基于Spring Boot的web设计与实现

杨家炜
(江苏科技大学环境工程与化学学院,江苏 镇江 212003)

【摘 要】 论述Spring Boot的产生背景及发展现状,讨论研究的目的与意义,介绍系统要求,使用Spring Boot进行简单Web开发。

【关键词】 Spring Boot;Web开发
【中图分类号】 TP31 【文献识别码】A 【文章编号】2095-3518(2016)07-86-04

Spring Boot是由Pivotal团队提供的全新框架,其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置,从而使开发人员不再需要定义样板化的配置。通过这种方式,Boot致力于在蓬勃发展的快速应用开发领域(rapid application development)成为领导者[1]。

1 背景及研究目的

1.1 背景及发展现状

多年以来,Spring IO平台饱受非议的一点就是大量的XML配置以及复杂的依赖管理。在13年的SpringOne 2GX会议上,Pivotal的CTO Adrian Colyer回应了这些批评,并且特别提到该平台将来的目标之一就是实现免XML配置的开发体验。Boot所实现的功能超出了这个任务的描述,开发人员不仅不再需要编写XML,而且在一些场景中甚至不需要编写繁琐的import语句。在对外公开的beta版本刚刚发布之时,Boot描述了如何使用该框架在140个字符内实现可运行的web应用,从而获得了极大的关注度,该样例发表在Twitter上。

然而,Spring Boot并不是要成为Spring IO平台里面众多“Foundation”层项目的替代者。Spring Boot的目标不在于为已解决的问题域提供新的解决方案,而是为平台带来另一种开发体验,从而简化对这些已有技术的使用。对于已经熟悉Spring生态系统的开发人员来说,Boot是一个很理想的选择,不过对于采用Spring技术的新人来说,Boot提供一种更简洁的方式来使用这些技术。

1.2 研究的目的与意义

Spring Boot被用于创建微服务,属于“微服务框架”的概念。微服务与分布式系统有点类似,但功能更单一。这为云部署创造了条件。云计算属于前沿技术。研究Spring Boot,可以为Web开发和云计算打好基础。

2 开发环境简介

2.1 系统要求

默认情况下,Spring Boot 1.3.0.BUILD-SNAPSHOT需要Java7和Spring框架4.1.3或以上。可以在Java6下使用Spring Boot,不过需要添加额外配置。构建环境明确支持的有Maven(3.2+)和Gradle(1.12+)。

尽管可以在Java6或Java7环境下使用Spring Boot,通常建议如果可能的话就使用Java8。

2.2 开发平台介绍

2.2.1 JDK介绍

JDK是Java语言的软件开发工具包,主要用于移动设备、嵌入式设备上的java应用程序。JDK是整个java开发的核心,它包含了JAVA的运行环境,JAVA工具和JAVA基础的类库[2]。

JDK(Java Development Kit) 是Java语言的软件开发工具包(SDK)。

SE(J2SE),standard edition,标准版,是我们通常用的一个版本,从JDK 5.0开始,改名为Java SE。

EE(J2EE),enterprise edition,企业版,使用这种JDK开发J2EE应用程序,从JDK 5.0开始,改名为Java EE。

ME(J2ME),micro edition,主要用于移动设备、嵌入式设备上的java应用程序,从JDK 5.0开始,改名为Java ME[3]。

没有JDK的话,无法编译Java程序,如果想只运行Java程序,要确保已安装相应的JRE。为了使用Spring Boot,系统安装了jdk1.8。在命令行窗口输入命令java-version进行查看。

```
C:\>java -version
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.91-b14, mixed mode)
```

图1 JDK版本

2.2.2 Maven介绍

Maven是一个项目管理工具,它包含了一个项目对象模型(Project Object Model),一组标准集合,一个项目生命周期(Project Lifecycle),一个依赖管理系统(Dependency Management System),和用来运行定义在生命周期阶段(phase)中插件(plugin)目标(goal)的逻辑。使用Maven的时候,用一个明确定义的项目对象模型来描述项目,然后Maven可以应用横切的逻辑,这些逻辑来自一组共享的(或者自定义的)插件[4]。

Maven有一个生命周期,当运行mvn install的时候被调用。这条命令告诉Maven执行一系列有序的步骤,直到到达指定的生命周期。遍历生命周期旅途中的一个影响就是,Maven运行了许多默认的插件目标,这些目标完成了像编译和创建一个JAR文件这样的工作。

