Ruta gradle

previo a la instalación se recomienda tener instalado el jdk

Instalar jdk 11

1. Instalar gradle 8.6: descargar, descomprimir y configurar gradle se recomienda la version 8.6

https://gradle.org/next-steps/?version=8.6&format=bin

otras versiones: https://gradle.org/releases/

instrucciones copiadas desde el sitio: https://docs.gradle.org/current/userguide/installation.html

Step 1 - Download the latest Gradle distribution

The distribution ZIP file comes in two flavors:

Binary-only (bin)

Complete (all) with docs and sources

We recommend downloading the bin file.

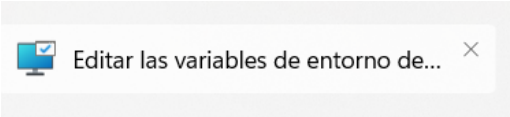
Step 2 - Unpack the distribution

Create a new directory C:\Gradle with File Explorer.

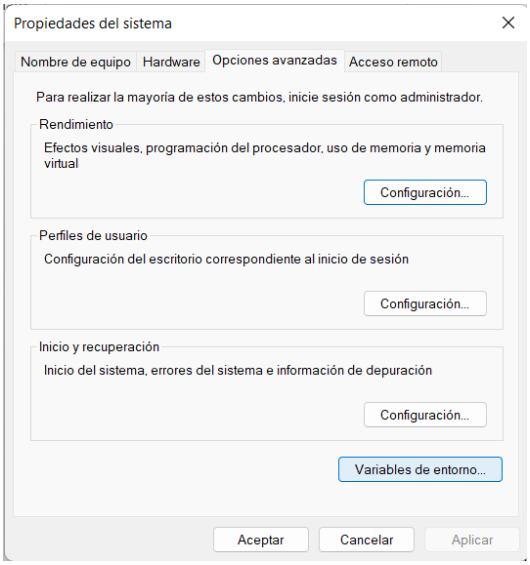
unpack the gradle distribution ZIP into C:\Gradle

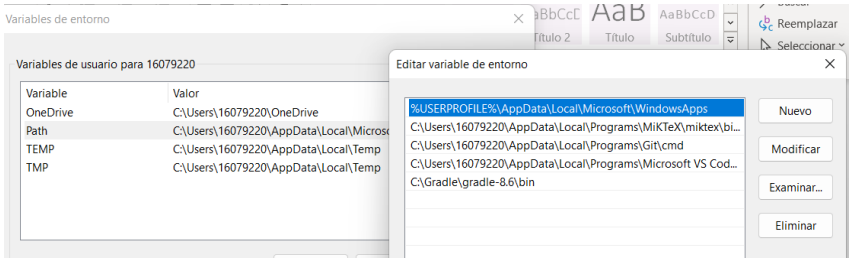
Step 3 - Configure your system environment: To install Gradle, the path to the unpacked files needs to be in your Path.

Search: variables de entorno y sistema



select: variables de entorno (System Variables): select Path, then click Edit.





Add an entry for **C:\Gradle\gradle-8.7\bin**. Click OK to save.

2. Verificar instalación

Abrir la consola de windows y ejecutar

gradle -v

debe mostrar algo como lo siguiente to run gradle and display the version, e.g.:

Gradle 8.6

Build time: 2024-02-02 16:47:16 UTC

Revision: d55c486870a0dc6f6278f53d21381396d0741c6e

Kotlin: 1.9.20

Groovy: 3.0.17

Ant: Apache Ant(TM) version 1.10.13 compiled on January 4 2023

JVM: 21.0.1 (Oracle Corporation 21.0.1+12-LTS-29)

OS: Windows 11 10.0 amd64

3. Probar el funcionamiento en un proyecto

1. crear una carpeta para su proyecto

2. ejecutar el comando gradle init y seleccionar las opciones 1, 2, las siguientes puede dejar las opciones por defecto

3. esto va a hacer que se genere una configuración inicial

el archivo build.gradle es un archivo que vamos a editar en la medida que avanzamos

4. ./gradlew help en linux o

.\gradlew help en windows (en powershell)

gradlew help en cmd

le permite ver los principales comandos a ejecutar

5. gradlew tasks permite ver las tareas disponibles

6. hay una serie de archivos entre ellos .gitignore que permite definir los archivos que

iran a control de version

7. Inicializar un repositorio usando git, agregar todos los archivos y aplicar el primer commit

```
git init
git add .
git status
git commit -m "Proyecto iniciado"
```

