Gradle

Benoît Charroux
benoit.charroux@efrei.fr

Definition

- Gradle is build automation *evolved*
- Gradle can automate the building, testing, publishing, deployment of software packages

http://www.gradle.org/

- Mainly Gradle is usefull to add libraries (jar files) to a project

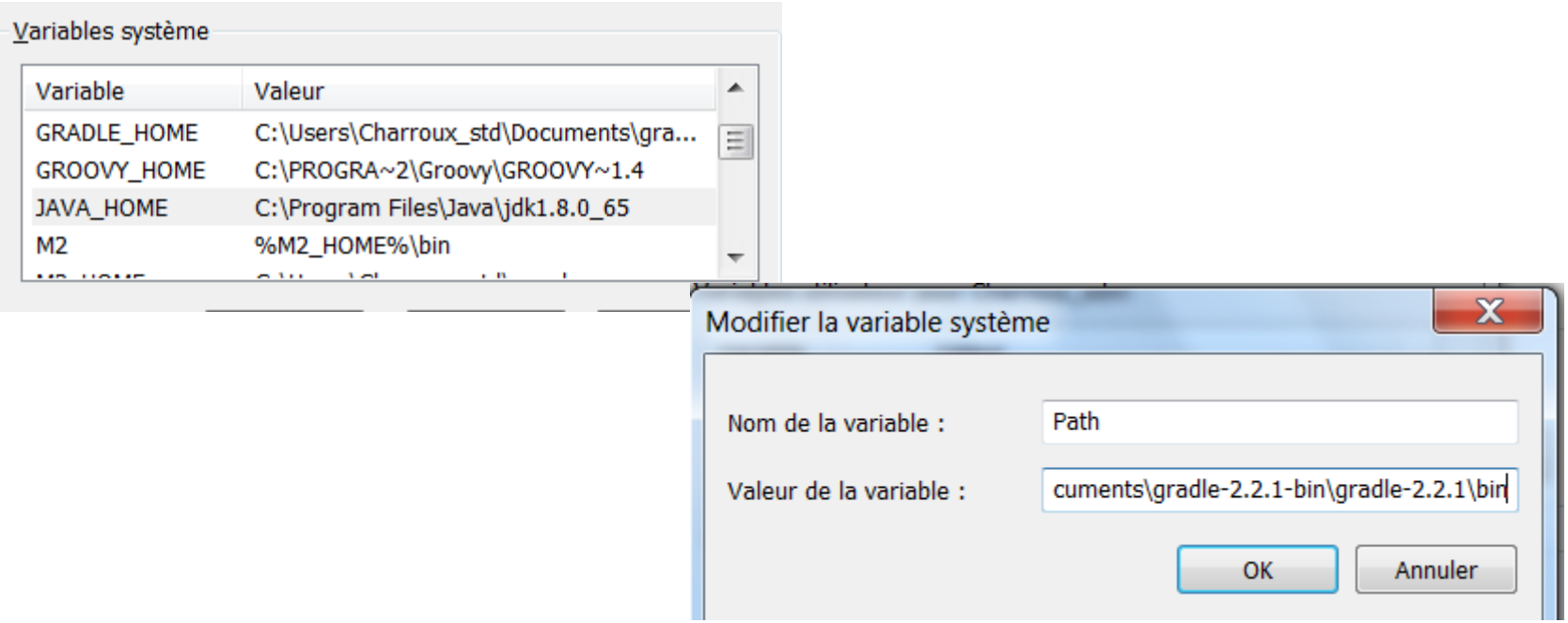
Installation (1/4)

- Download Gradle (binaries is enough)
- Uncompress the file
- Set environment variables
  - JAVA_HOME must point to a Java JDK (not a JRE)
  - Add GradleHome\bin to the PATH variable

```
Path=d:\programmes\CDF_Distribution\cdf34_1-dist;d:\programmes\CDF_Distribution\
cdf34_1-dist\lib;d:\programmes\CDF_Distribution\cdf34_1-dist\bin;C:\Python33\;C:
\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\Window
sPowerShell\v1.0\;C:\Program Files\Internet Explorer;C:\strawberry\c\bin;C:\stra
wberry\perl\site\bin;C:\strawberry\perl\bin;C:\Program Files\QuickTime\QTSystem\
;D:\programmes\Subversion\bin;C:\Program Files\MySQL\MySQL Utilities 1.3.6\;D:\P
rogrammes\gradle\gradle-2.1-all\gradle-2.1\bin
```

Installation (2/4)

- Under Linux :
  - export JAVA_HOME=path to java
  - export PATH=${PATH}:path to the bin directory of gradle
- Under Windows :
  - Configuration panel -> System -> Advanced parameters -> environment variables:



Installation (3/4)

- Using Gradle behind a proxy:
  - If the network configuration use a proxy, simply put the following property file named gradle.properties (adapt the proxy address and port) along with the file build.gradle (in the same directory) :

```
systemProp.http.proxyHost=proxy.efrei.fr
systemProp.http.proxyPort=3128
systemProp.http.proxyUser=
systemProp.http.proxyPassword=
systemProp.http.nonProxyHosts=*.nonproxyrepos.com|localhost
systemProp.https.proxyHost=proxy.efrei.fr
systemProp.https.proxyPort=3128
systemProp.https.proxyUser=
systemProp.https.proxyPassword=
systemProp.https.nonProxyHosts=*.nonproxyrepos.com|localhost
```

