Software experts for digital IOT enterprises the lab

DISI-Cesena// edited by Antonio Natali

Alma Mater Studiorum – University of Bologna
via Venezia 52, 47023 Cesena, Italy

Table of Contents

1 Demo

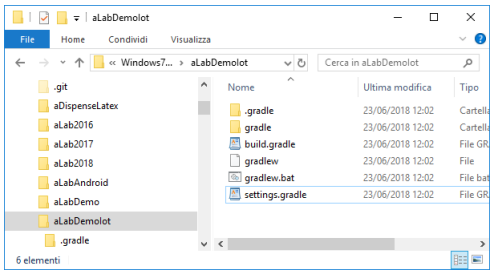
1.1 Working with gradle (and Eclipse)

1. Install gradle.
2. Install GIT.
3. Install Eclipse.
4. Create a working directory, e.g.`C:/aLabDemoIot`

1.2 Setting up a workspace

1. Open a terminal on `C:/aLabDemoIot` and execute

```
gradle init
```



2. Insert in the (generated, empty) `build.gradle` the sentence :

```
apply plugin: 'eclipse'
```

and execute:

```
gradle eclipse
```

3. Open Eclipse and import the project (actually named `aLabDemoIot`)
4. In the `Eclipse IDE`, edit the `build.gradle` as follows:

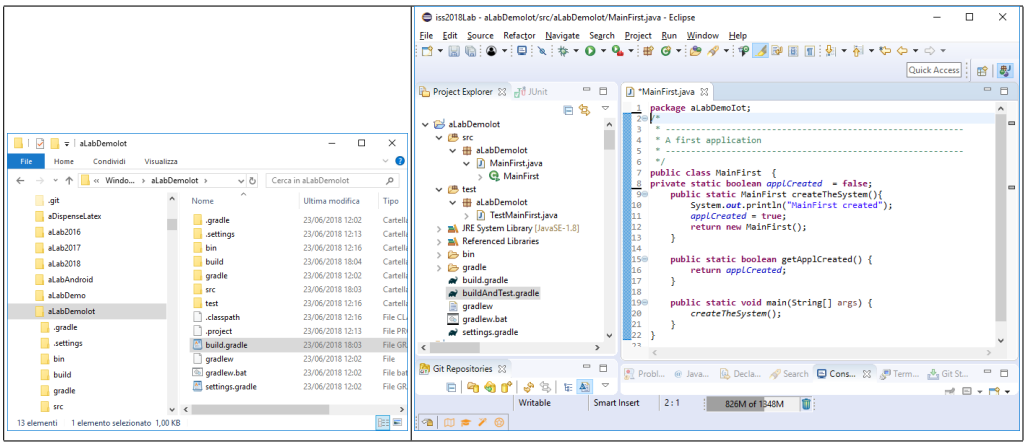
```
apply plugin: 'java'
apply plugin: 'eclipse'
version = "1.0"
sourceCompatibility = "1.8"
sourceSets {
    main {
        java { srcDirs = ['src' ]  }
    }
    test {
        java { srcDirs = ['test' ]   }
    }
}
repositories {
    mavenCentral()
}
dependencies {
    testCompile 'junit:junit:4.12'
compile group: 'com.typesafe.akka', name: 'akka-actor_2.11', version: '2.4.8'
compile group: 'com.typesafe.akka', name: 'akka-remote_2.11', version: '2.4.9-RC2'
}
jar {
    manifest {
        attributes "Class-Path": configurations.compile.collect { "lib/"+it.getName() }.join(' ')
        attributes 'Main-Class': 'aLabDemoIot.MainFirst'
    }
}
```

5. Open a `Shell` or the view `Termimal` and execute:

```
gradle eclipse //to update the classptah with the dependencies
```

1.3 A first application

1. Create the source directories `src` and `test`.
2. Define the application code.



3. Build

```
gradle build
```

Note the new created directory `build`.

4. Execute the generated jar

```
java -jar build/libs/aLabDemoIot-1.0.jar
```

1.4 Build and run

1. Insert in the `build.gradle` the code:

```
apply plugin: 'application'
mainClassName = "aLabDemoIot.MainFirst"
```

and execute:

```
gradle run
```

1.5 Automatic testing

1. Define the testing code (unit testing):

```
package aLabDemoIot;

import static org.junit.Assert.assertTrue;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class TestMainFirst {
    @Before
    public void setUp(){
        MainFirst.createTheSystem();
    }
    @Test
    public void testCreation(){
        assertTrue("", MainFirst.getApplCreated() );
    }
```

```
17      @After
18      public void after(){
19          System.out.println("End of test");
20      }
21  }
```

Listing 1.1. `test/aLabDemoIot/TestMainFirst.java`

2. Copy `build.gradle` into in the `buildAndTest.gradle` and add the following code:

```
test {
    testLogging {
        outputs.upToDateWhen { false }
        showStandardStreams = true
        events 'failed' , 'passed' //,  'started', 'skipped',
    }
    include '**/TestLed.class'
    test.afterSuite { TestDescriptor srtsuite, TestResult result -> //add closure to be notified
    if( !suite.parent && result.getTestCount() > 0) { //there is at least one test
      long elapsedTestTime = result.getEndTime() - result.getStartTime()
      System.out.println(
      """AFTERTEST Elapsed time for execution of ${result.getTestCount()} test(s):$elapsedTestTime ms""");
    }
    }
}
```