8. editar el archivo build.gradle y poner lo siguiente

```
plugins {
   id 'java'
}
```

9. los proyectos en java tienen una serie de requerimientos. Tales requerimientos se pueden satisfacer usando el plugin de java. En particular gradle espera que los proyectos tengan la siguiente estructura:

- src/main/java para clases
- src/main/resources para recursos
- src/test/java para clases de prueba
- src/test/resources para recursos de prueba

10. Crear las siguientes carpetas: src/main/java

crear un paquete con el nombre que usted prefiera: paquete1/myapp

mkdir -p src/main/java/paquete1/myapp

11. Agregar el archivo SayHello.java que debe contener el siguiente codigo

```
package paquete1.myapp;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.nio.charset.StandardCharsets;
public class SayHello {
    public static void main(String[] args) throws IOException {
            String language = args[0];

            InputStream resourceStream =

SayHello.class.getClassLoader().getResourceAsStream(language + ".txt");
            assert resourceStream != null;
            BufferedReader bufferedInputStream
                        = new BufferedReader(new

InputStreamReader(resourceStream,StandardCharsets.UTF_8));
            System.out.println(bufferedInputStream.readLine());

            }
}
```

12. agregar los recursos. crear un nuevo directorio en src/main llamado resources:

mkdir src/main/resources

en ese directorio poner los siguientes archivos:

en.txt que contiene la palabra HELLO!!!

es.txt que contiene la palabra HOLA!!!

deben estar ubicados aqui:

src/main/resources/en.txt

src/main/resources/es.txt

13. ver que tareas tenemos disponibles en este proyecto con el comando

.\gradlew tasks

gradlew tasks (cmd)

14. compilar las clases de java: .\gradlew compileJava

Esta acción va a generar la carpeta build y en la ruta: build/classes/java/main/paquete1/myapp

debe aparecer SayHello.class, lo que indica que la clase fue correctamente compilada

también puede ejecutar: .\gradlew processResources para procesar los recursos:

se genera la carpeta build/resources/main/

.\gradlew jar para generar el archivo jar que debe aparecer en build/libs

sin embargo este archivo no esta asociado con el nombre de la clase

15. Ejecutar la aplicacion:

java -jar build/libs/ex0.jar en

lo que nos genera el siguiente error en linux:

no main manifest attribute, in build/libs/ex0.jar

para corregir el problema, debemos agregar lo siguiente en el archivo build.gradle

```
tasks.named('jar', Jar) {
    manifest {
        attributes('Main-Class': 'paquete1.myapp.SayHello')
    }
}
```

crear el archivo jar de nuevo

.\gradlew jar

ejecutar de nuevo

java -jar build/libs/ex0.jar en

16. Probar el aplicativo (testing)

Crear el directorio test en src y agregar el directorio java:

mkdir src/test/java

dentro de este directorio crear la misma estructura de paquetes que creo en su proyecto:

paquete1/myapp

mkdir -p src/test/java/paquete1/myapp

agregar una clase para las pruebas: SayHelloTest.java

```
package paquete1.myapp;
import org.junit.jupiter.api.Test;
import paquete1.myapp.SayHello;
import java.io.IOException;
public class SayHelloTest {
    @Test
    public void testSayHello() throws IOException {

        SayHello.main(new String[]{"en"});
        }
}
```

ejecutar: .\gradlew test

esto genera un error. para corregir se debe agregar las dependencias necesarias en el archivo build.gradle

```
repositories {
       mavenCentral()
    }
   dependencies {
        testImplementation 'org.junit.jupiter:junit-
jupiter:5.10.0'
        }
```

ejecutar el test de nuevo: .\gradlew test

abrir el siguiente archivo: build/reports/tests/test/index.html

que muestra que no se ha corrido ningún test

para corregir ese problema agregar lo siguiente en el archivo build.gradle

```
tasks.named('test', Test) {
    useJUnitPlatform()
}
```

ejecutar el test de nuevo: .\gradlew test y actualizar el navegador

17. actualizar el repositorio

```
git add .
git status
git commit -m "Proyecto en java completo"
```

18. otras tareas que pueden resultar utiles

.\gradlew build assemble and check get run first

.\gradlew clean deletes the build directory

important tasks

• build is used when you just need to build and test the whole project.

• assemble and jar you can think of as equivalent, and are used when you only need to assemble the project without testing it.

• check and test you can think of as equivalent, and are used to test your code without assembling it