2017年1月21日 星期六

Spring Boot 入门教程

author：hushuang
email ： hd1611756908@gmail.com
adress：www.dajiangdahe.com

一.环境部署

1.Java 版本：Java version: 1.8.0_101

2.maven版本：Apache Maven 3.3.9

3.Spring Boot的版本号：1.4.3.RELEASE

4.IDE：采用IntelliJ IDEA

注意：maven 的setting.xml最好是配置上阿里云的maven仓库镜像，要不然第一次构建SpringBoot项目可能会等到地老天荒，当然你要是在公司里面有本地私服，那就无所谓了。阿里云maven镜像，百度一下就知道了，配置到maven配置文件mirrors标签中。

二.环境解读：为什么要采用这样的开发环境?有什么好处?

1.根据官网上面的一句话：做什么事情要有理有据，官网强调了这些环境，我们就要部署好这样的环境，并且SpringBoot版本之间差异化比较大，所以在项目开发中一定要对环境和版本进行统一。

System Requirements

By default, Spring Boot 1.4.3.RELEASE requires Java 7 and Spring Framework 4.3.5.RELEASE or above. You can use Spring Boot with Java 6 with some additional configuration. See Section 81.11, “How to use Java 6” for more details. Explicit build support is provided for Maven (3.2+) and Gradle (1.12 or 2.x). Support for Gradle 2.8 and earlier is deprecated. Gradle 3 is not supported.

上面的大概意思就是Spring Boot 1.4.3.RELEASE版本的需要Java 7和Spring Framework 4.3.5.RELEASE 以上的版本支持。当然你要是用Java6那还需要另外配置，看81.11使用Java6 配置的详情。明确支持构建工具为maven 3.2版本以上，Gradle 1.2或者是Gradle2.x版本，其余的早期版本为过时的版本，已经不在支持，并且Gradle 3 版本是不支持的，版本过高。

2.为什么采用IDEA这个集成开发工具：以前我也是不太愿意用这个集成开发工具的，因为用eclipse习惯了，熟悉快捷键，总之用着很顺手，但是现在eclipse的更新速度已经不能赶上Spring家族的开发步伐，并且新版本的eclipse又莫名其妙的bug，我用过的最稳定版本的eclispe是luna版本，但是现在用它开发SpringBoot已经不合适了，不支持yml文件。而IDEA恰好更新速度特别快，并且集成了最新的Spring插件，在git上支持的也特别好，操作简单，于是便舍弃了eclipse，原来搞开发以为集成开发工具就是那么回事，用什么开发都一样，但是随着越来越新的技术的发展才发现一个好的开发工具也越来越重要。

三.说说SpringBoot是什么?

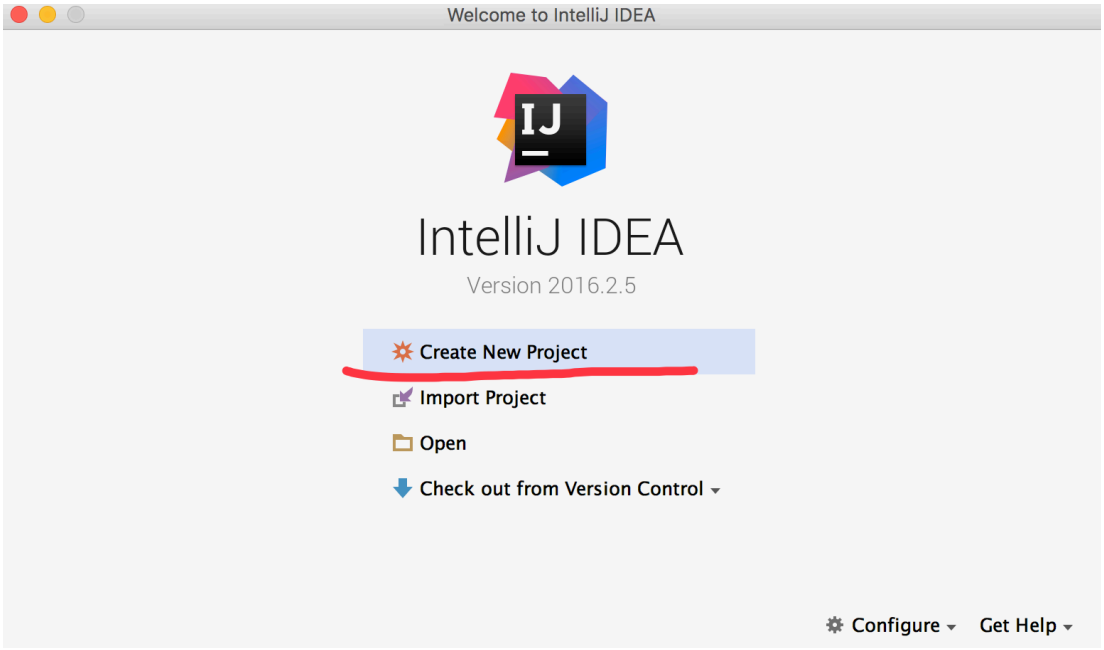
官网上是这么解释的：SpringBoot可以很容易的创建独立的，以生产为基础的应用程序，并且以Spring平台和第三方库作为基础开箱即用。就可以简单的进行开发操作，只需要很少的配置，你可以使用SpringBoot创建java应用并使用java -jar启动它，或者采用传统的war部署方式，我们也提供了一个运行spring脚本的命令行工具。