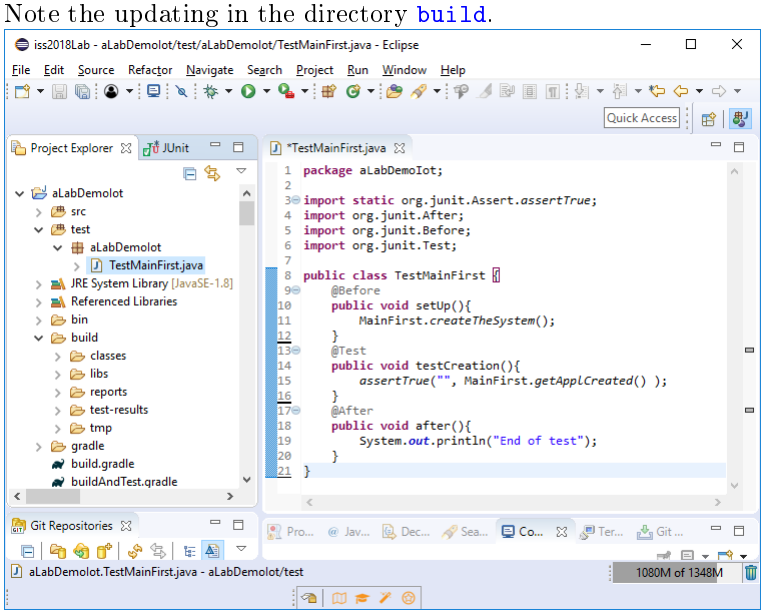
3. Build and Execute:

```
1       gradle -b buildAndtest.gradle build
```

The output should be:

```
aLabDemoIot.TestMainFirst > testCreation STANDARD_OUT
    MainFirst created
    End of test

aLabDemoIot.TestMainFirst > testCreation PASSED
AFTERTEST Elapsed time for execution of 1 test(s): 1581 ms

BUILD SUCCESSFUL in 5s
4 actionable tasks: 1 executed, 3 up-to-date
```

Note the updating in the directory `build`.



4. Look at the (created) test report: `build/reports/tests/test/index.html`

1.6 Jacoco

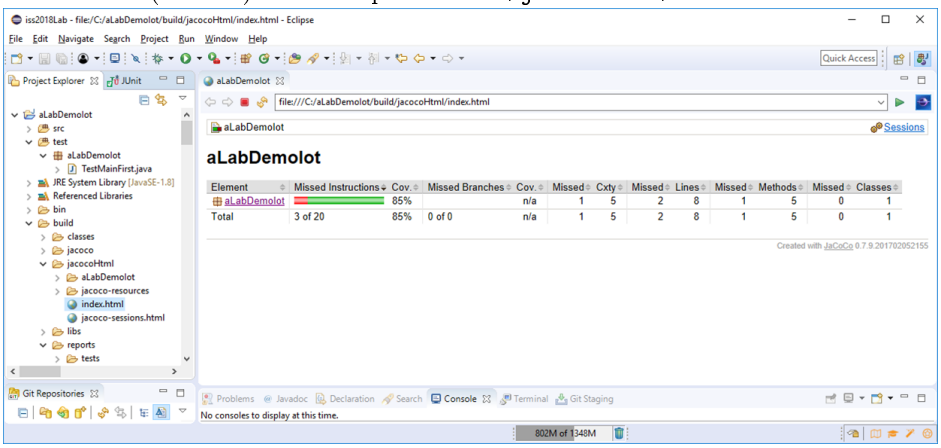
1. Add into in the `buildAndTest.gradle` the following code:

```
apply plugin: 'jacoco'

test.finalizedBy jacocoTestReport

jacocoTestReport {
    reports {
        xml.enabled false
        csv.enabled false
        html.destination "${buildDir}/jacocoHtml"
    }
}
```

2. Build and Execute:

```
1       gradle -b buildAndtest.gradle build
```

3. Look at the (created) Jacoco report: `build/jacocoHtml/index.html`



1.7 Working with GIT

1. Clone the `IotUniboCode` repository in a `REPODIR` by executing within `REPODIR` the command:

```
1    git clone https://github.com/anatali/IotUniboDemo
```

   To update the repository, the command is `git pull`.

2. Open an Eclipse working space and do:

```
Window -> ShowView -> Other -> Git -> Git Repositories
Add an existing local Git repository //(that loaded in  REPODIR)
```

3. Write the file `.gitignore` to exclude files in `GIT push`:

```
1  bin/
2  node_modules/
```

4. Import (by copying into your workspace)[1] the project `aLabDemoIot`.

[1] To avoid any conflict in project updating.