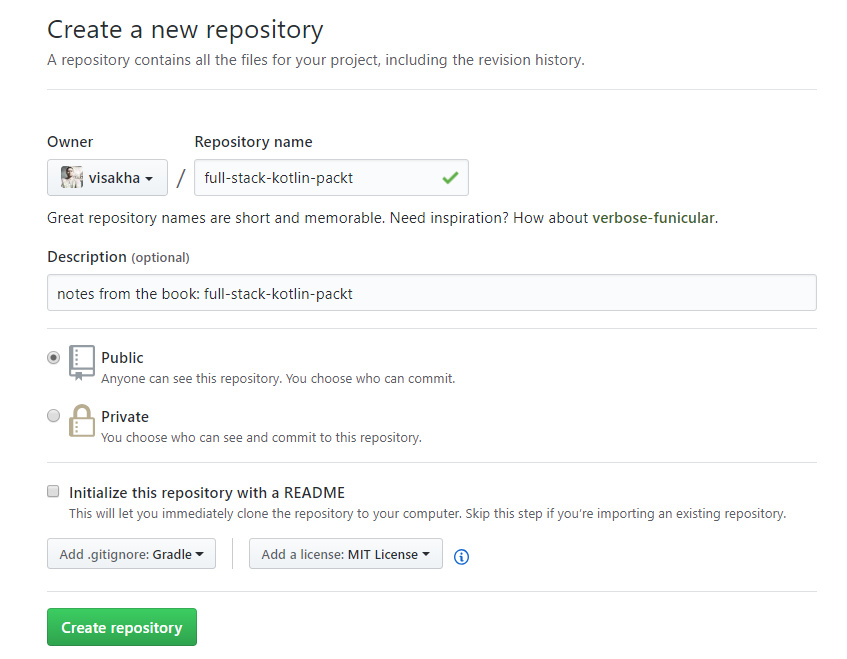
Notes: Full Stack Kotlin Development : from the Book in Packt Pub

ini: July 3rd , doc created

upd:

Steps

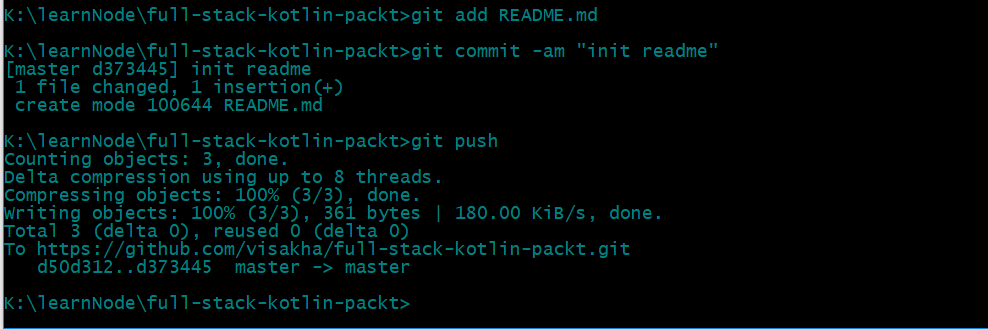
* Create a project in github



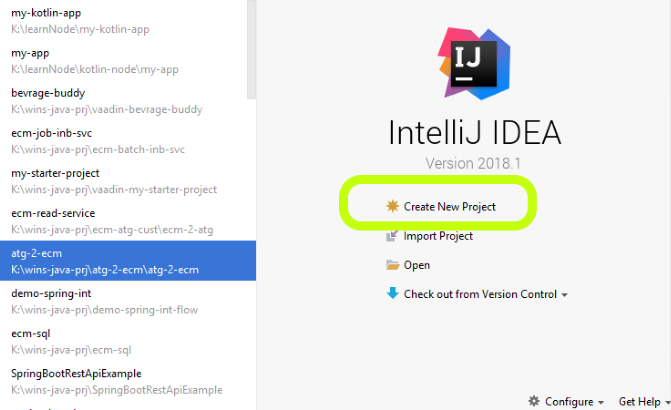
* Clone it



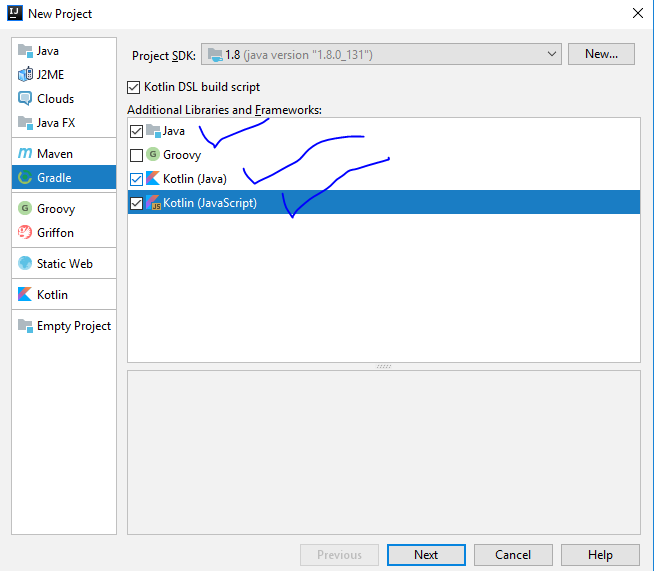
* add a readme to check that git can push to remote master



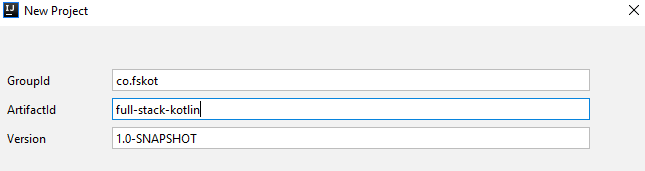
* open idea and create a new project



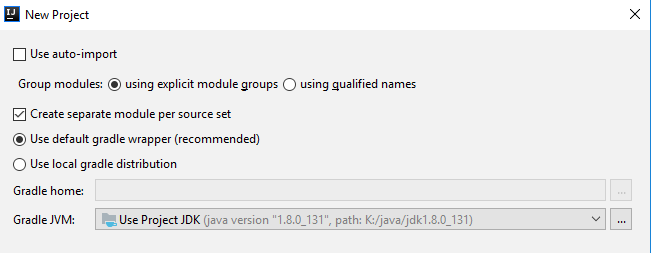
===next===



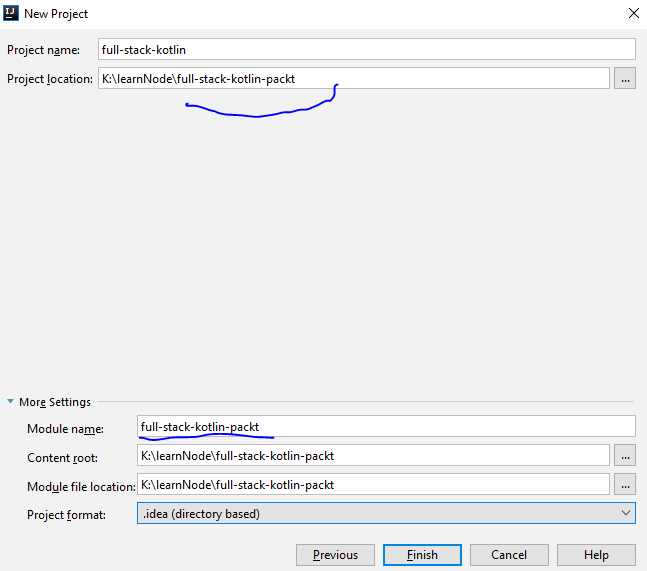
===next===



===next===



===next===



===next===

The project will try to initialize and will fail to build the project, I do not know why

therefore I copied the gradle wrapper from books source code on git hub

do the following

* clear the contents of the build.gradle.kts and instead use

|  |
| --- |
| import org.gradle.kotlin.dsl.dependencies import org.gradle.kotlin.dsl.repositories  allprojects {  group = "com.fstackkotlin"  version = "1.0"  repositories {  jcenter()  } }  plugins {  base }  dependencies {  subprojects.forEach {  archives(it)  } }   * copy: <https://github.com/Xantier/fullstack-kotlin/blob/master/gradle/wrapper/gradle-wrapper.jar> |
| same way – copy the gradlew and gradle files |
| now run: |
|  |
|  |
|  |

* Create a Module: backend

|  |
| --- |
|  |
|  |
|  |
| ERRORS:  you will see errors as below    So just delete the contents as use  <https://raw.githubusercontent.com/Xantier/fullstack-kotlin/Section-1-Video-1/backend/build.gradle.kts>  build it    See the fat jar in the `backend` dir    run it    it is failing on `no entry point` because we have not written any code yet  Do the following  under the root folder create a dir  src/main/kotlin  and restart intellij  you should now see    and the project structure should look like    at this point check the Kotlin version    change it to the latest stable    do the same for the backend\_test also  now create a package and write a simple hello world    Re-run the build and run it |

This is a logica stop for part 1

here is the zip file that can used as a starter project



====================================================================================================================================

