**Build Systems:**

1.)Compile the code->Run test(Unit & Integration Test)->Package (Jar & War)->Deploy & Run

All these tasks can be automated by gradle(Build Automation Tool)

2.)Gradle uses convention over configuration

3.)Gradle makes building & running applications very easy

**Gradle:**

It uses Groovy/Kotlin languages inside the build.gradle files

build.gradle

->written in Groovy / Kotlin language

->equivalent of Maven's pom.xml

**Java plugin: compiling classes**

Adds a task compileJava **: ./gradlew compileJava**

(.java into .class files -> build directory )

**Java plugin: managing resources**

Adds a task processResources : **./gradlew processResources**

(copies contents of resource directory into build directory )

**Java plugin: package into jar file**

Adds a task jar **: ./gradlew jar**

(adds compiled classes & resources to jar archive )

(jar file named <project-name>-<version>.jar )

**Java plugin: easily runs tests**

Adds a task test **: ./gradlew test**

(compiles tests, processes resources, runs test )

(creates a test report )

**Java plugin: define dependencies**

Dependencies configuration is used to generate classpath

dependencies{

Implementation ‘ ’

testImplementation ‘ ‘

}

**GRADLE Java Project Layout:**

src/main/java = java plugins have classes

src/main/resoucres = have resources

src/test/java = have test classes

src/test/resources = have test resources

**SUMMARY:**

* The Java plugin initializes a project as a Java project
* It adds tasks for compileJava, processResources, jar and test
* Dependencies are either implementation or testImplementation
* Always use standard Java project layout

**Task Graph essentials:**

* Tasks can depend on other tasks
* Build aggregates assemble and check
* You only need remember to run a single task

**XML File Format( Extensible Markup Language )**

used to store and exchange data between different applications and platforms.

**Build Automation System:**

1. software tool - automates the process of compiling, testing,
2. packaging software code into a deployable format
3. so that developers can focus on writing code instead of manually managing the build process.
4. Eg of build automation tools Apache Maven, Apache Ant, Gradle, and Make

**Software project management system :**

A software project management system helps manage and organize the various activities and tasks involved in developing a software project

**GRADLE:**

**->** build automation system

->Groovy

->does not use an XML file – for declaring projects

1. Create a directory : mkdir demo

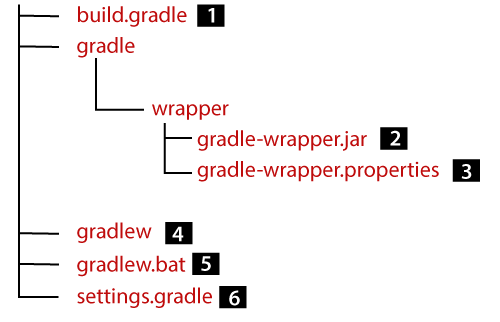
2. Change the directory : cd demo

3. Initialze a gradle project : gradle init

4. Type of project(basic, application, library, gradle plugins)

5. Select the DSL domain specific language (Groovy , Kotlin)

6. Give the project name -> Build will be successful



1. The **gradle** file is build script for configuring the current project.
2. An **executable JAR** file is used as a Gradle wrapper.
3. **Configuration properties** for Gradle Wrapper.
4. The **gradlew** is a Gradle wrapper script for UNIX based OS.
5. The **bat** is the Gradle Wrapper script for Windows.
6. **The settings script** for configuring the Gradle build.

**build.gradle**(main build configuration file).It consists of plugins, metadata,dependencies,repositories

The **build.gradle** file is typically located in the root directory of the project and is used by Gradle to configure and execute the build process.

1) Plugin:

Plugins {

id ‘java’ // depends on the specification that we choose (java/ Groovy / Kotlin)

id ‘application’

}

-> war , jar , jacoco, bootrun(springboot)

->additional build functionality

2) Metadata/ Task:

-> build metadata gives information about your build

-> compiling source code, running tests, or creating an executable JAR file.

3)Repositories:

->place where to find dependencies, libraries for a project

-> popular repositories jCenter() , mavenCentral() & mavenLocal()

4)Dependencies:

->dependencies or artifacts that are required to compile & run a project

-> Dependencies can be specified using various notations, such as **group:name:version** or **name:version**. Or **group:artifacts:version**

**-> dependencies{**

**implementation ‘group:artifacts:version’**  // this dependencies is used by the

application

**testImplementation ‘group:name:version’ //** this test framework is used

**}**

**gradle directory** (configuration for gradle wrapper)

Wrapper

* gradle-wrapper .jar(eventhough you have not installed gradle in your machine you can execute/build gradle projects)
* gradle-wrapper . properties

**gradle wrapper script**

In windows:->gradlew.bat build

**settings.gradle**

name of the project

->task represent the unit of work-> and are linked through the task group

->you can run the task using gradle wrapper script

* gradle tasks –all (shows all tasks)
* gradle compileJava (used to compile the code in project)
* gradle clean (deletes the build folder)
* gradle test (executes unit test)
* gradle jar (used to create a jar file under gradle libs folder)
* gradle run

BUILD GRADLE:

1. Initialization – gradle will identify whether the project is single or multiple projects and it will create objects for projects
2. Configuration – tasks are performed in an acyclic way
3. Execution – compilation , testing is done , jar/war files are built

**MAVEN:**

**->** software project management system

-> Java

-> use an XML file – for declaring projects