

原

初识Spring Boot框架

2016年12月19日 20:42:04

[_江南一点雨](#)

阅读数：406716

标签：

spring

java ee

SpringBoot

[更多](#)

版权声明：本文为sang原创文章，转载请注明出处。 <https://blog.csdn.net/u012702547/article/details/53740047>

前面的铺垫文章已经连着写了六篇了，主要是介绍了Spring和SpringMVC框架，小伙伴们在学习的过程中大概也发现了这两个框架需要我们手动配置的地方非常多，不过做JavaEE开发的小伙伴们肯定也听说过“约定大于配置”这样一句话，就是说系统，类库，框架应该假定合理的默认值，而非要求提供不必要的配置，可是使用Spring或者SpringMVC的话依然有许多这样的东西需要我们进行配置，这样不仅徒增工作量而且在跨平台部署时容易出问题。OK，由于这些已经存在的问题，Spring Boot应运而生，使用Spring Boot可以让我们快速创建一个基于Spring的项目，而让这个Spring项目跑起来我们只需要很少的配置就可以了。Spring Boot主要有如下核心功能：

1.独立运行的Spring项目

Spring Boot可以以jar包的形式来运行，运行一个Spring Boot项目我们只需要通过java -jar xx.jar类运行。非常方便。

2.内嵌Servlet容器

Spring Boot可以内嵌Tomcat，这样我们无需以war包的形式部署项目。

3.提供starter简化Maven配置

使用Spring或者SpringMVC我们需要添加大量的依赖，而这些依赖很多都是固定的，这里Spring Boot 通过starter能够帮助我们简化Maven配置。