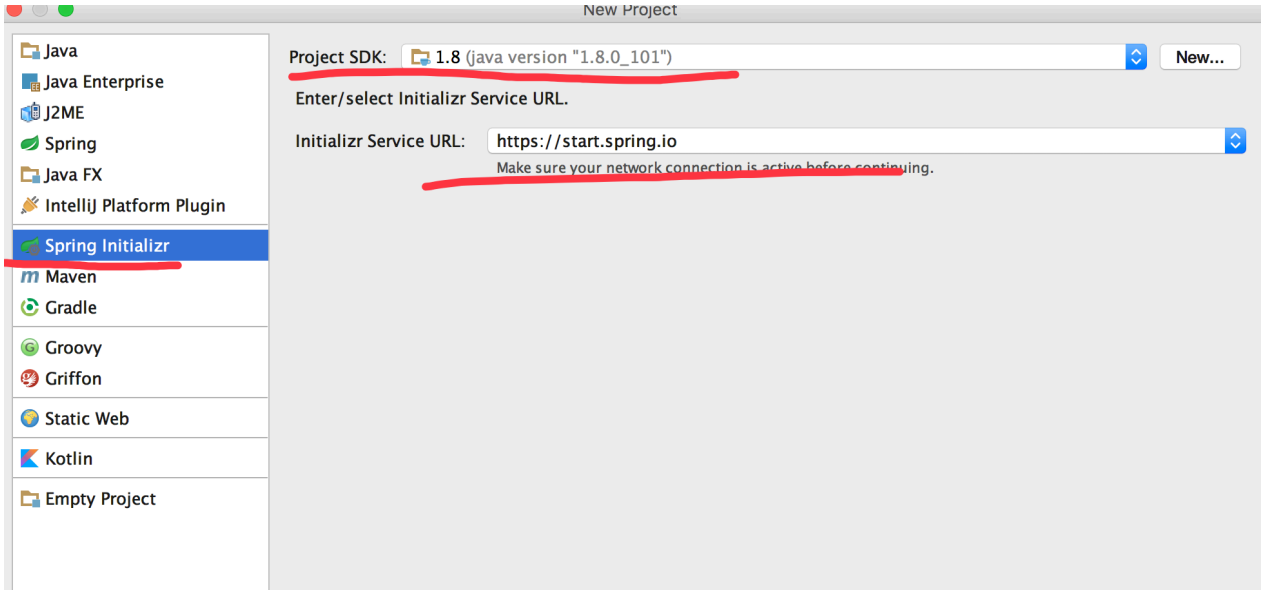
SpringBoot可以以 java -jar xxx.jar来启动项目，许多传统的应用还需要war部署，并且SpringBoot还提供了命令行工具来运行，在下面我会说启动SpringBoot项目的三种方式。

