Contents

[develop Grails/groovy via spring cli 1](#_Toc440973442)

[develop Spring/java via sts&gradle 1](#_Toc440973443)

[Knowledge about Gradle: 3](#_Toc440973444)

[Knowledge about Maven: 4](#_Toc440973445)

# develop Grails/groovy via spring cli

1. Download spring-boot-cli.zip
2. Unzip into H:\pilot\tools\spring\spring-boot-cli-1.2.3 and set EVN
3. > spring --version
4. > spring run \*.groovy

Note: if fails, please run the following cmd firstspr

> set JAVA\_OPTS=-Dhttp.proxyHost=161.92.51.225 -Dhttp.proxyPort=8080 -Dhttps.proxyHost=161.92.51.225 -Dhttps.proxyPort=8080

# develop Spring/java via sts&gradle

Updated version for spring boot application:

1. PS> mkdir gs-rest-service/src/main/java
2. 在gs-rest-service项目下创建build.gradle

//设置脚本的运行环境, 借用maven依赖库管理, 依赖包的定义

buildscript {

repositories {

mavenCentral()

}

dependencies {

classpath("org.springframework.boot:spring-boot-gradle-plugin:1.2.3.RELEASE")

}

}

//声明构建项目类型

//指定项目为java项目，项目编译(在项目提示符下执行：gradle build)时生成项目的jar包。

apply plugin: 'java'

// java项目的eclipse开发环境构建.生成所需要的.project,.classpath等文件。

apply plugin: 'eclipse'

apply plugin: 'idea'

apply plugin: 'spring-boot'

//指定jar包名字:gs-rest-service-0.1.0.jar

jar {

baseName = 'gs-rest-service'

version = '0.1.0'

}

//指定仓库使用

repositories {

mavenCentral()

}

sourceCompatibility = 1.7

targetCompatibility = 1.7

//项目依赖定义，compile为编译级别依赖，还有testCompile为测试级别的依赖等

//”group:name:version”

dependencies {

compile("org.springframework.boot:spring-boot-starter-web")

testCompile("junit:junit")

}

task wrapper(type: Wrapper) {

gradleVersion = '2.3'

}

1. Import project directory as gradle project, set source path

Right click -> Properties -> Java Build Path -> Source -> Add Folder: src/main/java

1. RESTful Service

创建Controller

**@RestController**

public class GreetingController {

**@RequestMapping("/greeting")**

public Greeting greeting(@RequestParam(value="name", defaultValue="World") String name) {

return new Greeting(counter.incrementAndGet(),

String.format(template, name));

}

}

创建Configuration

package hello;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

**@SpringBootApplication**

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

1. 启动服务

Right click -> run as -> Spring boot application

1. Deploy app to pivotal cloud foundry

Instance Server

Servers -> right click -> new servers -> Choose Cloud Foundry ->

Email, password, URL

Right click Package Explorer/project -> Configure -> Enable as Cloud Foundry App

Drag and drop project to Server instance (Pivotal Cloud Foundry)

1. Operate app of cloud foundry in Servers View.

App -> Right click -> remove

App -> Right click -> Open Home Page

App -> double click -> check your application’s health as well as scale your application’s memory and instances (in Cloud Foundry server instance editor)

# Knowledge about Gradle:

1. Install Gradle
2. Change Repositories (move user/.gradle to H:\pilot\tools\spring\Repositories\_Gradle\.gradle, then restart computer)
3. Add http proxy for project/gradle.properties

systemProp.http.proxyHost=165.225.96.34

systemProp.http.proxyPort=10015

systemProp.http.nonProxyHosts=\*.nonproxyrepos.com|localhost

systemProp.https.proxyHost=165.225.96.34

systemProp.https.proxyPort=10015

systemProp.https.nonProxyHosts=\*.nonproxyrepos.com|localhost

1. Create build.gradle
2. Add task createPom as follows:

apply plugin: 'maven'

task createPom << {

pom {

project {

groupId 'com.philips.pilot'

artifactId 'stereo-xmlconfig'

version '1.0.0'

inceptionYear '2008'

licenses {

license {

name 'The Apache Software License, Version 2.0'

url 'http://www.apache.org/licenses/LICENSE-2.0.txt'

distribution 'repo'

}

}

}

}.writeTo("pom.xml")

}

1. Convert gradle to maven

Project>> gradle createPom

1. Convert maven to gradle

Project>>gradle init –type java-library

# Knowledge about Maven:

**Maven command:**

// test maven installed or not?