4.自动配置Spring

5.准生产的应用监控

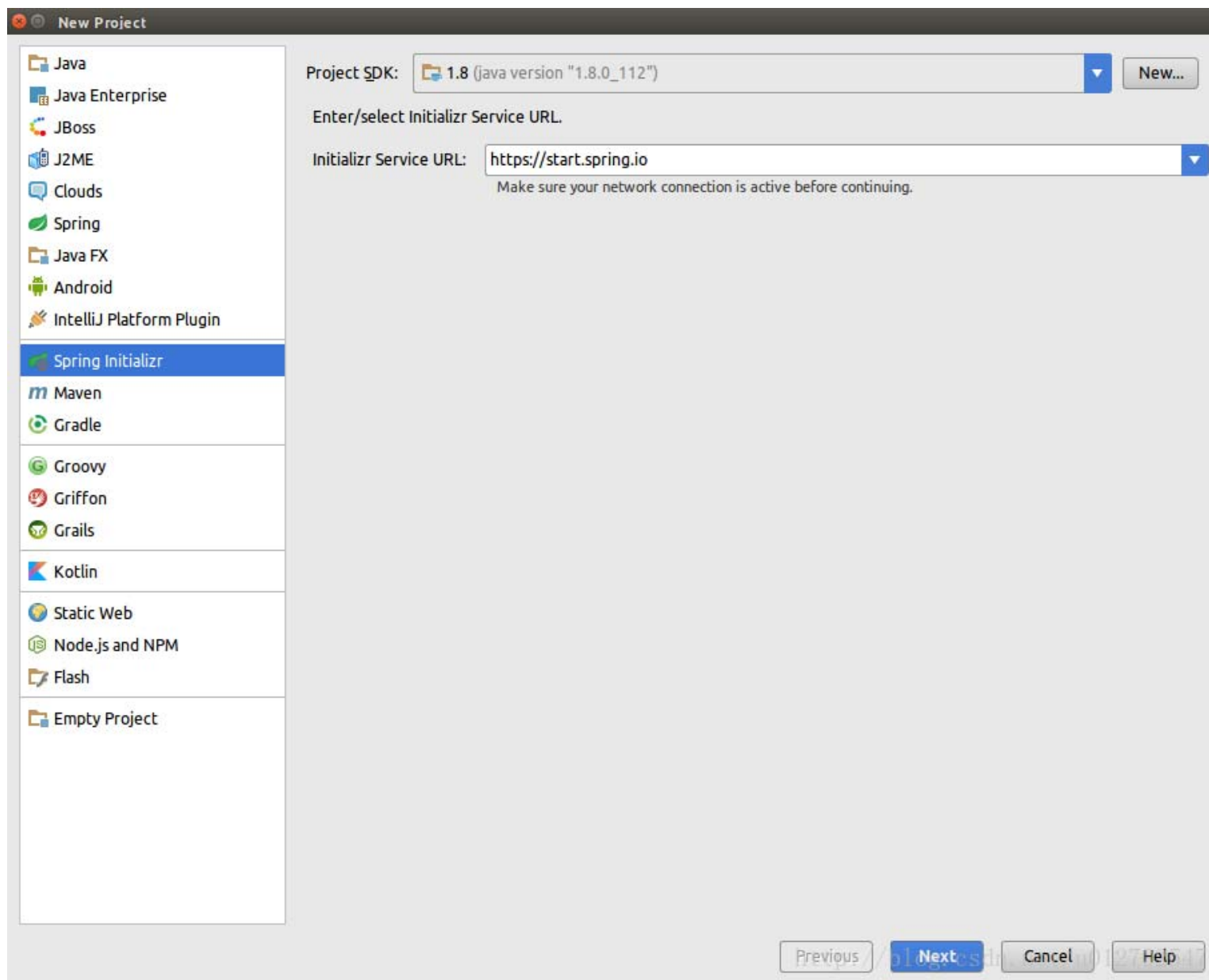
6.无代码生成和xml配置

OK，关于SpringBoot更详细的优缺点小伙伴们也可以自行搜索，我这里不再罗列，我们还是来看看代码。

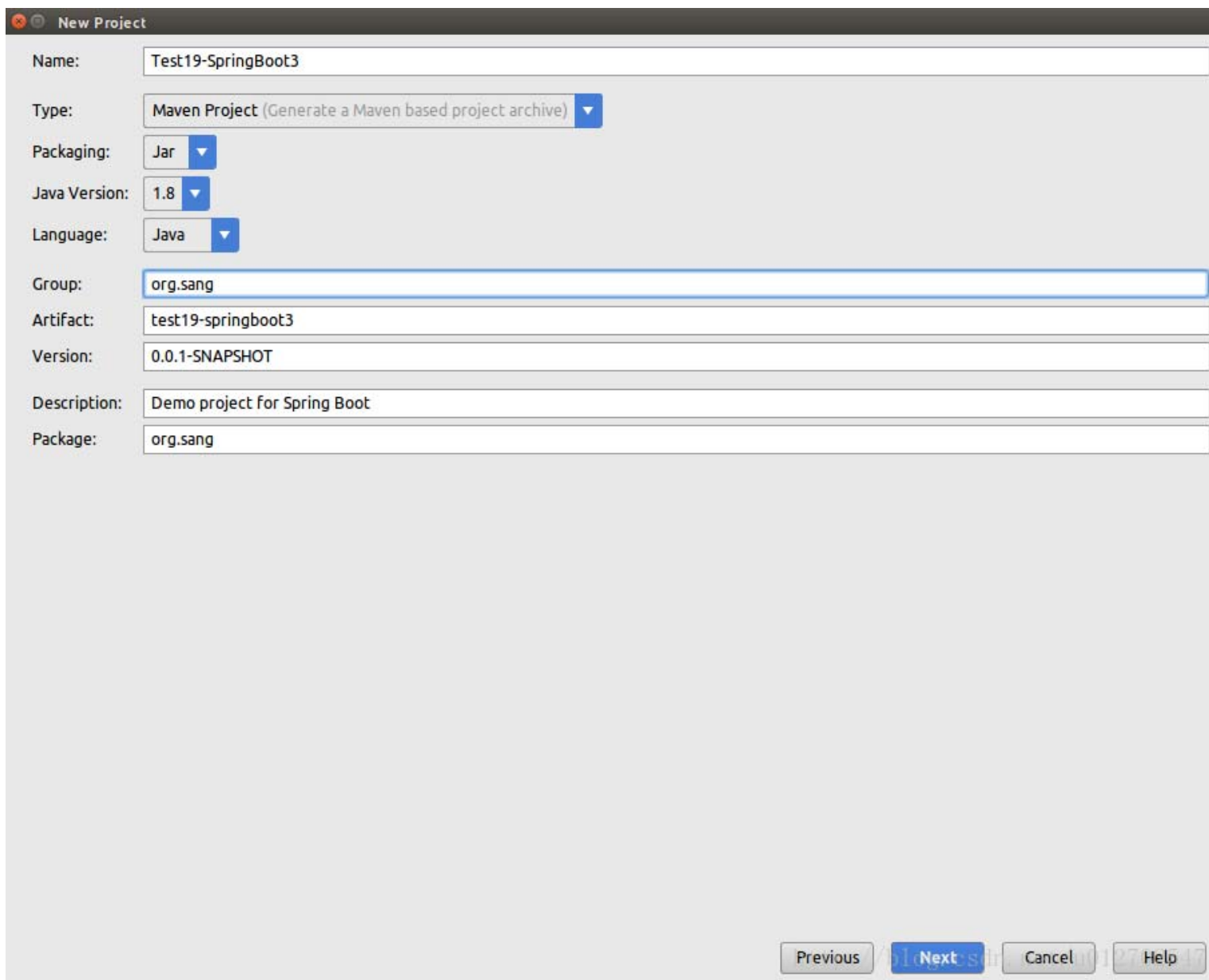
项目创建

初次接触，我们先来看看如何创建一个Spring Boot项目，这里以IntelliJ IDEA为例，其他的IDE工具小伙伴们自行搜索创建方式：

首先创建一个项目，创建时选择Spring Initializr，然后Next，如下图：



填写项目信息，如下图：

The image shows a 'New Project' dialog box in an IDE. It contains several fields for project configuration: Name, Type, Packaging, Java Version, Language, Group, Artifact, Version, Description, and Package. The 'Next' button is highlighted in blue.

Field	Value
Name:	Test19-SpringBoot3
Type:	Maven Project (Generate a Maven based project archive)
Packaging:	Jar
Java Version:	1.8
Language:	Java
Group:	org.sang
Artifact:	test19-springboot3
Version:	0.0.1-SNAPSHOT
Description:	Demo project for Spring Boot
Package:	org.sang

Buttons: Previous, Next, Cancel, Help

填写项目使用到的技术，上面的Spring Boot版本建议选择最新的稳定版，下面勾选上Web就可以了，如下图：

New Project

Spring Boot Version: 1.4.2

Dependencies

Core

☐ Security

☐ Narayana (JTA)

☐ Validation

☐ AOP

☐ Cache

☐ Session

☐ Atomikos (JTA)

☐ DevTools

☐ Retry

☐ Bitronix (JTA)

☐ Configuration Processor

☐ Lombok

Web

☒ Web

☐ Ratpack

☐ Rest Repositories HAL Browser

☐ Websocket

☐ Vaadin

☐ Mobile

☐ Web Services

☐ Rest Repositories

☐ REST Docs

☐ Jersey (JAX-RS)

