Hello Gradle

TestNG, Eclipse, IntelliJ IDEA

Óbuda University, Java Enterprise Edition
John von Neumann Faculty of Informatics
Lab 2

Dávid Bedők
2017.09.18.
v0.2

Java project structure

How to organize our source codes?

- ▷ **javac**
- ▷ **IDE** (Eclipse, IntelliJ IDEA, ...)
- ▷ **build tool**

Java Build tools:

- ▷ batch files / bash scripts
- ▷ Apache ANT (+ Apache IVY)
- ▷ Apache Maven
- ▷ Gradle

What kinds of things should be considered?

- ▷ There will be **unit tests**?
  - separate `source folder` for the unit tests (**test** and **main**)
- ▷ There will be **resources**?
  - separate `source folder` for the resources (**resources**)
- ▷ There will be **non-java source codes** too?
  - separate `source folders` for the disparate sources (**java**, **scala**, **groovy**, etc.)

Java philosophy

- ▷ transparency, clarity
- ▷ **classpath** (cp) usage
- ▷ usage of JAR, WAR, EAR, SAR*, APK** packagings

SAR*: JBoss specific
APK**: android specific

Java project structure

javac

We can do almost 'everything' with that (there are no rules)! We add all the source folders to the `javac` program via CLI.

Directory structure

`bin/`
`src1/` $\rightarrow$ **source folder**
`src2/` $\rightarrow$ **source folder**

The `Application` class uses an instance of `ImperialToMetricCalculator` class. Both classes are compiled into the same package, so application doesn't import the calculator class (at runtime both classes will be at the same place).

\hellotest

```
> javac -d ./bin ./src1/hu/qwaevisz/demo/Application.java
   ./src2/hu/qwaevisz/demo/ImperialToMetricCalculator.java
> java -cp ./bin hu.qwaevisz.demo.Application
```

Java project structure

Eclipse IDE

Configuration of the Eclipse's default 'Java project':

Directory structure

`bin/`
`src/` → **source folder**

Eclipse's 'Java project' with unit tests:

Directory structure

```
bin/
src/
    main/ → source folder
    test/ → source folder
```

In most of the IDEs these rules can be configure completely (Eclipse: `Project properties` | `Java Build Path` | `Source` tab).

Java project structure
Gradle

Default configuration of the Gradle's `java` plugin:

Directory structure

```
bin/
src/
    main/
        java/ → source folder
        resources/ → source folder
    test/
        java/ → source folder
        resources/ → source folder
```

Of course you are able to change these settings in Gradle. But if we use this default configuration we will have a very simple, clean and small build script to start the work.

Note: All the source folders and the `resources` directories are part of the classpath (we can reach its content during runtime).

JAR - Java ARchive

ZIP format which keeps (Java) `byte codes` (`*.class`), configuration files (e.g.: `*.properties`, `*.xml`, etc.) and a special metafile which contains key-value pairs (**MANIFEST.MF**).

Directory structure

```
META-INF/
   MANIFEST.MF
hu/
   qwaevisz/
      demo/
         HelloWorld.class
         Lorem.class
log4j.xml
```

It's structure is predefined, there is an option to store source files (e.g.: `*.java`, `*.groovy`, etc.) at the same place where the `byte codes` are located.

```
Manifest-Version: 1.0
Created-By: 1.7.0_67 (Oracle Corporation)
```

MANIFEST.MF

Executable JAR file

The Main-Class key has to be part of the MANIFEST.MF file, and the value of this key is the full qualified name of the entry point of the application.

```
Manifest-Version: 1.0
Created-By: 1.7.0_67 (Oracle Corporation)
Main-Class: hu.qwaevisz.demo.Application
```

MANIFEST.MF

```
> cd bin
> jar cvfe calculator.jar hu.qwaevisz.demo.Application
    hu/qwaevisz/demo/Application.class
    hu/qwaevisz/demo/ImperialToMetricCalculator.class
> cd ..
> java -jar bin/calculator.jar
```

create new archive
verbose
specify archive **f**ile name (2)
sepcify **e**ntry point (main class) (3)

Eclipse OXYGEN

IDE, 2017 June

Eclipse IDE for Java EE Developers

Download: `https://www.eclipse.org/downloads/`

Version: **4.7.0**

Install: unzip or installer

Integrated plugins:

- ▷ Gradle
- ▷ Maven
- ▷ Git
- ▷ EclEmma Java Code Coverage
- ▷ ...