四.构建SpringBoot入门的HelloWorld项目，构建一个web项目入门

1.用IDEA创建项目的步骤：

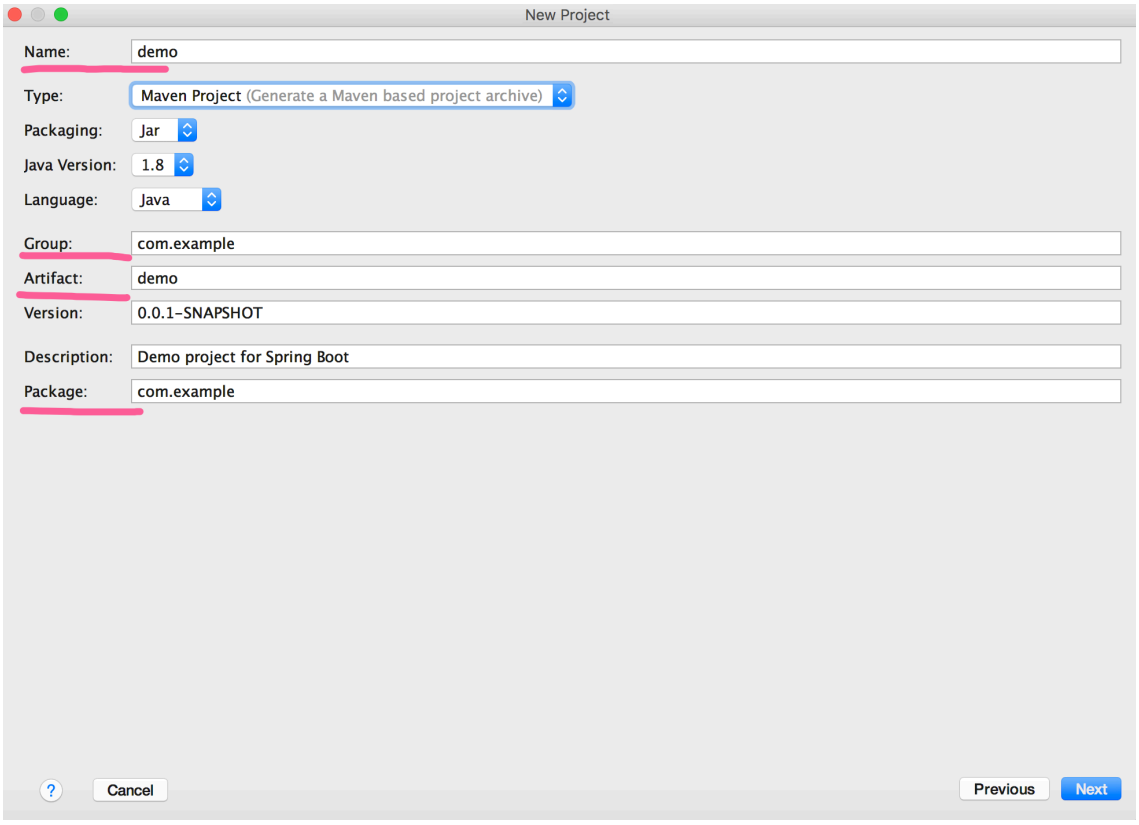
①第一步：打开IDEA,点击Create New Project



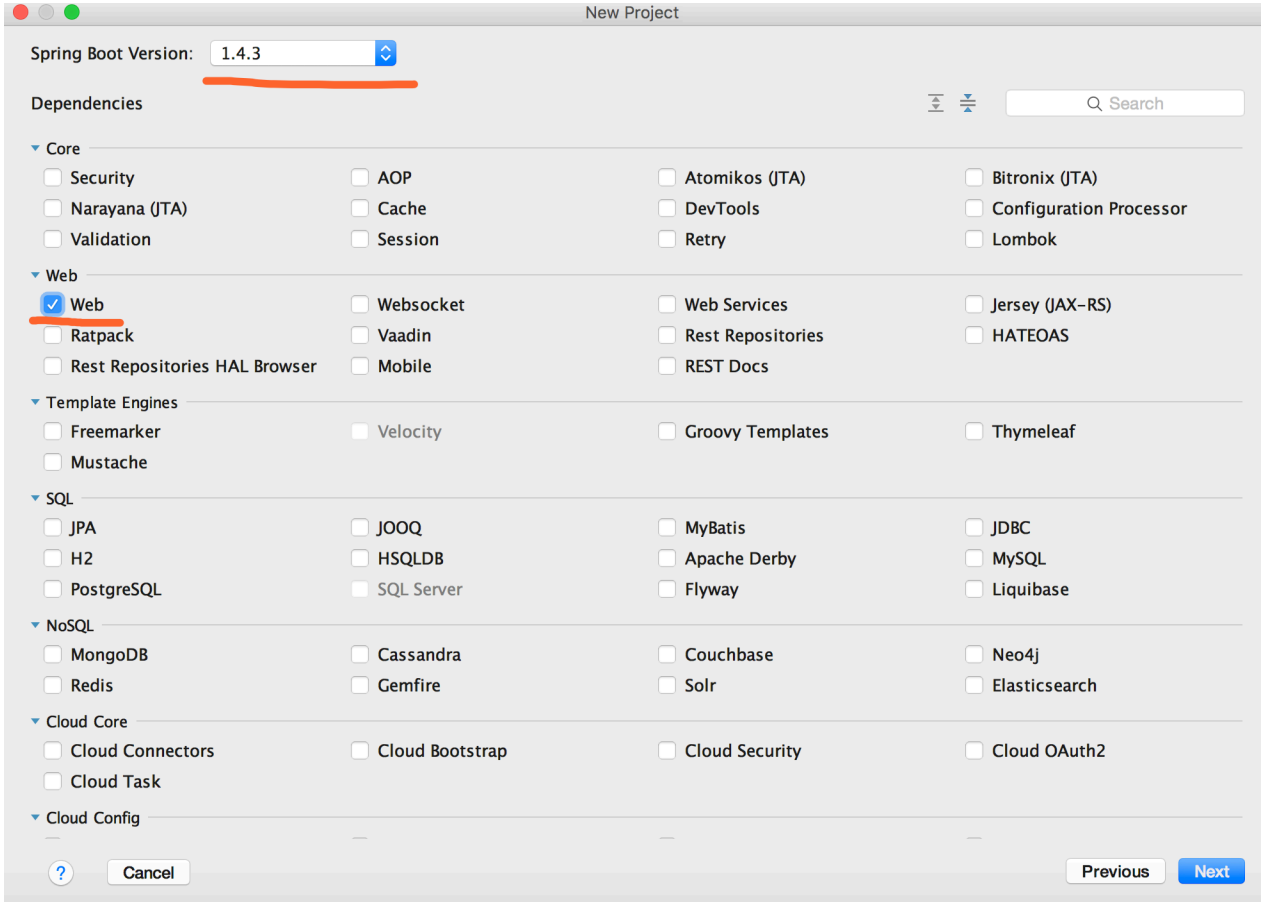


