Herramientas de Software para la Gestión y Construcción de Proyectos.

Guillermo Daniel Cruz Ortega
guillermo.daniel.cruz.ortega@gmail.com
Universidad de la Sierra Sur.

2022/04/09

1 Introducción.

Las herramientas de gestión y construcción de proyectos son una forma de nueva de trabjar en equipos grandes y dispersos, más que nada en estos tiempos modernos, el trabajar a distancias lejanas del demas equipo de trabajo es una realidad y mas en el área de la tecnologia, donde se puede trabjar con gente que esta al otro lado del globo terraqueo. Por eso el uso de estas herrameintas es un parte aguas que permite una mejor coordinación entre equipo y homogeneidad al momento de escoger las herramientas del proyecto.

2 *Maven.*

"*Maven* es una herramienta de software para la gestión y construcción de proyectos *Java* creada por *Jason van Zyl*, de *Sonatype*, en 2002. Es similar en funcionalidad a *Apache Ant* (y en menor medida a *PEAR* de *PHP* y *CPAN* de *Perl*), pero tiene un modelo de configuración de construcción más simple, basado en un formato XML. Estuvo integrado inicialmente dentro del proyecto Jakarta pero ahora ya es un proyecto de nivel superior de la Apache Software Foundation. Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores."
"Las partes del ciclo de vida principal del proyecto Maven son:

- *compile*: Genera los ficheros .class compilando los fuentes .java
- *test*: Ejecuta los test automáticos de JUnit existentes, abortando el proceso si alguno de ellos falla. package: Genera el fichero .jar con los .class compilados
- *install*: Copia el fichero .jar a un directorio de nuestro ordenador donde maven deja todos los .jar. De esta forma esos .jar pueden utilizarse en otros proyectos maven en el mismo ordenador.
- *deploy*: Copia el fichero .jar a un servidor remoto, poniéndolo disponible para cualquier proyecto maven con acceso a ese servidor remoto. (*Wikipedia Maven*, 2021)

Los proyectos en Maven son creados con una línea de comandos del siguiente tipo:

```
mvn archetype:generate -DgroupId="com.some.company" -DartifactId="some-project-Dversion="0.0.1
```

En la versión 1, basándose en un fichero de configuración en XML (project.xml) y una serie de extensiones (plugins), esta herramienta puede compilar el proyecto Java, ejecutar las pruebas unitarias, generar paquetes (jars, wars, ears o distribuciones en zip) y generar una serie de informes. La versión 2 usa también un fichero de configuración en XML llamado pom.xml (Project Object Model). Su funcionalidad es parecida a Apache Ant, de manera que permite compilar, ejecutar pruebas o realizar distribuciones, pero con la diferencia de que trata de forma automática las dependencias del proyecto que gestiona." En resumen, *Maven* es una gran

herramienta que permite una mejor gestión de proyectos y integracion de estos en equipos, para sacar el maximo provecho de esta herramienta en un inicio es necesario tener conexiónn en internet para descargar las herramientas necesarias. De ahi ya solo es conocer estas herramientas y saber ocuparlas, pero permiten mucha flexibilidad y facilidad para agregara los proyectos.

3 *Gradle.*

"*Gradle* es un sistema de automatización de construcción de código de software que construye sobre los conceptos de *Apache Ant* y *Apache Maven* e introduce un lenguaje específico del dominio (DSL) basado en *Groovy* en vez de la forma XML utilizada por *Apache Maven* para declarar la configuración de proyecto. *Gradle* utiliza un grafo acíclico dirigido (*"DAG"*) para determinar el orden en el que las tareas pueden ser ejecutadas. *Gradle* fue diseñado para construcciones multi-proyecto las cuales pueden crecer para s er bastante grandes, y da apoyo a construcciones incrementales determinando inteligentemente qué partes del árbol de construcción están actualizadas, de modo que cualquier tarea dependiente a aquellas partes no necesitarán ser reejecutada. Los *plugins* iniciales están principalmente centrados en el desarrollo y despliegue en *Java, Groovy y Scala*, pero existen más lenguajes y *workflows* de proyecto en el *roadmap*." (*Wikipedia Gradle*, 2022) Ejemplo en codigo Java: "Considera el caso donde la estructura de directorios Maven es usada para las fuentes utilizada para los recursos y el código fuente Java. Estos directorios son: src/main/java, src/main/resources, src/test/java y src/test/resources. build.gradle

```
apply plugin: 'java'
```

Ejecutar gradle build dará el siguiente resultado:

```
> gradle build
:compileJava
:processResources
:classes
:jar
:assemble
:compileTestJava
:processTestResources
:testClasses
:test
:check
:build

BUILD SUCCESSFUL
```

El plugin de Java emula muchos de los ciclos de vida Maven esperados como tareas en el grafo acíclico dirigido de dependencias para las entradas y salidas de cada tarea. Para este caso sencillo, la tarea de build depende de las salidas de las tareas check y assemble. Así mismo, check depende de test, y assemble depende de jar. Para proyectos que no siguen las convenciones Maven, Gradle permite que la estructura de directorios sea configurada. El ejemplo siguiente podría ser usado en un proyecto que contiene código fuente en src/java en lugar de src/main/java (que es la convención dictada por Maven).

```
build.gradle
apply plugin: 'java'
sourceSets.main.java.srcDirs = ['src/java']
```

