prácticamente cualquier lenguaje de programación que existan repositorios remotos que contengan de bibliotecas públicas a las que se pueda acceder e incluirlas como dependencias en un proyecto local.

1.5.1 *Ant + Ivy*.

Ant no cuenta por si mismo la funcionalidad de acceder a repositorios remotos. Sin embargo, de forma paralela se desarrolló el proyecto *Ivy* el cual permite gestionar repositorios remotos y locales de bibliotecas basadas en *Ant*.

1.5.2 Repositorios de *Maven*.

Maven es un proyecto que fue ideado para poder desplegar repostorios locales, centralizados y remotos para facilitar la gestión de dependencias. Los repositorios de Maven son muy populares para publicar proyectos comunitarios que cuentan con una gran cantidad de bibliotecas disponibles.

En el siguiente hipervínculo es posible consultar algunos de los repositorios de *Maven* más conocidos.

https://www.deps.co/guides/public-maven-repositories/

Maven Central. *Maven Central* es el repositorio que utiliza *Maven* por defecto.

Ejemplo:

Es necesario que cambie el kernel de la notebook a Java.

[ ]:
```
%maven org.knowm.xchart:xchart:3.5.2
import org.knowm.xchart.*;
double[] xData = new double[] {0.0, 1.0, 2.0};
double[] yData = new double[] {1.0, 5.0, 11.0};
XYChart chart =QuickChart.getChart("ejemplo", "x", "y", "f(x)", xData, yData);
BitmapEncoder.getBufferedImage(chart);
```

1.5.3 Integración de *Gradle*.

Gradle es capaz de integrar los repositorios de *Maven* y de *Ivy* de tal forma que puede acceder a una cantidad de bibliotecas sin necesidad de definir su propio esquema de repositorios.

1.6 Estructura de archivos de un proyecto.

A lo largo del tiempo, tanto *Maven* como *Gradle* han definido estructuras específicas para proyectos particulares que son creadas de forma automática dependiendo del tipo de aplicación de la que se trate. De ese modo, es posible que grandes grupos de trabajo puedan colaborar en integrar proyectos.

1.6.1 Arquetipos de *Maven*.

En el caso de *Maven* a los artefactos que definen estas estructuras se les conoce como *Arquetipos*.

https://maven.apache.org/guides/mini/guide-creating-archetypes.html

1.6.2 Plantillas de *Gradle*.

En el caso de *Gradle* el plugin de temolates permite definir la estructura de un proyecto de diversas índoles.

https://github.com/townsfolk/gradle-templates/wiki

1.7 Comparación entre un archivo *pom.xml* y un archivo *build.gradle*.

A contunuación se mostrarán los archivos *pom.xml* y un *build.gradle* para un proyecto de Spring Boot el cual se puede crear mediante el *Spring Initalizer*.

1.7.1 El archivo *pom.xml*.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>spring-boot</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>spring-boot</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
```

```
    </build>

</project>
```

1.7.2 El archivo *build.gradle*.

```
plugins {
    id 'org.springframework.boot' version '2.2.2.RELEASE'
    id 'io.spring.dependency-management' version '1.0.8.RELEASE'
    id 'java'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation('org.springframework.boot:spring-boot-starter-test') {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
    }
}

test {
    useJUnitPlatform()
}
```