Installation (4/4)

- Test gradle:

```
C:\Users\ben\Documents>gradle
:help

Welcome to Gradle 2.1.

To run a build, run gradle <task> ...

To see a list of available tasks, run gradle tasks

To see a list of command-line options, run gradle --help

BUILD SUCCESSFUL

Total time: 7.175 secs
```

Use Gradle without installation

- Gradle can use a wrapper avoiding any installation
- Just replace the gradle command by gradlew
- The first time this command can take a long time to complete since it downloads and caches Gradle

Java project creation

Java project creation

- Create a directory
- Create two sub directories:
  - src\main\java
  - src\test\java
- Add a file (at the same level than src !) named build.gradle containing (take care if you copy/paste):

```
apply plugin: 'java'
apply plugin: 'eclipse'     // allow to convert a Gradle project into an Eclipse project
repositories {
   mavenCentral()           // indicates where to find Java librairies
}
dependencies {              // list of libraries
   testCompile group: 'junit', name: 'junit', version: '4.+'
}
```

Gradle commands

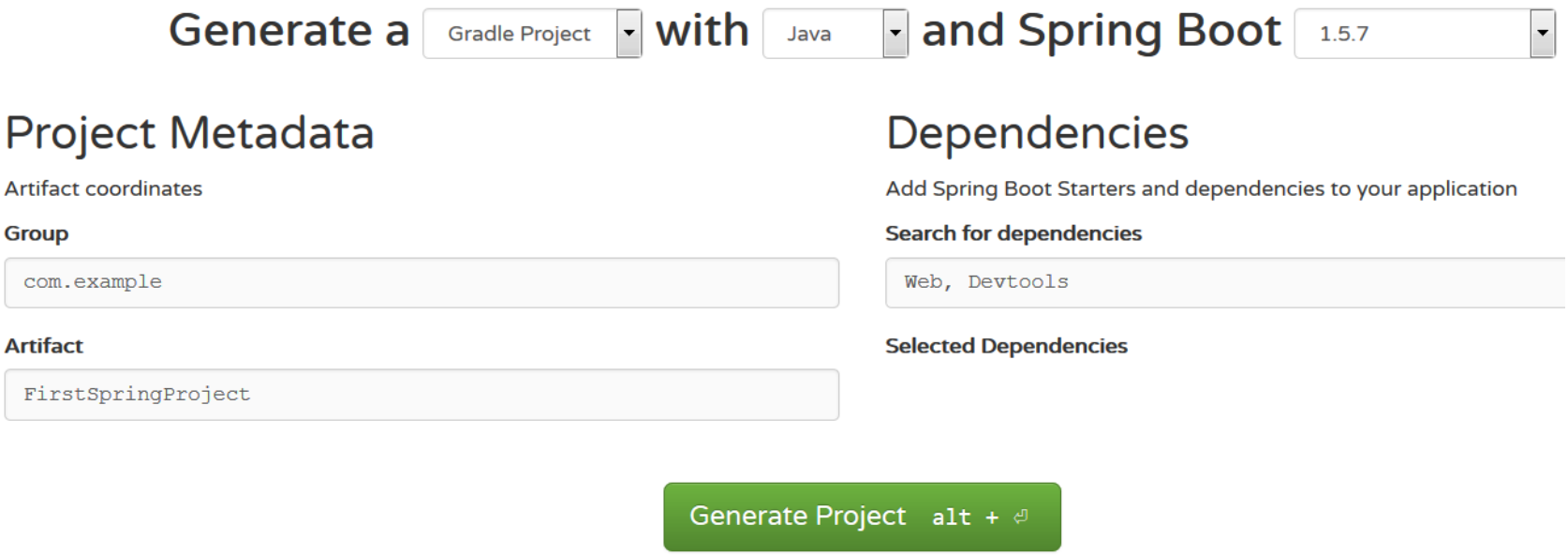
- Open a command line window to use Gradle
- Use the following commands :
  - Resolve dependencies: gradle build
  - Convert into an Eclipse project: gradle eclipse
  - Import the project into Eclipse (File -> Import –> General -> Existing project into workspace)
  - Enjoy coding ...

