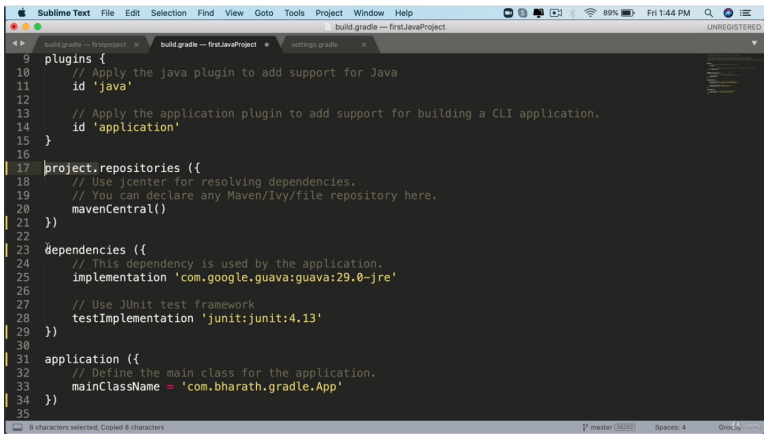
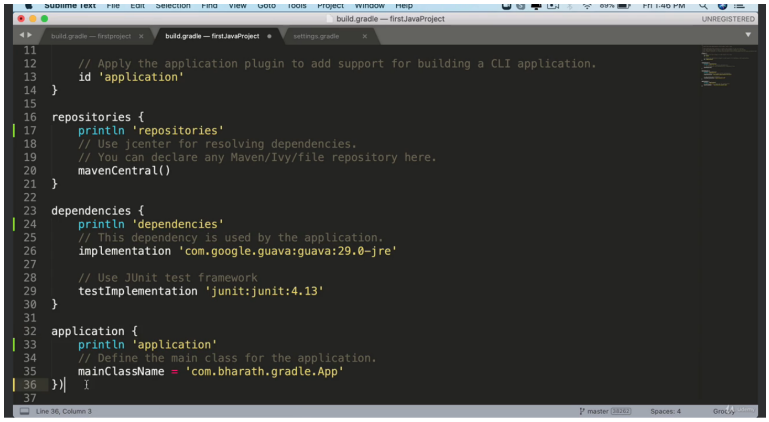
Everything that we saw is part of the project object. We do not need to call them explicitly and brackets are not required as well. Notice line number 17. Application method brackets are optional. Every method is part of the project object and it is not require call them using project object explicitly. Every block is closer which is nothing but a lambda.

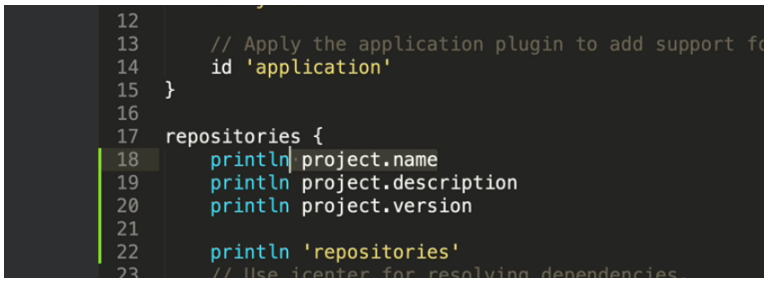


```
plugins {
    // Apply the java plugin to add support for Java
    id 'java'

    // Apply the application plugin to add support for building a CLI application.
    id 'application'
}

project.repositories ({
    // Use jcenter for resolving dependencies.
    // You can declare any Maven/Ivy/file repository here.
    mavenCentral()
})

dependencies ({
    // This dependency is used by the application.
    implementation 'com.google.guava:guava:29.0-jre'

    // Use JUnit test framework
    testImplementation 'junit:junit:4.13'
})

application ({
    // Define the main class for the application.
    mainClassName = 'com.bharath.gradle.App'
})
```

If you notice this file from a project perspective it is very easy to understand.

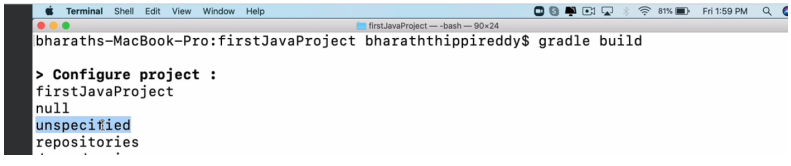


```
    // Apply the application plugin to add support for building a CLI application.
    id 'application'
}

repositories {
    println 'repositories'
    // Use jcenter for resolving dependencies.
    // You can declare any Maven/Ivy/file repository here.
    mavenCentral()
}

dependencies {
    println 'dependencies'
    // This dependency is used by the application.
    implementation 'com.google.guava:guava:29.0-jre'

    // Use JUnit test framework
    testImplementation 'junit:junit:4.13'
}

application {
    println 'application'
    // Define the main class for the application.
    mainClassName = 'com.bharath.gradle.App'
})
```



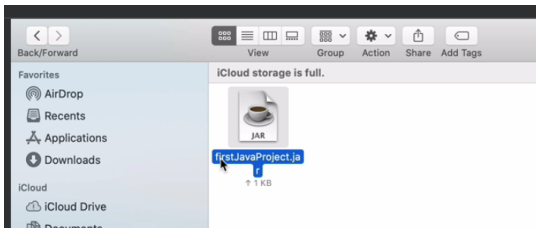
```
    // Apply the application plugin to add support fo
    id 'application'
}

repositories {
    println project.name
    println project.description
    println project.version

    println 'repositories'
```

Output of above print statement



```
bharaths-MacBook-Pro:firstJavaProject bharaththippireddy$ gradle build

> Configure project :
firstJavaProject
null
unspecified
repositories
```

Jar file it is creating without version.

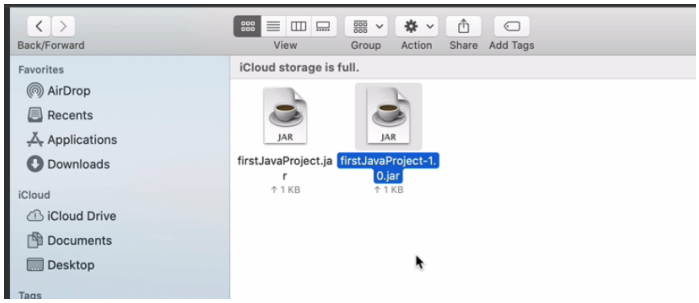


Project name comes from the settings file. Specify other values as specified below

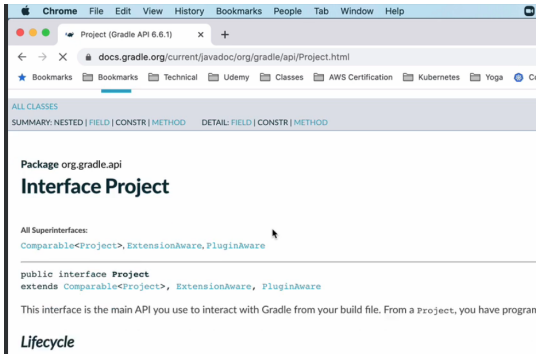
```
plugins {
    // Apply the java plugin to add support for
    id 'java'

    // Apply the application plugin to add supp
    id 'application'
}

project.description="First Java Project"
project.version=1.0

repositories {
    println project.name
    println project.description
    println project.version
```

```
BUILD SUCCESSFUL in 976ms
7 actionable tasks: 4 executed, 3 up-to-date
bharaths-MacBook-Pro:firstJavaProject bharaththippireddy$ gradle build

> Configure project :
firstJavaProject
First Java Project
1.0
repositories
dependencies
application
```