【第一作者】杨家炜(1997-),男,江苏南京人,本科生。

此外，Maven能够很方便地管理项目报告，生成站点，管理JAR文件，等等。

为了使用Spring Boot，系统安装了Apache Maven 3.3.9。在命令行输入命令mvn -v进行查看。

```
C:\>mvn -v
C:\
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-11T00:41:47+08:00)
Maven home: D:\OFC\apache-maven-3.3.9
Java version: 1.8.0_91, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_91\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 7", version: "6.1", arch: "amd64", family: "dos"
```

图2　Maven版本

3　简单Web开发

3.1　项目创建

开始一个新的Maven项目，在命令行使用Maven Archetype插件[5]。

mvn archetype:generate -DgroupId=com.sample -DartifactId=demo -DpackageName=com.sample

```
[INFO] ------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: D:\SpringBoot
[INFO] Parameter: package, Value: com.sample
[INFO] Parameter: groupId, Value: com.sample
[INFO] Parameter: artifactId, Value: demo
[INFO] Parameter: packageName, Value: com.sample
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: D:\SpringBoot\demo
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 21.740 s
[INFO] Finished at: 2016-05-20T00:03:21+08:00
[INFO] Final Memory: 14M/157M
[INFO] ------------------------------------------------------------------------
```

图3　项目搭建成功

archetype:generate称为一个Maven目标 (goal)，它描述了将会在构建中完成的工作单元 (unit of work)。而像-Dname=value这样的对是将会被传到目标中的参数，它使用-D属性这样的形式，类似于通过命令行向Java虚拟机传递系统属性。archetype:generate这个目标通过archetype快速创建一个项目。在这里，一个archetype被定义为“一个原始的模型或者类型，在它之后其它类似的东西与之匹配；一个原型(prototype)”。Maven有许多可用的archetype，从生成一个简单的Swing应用，到一个复杂的Web应用。

现在生成了项目demo，以下是创建的目录结构：

```
D:\SpringBoot>tree
文件夹 PATH 列表
卷序列号为 FEE2-48AC
D:.
└─demo
    └─src
        ├─main
        │  └─java
        │      └─com
        │          └─sample
        └─test
            └─java
                └─com
                    └─sample
```

图4　目录结构图

(1)Maven Archtype插件创建了一个与artifactId匹配的目录——demo。这是项目的基础目录；

(2)项目源码和资源文件放在src/main目录下；

(3)项目测试用例放在src/test目录下；

(4)pom.xml文件放在项目根目录下，这个文件描述了这个项目，配置了插件，声明了依赖。

Maven Archtype插件生成了一个简单的类com.sample.App，是一个仅有13行代码的Java，所做的只是在main方法中输出一行消息：

```
1  package com.sample;
2
3  /**
4   * Hello world!
5   *
6   */
7  public class App
8  {
9      public static void main( String[] args )
10     {
11         System.out.println( "Hello World!" );
12     }
13 }
14
```

图5　类App.java

这时可以构建打包这个应用查看效果。首先在pom.xml目录下运行mvn install来安装项目。然后再命令行运行它，验证这个程序能否正常工作。也可以用来测试Maven是否配置正确。

```
D:\SpringBoot\demo>java -cp target/demo-1.0-SNAPSHOT.jar com.sample.App
Hello World!
```

图6　运行结果

3.2　应用开发

3.2.1　需求分析

开发一个简单的web应用。在再现Pivotal团队发表在Twitter上的经典案例的基础上，能够更进一步，实现以下几点：

(1)使用浏览器打开localhost:8080，显示经典的问候语——Hello World；

(2)从uri中取得给定字段并打印输出；

(3)在get请求中取得请求参数并打印输出。

3.2.2　修改POM

pom.xml是用来构建项目的。要想使用Spring Boot，必须修改POM。打开文本编辑器，添加以下内容：

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.3.0.BUILD-SNAPSHOT</version>
</parent>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

图7　POM文件

spring-boot-starter-parent是一个特殊的starter，它提供了有用的Maven默认设置。同时，它也提供了一个dependency-management节点，这样对于“blessed”依赖可以省略version标记。而且这是一个web项目，所以添加一个spring-boot-starter-web依赖。如果要写Junit的话，junit依赖也是必须的。因为这是Maven自动创建的目录结构，junit依赖已经存在，所以这里不用考虑。

