1 spring-boot 入门

1.1 srping-boot 简介

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，Spring 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。

## 2 Spring Boot特点

创建独立的Spring应用程序

嵌入的Tomcat，无需部署WAR文件

简化Maven配置

自动配置Spring，没有冗余代码生成和XML配置的要求

1.3 系统要求：

Java 7及以上

Spring Framework 4.1.5及以上

1.4 使用Maven构建项目

1.4.1通过eclipse工具创建Maven Project项目

### 1.4.2引入spring-boot jar包依赖（注：为了简化依赖图，Boot的功能是模块化的，通过导入Boot所谓的“starter”模块，可以将许多的依赖添加到工程之中。为了更容易地管理依赖版本和使用默认配置，框架提供了一个parent POM，工程可以继承它。）

## Spring Boot工程的样例POM文件定义如程序如下所示：

## <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..nf.helloword</groupId>

<artifactId>helloword</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>helloword</name>

<parent>

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

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

<version>1.5.2.RELEASE</version>

<relativePath/>

</parent>

<properties>

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

<java.version>1.7</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

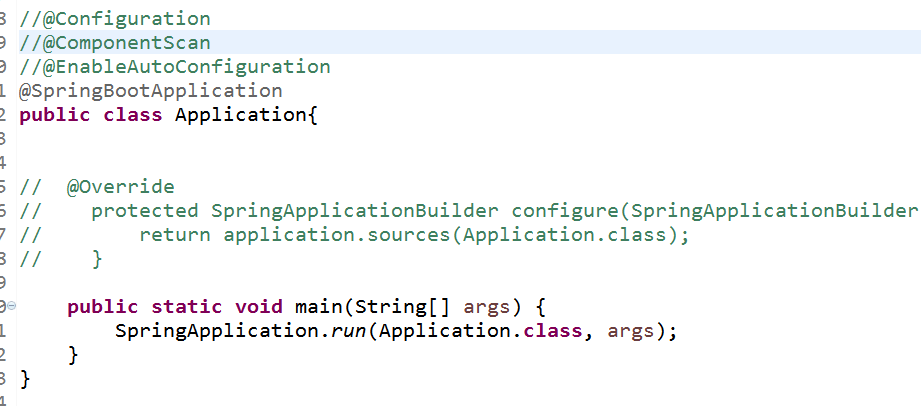
</dependencies>

</project>

### 1.4.3编写HelloWorld服务

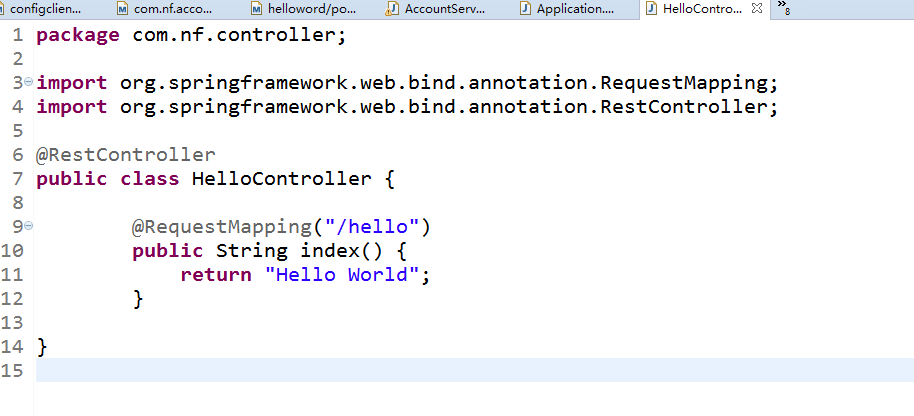
创建package命名为com.nf（根据实际情况修改）

创建Application.java主程序启动类，内容如下



创建package命名为com.nf.controller（根据实际情况修改）

创建HelloController类，内容如下



启动主程序，打开浏览器访问http://localhost:8080/hello，可以看到页面输出Hello World

@Controller：修饰class，用来创建处理http请求的对象

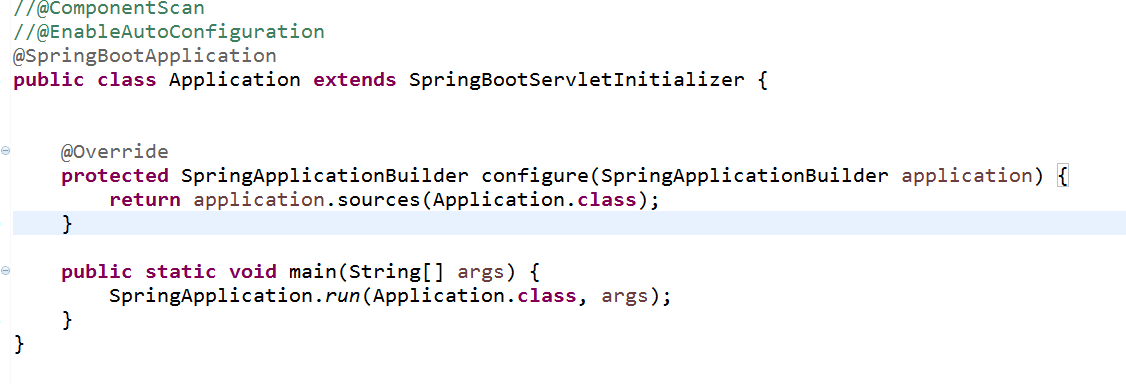
@RestController：Spring4之后加入的注解，原来在@Controller中返回json需要@ResponseBody来配合，如果直接用@RestController替代@Controller就不需要再配置@ResponseBody，默认返回json格式。

@RequestMapping：配置url映射

2 [spring-boot项目在外部tomcat环境下部署](http://blog.csdn.net/james_wade63/article/details/51009423)