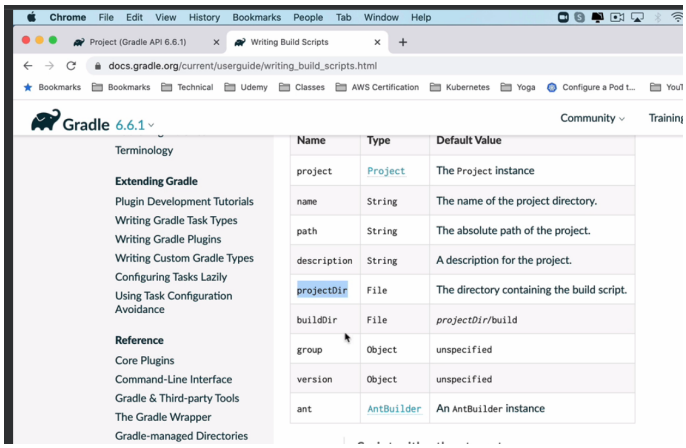
Jar created with the version we provided above.



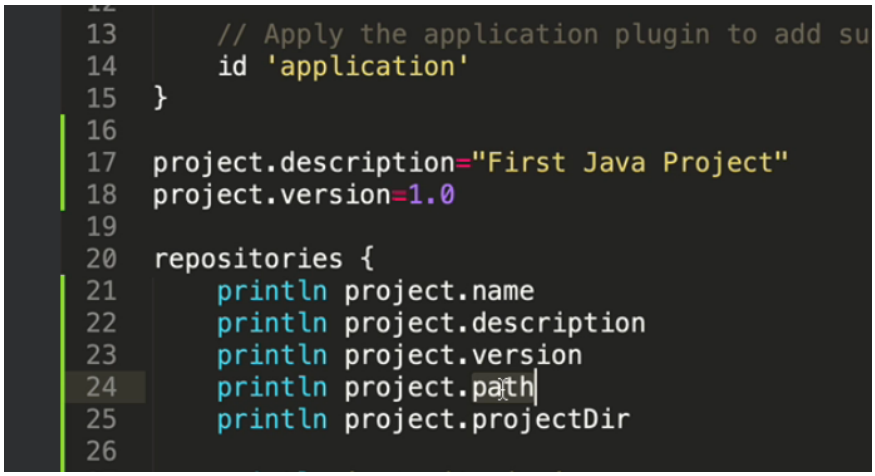
All info about this project object specified below



Project properties



Specify some of the properties




```
8 actionable tasks: 8 executed
bharaths-MacBook-Pro:firstJavaProject bharaththippireddy$ gradle clean build

> Configure project :
firstJavaProject
First Java Project
1.0
:
/Users/bharaththippireddy/Documents/gradle/firstJavaProject
repositories
```

We can also create user defined properties (xyz) and how can we access shown below. No need to use ext while retrieving values.

```
}

project.description="First Java Project"
project.version=1.0
project.ext.xyz="abc"

repositories {
    println project.name
    println project.description
    println project.version
    println project.path
    println project.projectDir
    println project.xyz
    println project.property("xyz")

    println 'repositories'
```

```
bharaths-MacBook-Pro:firstJavaProject bharaththippireddy$ gradle clean build

> Configure project :
firstJavaProject
First Java Project
1.0
:
/Users/bharaththippireddy/Documents/gradle/firstJavaProject
abc
abc
repositories
```

So far we have seen project objects. Now let us see the task.

Gradlew compile will create all the tasks required and run them implicitly.

```
Task task = new compile
```

```
*/

project.addTask(task)
```

```
Sublime Text  File  Edit  Selection  Find  View  Goto
build.gradle — firstproject  ×   build.gradle — firstJavaProject  ×
/*
 * This file was generated by the
 *
 * This is a general purpose Grad
 * Learn how to create Gradle bui
 */
task firstTask{
    println 'Gradle Rocks!!'
}
```

The above one equal to below one.

```
 6   */
 7  project.task (firstTask){
 8      println 'Gradle Rocks!!'
 9  }
```

There are so many overloaded methods

We can define variables and use them as follow. When variable name have conflict with project object properties, call them using project.abc.

```
 6   */
 7  task (firstTask){
 8      def abc=123
 9      println abc
10      println 'Gradle Rocks!!'
11  }
12
```

For example we gralde jar (jar is task - internally it run sequence of job like compile , run and install)

```
BUILD SUCCESSFUL in 600ms
bharaths-MacBook-Pro:firstproject bharaththippireddy$ gradle jar
```

```
 4   * This is a general purpose Gradle build.
 5   * Learn how to create Gradle builds at http
 6   */
 7  task (firstTask){
 8      def abc=123
 9      println abc
10      println 'Gradle Rocks!!'
11  }
12
13  task deployToStage{
14      doLast(){
15              println "Deployed to stage"
16      }
17  }
```

```
BUILD SUCCESSFUL in 621ms
bharaths-MacBook-Pro:firstproject bharaththippireddy$ gradle dTS

> Configure project :
123
Gradle Rocks!!

> Task :deployToStage
Deployed to stage
```

Notice how sequence is working from line number 25

```
 6  */
 7  task (firstTask){
 8      def abc=123
 9      println abc
10      println 'Gradle Rocks!!'
11  }
12
13  task deployToStage{
14      doLast(){
15              println "Deployed to stage"
16      }
17  }
18
19  task deployToProd{
20      doLast(){
21              println "Deployed to prod"
22      }
23  }
24
25  deployToProd.dependsOn deployToStage
```

Notice the sequence of execution below.

```
BUILD SUCCESSFUL in 600ms
1 actionable task: 1 executed
bharaths-MacBook-Pro:firstproject bharaththippireddy$ gradle dTP

> Configure project :
123
Gradle Rocks!!

> Task :deployToStage
Deployed to stage

> Task :deployToProd
Deployed to prod

BUILD SUCCESSFUL in 604ms
2 actionable tasks: 2 executed
```

```
bharaths-MacBook-Pro:firstproject bharaththippireddy$ gradle dTP

> Configure project :
123
Gradle Rocks!!

> Task :deployToStage
Deployed to stage

> Task :deployToProd
Deployed to prod

> Task :cleanupFiles
Cleaned Up Files

BUILD SUCCESSFUL in 617ms
3 actionable tasks: 3 executed
```

