Intellij has built in support for gradle , gradle developed in groovy

#build.gradle is like pom.xml which we used in maven file

Sample build.gradle file

|  |
| --- |
| group 'com.rsi.ingestion.services'  version '1.0-SNAPSHOT'  apply plugin: 'java'  sourceCompatibility = 1.8  repositories {  mavenCentral()  }  dependencies {  testCompile group:'io.vertx',name:'vertx-unit',version:'3.5.0'  testCompile group:'junit',name:'junit',version:'4.12'  compile group: 'com.google.code.gson', name: 'gson', version: '2.3.1'  compile project(':Utils')  compile project(':FileCopyEventProcessor')  compile project(':JobStatus')  compile project(':Producer')  compile rootProject  }  jar {  classifier = 'fat'  from configurations.compile.collect { zipTree it }  manifest {  attributes 'Main-class' : 'io.vertx.core.Launcher'  attributes 'Main-Verticle' : 'com.test.MyVerticle'  }  } |

First we need to define group and version

Gradle comes with lots of in build plugin like java so all the function is available if we include java plugin like jar, clean, compile etc

We can define java version using sourceCompatibility property

Here we are using maven repository

We can define dependencies using compile and tesCompile , test compile is only applicable for test phase

Also if we have multiple module we can add dependency for module using **compile project(':module-name')**

If we want to add root project dependency in child module we can use compile rootProject

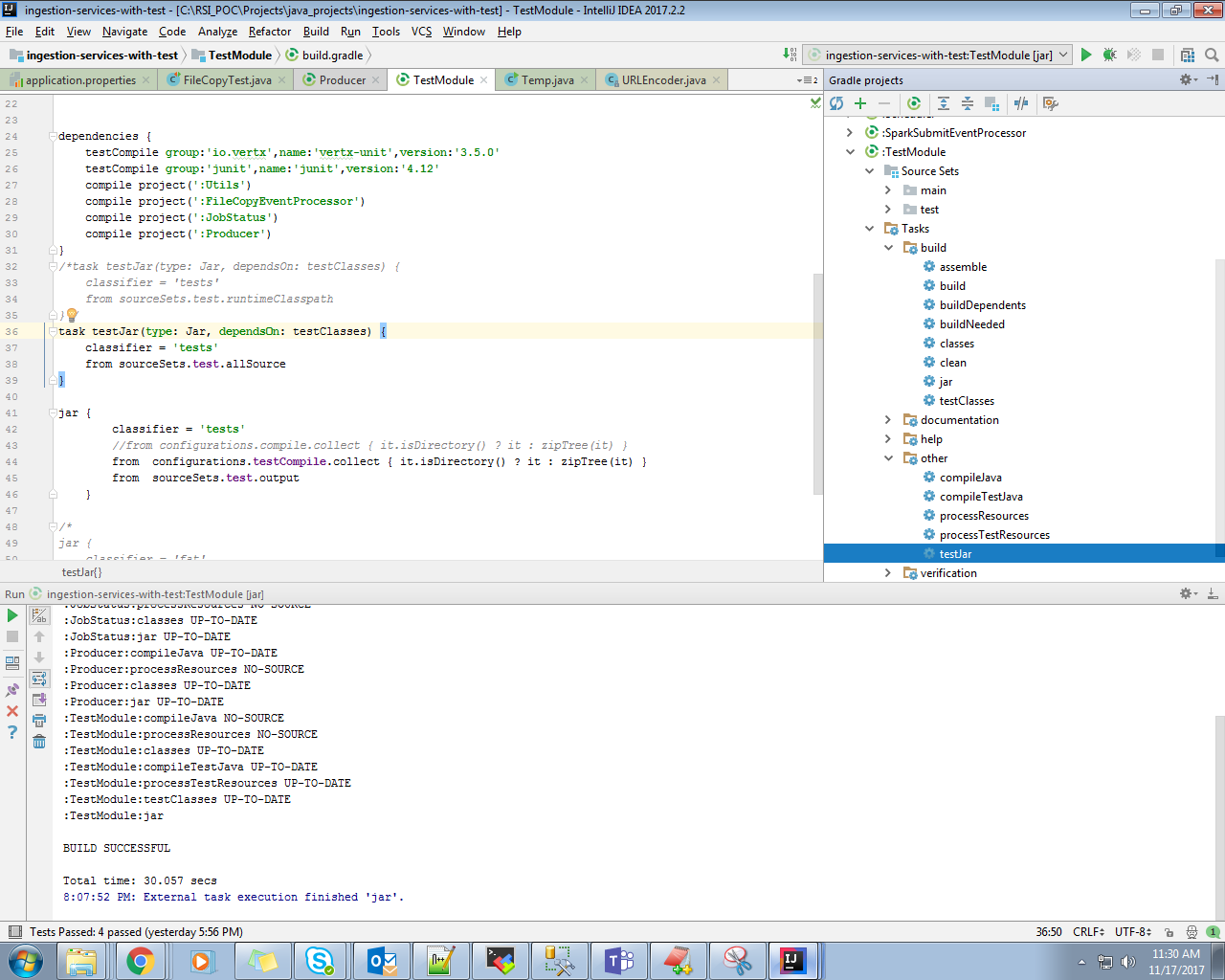
To create fat jar we can use jar task like shown above.

To build a jar which include all classes which are there under test folder we can use following two approaches

|  |
| --- |
| task testJar(type: Jar, dependsOn: testClasses) {  classifier = 'tests'  from sourceSets.test.allSource  }  jar {  classifier = 'tests'  from configurations.testCompile.collect { it.isDirectory() ? it : zipTree(it) }  from sourceSets.test.output  } |

In first approach we defined our custom task testJar , in this case we get only class files and resources which comes under Test folder

To create fat jar we need to use second approach



Screen shot for intellij for gradle java project

We can use intellij to direct run task as shown in above screen shot

#settings.gradle contains root project name as well as all modules name that we included under root project

Instead of hard coded version of dependencies we should use variables so that we can change easily in all modules by just changing rootproject variable

|  |
| --- |
| ext {  vertxVerstion = **'3.5.0'** } dependencies {  compile **"io.vertx:vertx-core:**$vertxVerstion**"** } |