1. Prerequisite : java installed

* java –version

1. Download the bin for gradle to your local machine.
2. Set the environment variable

gradle/gradle-6.1/bin

1. Check if you have gradle installed

* gradle –v

1. <https://guides.gradle.org/creating-new-gradle-builds/>

* Gradle init --- will start the gradle daemon.
* And will ask you what kind of project you have and will create the directory structure accordingly.

1. The structure will be like this.

The src folder will also be created which will have structure according to the application you want to create.

* ├── build.gradle
* ├── gradle
* │   └── wrapper
* │   ├── gradle-wrapper.jar
* │   └── gradle-wrapper.properties
* ├── gradlew
* ├── gradlew.bat
* └── settings.gradle

Src – main and test

\src\main\java\gradle\javaapp\App.java file

\src\test\java\gradle\javaapp\AppTest.java file

You can run these files java gradle.javaapp.App

[Create a task](https://guides.gradle.org/creating-new-gradle-builds/#create_a_task)

1. Create a folder by any name . Inside that folder gradle init .
2. Create a directory called src.
3. Add a file called myfile.txt in the src directory. The contents are arbitrary (it can even be empty), but for convenience add the single line Hello, World! to it.
4. In your build.gradle file, define a task called copy of type Copy (note the capital letter) in your build file that copies the src directory to a new directory called dest. (You don’t have to create the dest directory — the task will do it for you.)

task copy(type: Copy, group: "Custom", description: "Copies sources to the dest directory") {

from "src"

into "dest"

}

## gradle tasks

## 1.this command returns the default tasks in gradle.

## 2.If we add any custom task in build.gradle then we see that task also here.

## 3.If we add any plugin in build.gradle , they also may have some predefined tasks within them.

## => gradle copy

## Run the specified task in your build gradle.

## NOW WE ARE GOING TO BUILD A JAVA PROJECT USING GRADLE

## <https://spring.io/guides/gs/gradle/>

## Add a plugin that enables basic Java build functionality.

apply plugin: 'java’

## This plugin will add many of its own tasks for us.

## => gradle build

To see the results of the build effort, take a look in the build folder. Therein you’ll find several directories, including these three notable folders:

* classes. The project’s compiled .class files.
* reports. Reports produced by the build (such as test reports).
* libs. Assembled project libraries (usually JAR and/or WAR files).

## Declare dependencies

## Most applications, however, depend on external libraries to handle common and/or complex functionality. For example, suppose that in addition to saying "Hello World!", you want the application to print the current date and time. You could use the date and time facilities in the native Java libraries, but you can make things more interesting by using the Joda Time libraries.

## HelloWorld(your class) uses Joda Time’s LocalTime class to get and print the current time.

## you need to add a source for 3rd party libraries.

## 1.

repositories {

mavenCentral()

}

## The repositories block indicates that the build should resolve its dependencies from the Maven Central repository. Gradle leans heavily on many conventions and facilities established by the Maven build tool, including the option of using Maven Central as a source of library dependencies.

## 2.

sourceCompatibility = 1.8

targetCompatibility = 1.8

dependencies {

compile "joda-time:joda-time:2.2"

testCompile "junit:junit:4.12"

}

## dependencies block, you declare a single dependency for Joda Time. Specifically, you’re asking for (reading right to left) version 2.2 of the joda-time library, in the joda-time group.

## Another thing to note about this dependency is that it is a compile dependency, indicating that it should be available during compile-time

* providedCompile. Required dependencies for compiling the project code, but that will be provided at runtime by a container running the code (for example, the Java Servlet API).
* testCompile. Dependencies used for compiling and running tests, but not required for building or running the project’s runtime code.

3.

jar {

baseName = 'gs-gradle'

version = '0.1.0'

}

## The jar block specifies how the JAR file will be named. In this case, it will render gs-gradle-0.1.0.jar.