FinalizedBy

```
build.gradle — firstproject    build.gradle — firstJavaProject    settings.gradle
    def abc=123
    println abc
    println 'Gradle Rocks!!'
}

task deployToStage{
    doLast(){
            println "Deployed to stage"
    }
}

task deployToProd{
    doLast(){
            println "Deployed to prod"
    }
}

task cleanupFiles{
    doLast(){
        println 'Cleaned Up Files'
    }
}

deployToProd.dependsOn deployToStage
deployToProd.finalizedBy cleanupFiles

Line 34, Column 1
```

```
bharaths-MacBook-Pro:firstproject bharaththippireddy$ gradle dTP

> Configure project :
123
Gradle Rocks!!

> Task :deployToStage
Deployed to stage

> Task :deployToProd
Deployed to prod

> Task :cleanupFiles
Cleaned Up Files

BUILD SUCCESSFUL in 617ms
3 actionable tasks: 3 executed
```

```
}

deployToProd.dependsOn deployToStage
deployToProd.finalizedBy cleanupFiles
defaultTasks "deployToStage"

Line 33, Column 28; Saved ~/Documents/gradle/firstproject/build.gradle (UTF-8)
```

Just pass gradle - notice deploy to stage executes as it is the default task as we coded above.

```
bharaths-MacBook-Pro:firstproject bharaththippireddy$ gradle

> Configure project :
123
Gradle Rocks!!

> Task :deployToStage
Deployed to stage

BUILD SUCCESSFUL in 624ms
1 actionable task: 1 executed
```

Good job!

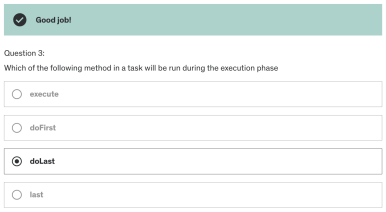
Question 1:
Every build.gradle has an object of which of the following Gradle api classes

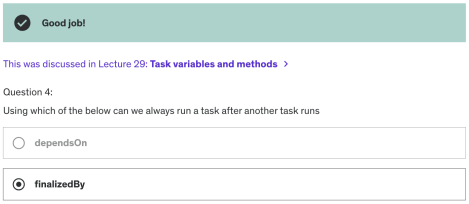
- Build
- Task
- Settings
- **Project**

Good job!

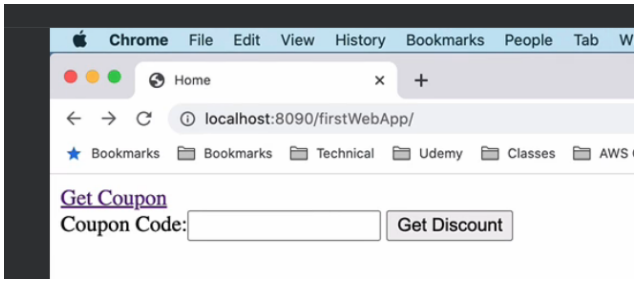
Question 2:
Which of the following should be used to add custom property called abc on the gradle project object

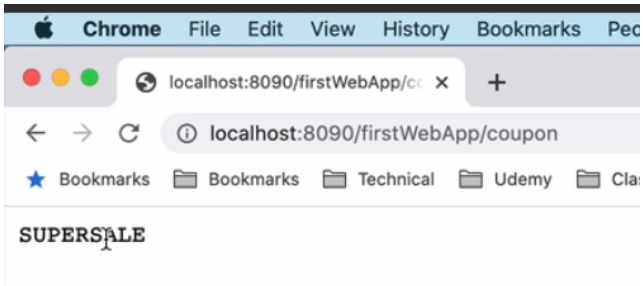
- project.props.abc
- project.abc
- project.abc.ext
- **project.ext.abc**

