Conceptos básicos Git, GitHub, Gradle y Maven

Autores:
Carlos Enrique Morales Rodríguez
Id: 1077420979
Alcides Antonio Zapata Méndez
Id: 8055453
Leonel Alfonso Zapata Méndez
Id: 1038100562
Dilsa Doriela Barrientos Ciro
Id: 1037237118

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas

Curso:
Arquitectura de Software

Docente:
Robinson Coronado García

Septiembre de 2020

Conceptos básicos de Git, GitHub, Gradle y Maven

Introducción

El mundo de la programación se encuentra en constante evolución, de tal manera que exige que los actores involucrados en este mundo, nos actualicemos permanentemente para ser mejores programadores o administradores y como usuarios sacarle el máximo provecho, además, los builds de los proyectos también requieren actualizarse constantemente; es aquí donde herramientas de software como Gradle y Maven a igual que los software de control como Git, Github toman gran importancia brindando una alta gama de posibilidades que facilitan el trabajo en el desarrollo y la programación.

Este informe sirve como introducción al conocimiento de algunos tipos de herramientas de software en forma general, haciendo énfasis en las herramientas Git, Github, Gradle y Maven que dinamizan y optimizan el trabajo de los desarrolladores, programadores y administradores en la actualidad, brindando un conocimiento somero del funcionamiento y las características de estos.

Además, se profundiza en uno de los sistemas de control de versiones existentes en la actualidad que se ha popularizado tremendamente, gracias al sitio Github. Se trata de Git, el sistema de control de versiones más conocido y usado actualmente, gracias al motor de Github. Al terminar su lectura entenderás qué es Git y qué es Github, dos términos distintos, cuya diferencia en ocasiones resulta confusa de entender por las personas que están dando sus primeros pasos en el mundo del desarrollo. Los sistemas de control de versiones son programas que tienen como objetivo controlar los cambios en el desarrollo de cualquier tipo de software,

permitiendo conocer el estado actual de un proyecto, los cambios que se le han realizado a cualquiera de sus componentes, las personas que intervinieron en ellos, etc.

Objetivo General

Conocer las distintas herramientas de software y sistemas de control más utilizados en la actualidad y sus características.

Objetivos específicos

- Comprender que es y para qué sirve los sistemas de control de software Git y Github.
- Conocer la importancia de las herramientas de software Gradle y Maven.
- Identificar las ventajas que tienen las herramientas de software Gradle y Maven en la actualidad.

Herramientas de desarrollo

¿Qué es GIT?

Git es un software de control de versiones diseñado por Linux Torvalds, el autor de Linux. El control de versiones es la gestión de los cambios que se realizan sobre los elementos de un proyecto.

Git es multiplataforma, se pueden crear repositorios locales en todos los sistemas operativos más comunes, Windows, Linux o Mac. Existen algunos GUIs (Graphical User Interface o Interfaz de Usuario Gráfica) para trabajar con Git, no obstante para el aprendizaje se recomienda usarlo con línea de comandos, de modo que se pueda dominar el sistema desde su base.

Una versión o revisión es el estado completo en el que se encuentra el proyecto en un instante dado. A diferencia de un editor, cuando se da la orden de guardar (denominada commit), no se guarda un fichero, se guarda todo el estado del proyecto. Para algunas personas, Git es algo parecido a Dropbox o Google Drive pero con un grado de control sobre las versiones mucho más fino.

Comandos más importantes en GIT

git init (Inicializa un repositorio desde cero)
git clone URL (Hace copia del repositorio en la maquina local)
git add [archivo] (Agrega el archivo al Stage)
git commit -m "[mensaje]" (Realiza commit al archivo y envia un mensaje para saber que se hizo
git log (Muestra el historial de todos los commit que hemos realizado)
git branch [nombre] (Crea una nueva rama)
git checkout [branch] (Nos permite pasearnos entre las ramas)
git push origin [branch] (Envía los cambios al repositorio)
git pull (Actualiza el repositorio)

¿Qué es GitHub?

GitHub es una plataforma de desarrollo colaborativo que es utilizada para alojar proyectos utilizando el sistema de control de versiones Git. GitHub ofrece una infinidad de servicios para el trabajo colaborativo como: repositorios, documentación, gestión de incidencias, gestión de equipos, chats, notificaciones, páginas web, tableros, asignación de tareas, etc. que pueden ser utilizados para el desarrollo de la actividad académica.

Por tanto, Git es algo más general que nos sirve para controlar el estado de un desarrollo a lo largo del tiempo, mientras que Github es algo más particular: un sitio web que usa Git para ofrecer a la comunidad de desarrolladores repositorios de software. En definitiva, Github es un sitio web pensado para hacer posible el compartir el código de una manera más fácil.