☐ HATEOAS

Template Engines

☐ Freemarker

☐ Mustache

☐ Velocity

☐ Groovy Templates

☐ Thymeleaf

SQL

☐ JPA

☐ H2

☐ PostgreSQL

☐ JOOQ

☐ HSQLDB

☐ SQL Server

☐ MyBatis

☐ Apache Derby

☐ Flyway

☐ JDBC

☐ MySQL

☐ Liquibase

NoSQL

☐ MongoDB

☐ Redis

☐ Cassandra

☐ Gemfire

☐ Couchbase

☐ Solr

☐ Neo4j

☐ Elasticsearch

Cloud Core

☐ Cloud Connectors

☐ Cloud Task

☐ Cloud Bootstrap

☐ Cloud Security

☐ Cloud OAuth2

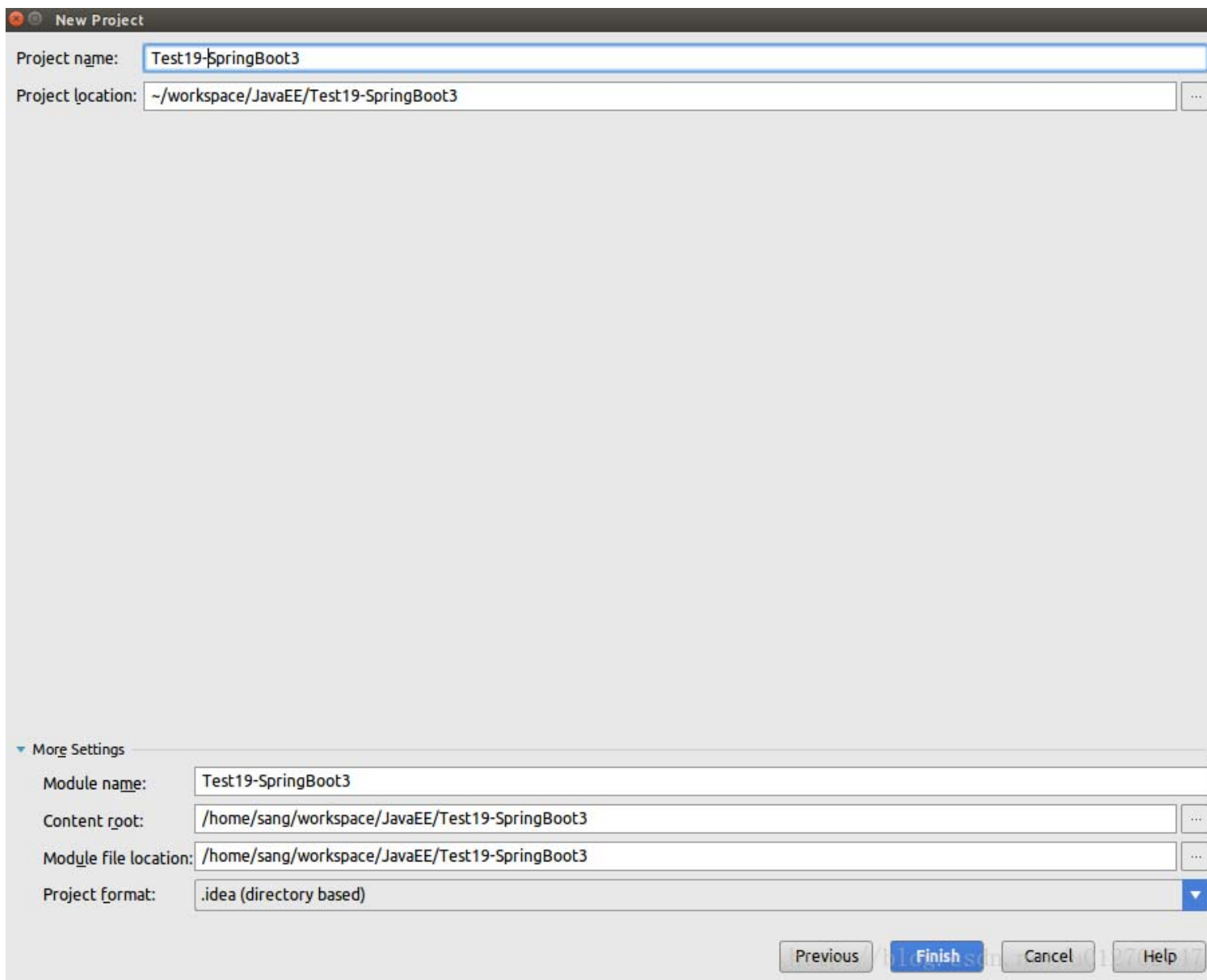
Previous

Next

Cancel

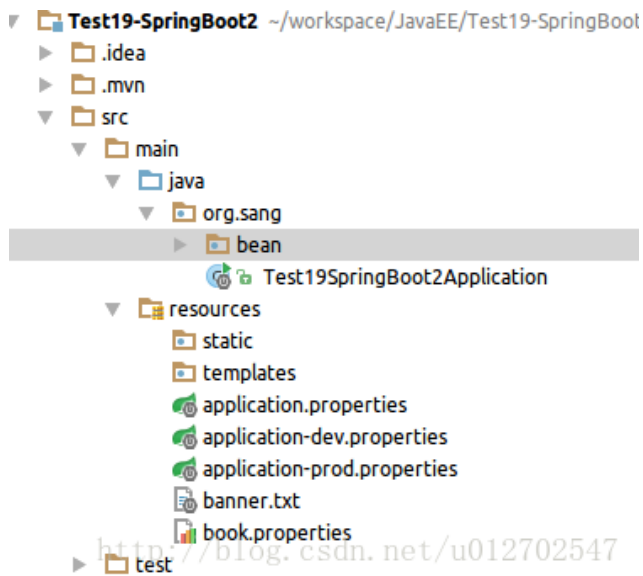
Help

最后一步，填写工程名字点击finish：



OK，第一次创建时系统会去下载需要的依赖等，耗时稍长，以后每次都会很快创建好。

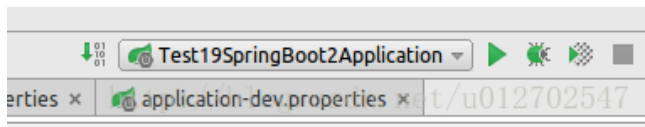
OK，项目创建成功之后接下来我们来看看这个东西要怎么样去运行。首先我们看到在项目创建成功之后，在项目的根目录下会有一个artifactId+Application命名规则的入口类，如下图：



