Module1: Introduction What is Gradle? • Gradle has a bunch of building conventions that we can follow when we do the build and this allows other developers who are also using Gradle to understand our builds. However, these conventions are easily overwritable and we will see some of this tool in the class. • Gradle is written in Java, but the build language is a DSL, a Domain Specific Language that's written in Groovy and this language makes it very easy to configure our builds. • It supports dependencies, so just like Maven supports dependencies, Gradle also support dependencies, and it supports both Ivy and Maven dependencies. • We also have out-of-the-box support from multi-project builds, so if you have an application that requires many different projects to build, Gradle supports that very easily. • Gradle also allows us to build things other than Java projects, so if you want to manage your JavaScript projects or your C++ projects, Gradle will let you do that as well. • Gradle is also highly customizable. • Gradle has this declarative build language, and the declarative build language expresses the intent of the build. • Gradle also makes it higly maintainable, because they are readable and understandable. Gradle compared to Ant and Maven • Which stand for Another Neat Tool. Probably the first Java build tool written when people got very frustrated using with Make, realized that Make was failing us and decided to write a build tool. • At the time XML was used for everything, so he used XML to create the tool, and the tool he came up with was Ant. • So Ant provides us with an XML build script that's hard to read and difficcult to maintain. • One of the biggest issues with Ant is the lack of conventions, so we have to specify everything from where our source directory lives to where the classes will be built, where we are going to get libraries from, to where we are going to output anything that we create here. • Ant files can get very large, very quickly and become very unmaintainable. • Maven has several advantages over Ant. Firstly, it has many, many conventions so we dont have to specify where we are going to get our sources from and where to build output to. Maven already understands that. Maven also supports dependency management. • Maven is also highly extensible, so we can configure Maven plugins, unlike Ant, where everything has to be defined by us. • However, like Ant, Maven is written in XML, and like Ant, it is hard to read and can be unmaintainable. • The DSL is a gradle one of the biggest strength. Installation of Gradle • From the website http://gradle.org Using gvm Groovy Environment Manager http://gvmtool.net/ Command: brew install gradle gradle -v **Running Gradle for the First Time** Gradle has a build file • typically build.gradle: The build file contains tasks, it can also contain plugins, it can also contain dependencies, but mostly we have tasks in the building file. So let's look at a very simple build file [bash-3.2\$ pwd /usr/local/Cellar/gradle/3.5 command vim build.gradle Total time: 1.041 secs bash-3.2\$ cat build.gradle task hello { doLast { println "Hello, Gradle" \$ gradle hello bash-3.2\$ gradle hello :hello Hello, Gradle BUILD SUCCESSFUL Total time: 1.041 secs • https://spring.io/guides/gs/gradle/ —> Please follow the instruction and then find the class path and execute. • java -cp build/classes/main/ hello.HelloWorld [Memans-MacBook-Pro:gradle aamir2292\$ java -cp build/classes/main/ hello.HelloWorld Hello, world! **Introduction to Gradle Wrapper** • vim gradle [Memans-MacBook-Pro:gradle aamir2292\$ cat build.gradle apply plugin: 'java' task wrapper(type: Wrapper) { gradleVersion = '3.5' • gradle wrapper ./gradlew build **Module2: Basic Gradle Tasks** • Introduction Objectives: Defining and Using Tasks Task Domain Specific Language(DSL) Task lifecycle Gradle properties Gradle DSL is written in Groovy. http://www.pluralsight.com/courses/groovy-fundamentals Writing Simple Tasks • What is a Task? Code that Gradle executes Has a lifecycle: intialization phase, execution phase and configuration phase. Has properties Has 'actions' Has dependencies • Gradle uses Groovy and Groovy is a JVM language, and Groovy, like other JVM languages, is object-oriented so it has the content of objects. In a Gradle build script, the top level of object is called a project and we use the project object to define everything within that build script. • So to create a task within my build script, I can say project.task, and I give this task a name. • Gradle knows that project is a top-level object and it delegates everything to that object. We can create the task by 4 ways: project.task("Task1") Memans-MacBook-Pro:demos aamir2292\$ gradle tasks --all All tasks runnable from root project |Memans-MacBook-Pro:demos aamir2292\$ cat build.gradle |project.task("Task1") task("Task2") task "Task3" Task4.description =" Task 4 Descriptiion" gradle tasks --all. Running Tasks Memans-MacBook-Pro:demos aamir2292\$ cat build.gradle project.task("Task1") task("Task2") task "Task3" task Task4 Task4.description =" Task 4 Description" Task4.doLast {println "This is Task 4"} Task3 << {println "This is task 3"}</pre> task Task5 << {println "This is task5"}</pre> [Task5.doLast {println "Another closure"} task Task6 { description "This is task 6" doLast { println "This is task 6" Task Phases Build Phases: Initialization Phase : Use to configure multi project builds Configuration Phase : • The Configuration Phase, and the Configuration Phase executes any code that's in the task that's not the task action. So if I set the description, for example, that's executed as part of the configuration Phase. Memans-MacBook-Pro:demos aamir2292\$ cat build.gradle project.task("Task1") task("Task2") task "Task3" task Task4 Task4.description = " Task 4 Descriptiion" Task4.doLast {println "This is Task 4"} Task3 << {println "This is task 3"} task Task5 << {println "This is task5"}</pre> Task5.doLast {println "Another closure"} task Task6 { description "This is task 6" doFirst { println "this is Task-6 first" doLast { println "This is task 6" Task6.doFirst { println " Another closure for Task 6 " [Memans-MacBook-Pro:demos aamir2292\$ gradle Task6 The Task.leftShift(Closure) method has been deprecated and is scheduled to be removed in Gradle 5.0. Please use Task.doLast(Action) instead. at build_d1n7lmkgr8ewmx1iw0dhq2bt3.run(/Users/aamir2292/Desktop/gradle/demos/build.gradle:13) :Task6 Another closure for Task 6 this is Task-6 first This is task 6 Task Dependencies task taskA << {println "This is TaskA"} task taskB << {println "This is TaskB"} task taskC << {println "This is TaskC"}</pre> task taskD << {println "This is TaskD"}</pre> task taskE << {println "This is TaskE"}</pre> taskA.dependsOn taskB taskA.dependsOn taskC,taskD taskC.dependsOn taskE taskD.dependsOn taskE taskA.dependsOn taskE task taskG { dependsOn taskE doLast { println "TaskG" }}

taskA.dependsOn taskG

Assignment done and submission on a git-repo.