3.2.3　编写代码

因为要开发一个简单的web应用，所以我参考经典案例，直接对生成的App作出如下修改：

```java
package com.sample;

import org.springframework.boot.*;
import org.springframework.boot.autoconfigure.*;
import org.springframework.stereotype.*;
import org.springframework.web.bind.annotation.*;

/**
 * Hello world!
 *
 */
@RestController
@EnableAutoConfiguration
public class App
{
    @RequestMapping("/")
    String home() {
        return "Hello World!";
    }

    @RequestMapping("/{name}")
    String var(@PathVariable String name) {
        return "Hello World, " + name;
    }

    @RequestMapping("/demo")
    String param(@RequestParam("name") String name) {
        return "Hello World, " + name;
    }

    public static void main( String[] args ) throws Exception {
        SpringApplication.run(App.class, args);
    }
}
```

图 8　App.java

（1）@RestController 被称为构造型（stereotype）注解。它为阅读代码的人们提供建议。对于 Spring，该类扮演了一个特殊角色。当处理进来的 web 请求时，Spring 会询问它；

（2）@RequestMapping 注解提供路由信息。它告诉 Spring 任何来自 "/" 路径的 HTTP 请求都应该被映射到 home 方法。@RestController 注解告诉 Spring 以字符串的形式渲染结果，并直接返回给调用者；

（3）@EnableAutoConfiguration 注解告诉 Spring Boot 根据添加的 jar 依赖猜测你想如何配置 Spring。由于 spring-boot-starter-web 添加了 Tomcat 和 Spring MVC，所以 auto-configuration 将假定你正在开发一个 web 应用并相应地对 Spring 进行设置；

（4）@PathVariable 是处理 requet uri 部分（这里指 uri template 中 variable，不含 queryString 部分）的注解；@RequestParam 是处理 request body 部分的注解。都是方法参数绑定常用的注解，只是处理 Request 的内容部分不同；

（5）作为应用程序入口点的 main 方法通过调用 run，将业务委托给了 Spring Boot 的 SpringApplication 类。SpringApplication 将引导应用，启动 Spring，相应地启动被自动配置的 Tomcat web 服务器。我们需要将 App.class 作为参数传递给 run 方法来告诉 SpringApplication 谁是主要的 Spring 组件。为了暴露任何的命令行参数，args 数组也会被传递过去。

3.3　用例描述

3.3.1　默认请求用例描述

[引言]

使用浏览器打开 localhost:8080。

[主事件流]

打开浏览器，输入 uri（localhost:8080），显示“Hello World!”，该用例结束。

[异常事件流]

打开浏览器，输入 uri（localhost:8090），不能打开页面，该用例结束。

3.3.2　URI 取值用例描述

[引言]

使用浏览器打开 localhost:8080/{name}。

[主事件流]

打开浏览器，输入 uri（localhost:8080），显示“Hello World, SpringBoot”，该用例结束。

[异常事件流]

打开浏览器，输入 uri（localhost:8080/demo），返回状态码 400，该用例结束。

3.3.3　请求体取值用例描述

[引言]

使用浏览器打开 localhost:8080/demo?name={name}。

[主事件流]

打开浏览器，输入 uri（localhost:8080/demo?name=SpringBoot），显示“Hello World, SpringBoot”，该用例结束。

[异常事件流]

打开浏览器，输入 uri（localhost:8080/demo），返回状态码 400，该用例结束。

3.4　小结

本章主要介绍了开发工具，开发流程和测试用例。本次开发未使用 IDE 工具，主要使用文本编辑 java 代码和 pom 配置信息。项目的编译、打包、执行都是依靠 Maven Archtype 插件。整个代码就显得整洁清晰，“约定优于配置”的好处得以显现。

4　测试

4.1　启动

在项目根目录下输入 mvn spring-boot:run 来启动应用：

```
D:\SpringBoot\demo>mvn spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building demo 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> spring-boot-maven-plugin:1.3.0.BUILD-SNAPSHOT:run (default-cli) > tes
t-compile @ demo >>>
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ demo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory D:\SpringBoot\demo\src\main\resources

[INFO] skip non existing resourceDirectory D:\SpringBoot\demo\src\main\resources

[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ demo ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ de
mo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory D:\SpringBoot\demo\src\test\resources

[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ demo --
-
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] <<< spring-boot-maven-plugin:1.3.0.BUILD-SNAPSHOT:run (default-cli) < tes
t-compile @ demo <<<
[INFO]
[INFO] --- spring-boot-maven-plugin:1.3.0.BUILD-SNAPSHOT:run (default-cli) @ dem
o ---

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::  (v1.3.0.BUILD-SNAPSHOT)
```