\*\*\* Create a New GIT branch \*\*\*

====================================================================================================================================

Since this is a logical point lets create a GIT branch

Do the following

* at git hub create a new branch called `part-02-apply-spring`
* in the intelliJ refresh the git

|  |
| --- |
|  |
|  |
|  |

* change the read me and test it

Now lets incorporate Spring Boot into the BackEnd module

Edit the file: backend/build.gradle.kts as below

[

|  |
| --- |
| import org.gradle.api.file.CopySpec import org.gradle.api.tasks.bundling.Jar import org.gradle.kotlin.dsl.\* import org.jetbrains.kotlin.diagnostics.PositioningStrategies.UNUSED\_VALUE import org.jetbrains.kotlin.gradle.tasks.KotlinCompile  buildscript **{** @Suppress("ASSIGNED\_BUT\_NEVER\_ACCESSED\_VARIABLE")  var kotlinVersion: String by extra  @Suppress("ASSIGNED\_BUT\_NEVER\_ACCESSED\_VARIABLE")  var kotlinTestVersion: String by extra  @Suppress("ASSIGNED\_BUT\_NEVER\_ACCESSED\_VARIABLE")  var springBootVersion : String by extra   @Suppress("UNUSED\_VALUE")  kotlinVersion = "1.2.41"  @Suppress("UNUSED\_VALUE")  kotlinTestVersion = kotlinVersion  @Suppress("UNUSED\_VALUE")  springBootVersion = "2.0.3.RELEASE"    repositories **{** mavenCentral()  **}** dependencies **{** classpath("org.springframework.boot:spring-boot-gradle-plugin:$springBootVersion")  classpath("org.jetbrains.kotlin:kotlin-gradle-plugin:${kotlinVersion}")  classpath("org.jetbrains.kotlin:kotlin-allopen:${kotlinVersion}")  **} }** repositories **{** mavenCentral() **}** plugins **{** kotlin("jvm") version embeddedKotlinVersion  id("io.spring.dependency-management") version "1.0.5.RELEASE" **}** apply **{** plugin("org.springframework.boot") **}** dependencies **{** compile("org.springframework.boot:spring-boot-starter")  compile(kotlin("stdlib-jdk8"))  compile(kotlin("reflect"))  testCompile("org.springframework.boot:spring-boot-starter-test") **}** val project = mapOf(  name to "backend" )  tasks.withType<KotlinCompile> **{** kotlinOptions **{** jvmTarget = "1.8"  freeCompilerArgs = "-Xjsr305=strict".lines()  **} }** // below from start.spring.io -- gradle DSL //compileTestKotlin { // kotlinOptions { // freeCompilerArgs = ["-Xjsr305=strict"] // jvmTarget = "1.8" // } //} |

and then create the Spring app

Note: the previously fat jar with manifest HelloWorld.class is removed

run the clean build and run the jar as shown

|  |
| --- |
|  |

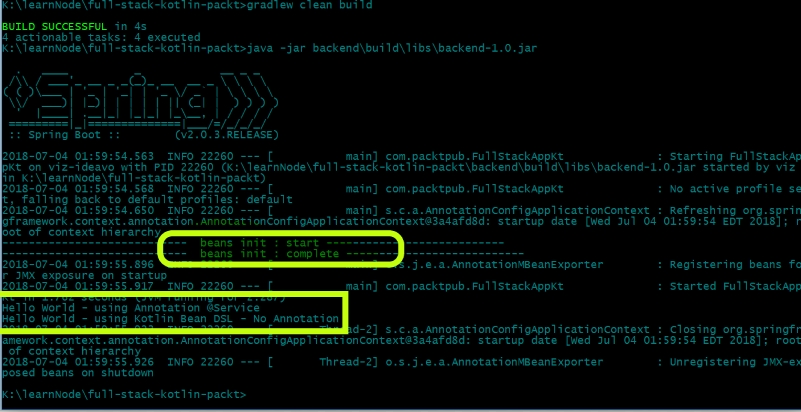
]

========================================================================= Add Kotlin DSL =======



git commit here

run it



* m
* m
* m
* m
* install pyCharm
* create a git repo for python prjoect
* in terminal clone the project here:
  + K:\wins-py-rpj\ecm-jupyter
* Open pyCharm and create a New Project in that Dir
  + when prompted to use existing say yes
* at this point do not worry about project settings
  + because the Python interpretor is not configured yet – if that what you are thinking
  + create a file called hello.py and type print(‘hello’)
  + try to run it, pyCharm will prompt you to set the interpretor now
    - I used a venv instead of Conda here

Images:

create project in git hub

create project in git hub

create project in git hub

create project in git hub