In case of Hungarian keyboard layout (and usage) you have to turn off some shortcut keys (e.g. "{" (Ctrl + B): `Preferences` | `General` | `Keys` | `Skip all brakepoints` (Ctrl + Alt + B) → Unbind

Basic usage of Eclipse IDE: `http://users.nik.uni-obuda.hu/bedok.david/jse.html`

Additional plugins (`Help` / `Eclipse Marketplace`):

- ▷ **TestNG** (filter: `testng`)
  - `http://beust.com/eclipse`

Eclipse configuration
Code Style Formatter

`Window` | `Preferences` (type: `formatter`)

▷ `Java` | `Code Style` | `Formatter`

- New... / Import...: **uni-obuda-java-formatter**
  - Initialize: Eclipse [build-in]
  - Indentation | Indent | Statement within ‘switch’ body
  - Line Wrapping | General | Maximum line width: 160
  - Line Wrapping | Enum declaration
    - * Policy: Wrap all elements, every element on a new line
    - * Constants policy: Wrap all elements, every element on a new line + Force split
  - Comments | Line width | Maximum: 120

```
\eclipse\uni-obuda-java-formatter.xml
```

Eclipse configuration

Save Actions

`Window | Preferences (type: save actions)`

- ▷ `Java | Editor | Save Actions`
  - Perform the selected actions on save
    - **Format source code** (all lines)
    - Organize imports
    - Additional actions - Configure
      - Code Organaizing: Remove trailing whitespaces
      - Code Style: Use blocks in if/while/for/do statements
      - Member Accesses: Use ‘this’ qualifier for field accesses: Always
      - Member Accesses: Use ‘this’ qualifier for method accesses: Always
      - Unnecessary Code: Remove unused imports

IntelliJ IDEA
IDE

Download: https://www.jetbrains.com/idea/

- **Commercial product**
- The community version doesn't support the JavaEE, but without this feature it is usable for professional work as well (event JavaEE projects).
- Sometimes it is faster than Eclipse
- Different shortcut keys, hard to get used to
- Integrated Maven/Gradle/Git plugins

Hello World

src | main | java | hu | qwaevisz | hello | Application.java

```java
package hu.qwaevisz.hello;

public class Application {

  public static void main(final String[] args) {
    System.out.println("Hello World");
  }

  public int add(final int a, final int b) {
    return a + b;
  }

}
```

Application.java

[gradle|maven]\helloworld

TestNG

3rd party library

- http://testng.org/
- GitHub: https://github.com/cbeust/testng
- Version: **6.11**
- Artifactory URL:
  - `'org.testng:testng:6.11'`
  - group/groupId: `org.testng`
  - name/artifactId: `testng`
  - version: `6.11`

Unit Test with TestNG

src | test | java | hu | qwaevisz | hello | ApplicationTest.java

```java
package hu.qwaevisz.hello;

import org.testng.Assert;
import org.testng.annotations.Test;

public class ApplicationTest {

  @Test
  public void addNumbers() {
    Application app = new Application();
    Assert.assertEquals(app.add(2, 3), 5);
  }

}
```

ApplicationTest.java

Gradle
Build tool

- https://gradle.org/
- Download: https://gradle.org/releases/
- Version: **4.1**
- supports Java, C++, Python and more programming languages
- supports monorepo and multi-repo as well
- multi-language, multi-platform, multi-project and multi-channel software development (SaaS: Software as a Service)
- Install: unzip

Environment variables:

- **GRADLE_HOME** →`c:\apps\gradle−2.6`
- **Path** modification → `%Path%;%GRADLE_HOME%\bin`

Gradle

major properties

- ▷ Apache **Ant** based build system
  - http://ant.apache.org/
- ▷ Apache **Ivy** styled dependency management
  - http://ant.apache.org/ivy/
- ▷ intelligent default environments like Apache **Maven**
  - https://maven.apache.org/
- ▷ speed and hashing of **Git**
- ▷ metaprogramming with Apache **Groovy** / JetBrains **Kotlin**
  - http://groovy-lang.org/
  - https://kotlinlang.org/
- ▷ Directed Acyclic Graph (DAG)

Gradle

```
> gradle --version

------------------------------------------------------------
Gradle 4.1
------------------------------------------------------------

Build time:   2017-08-07 14:38:48 UTC
Revision:     941559e020f6c357ebb08d5c67acdb858a3defc2

Groovy:       2.4.11
Ant:          Apache Ant(TM) version 1.9.6 compiled on June 29
    2015
JVM:          1.8.0_101 (Oracle Corporation 25.101-b13)
OS:           Mac OS X 10.12.6 x86_64
```

Gradle - Eclipse integration

- Gradle's **eclipse plugin** constructs all the required configuration files and with these files you can import your project into Eclipse with ease.
- Eclipse's **Gradle plugin** recognizes the gradle's build files and Eclipse can handle the project properly.
- In the past it was recommended to use both ways at the same time, but now the official Eclipse plugin (buildship) is good enough.

Hello Gradle!

Hello World application with Gradle

[gradle|maven]\helloworld

```
apply plugin: 'java'
```

build.gradle

```
> gradle clean build

> Configure project :
Task name: info
Project name: helloworld


BUILD SUCCESSFUL in 1s
5 actionable tasks: 5 executed
```

- ▷ output: `build\libs\helloworld.jar` (non-executable)
- ▷ clean, build → tasks of 'java' plugin
  - dependent tasks of build: compileJava, jar, assemble, test, check, ...