4 *Apache Ant.*

"*Apache Ant* es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción (*build*). Es, por tanto, un software para procesos

de automatización de compilación, similar a *Make* pero desarrollado en lenguaje *Java* y requiere la plataforma *Java*, así que es más apropiado para la construcción de proyectos *Java*. Esta herramienta, hecha en el lenguaje de programación *Java*, tiene la ventaja de no depender de las órdenes del *shell* de cada sistema operativo, sino que se basa en archivos de configuración *XML* y clases *Java* para la realización de las distintas tareas, siendo idónea como solución multi-plataforma. Para utilizar *ANT* basta con disponer de una distribución binaria de *ANT* y tener instalado la versión 1.4 o superior del JDK. La distribución binaria consiste en la siguiente estructura de directorios: La carpeta "*ant*" contiene las carpetas "*bin*" (que asimismo contiene scripts de lanzador), "*lib*" (que contiene las dependencias necesarias y los *.JAR* de *Ant*), "*docs*", (que contiene la documentación de *Ant*, incluyendo una descripción, imágenes y un manual), y "*etc*" (que contiene valiosos archivos *.XSL* para crear informe de mejora de la salida *XML* de varias tareas, migrar los archivos de creación y deshacerse de la "obsoleta" alerta, y más). Pero solo se necesitan los directorios *bin* y *lib* para ejecutar *ANT*. Las limitaciones de ANT se pueden decir que son:

- Al ser una herramienta basada en *XML*, los archivos *Ant* deben ser escritos en *XML*. Esto es no sólo una barrera para los nuevos usuarios, sino también un problema en los proyectos muy grandes, cuando se construyen archivos muy grandes y complejos. Esto quizá sea un problema común a todos los lenguajes *XML*, pero la granularidad de las tareas de *Ant* (comparado con Maven, por decir alguno), significa que los problemas de escalabilidad llegan pronto.

- La mayoría de las antiguas herramientas — las que se usan todos los días, como *¡javac¿, ¡exec¿ y ¡java¿* — tienen malas configuraciones por defecto, valores para opciones que no son coherentes con las tareas más recientes. Esta es la maldición de la compatibilidad hacia atrás: cambiar estos valores supone estropear las herramientas existentes.

- Cuando se expanden las propiedades en una cadena o un elemento de texto, las propiedades no definidas no son planteadas como error, sino que se dejan como una referencia sin expandir. De nuevo, esta es una cuestión de la compatibilidad hacia atrás, incluso se reconoce que tener la herramienta desactivada es normalmente la mejor opción, al menos hasta el punto que el mítico producto "*Ant2.0*" falle en propiedades no asignadas.

- No es un lenguaje para un flujo de trabajo general, y no debería ser usado como tal. En particular, tiene reglas de manejo de errores limitadas, y no tiene persistencia de estado, así que no puede ser usado con confianza para manejar una construcción de varios días." (*Wikipedia Apache Ant*, 2021)

5 *Ivy.*

"*Ivy* es el conducto de renderizado y compilación de la próxima generación. Es muy avanzado y ofrece funciones avanzadas que antes no estaban disponibles. La velocidad que proporciona es increíble. La carga es muy rápida incluso en las redes que son lentas. Es muy simple de usar sin ninguna complicación. El tamaño del paquete también se reduce con su ayuda. Ivy es una reescritura completa del motor de renderizado de Angular. De hecho, es la cuarta reescritura del motor y la tercera desde *Angular* 2. Pero a diferencia de las reescrituras dos y tres, Ivy promete grandes mejoras para su aplicación. Con *Ivy*, puede compilar componentes de manera más independiente entre sí. Esto mejora los tiempos de desarrollo ya que recompilar una aplicación solo implicará compilar los componentes que cambiaron. La localidad y el movimiento de los árboles son dos aspectos clave que *Ivy* siempre considera. Ambos pueden hacer que *Ivy* sea capaz de lo que puede hacer. El proceso de compilación independiente de cada componente con su información. Los cambios parciales se compilan en el proceso que hace que el proceso sea más rápido al no cambiar todos los archivos del proyecto." (*Angular Minds*, 2021)

6 Conclusiones.

En el campo laboral hay varias herramientas que ayudan a los equipo de programadroes a organizarse en sus codigos y que todos tengas las mismas herramientas disponibles, en su mayoria las herramientas que se analizan en este trabajo tiene mayor integración con *JAVA*, pero igual hay varias herramientas para diferentes programas. La gran ventaja de estas heramientas es la homogeneidad que tendran los trabajos y la compatibilidad entre haces, haciendoq eu la comunicacion con respositorios como git, sean mas eficaces.

Es de suma importancia entender como funcionana estas herramientas, y aunque estas herramientas en su mayoria son similares, hayq eu investigar bien su funcionamiento como ventajas y desventajas que estas den a tu trabajo.

7 Referencias.

Maven. (2021, 14 de septiembre). Wikipedia, La enciclopedia libre. Fecha de consulta: 20:29, abril 14, 2022 desde https://es.wikipedia.org/w/index.php?title=Mavenoldid=138342974.
Gradle. (2022, 5 de abril). Wikipedia, La enciclopedia libre. Fecha de consulta: 21:13, abril 14, 2022 desde https://es.wikipedia.org/w/index.php?title=Gradleoldid=142734806.
Apache Ant. (2021, 30 de noviembre). Wikipedia, La enciclopedia libre. Fecha de consulta: 21:36, abril 14, 2022 desde https://es.wikipedia.org/w/index.php?title=ApacheAntoldid=140065758.
Khirale, A. (2021, 17 febrero). All About Angular Engine Ivy In 5 Mins. Angular Minds. Recuperado 14 de abril de 2022, de https://www.angularminds.com/blog/article/what-is-angular-ivy.html