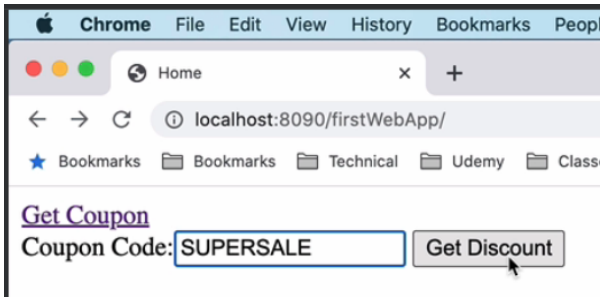


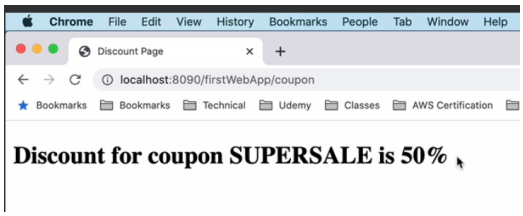


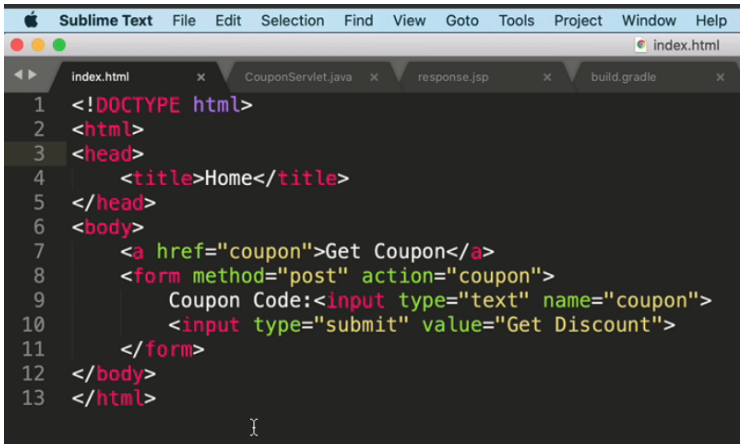
Create java web application

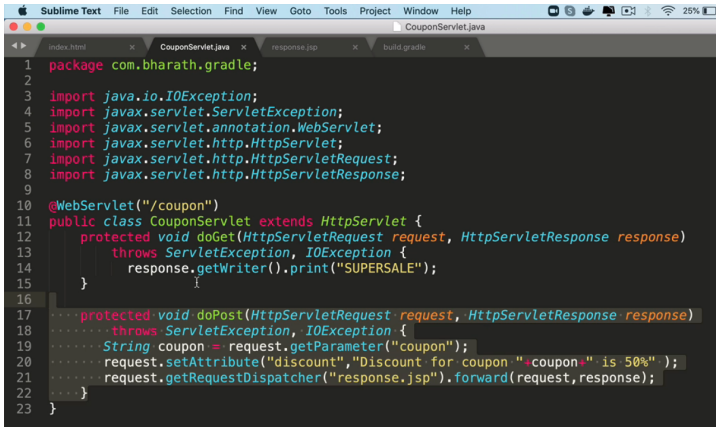


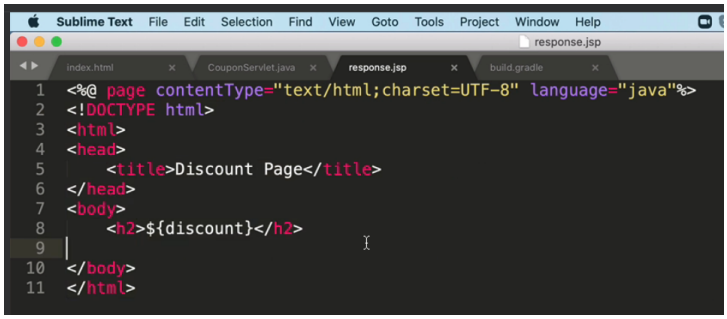




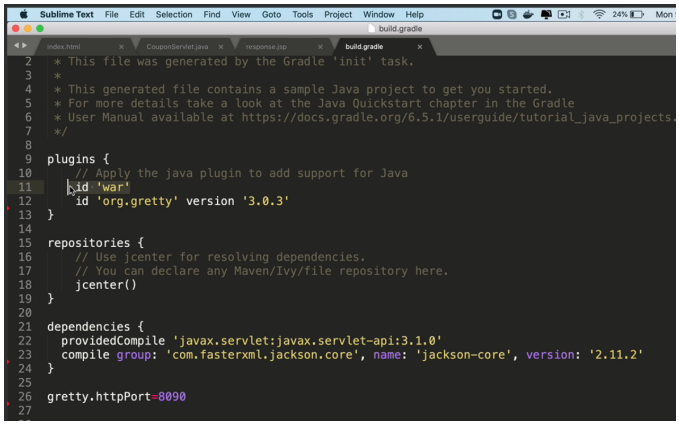




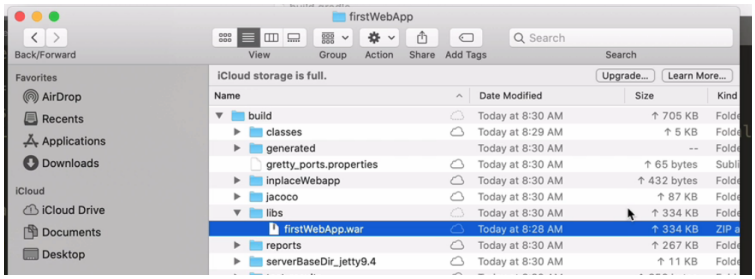




Notice the plugin ‘war’



Gradlew build generate the war file so we can deploy in tomcat server.



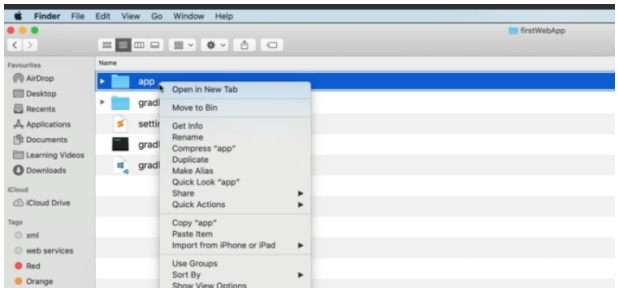
How to overcome this.

Or use the gretty plugin- this helps us to run the application as part of gradle build.

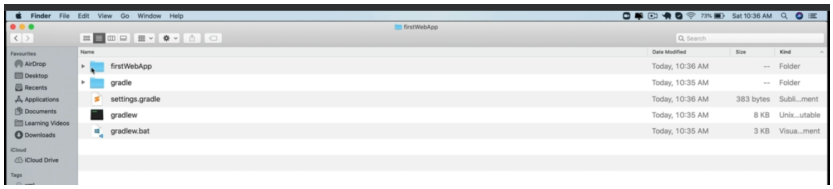
```
*/

plugins {
    // Apply the java plugin to add support for Java
    id 'war'
    id 'org.gretty' version '3.0.3'
}
```

```
bharaththippireddy@bharaths-MacBook-Pro-2 gradle % cd firstWebApp
bharaththippireddy@bharaths-MacBook-Pro-2 firstWebApp % gradle init

Select type of project to generate:
  1: basic
  2: application
  3: library
  4: Gradle plugin
Enter selection (default: basic) [1..4] 2

Select implementation language:
  1: C++
  2: Groovy
  3: Java
  4: Kotlin
  5: Scala
  6: Swift
Enter selection (default: Java) [1..6] 3

Split functionality across multiple subprojects?:
  1: no - only one application project
  2: yes - application and library projects
Enter selection (default: no - only one application project) [1..2] 1
```

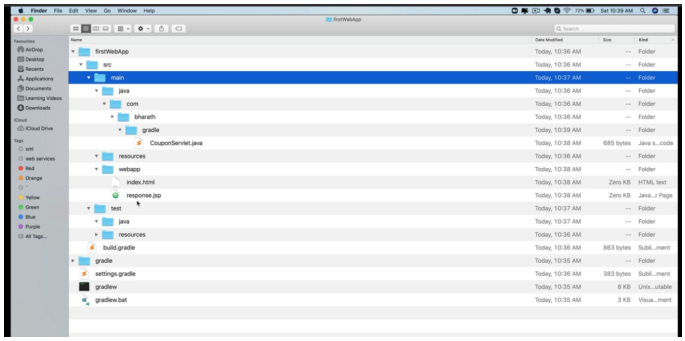
```
Select build script DSL:
  1: Groovy
  2: Kotlin
Enter selection (default: Groovy) [1..2] 1

Generate build using new APIs and behavior (some features may change in the next minor re

Select test framework:
  1: JUnit 4
  2: TestNG
  3: Spock
  4: JUnit Jupiter
Enter selection (default: JUnit Jupiter) [1..4] 1

Project name (default: firstWebApp):
Source package (default: firstWebApp): com.bharath.gradle
```



Rename app folder to firstWebApp as shown below. Similarly in settings file.

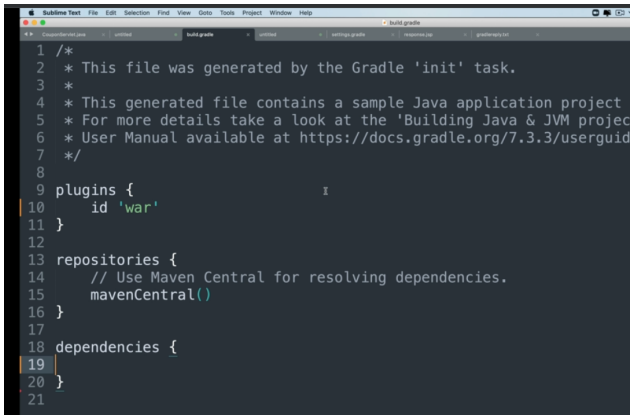


Create webapp folder and create index.html and response.jsp skeleton files and in gradle file copy and paste Couponservlet.java



Copy the corresponding code and past it.