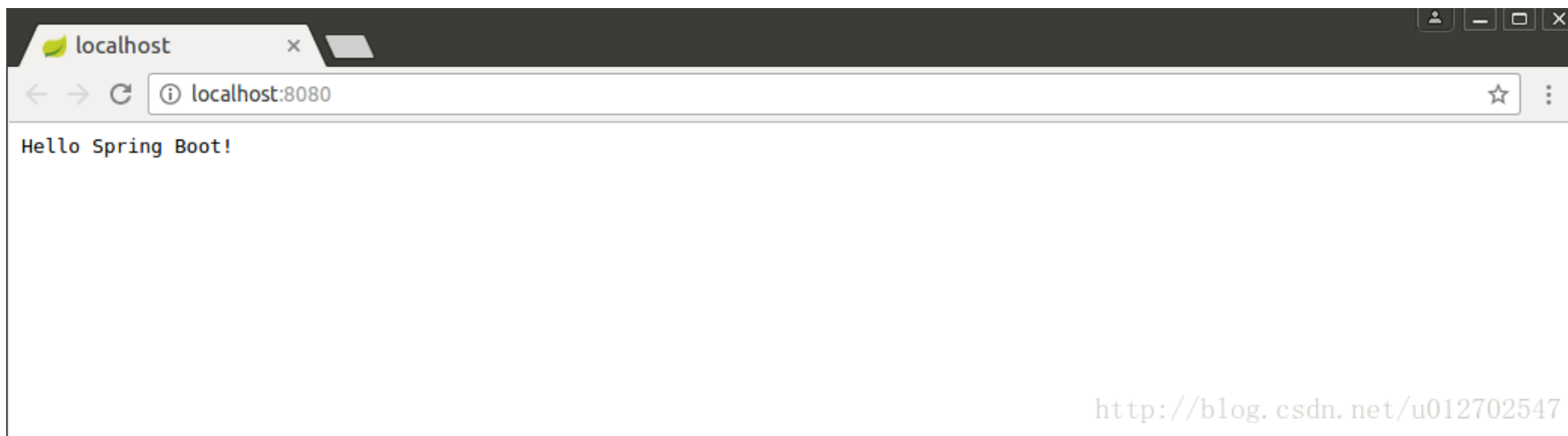
，就是这个Test19SpringBoot2Application类，这是我们整个项目的入口类，这个类有一个@SpringBootApplication注解，这是整个Spring Boot的核心注解，它的目的就是开启Spring Boot的自动配置。OK，那么我在这个类上再添加一个@RestController注解，使之变为一个Controller，然后里边提供一个地址转换方法，如下：

```
1 @RestController
2 @SpringBootApplication
3 public class Test19SpringBoot2Application {
4
5     public static void main(String[] args) {
6         SpringApplication.run(Test19SpringBoot2Application.class, args);
7     }
8
9     @RequestMapping(value = "/", produces = "text/plain;charset=UTF-8")
10    String index(){
11        return "Hello Spring Boot!";
12    }
13 }
```

然后点击项目启动按钮运行，在IntelliJ中就是这个按钮：



启动成功之后我们就可以直接在浏览器中访问了，如下：



OK，至此，我们一个简单的Spring Boot工程已经创建出来了，并且成功的从浏览器中访问到了，但是为什么它最终会跑起来呢？想必小伙伴们还有许多疑问，我们来分析下。

入口类和@SpringBootApplication注解

上文说过，我们新建一个Project系统都会帮我们创建一个名为artifactId+Application的入口类，这个类中有一个main方法，这个main方法就是一个标准的Java应用程序的入口方法。而这里的@SpringBootApplication则是一个组合注解，我们可以看看它的源码：

```
1 @Target({ElementType.TYPE})
2 @Retention(RetentionPolicy.RUNTIME)
3 @Documented
4 @Inherited
5 @SpringBootConfiguration
6 @EnableAutoConfiguration
7 @ComponentScan(
8     excludeFilters = {@Filter(
9         type = FilterType.CUSTOM,
10         classes = {TypeExcludeFilter.class}
11     )}
```

```
12  )
13  public @interface SpringBootApplication {
14
15  }
```

我们可以看到它组合了@SpringBootConfiguration、@EnableAutoConfiguration以及@ComponentScan，我们在开发的过程中如果不使用@SpringBootApplication，则可以组合使用这三个注解。这三个注解中，@SpringBootConfiguration实际上就是我们前面几篇博客提到的@Configuration注解，表明这个类是一个配置类，@EnableAutoConfiguration则表示让Spring Boot根据类路径中的jar包依赖为当前项目进行自动配置，最后一个@ComponentScan的作用我也不赘述了，唯一要注意的是如果我们使用了@SpringBootApplication注解的话，系统会去入口类的同级包以及下级包中去扫描实体类，因此我们建议入口类的位置在groupId+artifactID组合的包名下。