②第二部：选择Spring Initializr 并且确定自己的jdk版本，和 Initializr Service URL:地址，一定要为这个地址 https://start.spring.io ，(一定要连接外网) 确定好，点Next

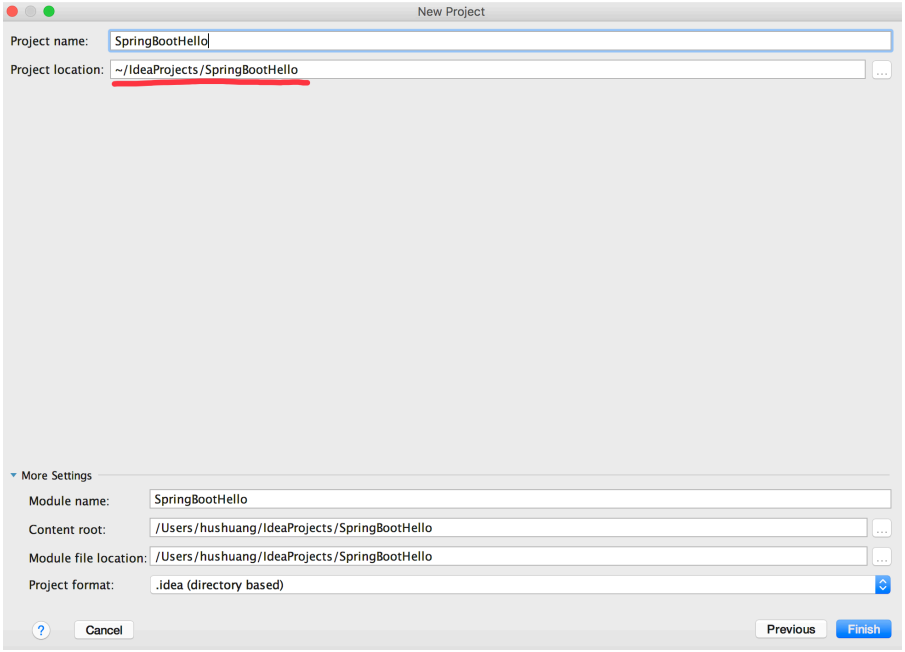
③第三部：name:项目名称可以自定义；Group,Artifact,package这些都是自定义，其余的采用默认值。