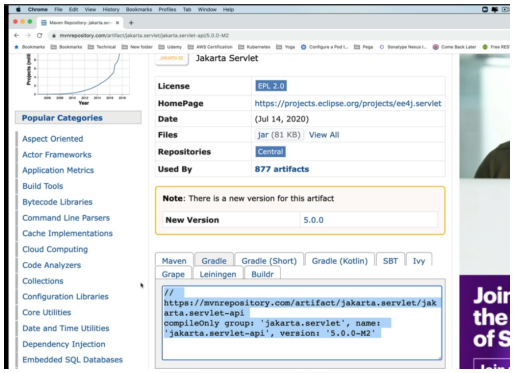
Change the plugin name from application to war. War is child plugin of war and cleanup build.gradle



```
/*
 * This file was generated by the Gradle 'init' task.
 *
 * This generated file contains a sample Java application project
 * For more details take a look at the 'Building Java & JVM projec
 * User Manual available at https://docs.gradle.org/7.3.3/userguid
 */

plugins {
    id 'war'
}

repositories {
    // Use Maven Central for resolving dependencies.
    mavenCentral()
}

dependencies {

}
```

We wil see war task as shown below.

```
bharaththippireddy@bharaths-MacBook-Pro-2 firstWebApp % gradle tasks --all

> Task :tasks

------------------------------------------------------------
Tasks runnable from root project 'firstWebApp'
------------------------------------------------------------

Build tasks
-----------
firstWebApp:assemble - Assembles the outputs of this project.
firstWebApp:build - Assembles and tests this project.
firstWebApp:buildDependents - Assembles and tests this project and all projects
nd on it.
firstWebApp:buildNeeded - Assembles and tests this project and all projects it
.
firstWebApp:classes - Assembles main classes.
firstWebApp:clean - Deletes the build directory.
firstWebApp:jar - Assembles a jar archive containing the main classes.
firstWebApp:testClasses - Assembles test classes.
firstWebApp:war - Generates a war archive with all the compiled classes, the we
ent and the libraries.
```



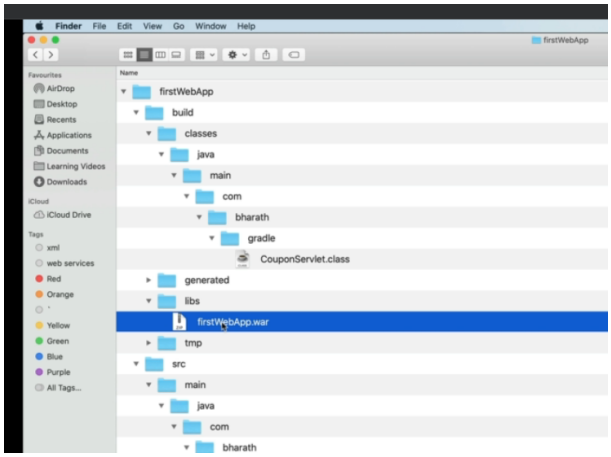
Copy this dependency and past it as shown below

```
16 }
17
18 dependencies {
19     compileOnly group: 'jakarta.servlet', name: 'jakarta.servlet-api', version:
20 }
21
```

We are using this only for compile and we do not want this packaging that is why we had to keep it as compileOnly.

```
bharaththippireddy@bharaths-MacBook-Pro-2 firstWebApp % gradle war

BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 executed
bharaththippireddy@bharaths-MacBook-Pro-2 firstWebApp %
```

It creates war file.



Add gretty plugin as well as shown below. It make our web application run on jetty server.

```
plugins {
    id 'war'
    id 'org.gretty' version '4.0.3'
}
```

```
firstWebApp:showClassPath - Shows classpath information
firstWebApp:tomcatRestart - Sends 'restart' command to a running server.
firstWebApp:tomcatRun - Starts web-app inplace, in interactive mode.
firstWebApp:tomcatRunDebug - Starts web-app inplace, in debug and interactive mode.
firstWebApp:tomcatRunWar - Starts web-app on WAR-file, in interactive mode.
firstWebApp:tomcatRunWarDebug - Starts web-app on WAR-file, in debug and interactive mode
```

Just run gradle appRun as shown below

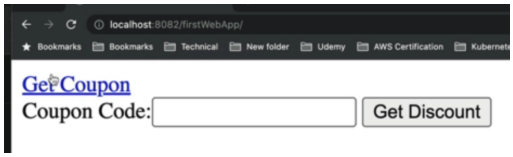
```
21:33:09.163 [main] DEBUG o.akhikhl.gretty.JettyServerManager - Jetty 11.0.3 starting.
21:33:09.420 [main] DEBUG o.a.gretty.JettyServerConfigurer - jetty context temp directory
: /Users/bharaththippireddy/Documents/gradle/firstWebApp/firstWebApp/build/serverBaseDir_
jetty11/webapps-exploded/firstWebApp
21:33:09.642 [main] INFO  o.a.gretty.JettyServerStartInfo - Jetty 11.0.3 started and list
ening on port 8080
21:33:09.653 [main] INFO  o.a.gretty.JettyServerStartInfo - firstWebApp runs at:
21:33:09.653 [main] INFO  o.a.gretty.JettyServerStartInfo -   http://localhost:8080/first
WebApp
21:33:09.658 [main] DEBUG org.akhikhl.gretty.ServiceProtocol - ServiceProtocol.send(52205
, {"status":"successfully started","host":"localhost","httpPort":8080,"contexts":[{"proto
col":"http","host":"localhost","port":8080,"contextPath":"/firstWebApp","baseURI":"http:/
/localhost:8080/firstWebApp"}]})
21:33:09.658 [main] DEBUG o.akhikhl.gretty.JettyServerManager - Jetty 11.0.3 started.

> Task :firstWebApp:appRun
Press any key to stop the server.
<-----------> 87% EXECUTING [9s]
> :firstWebApp:appRun
```

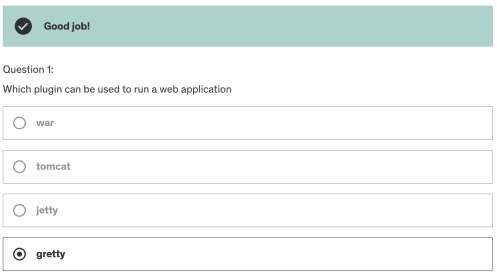
We can also run on tomcat as shown below.

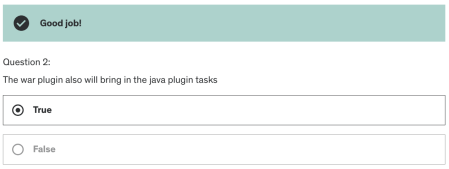
```
bharaththippireddy@bharaths-MacBook-Pro-2 firstWebApp % gradle tomcatRun
<-----------> 87% EXECUTING [618ms]
> :firstWebApp:tomcatRun
```

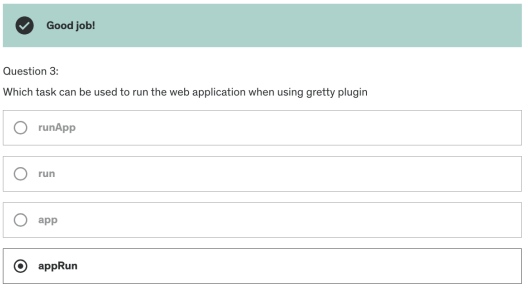
```
19
20 dependencies {
21     // Use JUnit test frame
22     compileOnly(group:"jaka
23 }
24
25 gretty.httpPort = 8082
```