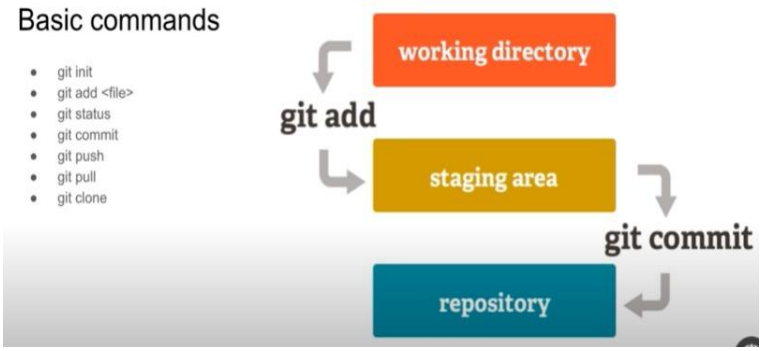

Figura 1. Basic commands Git. Tomada de la web

¿Qué es Gradle?

Gradle, es una herramienta que permite la automatización de compilación de código abierto, la cual se encuentra centrada en la flexibilidad y el rendimiento. Los scripts de compilación de Gradle se escriben utilizando Groovy o Kotlin DSL (Domain Specific Language).

Gradle tiene una gran flexibilidad y nos deja hacer usos de otros lenguajes y no solo de Java, también cuenta con un sistema de gestión de dependencias muy estable. Gradle es altamente personalizable y rápido ya que completa las tareas de forma rápida y precisa, reutilizando las salidas de las ejecuciones anteriores, sólo procesar las entradas que presentan cambios en paralelo.

Características de Gradle

Algunas características de Gradle que podemos destacar son las siguientes:

Depuración colaborativa: Permite compartir los resultados de la compilación para resolver en equipo de forma eficiente posibles problemas que aparezcan.

Construcción incremental: Valida en el proceso de compilación si la entrada, salida o implementación de una tarea ha cambiado, en caso de no existir algún cambio la considera actualizada y no se ejecuta.

Diseño de repositorio personalizado: Podremos tratar prácticamente cualquier estructura de directorios del sistema de archivos como un repositorio de Artifacts.

Dependencias transitivas: Es uno de los principales beneficios que ofrece al utilizar la gestión de dependencias ya que se encarga de descargar y administrar las dependencias transitivas.

Soporte a Groovy y Scala incorporado: Compatibilidad con los proyectos de Groovy, permitiendo trabajar con código Groovy o código Scala e inclusive desarrollar código mixto Java y Groovy o Java y Scala.

Compilación incremental para Java: En caso de que el código fuente o la ruta de clase cambien, Gradle cuenta con la capacidad para detectar todas las clases que se vean afectadas por dicho cambio y procederá a recompilarlas.

Embalaje y distribución de JAR, WAR y EAR: Cuenta con herramientas para empaquetar el código basado en JVM (Java Virtual Machine) en archivos de archivo comunes.

Integración con Android Studio: Android Studio no cuenta con un generador interno, sino que delega todas las tareas de compilación en Gradle, garantizando la corrección en todas las construcciones, ya sea que se ejecuten desde Android Studio, la línea de comandos o un servidor de construcción de integración continua.

Soporte de MS Visual C ++ y GoogleTest: Gradle acepta la construcción con el compilador de Visual C de Microsoft en Windows. (VS 2010, VS 2013 y VS 2015 compatibles), así como también realizar pruebas de aplicaciones C con GoogleTest.

Publicar en repositorios Ivy y Maven: Permite publicar Artifacts en repositorios Ivy con diseños de directorios completamente personalizables. De igual modo, sucede con Maven en Bintray o Maven Central.

TestKit para pruebas funcionales: Permite la ejecución prágramática de builds inspeccionando los resultados de compilación, ésta es una prueba de compatibilidad entre versiones.

Distribuciones personalizadas: En Gradle cada distribución cuenta con un directorio init.d en el que se pueden colocar scripts personalizados que pre configuran su entorno de compilación.

Lee el formato POM: Es compatible con el formato de metadatos POM, por lo que es posible recuperar dependencias de cualquier repositorio compatible con Maven.

Compara builds: Resalta de forma rápida las diferencias entre compilaciones, lo que hace que el análisis de la causa raíz sea mucho más rápido y eficaz.

Compilador daemon: Gradle crea un proceso de daemon que se reutiliza dentro de una compilación de múltiples proyectos, cuando necesita bifurcar el proceso de compilación, mejorando la velocidad de compilación.

Personalizar y extender escaneos: Ofrece la opción de agregar sus propios datos para construir escaneos como etiquetas, valores y enlaces, integrando escaneos de compilación en la cadena de herramientas.

Caché de dependencia de terceros: Las dependencias de repositorios remotos se descargan y almacenan en caché localmente, las compilaciones posteriores utilizan los artifacts almacenados en caché para evitar el tráfico de red innecesario.