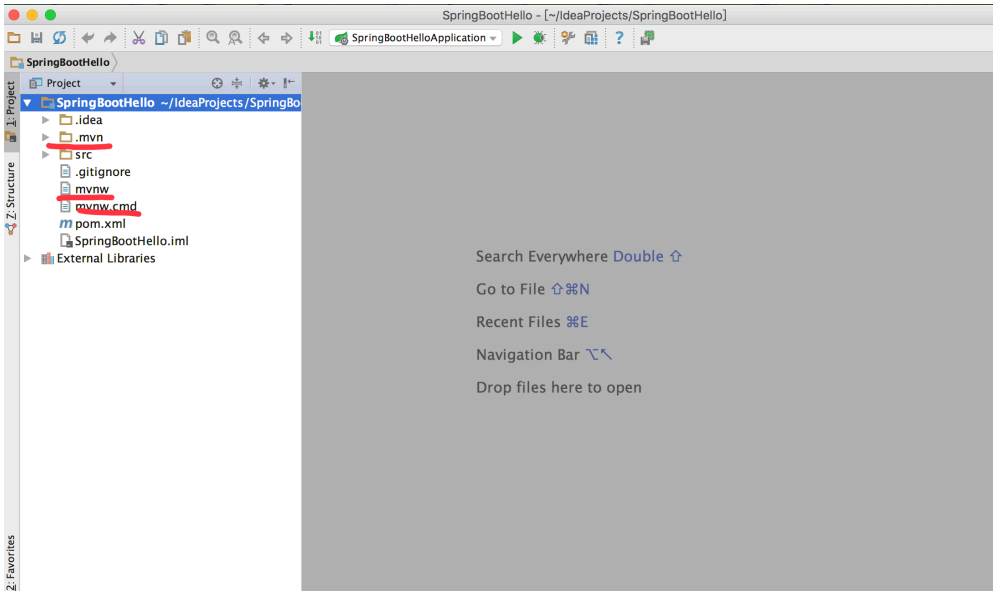
④第四部：确定SpringBoot版本，勾选Web下的Web选项。然后Next



⑤第五步：这一步没什么说的，只要项目路径没有中文，空格就可以，然后finish



⑥第六步：然后就等待吧，等待创建完成，第一次创建如果你不是在公司的内部私服或者自己没配置阿里云的maven镜像，那你就可以该忙什么就忙什么去了，时间会特别长。他会自己下载很多依赖的jar,比如Spring的jar。所以pom里面是不需要再加入Spring框架的依赖的。项目创建好之后，删除没有用的选项，如图。



以上步骤完成，一个简单的SpringBoot项目就创建成功了。

点开pom.xml文件，做一个简单介绍：

※里面有一个父依赖，这个就不解释了，有兴趣的可以看看，里面初始化很多基础的配置

```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.4.3.RELEASE</version>
        <relativePath/>
</parent>
```

※还有个关于web项目的依赖，这个是开发web项目用的依赖，就是咱们创建项目时候勾选的那个web选项。SpringBoot他会自动生成所需要的依赖。当然你要是自己构建项目不采用IDEA这种创建模式，用maven创建也是可以的，只要将这些依赖加入即可。

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

※这个是SpringBoot的测试依赖

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
</dependency>
```

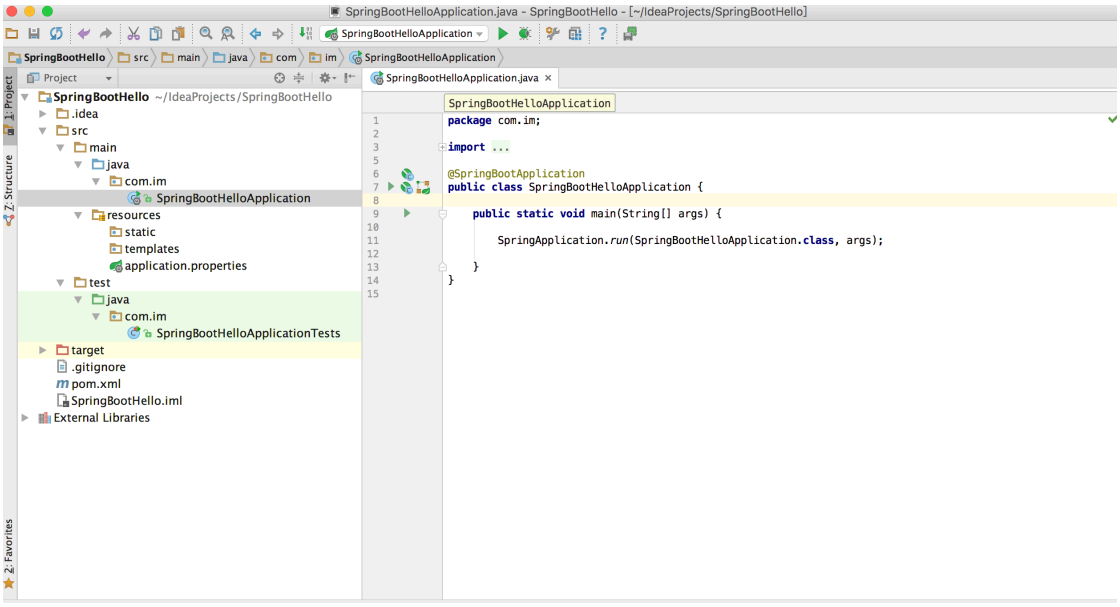
※有一个特别重要的插件