[spring](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "http://blog.csdn.net/james_wade63/article/details/_blank)-boot默认提供内嵌的tomcat，所以打包直接生成jar包，用[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "http://blog.csdn.net/james_wade63/article/details/_blank) -jar命令就可以启动。但是，有时候我们更希望一个tomcat来管理多个项目，这种情况下就需要项目是war格式的包而不是jar格式的包。spring-boot同样提供了解决方案，只需要简单的几步更改就可以了，这里提供maven项目的解决方法：

2.1将项目的启动类Application.java继承SpringBootServletInitializer,并重写configure方法，如下截图：



2.2在pom.xml文件中，project下面增加package标签

<packaging>war</packaging>

2.3.还是在pom.xml文件中，dependencies下面添加

<dependency>

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

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

<!-- 打包部署到tomcat上面时，不需要打包tmocat相关的jar包，否则会引起jar包冲突 -->

<exclusions>

<exclusion>

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

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

</exclusion>

</exclusions>

</dependency>

<dependency>

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

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

<scope>provided</scope>

</dependency>

2.4还是在pom.xml文件中，build下面添加

<build>

<plugins>

<plugin>

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

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

</plugin>

<plugin>

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

<artifactId>maven-war-plugin</artifactId>

<configuration>

**<warName>springboot</warName>**

</configuration>

</plugin>

</plugins>

</build>

这样，只需要以上3步就可以打包成war包，并且部署到tomcat中了。需要注意的是这样部署的request url需要在端口后加上项目的名字才能正常访问,项目的名字即<warName>springboot</warName>。spring-boot更加强大的一点，就是：即便项目是以上配置，依然可以用内嵌的tomcat来调试。

按上面的例子，启动主程序，打开浏览器访问http://localhost:8080/springboot/hello，可以看到页面输出Hello World

# 3 Spring Boot中使用Swagger2构建强大的RESTful API文档

Swagger2可以轻松的整合到Spring Boot中，并与Spring MVC程序配合组织出强大RESTful API文档。它既可以减少我们创建文档的工作量，同时说明内容又整合入实现代码中，让维护文档和修改代码整合为一体，可以让我们在修改代码逻辑的同时方便的修改文档说明。另外Swagger2也提供了强大的页面测试功能来调试每个RESTful API。下面来具体介绍，如果在Spring Boot中使用Swagger2