Figura 2. Gradle. Tomada de la web

Complementos Gradle

Gradle permite construir desde micro servicios hasta aplicaciones móviles, puede ser utilizado por pequeñas startups como por grandes empresas, ya que ayuda a los equipos a desarrollar, automatizar y entregar software de calidad en un menor tiempo (dependiendo de su complejidad y planificación previa del proceso de desarrollo).

Esta herramienta cuenta con una gran variedad de complementos o plugins que ayudan agilizar la construcción, entre los cuales destacamos los siguientes:

Javamuc.gradle-semantic-build-versioning: Proporciona soporte para vel ersionado semántico de las compilaciones, es fácil de usar y configurar.

Io.freefair.maven-publish-war: Permite crear una publicación de mavenWeb.

Io.freefair.maven-publish-java: Crea una publicación mavenJava.

Org.mozilla.rust-android-gradle.rust-android: Un complemento que ayuda a construir bibliotecas Rust JNI con Cargo para su uso en proyectos de Android.

Net.wooga.build-unity: Este complemento proporciona tareas para exportar proyectos de plataforma desde los proyectos de Unity3D.

de.db.vz.msintplugin: Este complemento nos permitirá ejecutar pruebas de integración de microservicio con docker.

Com.bmuschko.docker-remote-api: Nos facilita la gestión de imágenes y contenedores Docker.

com.google.cloud.tools.jib: Crea un contenedor para tu aplicación Java.

Gradle cuenta con paquetes para ser implementado en cualquier plataforma su gran versatilidad permite trabajar con monorepos o multirepositorios, modelando, sistematizando y construyendo soluciones exitosas, de forma rápida y precisa.

¿Qué es Maven?

Maven es, de facto, la herramienta Java para construir y gestionar proyectos de desarrollo. En su día ocupó el lugar de Apache Ant, debido sobre todo a su modelo "convention over configuration". La gran diferencia que aportaba sobre aquella es la facilidad para configurar un proyecto basándonos en estándares, que si no queríamos sobrescribir facilitaba enormemente el proceso de puesta en marcha inicial. Actualmente, Gradle va ganando popularidad, y aunque personalmente le veo potencial, su flexibilidad (que remite a Ant) puede llegar a ser un hándicap a la hora de comprender una configuración determinada. Maven sigue siendo, por tanto, un must si desarrollas con Java.

Maven se utiliza en la gestión y construcción de software. Posee la capacidad de realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. Es decir, hace posible la creación de software con dependencias incluidas dentro de la estructura del JAR. Es necesario definir todas las dependencias del proyecto (librerías externas utilizadas) en un fichero propio de todo proyecto Maven, el POM (Project Object Model). Este es un archivo en formato XML que contiene todo lo necesario para que a la hora de generar el fichero ejecutable de nuestra aplicación este contenga todo lo que necesita para su ejecución en su interior.

Conclusiones

La existencia de herramientas como Git que permiten el trabajo colaborativo entre grupos de personas o equipos, que enfrentan la limitante de no poder compartir espacios comunes. Abre la posibilidad a nuevos usuarios que incluso desde sus casas pueden aportar al desarrollo de nuevas tecnologías o proyectos que de otra manera no tendrían ninguna posibilidad.

En la actualidad proliferan herramientas tecnológicas que posibilitan el desarrollo de infinidad de softwares que atienden las necesidades de un mundo que crece exponencialmente, esto se potencia con la colaboración de diferentes saberes que se conjugan gracias a la existencia de plataformas como Git, Gradle y Maven.

GitHub ofrece múltiples servicios para el trabajo colaborativo no sólo de repositorios, sino también de documentación, gestión de incidencias, gestión de equipos, chats, notificaciones, páginas web, tableros, asignación de tareas, etc. que pueden ser utilizados para el desarrollo de la actividad académica.

- Podemos decir que Git es algo más general que nos sirve para controlar el estado de un desarrollo a lo largo del tiempo, mientras que Github es algo más particular: un sitio web que usa Git para ofrecer a la comunidad de desarrolladores repositorios de software.

Referencias

Normas *APA. (s.f.) Recuperado de https://normasapa.in/*

Robledano, A. (14 de Mayo de 2019). *openwebinars: Gradle vs Maven. Recuperado de https://openwebinars.net/blog/gradle-vs-maven/*

Del Hoyo, C. (31 Oct 2018). *autentia: Por qué trabajar con Maven y Gradle. Recuperado de https://www.autentia.com/2018/10/31/por-que-trabajar-con-maven-y-gradle/*

Kinsta: *¿Qué es GitHub? Una Guía para Principiantes sobre GitHub (23 marzo 2020). Recuperado de https://kinsta.com/es/base-de-conocimiento/que-es-github/*

Gutiérrez, I. [Bluuweb !]. (2019 jun. 26) *GIT / GITHUB [Tutorial en Español - Parte 1]* ♥ *Inicio Rápido para Principiantes [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=hWglK8nWh60&t=385s*