关闭特定的自动配置

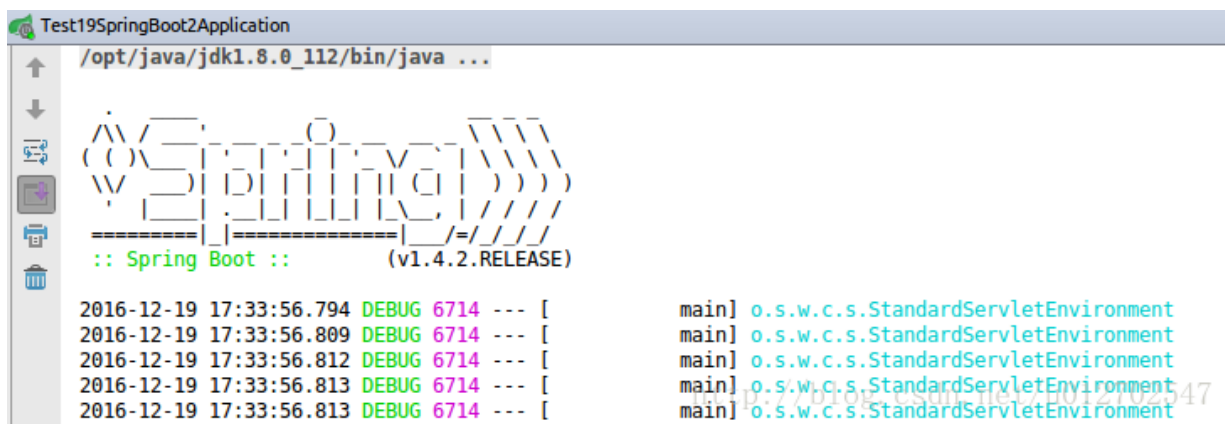
在上面一小节中我们看到@ComponentScan注解是有一个过滤器的，如果我们只想要@SpringBootApplication去扫描特定的类而不是全部类，那么就可以关闭自动配置，如下：

```
1 @SpringBootApplication(exclude = DataSourceAutoConfiguration.class)
```

定制Banner

修改Banner

我们在启动Spring Boot项目的时候，在控制台会默认输出一个启动图案，如下：

A screenshot of a Java IDE window titled 'Test19SpringBoot2Application'. The command line shows the path to the Java executable: '/opt/java/jdk1.8.0_112/bin/java ...'. The output displays the Spring Boot startup banner, which consists of a large, stylized ASCII art logo for 'Spring Boot' and the text 'Spring Boot :: (v1.4.2.RELEASE)'. Below the banner, there are several lines of log output, each starting with a timestamp, log level (DEBUG), and thread name (main), followed by the class 'o.s.w.c.s.StandardServletEnvironment'.

当然，这个图案如果你需要的话是可以自己修改的，修改方式很简单：

- 1.在src/main/resources下新建一个banner.txt文档
- 2.通过<http://patorjk.com/software/taag>网站生成需要的字符，将字符拷贝到步骤1所创建的txt文档中，比如我这里为Hello Sang！生成字符，如下：

Main Controls - *FIGlet and AOL Macro Fonts Supported*

Font: Graffiti

Character Width: Default

Character Height: Default

Test All

More Opts

About

Hello Sang!

EF

不敢开口说英语？看看胡歌怎...

Other Stuff From patorjk.com That You Might Like:

• [Typing Speed Test](#)

• [Keyboard Layout Analyzer](#)

• [Text Color Fader](#)

• [Snake Game](#)

• [Pat's Photography](#)

Hello Sang!

