29_introduccion_a_gradle

May 15, 2020

1 Introducción a `Gradle`.

`Gradle` es una herramienta que permite la automatización de construcción de binarios y elaboración de pruebas. Aún cuando el código de *Gradle* está escrito en *Java* (primordialmente), *Groovy* y *Kotlin* (a partir de *Gradle 5.0*), es una herramienta que soporta diversos lenguajes frameworks y plataformas.

La documentación de Gradle está disponible en:

https://docs.gradle.org/current/userguide/userguide.html

https://docs.gradle.org/current/userguide/userguide.pdf

1.1 `Gradle 6.x`.

Gradle es una herramienta que fue creada en 2007 y actualmente se encuentra el la *versión 6.4*.

Aún cuando existen diferencias que hacen incompatible la sintaxis de versiones previas de *Gradle*, es posible migrar el código de forma automatizada desde *Gradle 5.x*.

1.2 Requisitos de instalación.

Gradle 6.x funciona en prácticamente cualquier plataforma que tenga una *JVM* superior a la version 8.

Las instrucciones de instalación están disponibles en:

https://docs.gradle.org/current/userguide/installation.html

La instalación de *Gradle* ya incluye una versión de *Kotlin*, *Groovy* y *Apache Ant*.

1.3 Configuración en el entorno de trabajo.

Para fines de este curso, el servidor `o-i.mx` tiene instalada una instancia de *Gradle* en el directorio `/opt/gradle/gradle-6.4/`.

Para poder ejecutar los comandos de gradle es necesario añadir esta ruta a la variable de entorno `PATH` mediante la siguiente instrucción:

[ ]:
```
export PATH=$PATH:/opt/gradle/gradle-6.4/bin
```

[ ]:
```
echo $PATH
```

En caso de que se desee que esta configuración sea permanente se puede ejecutar la siguiente instrucción:

Advertencia: Modificar de forma incorrecta al archivo `~/.bashrc` puede ser riesgoso.

[ ]:
```
echo "export PATH=$PATH:/opt/gradle/gradle-6.4/bin" >> ~/.bashrc
```

Para validar que la configuración es correcta puede ejecutar:

[ ]:
```
gradle -v
```

1.4 Funcionalidades y características.

https://gradle.org/features

- Ejecución de corridas (*Gradle builds*).
  - Optimización de desempeño.
    - Corridas incrementales.
    - Corridas en cache.
    - Subtareas incrementales.
    - Procesamiento incremental de anotaciones.
    - Daemon de compliación.
    - Ejecución en paralelo.
    - Descarga de dependenciasd en paralelo.
    - Restriccion de tiempo por tarea.
  - Análisis de la corrida (*build scans*).
    - Visualización basada en web.
    - Depuración colaborativa.
    - Extensible y ajustable.
  - Opciones de ejecución.
    - Corridas continuas.
    - Corridas compuestas.
    - Exclusión de tareas.
    - Corridas "en seco" (*dry run*).
    - Ejecución continua después de fallos.
    - Ejecución de pruebas de fallo rápidas.
    - Sincronización del cache de depandencias con repositorios.
- Creación de corridas.
  - La lógica es código susceptible de ser probado.
    - *DSL* con *Groovy* y *Kotlin*.
    - El plugin de inicialización de *Gradle* (*Gradle Init Plugin*).
  - Gestión de dependencias.
    - Dependencias transitivas.
    - Ámbitos de dependencias ajustables.
    - Dependencias basadas en archivos.
    - Plantillas ajustables de repositorios.
    - Compatibilidad con repositorios de *Ivy* y *Maven*.
    - Soporte nativo de *BOM*.
    - Dependencias dinámicas.