3.1添加Swagger2依赖

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.2.2</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

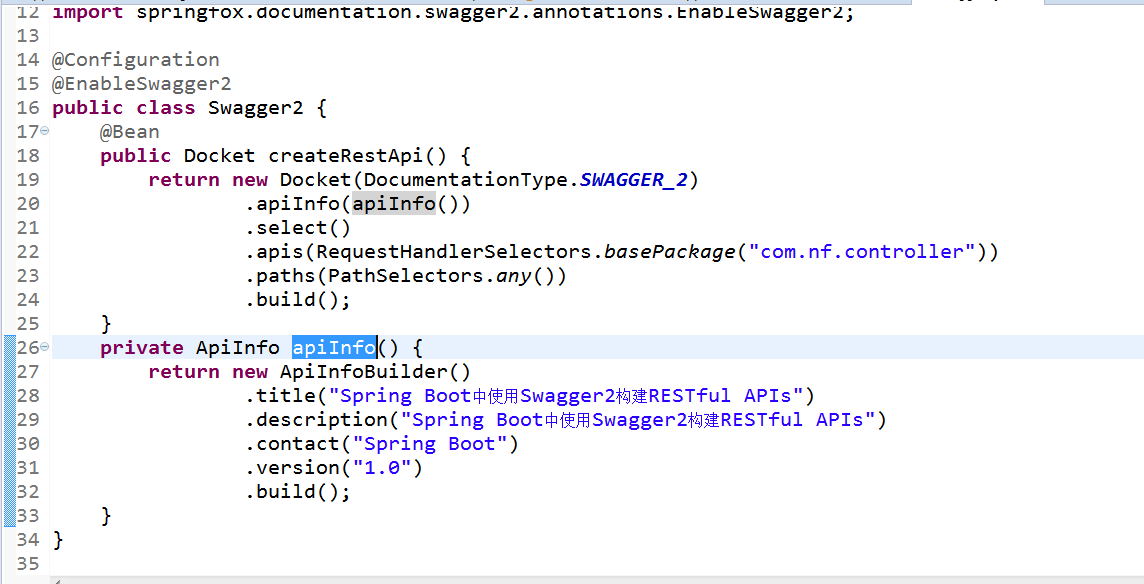
<artifactId>springfox-swagger-ui</artifactId>

<version>2.2.2</version>

</dependency>

3.2创建Swagger2配置类

在Application.java同级创建Swagger2的配置类Swagger2。代码如下图所示：



如上代码所示，通过@Configuration注解，让Spring来加载该类配置。再通过@EnableSwagger2注解来启用Swagger2。

再通过createRestApi函数创建Docket的Bean之后，apiInfo()用来创建该Api的基本信息（这些基本信息会展现在文档页面中）。select()函数返回一个ApiSelectorBuilder实例用来控制哪些接口暴露给Swagger来展现，本例采用指定扫描的包路径来定义，Swagger会扫描该包下所有Controller定义的API，并产生文档内容（除了被@ApiIgnore指定的请求）。

3.3 添加文档内容

在完成了上述配置后，其实已经可以生产文档内容，但是这样的文档主要针对请求本身，而描述主要来源于函数等命名产生，对用户并不友好，我们通常需要自己增加一些说明来丰富文档内容。如下所示，我们通过@ApiOperation注解来给API增加说明、通过@ApiImplicitParams、@ApiImplicitParam注解来给参数增加说明。

完成上述代码添加上，启动Spring Boot程序，访问：[http://localhost:8080/swagger-ui.html](http://localhost:8080/swagger-ui.html" \t "http://blog.didispace.com/springbootswagger2/_blank)  
。就能看到前文所展示的RESTful API的页面。