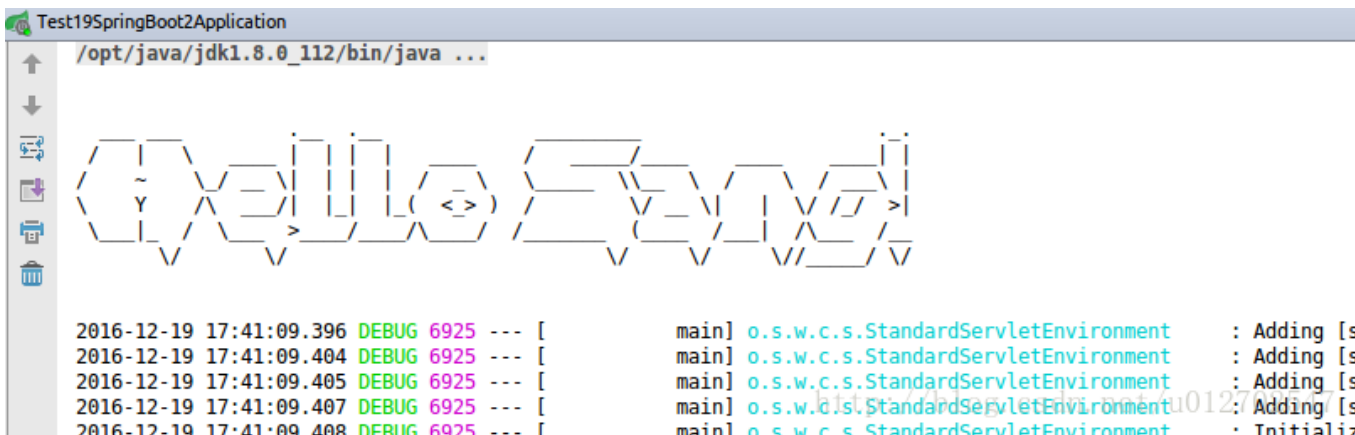
Share Link

Select & Copy

Font Info

<http://blog.csdn.net/patorjk>

点击左下角的选择和拷贝按钮，将这个字符拷贝到txt文档中，然后再启动项目，这个时候控制台输出的文本就会自动改变，如下：



关闭Banner

可以修改当然也可以关闭，关闭Banner需要我们稍微修改一下main方法中的代码，如下：

```
1 public static void main(String[] args) {
2     SpringApplication builder = new SpringApplication(Test19SpringBoot2Application.class);
3     //修改Banner的模式为OFF
4     builder.bannerMode(Banner.Mode.OFF).run(args);
5 }
```

OK，如此修改之后当我们再次启动Project的时候就看不到Banner了。

Spring Boot的配置文件

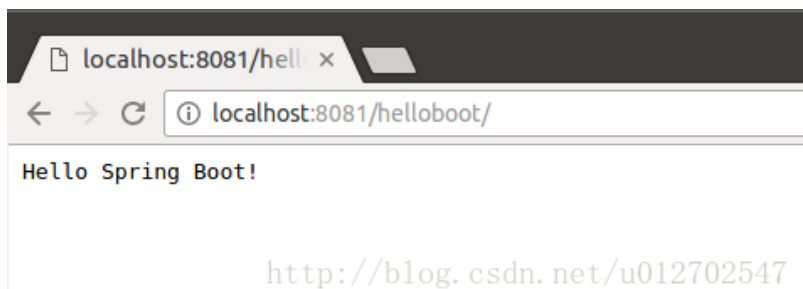
Spring Boot使用一个全局的配置文件application.properties或者application.yml，配置文件放在src/main/resources目录下。properties是我们常见的一种配置文件，Spring Boot不仅支持properties这种类型的配置文件，也支持yaml语言的配置文件，我这里以properties类型的配置文件为例来看几个案例。

1.修改Tomcat默认端口和默认访问路径

Tomcat默认端口是8080，我将之改为8081，默认访问路径是<http://localhost:8080>，我将之改为<http://localhost:8081/helloboot>,我们来看看这两个需求要怎么样通过简单的配置来实现。很简单，在application.properties文件中添加如下代码：

```
1 server.context-path=/helloboot
2 server.port=8081
```

然后再启动Project，在浏览器中就得这样来访问了：



常规属性配置

在前面的博客([Spring常用配置](#))中我们介绍了如何在使用Spring容器框架下注入properties文件里的值。如果我们使用了Spring Boot，这项工作将会变得更加简单，我们只需要在application.properties中定义属性，然后在代码中直接使用@Value注入即可。

如下：

```
1 book.author=罗贯中
2 book.name=三国演义
3 book.pinyin=sanguoyanyi
```

我这里专门设置了中文，因为中文不做特殊处理会乱码，处理方式为继续在application.properties中添加如下代码：

```
1 server.tomcat.uri-encoding=UTF-8
2 spring.http.encoding.charset=UTF-8
3 spring.http.encoding.enabled=true
4 spring.http.encoding.force=true
5 spring.messages.encoding=UTF-8
```

然后 在IntelliJ IDEA中依次点击File -> Settings -> Editor -> File Encodings

将Properties Files (*.properties)下的Default encoding for properties files设置为UTF-8，将Transparent native-to-ascii conversion前的勾选上。（参考【[Springboot 之 解决IDEA读取properties配置文件的中文乱码问题](#)】【[Springboot 之 解决IDEA读取properties配置文件的中文乱码问题](#)】）。

然后在变量中通过@Value直接注入就行了，如下：

```
1 @Value(value = "${book.author}")
2 private String bookAuthor;
3 @Value("${book.name}")
4 private String bookName;
```

```
5 @Value("${book.pinyin}")
6 private String bookPinYin;
```