```
<plugins>
        <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
</plugins>
```

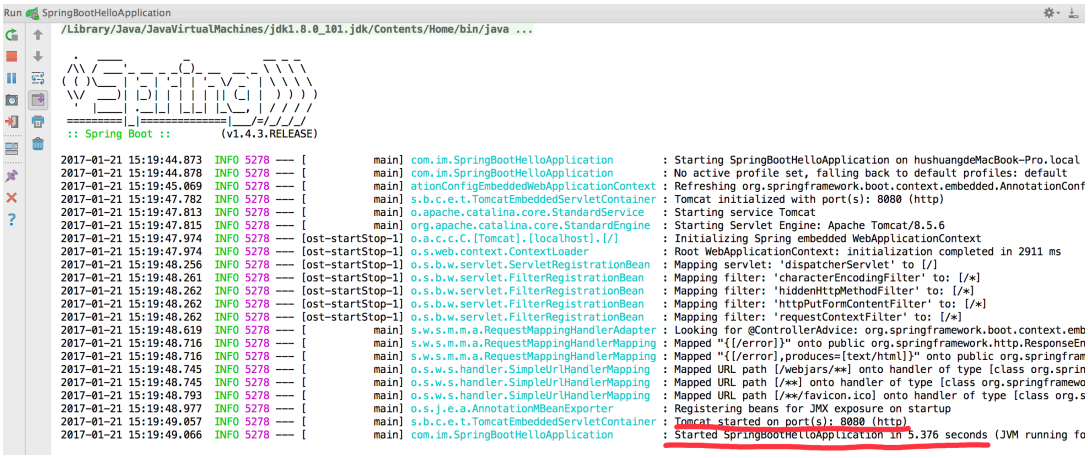
这个插件的作用就是为jar项目配置执行入口。如果没有这个插件那么你的项目运行会报错，因为找不到入口程序。如果你要是不知道什么是**可执行jar**那么请先学习maven。

特别注意：创建项目时一定要连接外网，否则会创建失败。

五.项目创建成功了之后是这样的，由于你们自定义的包名和项目名和我的不一样，略有差别。找到main方法，右键运行main方法。这样项目就运行起来了。



运行起来的状态，内部启动的是tomcat，默认端口号是8080



通过浏览器访问：

http://localhost:8080 会报错，因为我们什么都没写，下面我们开始写第一个入门HelloWorld

六.写第一个入门的HelloWorld方法。

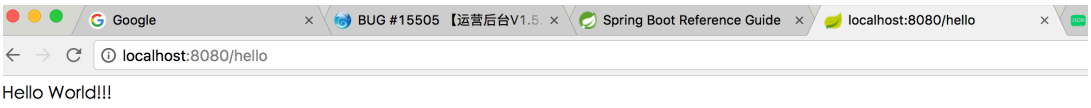
我就直接在带main方法这个类中写了，不在创建一个新类，但是在生产上我们一定会根据业务需求来创建一个个的Controller类,比如新建一个UserController类，在里面加一个login方法。

我这由于只是写一个入门程序，所以就简答的在这个类里面加一个helloworld方法。

```
@RequestMapping(value = "/hello",method = RequestMethod.GET)
public String helloworld(){
        return "Hello World!!!";
}
```

加了这个方法之后还需要在这个类上加一个注解 @RestController

然后重启项目 在浏览器中输入： http://localhost:8080/hello



这样第一个SpringBoot 入门HelloWorld就创建成功了。

下面我说说启动SpringBoot项目的三种方式：

第一种我们已经用了，就是main方法启动。

第二种命令启动：mvn spring-boot:run 在项目的pom文件目录下。

第三种方式：java -jar xxx.jar方式，首先编译一下项目生成jar文件，然后进入jar文件所在的文件目录通过命令行执行 java -jar xxx.jar

以上就是SpringBoot1.4.3.RELEASE 版本的入门教程。

说一下这个项目中用到的注解

@SpringBootApplication

官网的解释：一个顶三个，当然你可以用那三个，我测试过，可以启动成功。

18. Using the @SpringBootApplication annotation

Many Spring Boot developers always have their main class annotated with `@Configuration`, `@EnableAutoConfiguration` and `@ComponentScan`. Since these annotations are so frequently used together (especially if you follow the best practices above), Spring Boot provides a convenient `@SpringBootApplication` alternative.

The `@SpringBootApplication` annotation is equivalent to using `@Configuration`, `@EnableAutoConfiguration` and `@ComponentScan` with their default attributes:

```
package com.example.myproject;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication // same as @Configuration @EnableAutoConfiguration @ComponentScan
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}
```

一个顶三个那这三个注解都是什么意思呢?

首先我们不使用@SpringBootApplication注解，先换成那三个注解运行试一下，发现运行正常。

接下来我们先注视掉其中的一个@EnableAutoConfiguration再启动试试，发现启动报错，Unable to start embedded container无法启动嵌入容器，官网解释：这个注解的意思就是会根据你的pom文件中添加的像spring-boot-starter-web类型的依赖来猜测你建的这个项目是什么类型的项目，根据这个依赖他会认为你建的是一个web项目，就会初始化一些关于web项目的配置，当启动项目的时候，将web环境初始化好。

@Configuration ：

官网说明：配置类，项目中我们可以通过加载xml的形式添加配置，但是SpringBoot推荐使用@Configuration来配置一个类来加载配置（比如我需要集成数据库连接池Druid，而SpringBoot内置的DBCP，我需要将其替换掉，如果不是用SpringBoot项框架我们直接建一个XML文件然后配置就可以了，但是在SpringBoot中，官方推荐使用java类的方式配置，在类上加这个注解@Configuration，并且通过一些注解来替换掉SpringBoot原来自带的数据库连接池）。

@ComponentScan:

意思是如果你的主类是在root包，就是根报，那么你就不需要使用这个注解，如果你的类不是根/root包，那么就需要进行扫描。

@RestController :这是一个组合注解，如果你要是熟悉SpringMVC的话，那么你就会很好理解 这个注解可以分成@Controller and @ResponseBody 给前端返回 Json 对象

提示：SpringBoot其实在web表现上就是遵循的SpringMVC，所以SpringMVC的注解他都是可以用的只是SpringBoot提供了更简单易用的注解方式。还有就是这个入门程序只是简单的像前端发送了一段json字符串，没有视图渲染，SpringBoot也是支持视图渲染的，只是还需要加一个其他的视图模板依赖，此入门教程不再赘述，以后继续会说到。其实这个视图渲染的模板功能没有太大的意义，因为现在的大多数的项目都是采用前后端分离的，前端的静态页面都会放在CDN上，或者Apache或者Nginx上，很少有大中型的项目还会将页面和服务放在一起的。其实这个教程主要是介绍一下SpringBoot的一些使用方式和注意事项，也算不上入门，等有时间我在写一个增删改查来入门。

SpringBoot入门之增删改查

一.SpringBoot的配置管理

在说增删改查之前一定要说说SpringBoot的配置管理，SpringBoot框架支持两种配置方式，一种是.properties文件形式，就是我们用IDEA创建上面项目时候在resources目录下生成的那个。另一种是.yml文件形式，这也是我为什么不用eclipse其中的一个原因，因为eclipse不支持.yml文件。而且SpringBoot大多数都是采用yml文件配置。

二.集成spring-data-jpa 和 配置mysql数据库

1.配置SpringBoot环境，还是以上面的项目为基础，因为要测试增删改查，所以得用上数据库，这里我用spring-data-jpa + hibernste作为持久化框架和SpringBoot整合，数据库采用mysql数据库。所以我们要两个依赖一个是mysql数据库驱动依赖，还有一个就是spring-data-jpa依赖，这里不用加入Hibernate依赖，因为加上spring-data-jpa依赖之后jpa会根据自身的依赖关系加载上相关依赖。

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
</dependency>
```

为什么没有版本号呢？因为SpringBoot的终极目标是开箱即用，所以很多依赖的版本他都已经固定了，因为咱们在建项目的时候pom里面生成的一个父pom

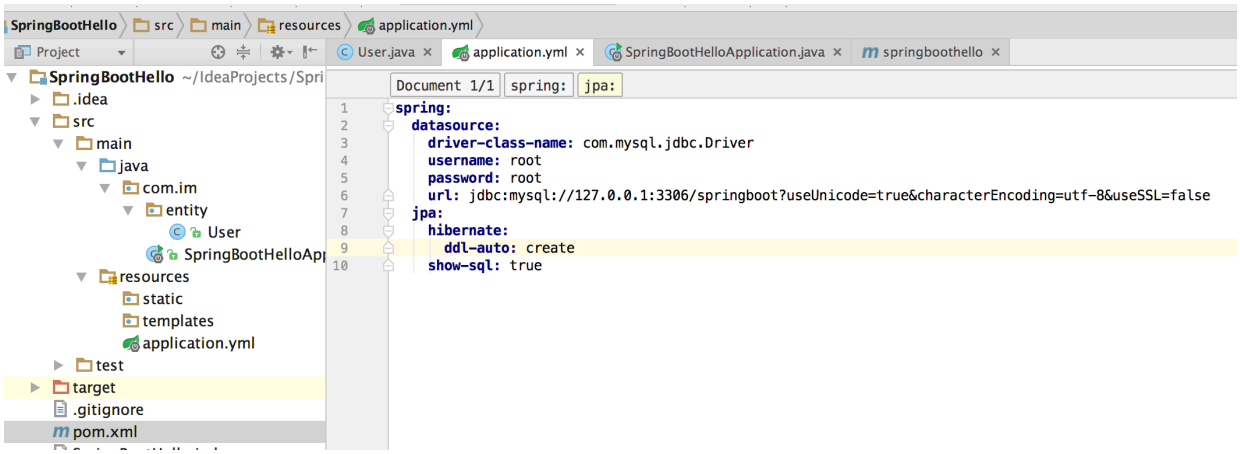
```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.4.3.RELEASE</version>
        <relativePath/>
</parent>
```

所以这两个依赖不需要版本号，最好不要加上一些高版本的驱动，最后要是不兼容，报一些解决不了的错，比较麻烦。如果非要使用高版本的，那么请将SpringBoot版本也提高，自然就集成了其他高版本的依赖。