Java Web project creation

(Using Spring Boot framework)

Spring Initializer

- Creation of a Spring project using Spring Initializer: http://start.spring.io/



- Download and uncompress the zip file
- Check if the web starter and dev tools are present into the file build.gradle:

```
dependencies {
    compile('org.springframework.boot:spring-boot-starter')
    compile('org.springframework.boot:spring-boot-starter-web')
    compile("org.springframework.boot:spring-boot-devtools")
    testCompile('org.springframework.boot:spring-boot-starter-test')
}
```

Spring project

- First build of the project:

```
C:\Windows\system32\cmd.exe

C:\Users\Charroux_std\Documents\tp\SpringBoot\FirstSpringProject\FirstSpringProj
ect>gradlew  build
Starting a Gradle Daemon, 1 incompatible and 1 stopped Daemons could not be reus
```

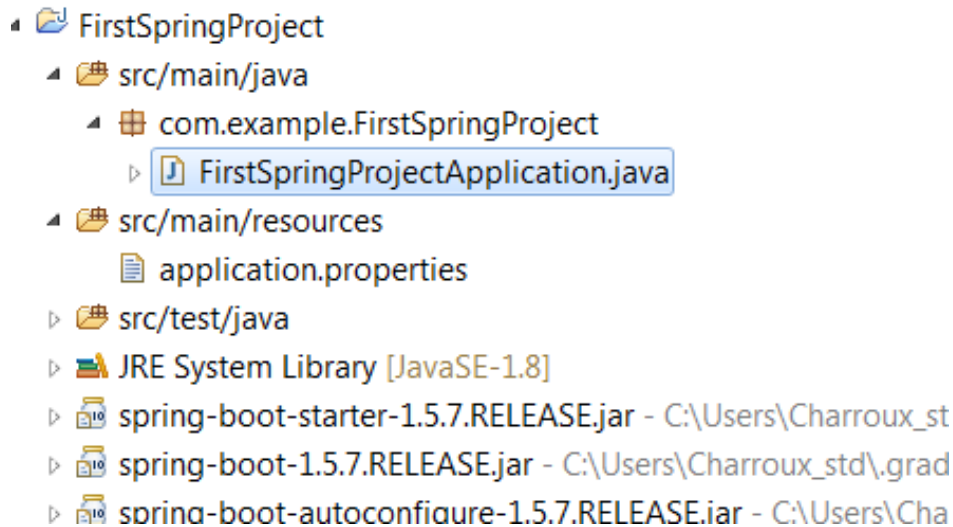
- Can take a long time to complete the first time this command is used:
  - Gradle is downloaded and cached (gradleW)
  - All the Spring libraries are downloaded
- Create an Eclipse project:

```
C:\Windows\system32\cmd.exe
C:\Users\Charroux_std\Documents\tp\SpringBoot\FirstSpringProject\FirstSpringProj
ect>gradlew eclipse
```

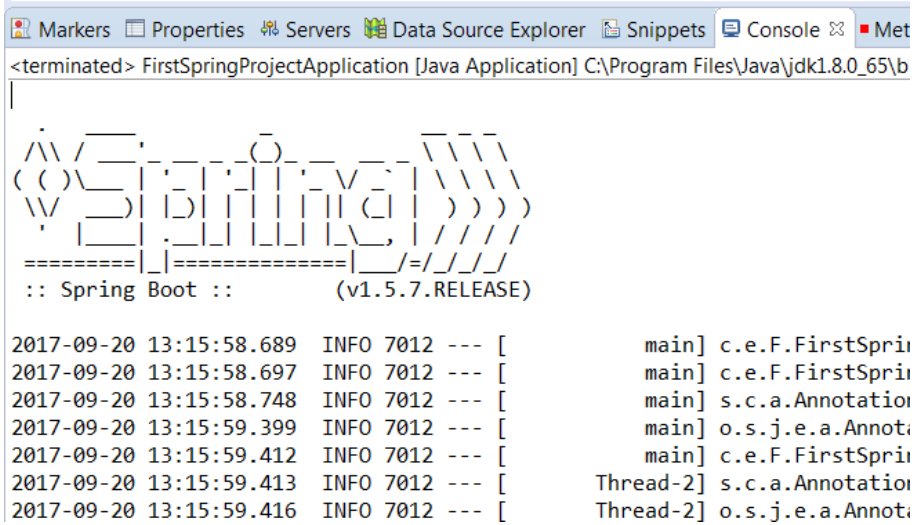
- Import the project into Eclipse (version JEE developer recommended): File -> Import –> Existing project into workspace…

Structure of the project

- The project is composed of a main program



- Launch it as a Java Application:



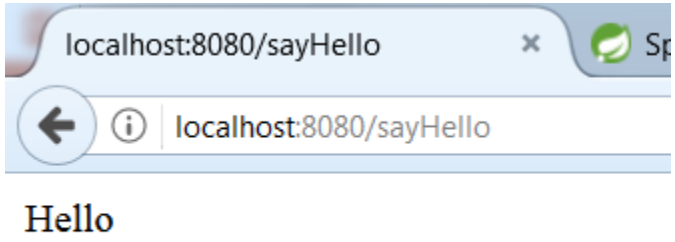
Adding a web content

- Create a sub directory named web and add the following class:

```
package com.example.FirstSpringProject.web;

import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;

@Controller
public class MyFirstController {

        @RequestMapping(value = "/sayHello", method = RequestMethod.GET)
        @ResponseStatus(HttpStatus.OK)
        @ResponseBody
        public String sayHello(){
                return "Hello";
        }

}
```

Start the web server

- Launch again the main program
- Test it in a web browser:



- How does it work ?
  - Spring looks for and start an embedded web container (tomcat by default) into the libraries :