> mvn -v

// compile the project's code, the compiled .class files are put in the target/classes/

maven project> mvn compile

// compile java code, run tests and package the code up in a JAR file in the target/artifactId-version.jar

maven project> mvn package

// install project's JAR to local repository

maven project> mvn install

more maven command:

--mvn help:effective-pom 查看项目的有效POM

--mvn test：运行应用程序中的单元测试

--mvn package：依据项目生成jar文件

--mvn install：在本地Repository中安装jar

--mvn site：生成项目相关信息的网站

--mvn clean：清除目标目录中的生成结果

--mvn eclipse:eclipse：生成Eclipse项目文件

--mvn compile:编译相应的java文件

--mvn test-compile:只编译test

--mvn archetype:create 创建Maven项目

--mvn -Dplugin=groupId:artifactId help:describe 查看一个插件所用的版本

--mvn -Dmaven.test.skip=true 忽略编译test

--mvn install:install-file -Dfile=<path-to-file> -DgroupId=<group-id> -DartifactId=<artifact-id> -Dversion=<version> -Dpackaging=<packaging> -DgeneratePom=true maven的一个插件，安装到本地库

说明："-D<name>=<value>"这种格式不是Maven定义的，它其实是Java用来设置系统属性的方式，可以通过“java -help”查看Java的解释。Maven的bin目录下的脚本文件仅仅是把属性传入Java而已。

**Install Maven and Maven plugin for eclipse:**

1. install JDK and config system environment
2. install Maven and config system environment

check install success or not:

cmd>> mvn -v

1. change Repositories of Maven

open H:\pilot\tools\spring\apache-maven-3.2.5\conf\settings.xml

add <localRepository>H:\pilot\tools\spring\Repositories\Maven</localRepository>

download repository

cmd>> mvn help:system

if download fails, please add proxy into H:\pilot\tools\spring\apache-maven-3.2.5\conf\settings.xml

1. install eclipse plugin for Maven

marketplace: maven

1. config maven for eclipse

eclipse -> preferences -> Maven -> Installations -> Add -> H:\pilot\tools\spring\apache-maven-3.2.5

eclipse -> preferences -> Maven -> User Settings -> Browse -> H:\pilot\tools\spring\apache-maven-3.2.5\conf\settings.xml

**create Maven web project:**

1. new -> Maven project -> maven-archetype-webapp
2. add new source folder:

right click project -> Java Build Path -> Source -> Add Folder: src/main/java, src/test/java, src/test/resources

1. change the output folder of source:

Java Build Path -> Source -> \*/src/test/java -> output folder -> target/test-classes

Java Build Path -> Source -> \*/src/test/resources -> output folder -> target/test-classes

1. change the order and export:

Java Build Path -> Order and Export -> ...

1. add Tomcat Lib

Java Build Path -> Libraries -> Server Runtime -> Apache Tomcat v7.0

(if Apache Tomcat v7.0 doesn't exist, setup as follows:

window -> preferences -> Server -> Runtime Environments -> Add... (your tomcat path)

**create Maven + Spring for JAR file:**

1. new Maven Project -> maven-archetype-quickstart
2. pom.xml:

* for POJO via JAR file

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.philips.spring</groupId>

<artifactId>springPOJO</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.1.4.RELEASE</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-shade-plugin</artifactId>

<version>2.1</version>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>shade</goal>

</goals>

<configuration>

<transformers>

<transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">

<mainClass>com.philips.spring.springPOJO.App</mainClass>

</transformer>

</transformers>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

**Key points:**

The text in red should be changed! the JAR file executable is made via using the [Maven Shade Plugin](http://maven.apache.org/plugins/maven-shade-plugin/),

for detail please refer to https://spring.io/guides/gs/maven/

* for spring restful web service via jar file

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.springframework</groupId>

<artifactId>gs-rest-service</artifactId>

<version>0.1.0</version>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.1.10.RELEASE</version>

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<start-class>com.philips.spring.spring\_rest\_service.App</start-class>

</properties>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

* for spring restful web service via war file

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.springframework</groupId>

<artifactId>gs-rest-service</artifactId>

<version>0.1.0</version>

<packaging>war</packaging>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.1.10.RELEASE</version>

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**Key Points:**

Create maven project via m2e, JRE System Library should be switched to Workspace default JRE

**How to debug and run?**

* via Maven and cmd way:

//package maven project as jar or war

1. maven project> mvn clean package

// run

1. maven project> java -jar target/\*.jar

* via Maven and eclipse:

for JAR:

// clean

1. right click Maven project -> Run as -> Maven clean

// create Maven Build

1. right click Maven project -> Run as -> Maven Builds... ->

Name: package-project-name

Goals: package

debug out, Skip Tests

// package

1. run package-project-name

// debug

1. right click Maven project -> Debug as -> Java Application

Note:

* Maven Build and Java Application could be rename in Debug Configuration -> Maven Build/Java Application
* if there already is Maven Build and Java Application in Debug/Run Configuration, run them directly

for WAR:

//create Maven Build

//library project

Name: install-project-name

Goals: install

debug out, Skip Tests

// web project

Name: run-project-name

Goals: tomcat7:run

Note: first run Maven Build of library project, then run Maven Build of web project