We can also port like this

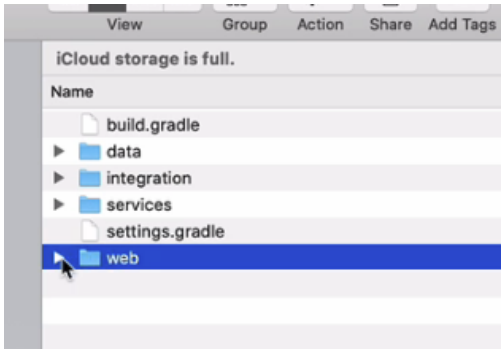




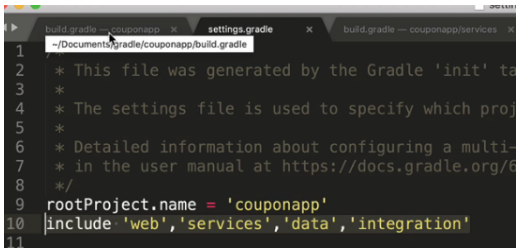




Different layers in the project web, service, integration and data layers



Include those modules in setting file



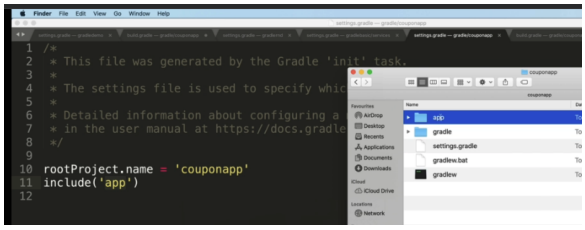
Or

```
project(':web'){
    dependencies{
        implementation project(':services')
    }
}

project(':services'){
    dependencies{
        implementation project(':data')
    }
}
```

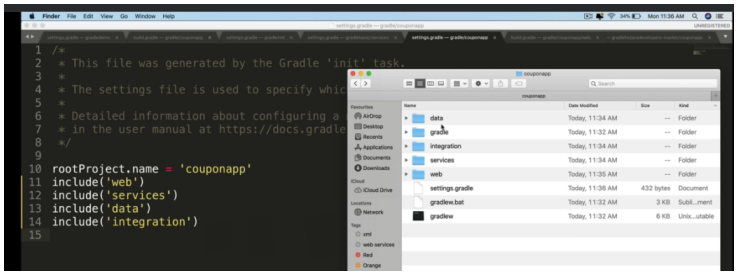
```
bharaththippireddy@bharaths-MacBook-Pro couponapp % gradle init

Select type of project to generate:
  1: basic
  2: application
  3: library
  4: Gradle plugin
Enter selection (default: basic) [1..4] 2

Select implementation language:
  1: C++
  2: Groovy
  3: Java
  4: Kotlin
  5: Scala
  6: Swift
Enter selection (default: Java) [1..6] 3

Split functionality across multiple subprojects?:
  1: no - only one application project
  2: yes - application and library projects
Enter selection (default: no - only one application project) [1..2]
```

```
  2: yes - application and library projects
Enter selection (default: no - only one application project) [1..2] 1

Select build script DSL:
  1: Groovy
  2: Kotlin
Enter selection (default: Groovy) [1..2] 1

Select test framework:
  1: JUnit 4
  2: TestNG
  3: Spock
  4: JUnit Jupiter
Enter selection (default: JUnit 4) [1..4]

Project name (default: couponapp):
Source package (default: couponapp): com.bharath.gradle
```

It creates app folder by default from from gradle 6.6 onwards

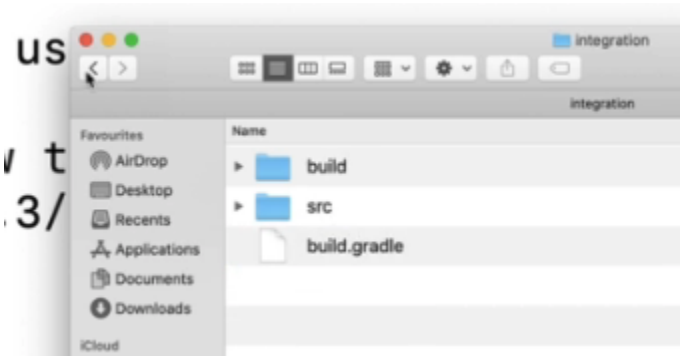


Create folders as shown below and include them in settings



```
bharaththippireddy@bharaths-MacBook-Pro couponapp % gradle build

Deprecated Gradle features were used in this build, making it incompatible
 with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.8.3/userguide/command_line_interface.html#se
c:command_line_warnings

BUILD SUCCESSFUL in 2s
23 actionable tasks: 23 executed
```

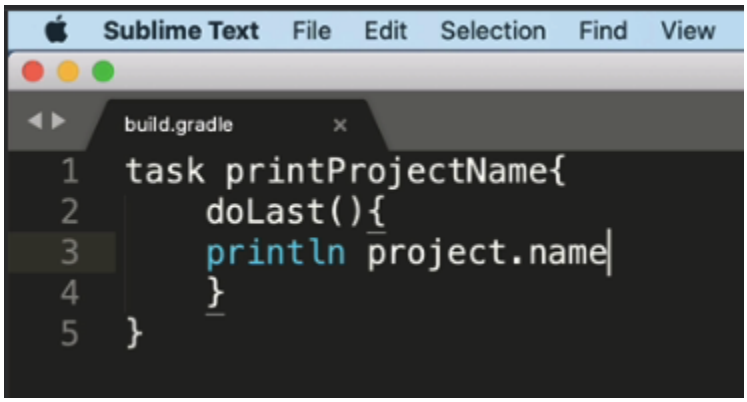
After this we will see the build folder and build.gradle file in every layer.



Gradle clean will remove the build folder from all the folders.

If we run ‘gradle task’ from the root folder it will do all the tasks of every module. In order to see specific module tasks run the same command from the module folder.

Create a task in the main build.gradle file



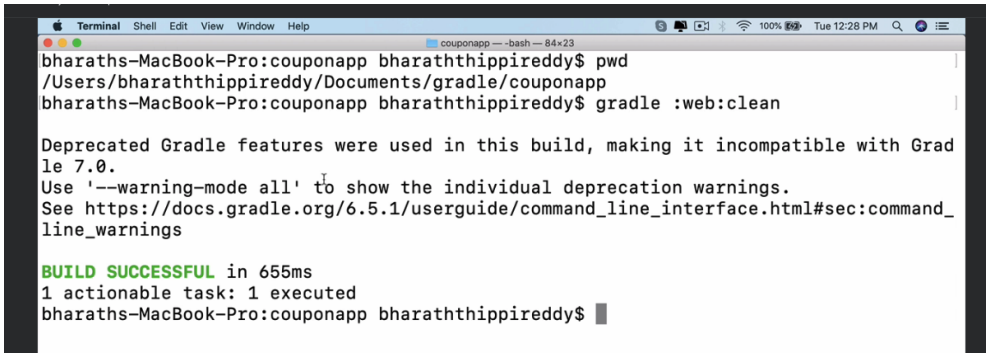

```
bharaths-MacBook-Pro:couponapp bharaththippireddy$ gradle pPN

> Task :printProjectName
couponapp
```

When you try same command from web we get the error

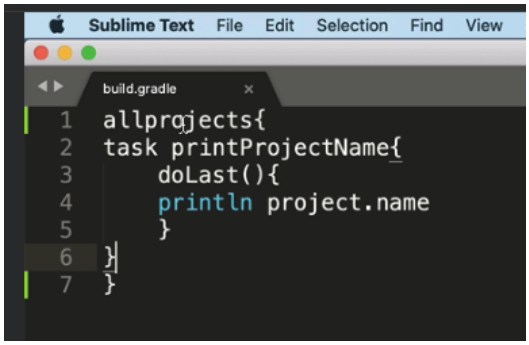
```
BUILD SUCCESSFUL in 688ms
1 actionable task: 1 executed
bharaths-MacBook-Pro:couponapp bharaththippireddy$ cd web
bharaths-MacBook-Pro:web bharaththippireddy$ gradle pPN
```

```
FAILURE: Build failed with an exception.

* What went wrong:
Task 'pPN' not found in root project 'web'.
```

How to run the submodule tasks from the root module



Similarly if we want to apply for all the projects use allproject method as shown below.