        * Bloqueo de dependencias.
        * Reglas de selección de dependencias dinámicas.
        * Resolución de conflictos de versiones.
        * Sustitución de bibliotecas compatibles.
        * Soporte mejorado de resolución de metadatos.
        * Sustitución de dependencias externas y de proyecto.
    - Estandarización entre gurpos de trabajo.
        * Entorno de corrida autogenerado.
        * Configuración de los entornos de corrida cotrolados por versiones.
        * Distirbuciones a la medida.
    - *Software Domain Modeling*.
        * Contenedores de objetos de dominio.
        * Publicaciónd de múltiples artefactos.
        * Orden de tareas avanzado.
        * Inferencia de las dependencias de una tarea.
        * Finalizadores de tareas.
        * Creación dinámica de tareas.
        * Sensor de eventos a detalle.
        * Inyección de comportamientos basados en usuarios.
        * Inyección de comportamiento por corrida.
- Funcionalidades específicas de ecosistema.
    - Aplicaciones para la *JVM*.
        * Compilaciones incrementales de *Java*.
        * Soporte de plugins para herramientas de calidad de código de Java (*JaCoCo*).
        * Empaquetado y distribución de *JARs*, *WARs* y *EARs*.
        * Publicación en repositorios de *Maven* e *Ivy*.
        * Integración con *Ant*.

1.5 El comando `gradle`.

La ejecución de los proyectos y tareas de *Gradle* se realiza medainte el comando:

`gradle <tarea 1> <tarea 2> ... <tarea n> <opción 1> <opción 2> ... <opción n>`

Donde:

- `<tarea i>` corresponde a una tarea definida.
- `<opción i>` corresponde a una opción de ejecución de `gradle`.

Ejemplos:

- El siguiente comando desplegará la ayuda de `gradle`.

[ ]:
```
gradle --help
```

- El siguiente comando levantará un *daemon* desplegará las tareas de construcción de binarios por defecto.

[ ]:
```
gradle tasks
```

1.6 Los daemon de *Gradle*.

Gradle es capaz de levantar *daemons*, los cuales realizarán la construcción de binarios de un proyecto en particular.

Es posible tener en ejecución varios daemons de forma paralela.

La documentación de los daemons de *Gradel* puede ser consultada desde:

https://docs.gradle.org/current/userguide/gradle_daemon.html

Ejemplos:

- El siguiente comando deplegará el estado de todos los daemos activos de un usuario.

[ ]:
```
gradle --status
```

- El siguiente comando detendrá al *daemon* que ya se encuentra en ejecución.

[ ]:
```
gradle --stop
```

[ ]:
```
gradle --status
```

1.7 El archivo `gradle.build`.

El archivo `gradle.build` es un script escrito en *Groovy* o *Kotiln*, el cual contiene el código de construcción de uno o varios proyectos y tareas. Es el documento básico para la ejecución de las diversas funcionalidades de *Gradle*.

1.8 El *wrapper* de *Gradle*.

El empaquetador (*wrapper*) es un script que permite descargar es instalar gradle y otros componentes, con la finalidad de facilitar la costrucciónd e binarios desde un repositorio.

El script que ejecuta estas tareas es `gradlew`.

La documentación del *wrapper* puede ser consultada desde:

https://docs.gradle.org/current/userguide/gradle_wrapper.html

1.9 Configuración de los entornos.

Gradle cuenta con diversos recursos para configurar la construcción de binarios.

La documentación correspondiente puede ser consultada desde:

https://docs.gradle.org/current/userguide/build_environment.html

1.10 Componentes.

Todo el sistema de automatización y construcción de binarios de *Gradle* está conformado por:

- Proyectos.
- Tareas.
- Plugins.

1.10.1 Los proyectos de *Gradle*.

Un proyecto es un conjunto de tareas que da por resultado uno o varios "artefactos".

1.10.2 Las tareas (tasks) de *Gradle*.

Una tarea es una pieza de código que realiza una acción específica en función de ciertas condiciones.

1.10.3 Los *plugins* de *Gradle*.

Un *plugin* es un script que extiende las funcionalidades de *Gradle*.

La referencia de *plugins* puede ser consultada desde:

https://docs.gradle.org/current/userguide/plugins.html