图 9　启动画面

4.2　测试

4.2.1　默认请求测试

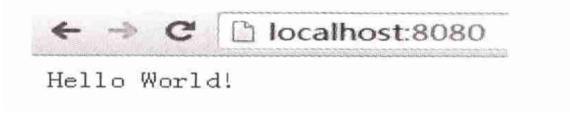

图 10　默认请求测试

例：http://localhost:8080。

异常事件：

例 1：输入端口号不正确时：

图11　输入端口号不正确

Tomcat默认端口号是8080，如果想改成其他端口号，可以修改配置文件。

4.2.2　URI取值测试

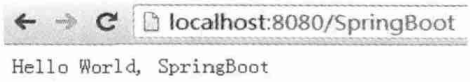

图12　URI取值测试

例：http://localhost:8080/SpringBoot。

异常事件：

例1：输入http://localhost:8080/demo时：

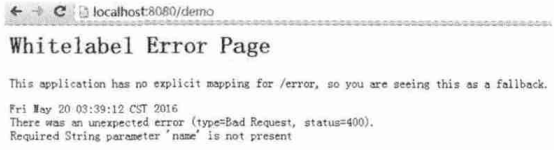

图13　demo

当输入http://localhost:8080/demo时，它会匹配入口param (String name)，但是没有请求参数，所以返回状态码400。

4.2.3　请求体取值测试

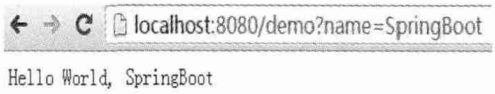

图14　请求体取值测试

例：http://localhost:8080/demo?name=SpringBoot。

异常事件：

例1：输入http://localhost:8080/demo时：同2。

4.3　小结

本章主要介绍了如何启动服务和测试。因为开发比较简单，测试没有什么问题。启动的话，也可以创建一个完全自包含的可执行jar文件，然后使用java -jar命令来运行应用程序。

5　结论

在开发初期，查阅了大量相关资料，相较于其它框架，Spring Boot虽然比较新，但是上手难度不大。

从总体上看，该应用简单实现了请求响应功能。但是，功能非常简单，也没有数据库操作。但是，万事开头难，现在已经成功开发了一个web项目。以后我会继续提升自己的能力，努力完善它的功能。

参考文献

[1]深入学习微框架：Spring Boot.http://www.infoq.com/cn/articles/micro-frameworks1-spring-boot/.

[2]JAVA8十大新特性详解.脚本之家.2014-03-21.http://www.jb51.net/article/48304.htm.

[3]What´s New in JDK 8.oracle.http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html.

[4]项目构建工具Maven开源社区网.http://www.oschina.net/p/maven

[5]Sonatype.Maven权威指南[A].2008.

（上接第33页）

3　产品化设计

3.1　配件清单

试剂A瓶为丙酮5ml；试剂B瓶为0.05%溴酚蓝；一次性注射器；0.45 μm 滤膜；反应管。

3.2　样品取样

胶囊剂取1粒内容物；片剂，取1片；丸剂和散剂，取1/2服用量；液体制剂，蒸干。

3.3 操作步骤

A.将样品加入试剂A瓶中，盖紧瓶盖，大力振摇约1分钟，静置约30s；

B.取1支注射器，拔出活塞，接滤膜，倒入试剂A上层溶液约2ml，用活塞挤压，收集约1ml至反应管中；

C.滴加试剂B滴，轻摇，观察是否产生蓝色溶液。

3.4　结果判断

A.溶液为蓝色，判为阳性；B.溶液为其他颜色，判为阴性；C.溶液初为蓝色，轻摇后蓝色消失，应加大样品量重新检测。

4　讨论

（1）本文对样品提取溶剂进行了优化选择，选择没有干扰的丙酮作为提取溶剂。

（2）本文方法准确方便，可以快速检测出抗风湿类健康产品中是否添加双氯芬酸钠成分。

（3）对方法进行了产品化设计，操作性强，可以为量产提供参考。

参考文献

[1]王仕平，黄炳泉，刘卿，等.抗风湿类中成药及保健食品中非法添加双氯芬酸钠与尼美舒利的快速筛查[J].中国药事，2010，(12)：1207-1209.

[2]蓝献泉，黄义纯，黄红雯，等.抗风湿类中成药及保健品中非法添加双氯芬酸钠快速筛查研究[J].中南药学，2014，(9)：902-905.