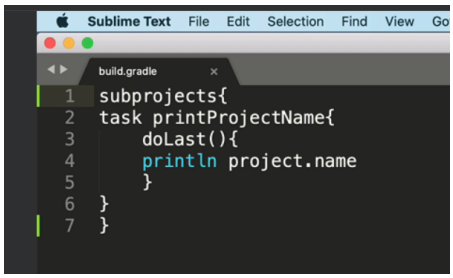
Now when i run the web module - that time we can see above task executing.

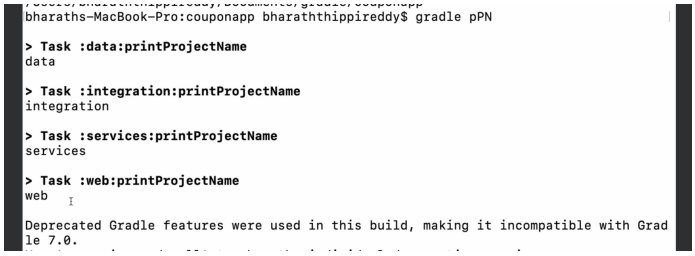
```
BUILD FAILED in 640ms
bharaths-MacBook-Pro:couponapp bharaththippireddy$ gradle :web:pPN

> Task :web:printProjectName
web
```

We get the same output for other projects also.

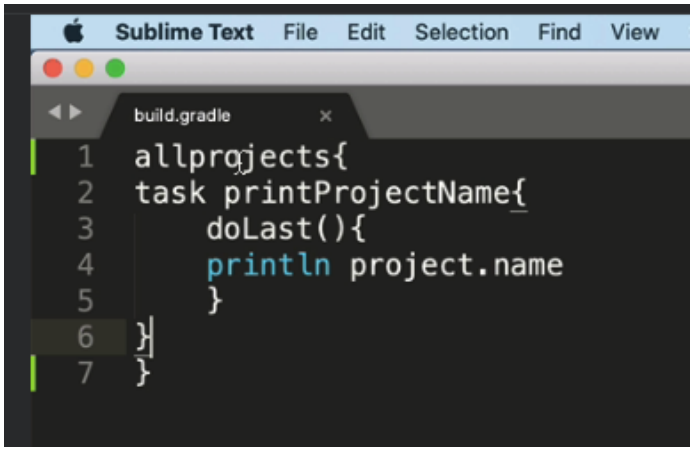
```
1 actionable task: 1 executed
bharaths-MacBook-Pro:couponapp bharaththippireddy$ gradle :services:pPN

> Task :services:printProjectName
services
```

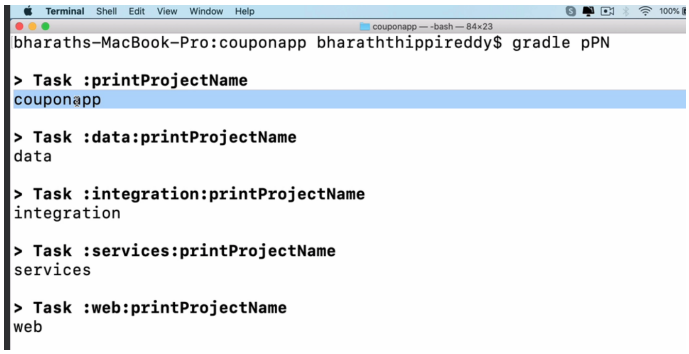
If we want to apply only for sub projects follow the syntax. It will not execute route folder tasks



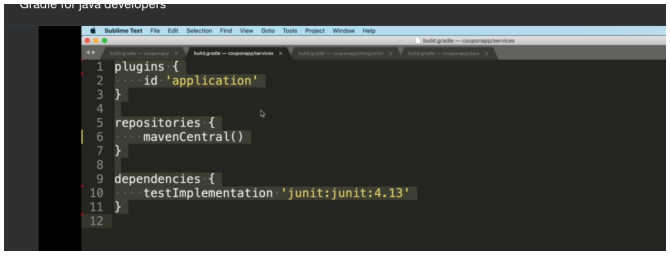


If you run after changing to allprojects. It will run the root project also.

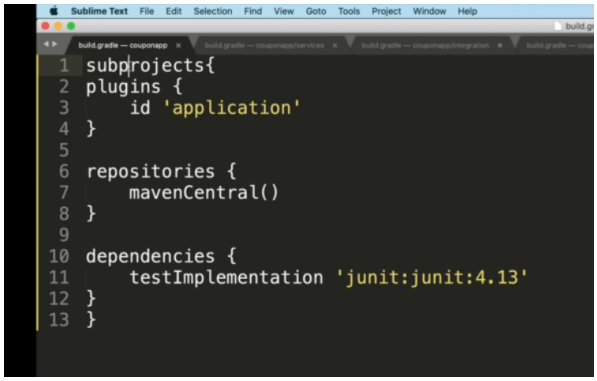




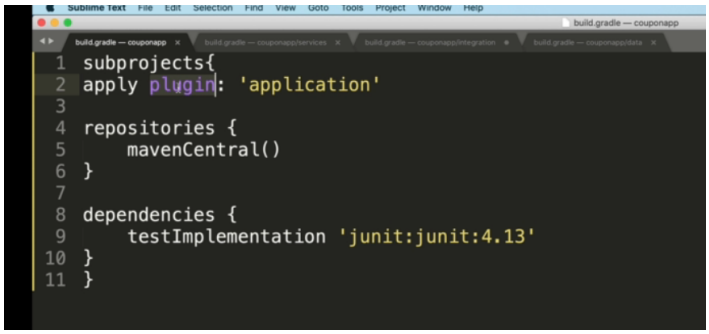
If we have common logic across all the projects - How to reuse.