2.配置连接数据库

将项目自动生成的application.properties配置文件改一下后缀application.yml，并且加上以下的配置信息，这只是简单的配置，因为咱们的重点不在这，下面的配置信息，相信不用我来说，大家都能看懂吧。不要忘记自己首先建一个数据库，我的数据库名称是springboot，hibernate可以自动创建表，但是不能自动创建数据库。

```
spring:
  datasource:
    driver-class-name: com.mysql.jdbc.Driver
    username: root
    password: root
    url: jdbc:mysql://127.0.0.1:3306/springboot?useUnicode=true&characterEncoding=utf-8&useSSL=false
  jpa:
    hibernate:
      ddl-auto: create
    show-sql: true
```



以上步骤完成，然后从main方法重启项目，项目还是可以正确运行起来的，不过会报错。接下来我们开始创建一个类，然后运行项目，让JPA自动给我们创建一个表。

三.创建关联数据库表的类

创建一个User类用于自动创建数据库表，所以必须要给User类加上标志注解，好让JPA识别这个类是数据库映射实体类。

注解的包路径；注意：不要导错包

```
javax.persistence.Entity;                //映射数据库实体类注解
javax.persistence.GeneratedValue;    //主键自增的注解
javax.persistence.Id;                    //设置Id主键为主键的注解
```

```
@Entity
public class User {
        @Id
        @GeneratedValue
        private Integer id;
        private String name;
        //… …
        // get/set

}
```

实体类建好了之后在启动项目，如果没有报错，项目启动成功，那么进去数据库查看，是不是user表已经建好，如果建好了，那么开始就说明成功了。

四.增删改查操作

在进行增删改查之前首先我们得准备一个工具类，用于发送和接受请求。postman
为什么要用这个工具，因为我们没有前端的页面渲染，没有创建表单用来提交数据，所以我们要有一套工具进行增删改查操作。postman简单易用，我用这个工具类来操作，如果你要是没有听过这个工具或者没有用过，那么请先熟悉一下这个工具，我用的是chrome集成的一个postman插件.他也是有客户端的，你可以直接下载一个客户端，装到本地,总之看个人喜好。

1.对于操作数据库我简单的写一个接口UserRepository 以为没有涉及到事物，只是写一个简单的入门程序，所以我这里不写service层，直接dao层，注入到controller即可

```
@Repository
public interface UserRepository extends JpaRepository<User,Integer> {

}
```

注意：这就是我自定义的一个接口，他里面什么都不用做，只需要继承JpaRepository<User,Integer>　泛型类型是User, id类型为用户ID类型。JPA有很多这样的接口，实现不同的功能，如果你想要了解JPA可以去官网查看。

2.web层定义一个类UserHandler，并且加上@RestController注解，因为没有视图渲染所以加的这个注解，如果你要是想用页面来进行操作，那么就正常像SpringMVC那样就可以，这注解要换成@Controller注解。

```
@RestController
public class UserHandler {

        // ….. …..
        // ….. …..
```

}

具体的 UserHandler类如下（因为上面写不下了）

```
@RestController
public class UserHandler {

    @Autowired
    private UserRepository userRepository;

    /**
     * 获取用户列表
     * @return
     */
    @GetMapping("/user")
    public List<User> getUsers() {
        return userRepository.findAll();
    }

    /**
     * 根据用户ID获取单个用户详情
     * @param id
     * @return
     */
    @GetMapping("/user/{id}")
    public User getUserById(@PathVariable("id") Integer id) {
        return userRepository.findOne(id);
    }

    /**
     * 添加一个用户
     * @param name
     * @return
     */
    @PostMapping("/user")
    public User userAdd(@Param("name") String name){
        User user = new User();
        user.setName(name);
        return userRepository.save(user);
    }

    /**
     * 根据用户ID删除用户
     * @param id
     */
    @DeleteMapping("/user/{id}")
    public void userDel(@PathVariable("id") Integer id){
        userRepository.delete(id);
    }
```

```
    /**
     * 更新用户信息
     * @param id
     * @param name
     * @return
     */
    @PutMapping("/user")
    public User userUpdate(@Param("id") Integer id,@Param("name") String name){
        User user = new User();
        user.setId(id);
        user.setName(name);
        return userRepository.save(user);
    }

}
```

这就是web端具体的操作，要是没有学过JPA的人可能会有疑惑，为什么我的DAO层根本就没有实现任何方法，但是还是可以操作数据库呢，具体原因，看看JPA就知道了，这里不再赘述，因为这个重点是SpringBoot。

以上这些操作完全可以用Postman去进行操作。如果Postman你还不会用那么，请学习Postman

好了，以上就是SpringBoot的入门教程，不管写的好不好也算是写完了，顺便的写了一点关于JAP的操作，其实上面介绍的实在是太少了，要想学好一个框架得得花费特别多的时间，特别是英文不好的程序员更要花费特别多的时间。