Create a custom Gradle task

```
apply plugin: 'java'

sourceCompatibility = 1.7
version = '1.0'

task info() {
  println "Task name: " + name
  println "Project name: " + project.name
  println "Project version: " + version
}
```

build.gradle

```
> gradle info

> Configure project :
Task name: info
Project name: helloworld
Project version: 1.0


BUILD SUCCESSFUL in 0s
```

Executable JAR

Modify a Gradle jar task

```
[..]
jar {
    manifest {
        attributes 'Implementation-Title': 'Gradle Demo
            Application',
                   'Implementation-Version': version,
            'Main-Class': 'hu.qwaevisz.hello.Application'
    }
}

task run( type: Exec ) {
  workingDir 'build/libs'
  commandLine 'java', '-jar', "${project.name}-${version}.jar"
}
```

build.gradle

```
> gradle clean build run

[..]

> Task :run
Hello World

BUILD SUCCESSFUL in 2s
6 actionable tasks: 6 executed
```

Gradle TestNG integration

```
[..]
def testngVersion = '6.9.10'

repositories { mavenCentral() }

dependencies {
  testCompile group: 'org.testng', name: 'testng' , version:
      testngVersion
}

test {
  useTestNG()
}
```

Default test framework: junit.

build.gradle

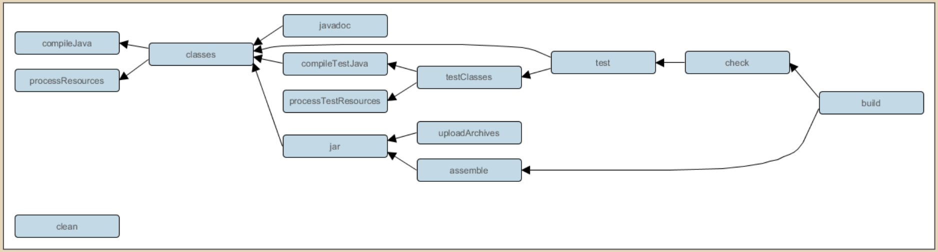
```
> gradle test
[..]
> Task :test

Gradle suite > Gradle test > hu.qwaevisz.hello.ApplicationTest.addNumbers FAILED
    java.lang.AssertionError at ApplicationTest.java:11

1 test completed, 1 failed
[..]
BUILD FAILED in 1s
3 actionable tasks: 3 executed
```

Gradle Java plugin

Tasks

https://docs.gradle.org/current/userguide/java_plugin.html

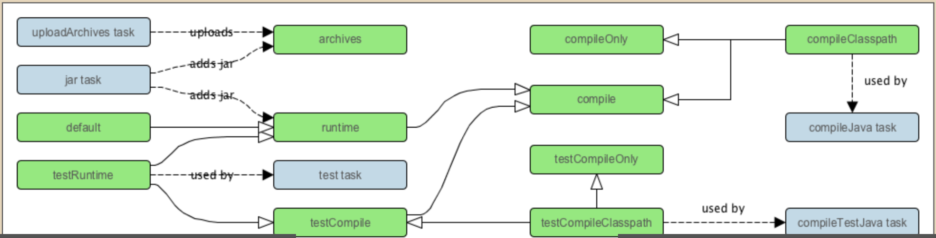
Task: compileJava, compileTestJava, jar, javadoc, test, clean, ...



Gradle Java plugin

Dependency Configurations

compile	Compile and runtime dependency.
compileOnly	Dependency only for compilation.
runtime	Runtime dependency.
testCompile	Additional compile and runtime dependency for tests (`test` task).
testRuntime	Additional runtime dependency for tests (`test` task).
archives	The artifactory of the project (e.g. `jar`)
...	...



Gradle Eclipse integration

```
[..]
apply plugin: 'eclipse'
[..]
```

build.gradle

```
> gradle eclipse
```

Additional tasks: eclipseClasspath, eclipseJdt, eclipseProject

Directory structure

```
settings/
   org.eclipse.jdt.core.prefs
.classpath
.project
```

Eclipse Gradle project

`File | Import... | Gradle | Existing Gradle Project`

- ▷ Project root directory: `\helloworld`
- ▷ Import options: Gradle wrapper

Where do I execute Gradle's tasks?

The Eclipse's Gradle plugin (`buildship`) is good to organize the Gradle project, but you can execute any Gradle tasks inside Eclipse after install that plugin. But I suggest to execute Gradle tasks via a separate terminal, without Eclipse IDE.

Gradle IntelliJ integration

```
[..]
apply plugin: 'idea'
[..]
```

build.gradle

```
> gradle idea
```

További taskok: `cleanIdea`, `ideaModule`, `ideaProject`, `ideaWorkspace`

Directory structure

```
helloworld.iml
helloworld.ipr
helloworld.iws
```