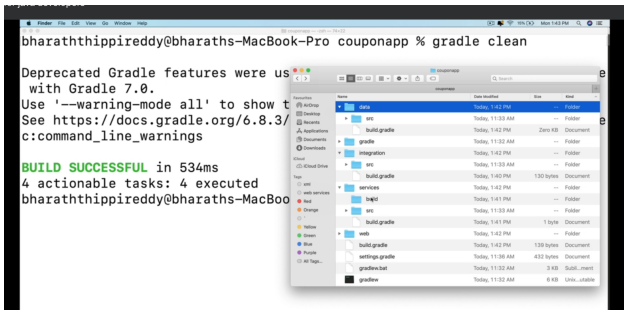
Move it to main build.gradle file



Syntax change for plugin



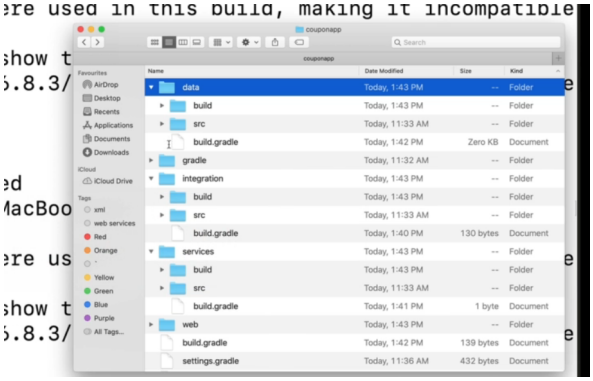
After gradle clean we wont find any build folder



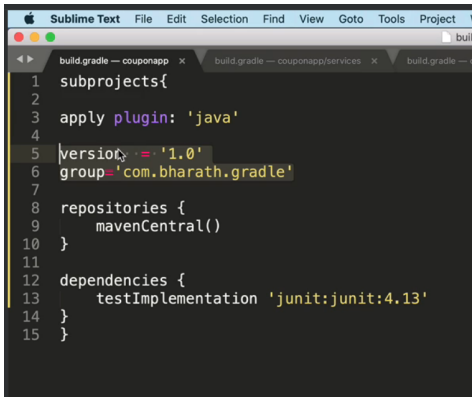
Now build

```
4 actionable tasks: 4 executed
bharaththippireddy@bharaths-MacBook-Pro couponapp % gradle build

Deprecated Gradle features were used in this build, making it incompatible
 with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.8.3/userguide/command_line_interface.html#se
c:command_line_warnings

BUILD SUCCESSFUL in 4s
```

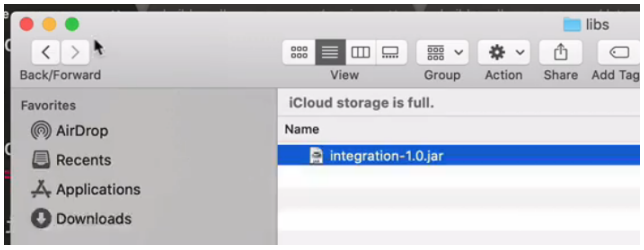
All build folder appears



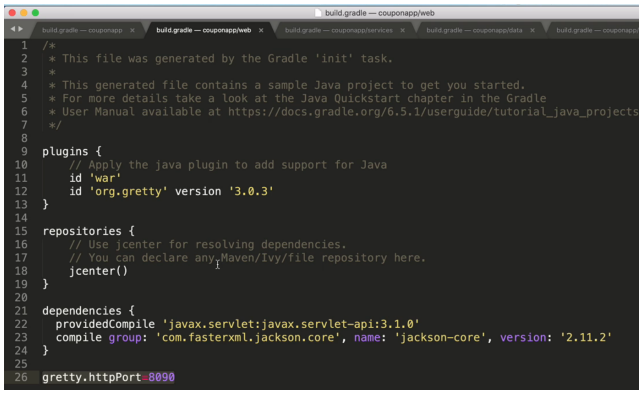
We can also apply properties for sub folders as follow. Line number 5 and 6



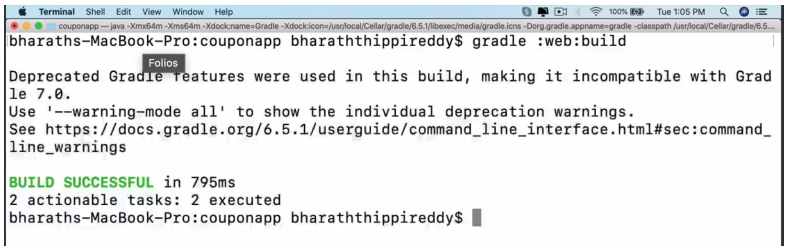
See how the jar was created for the integration folder.



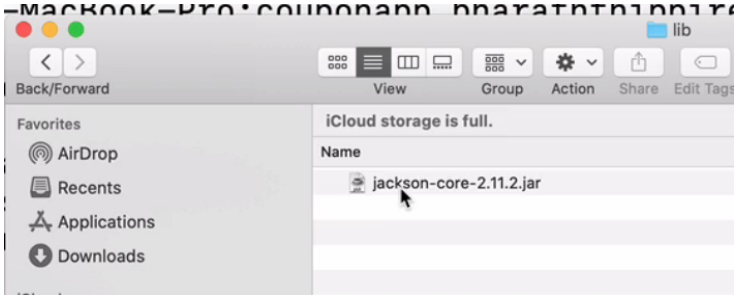
We have separate logic for web layer



Run build command for build folder



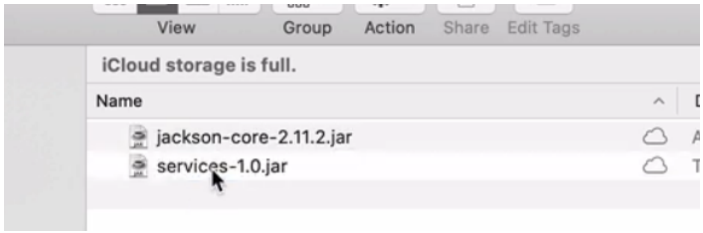
Extract the jar file and go to build folder you will find only dependencies that are part of web build.gradle.



But our web also depends on the service layer and other layers. So specify them in the main build.gradle as shown below. (dependencies between module)

```
subprojects{

apply plugin: 'java'

version  = '1.0'
group='com.bharath.gradle'

repositories {
    mavenCentral()
}

dependencies {
    testImplementation 'junit:junit:4.13'
}
}

project(':web'){
    dependencies{
        implementation project(':services')
    }
}
```

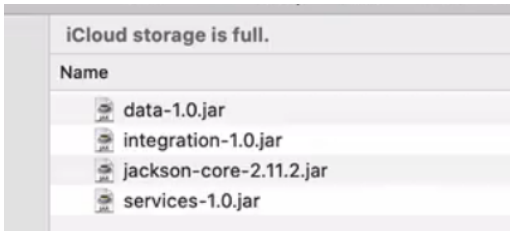
Now build web project again and check lib folder after extracting jar file



Now introduce some more project dependencies as shown below

```
dependencies {
    testImplementation 'junit:junit:4.13'
}
}

project(':web'){
    dependencies{
        implementation project(':services')
    }
}

project(':services'){
    dependencies{
        implementation project(':data')
    }
}



project(':data'){
    dependencies{
        implementation project(':integration')
    }
}
```

Now build and check lib folder



The above syntax is for root level dependencies.

Whereas in project level when one module depends on the other module we can also specify like this. Service module depends on the data module.

Show dependencies

```
BUILD SUCCESSFUL in 800ms
1 actionable task: 1 executed
bharaths-MacBook-Pro:couponapp bharaththippireddy$
bharaths-MacBook-Pro:couponapp bharaththippireddy$ gradle web:dependencies
```

It shows what are all the dependencies and its transitive dependencies as well as shown below.

```
springBoot
+--- com.fasterxml.jackson.core:jackson-core:2.11.2
+--- javax.servlet:javax.servlet-api:3.1.0
+--- javax.websocket:javax.websocket-api:1.0
\--- project :services
     \--- project :data
          \--- project :integration
```