修改index方法，使之返回这些值：

```
1 @RequestMapping(value = "/",produces = "text/plain;charset=UTF-8")
2     String index(){
3         return "Hello Spring Boot! The BookName is "+bookName+";and Book Author is "+bookAuthor+";and Book PinYin is "+bookPinYin;
4     }
```

然后在浏览器中访问，结果如下：



很简单吧。

类型安全的配置

刚刚说的这种方式我们在实际项目中使用的时候工作量略大，因为每个项目要注入的变量的值太多了，这种时候我们可以使用基于类型安全的配置方式，就是将properties属性和一个Bean关联在一起，这样使用起来会更加方便。我么来看看这种方式怎么实现。

- 1.在src/main/resources文件夹下创建文件book.properties

文件内容如下：

```
1 book.name=红楼梦
2 book.author=曹雪芹
3 book.price=28
```

- 2.创建Book Bean,并注入properties文件中的值

代码如下：

```
1  @Component
2  @ConfigurationProperties(prefix = "book",locations = "classpath:book.properties")
3  public class BookBean {
4      private String name;
5      private String author;
6      private String price;
7
8      public String getName() {
9          return name;
10     }
11
12     public void setName(String name) {
13         this.name = name;
14     }
15
16     public String getAuthor() {
17         return author;
18     }
19
20     public void setAuthor(String author) {
21         this.author = author;
22     }
23
24     public String getPrice() {
25         return price;
26     }
27
28     public void setPrice(String price) {
29         this.price = price;
30     }
31 }
```

prefix是指前缀，location指定要注入文件的位置。

3.添加路径映射

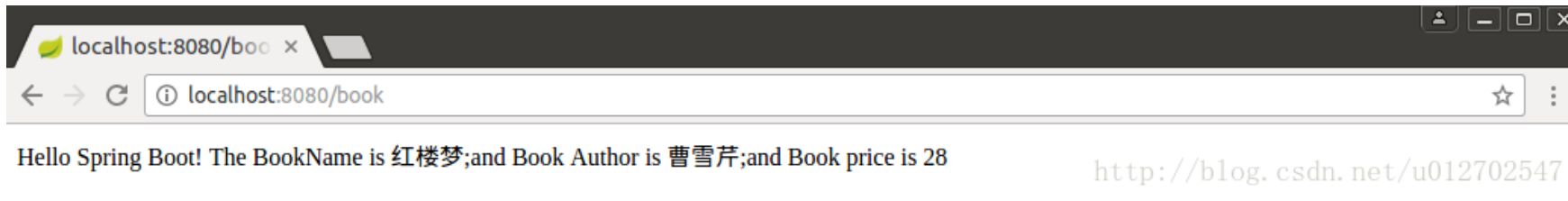
在Controller中添加如下代码注入Bean：

```
1 @Autowired
2     private BookBean bookBean;
```

添加路径映射：

```
1 @RequestMapping("/book")
2     public String book() {
3         return "Hello Spring Boot! The BookName is "+bookBean.getName()+"and Book Author is "+bookBean.getAuthor()+"and Book price is "+bookBean.getPrice();
4     }
```

运行效果如下：



日志配置

默认情况下Spring Boot使用Logback作为日志框架，也就是我们前面几篇博客中用到的打印日志方式，当然如果有需要我们可以手动配置日志级别以及日志输出位置，相比于我们在Spring容器中写的日志输出代码，这里的配置简直就是小儿科了，只需要在application.properties中添加如下代码：

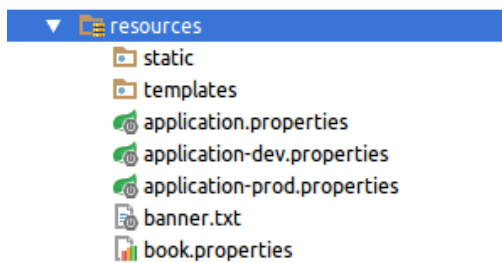
```
1 logging.file=/home/sang/workspace/log.log
2 logging.level.org.springframework.web=debug
```

上面表示配置日志输出位置，下面配置日志级别。

Profile配置问题

在 [Spring常用配置](#) 这篇文章中，我们已经介绍了Profile的作用，已经如何在Spring框架下使用Profile，但是当时小伙伴们看到了还是稍微有点麻烦，在Spring Boot 中系统提供了更为简洁的方式。全局Profile配置我们使用application-{profile}.properties来定义，然后在application.properties中通过spring.profiles.active来指定使用哪个Profile。OK，那么接下来我们来看一个简单的案例。

1.在src/main/resources文件夹下定义不同环境下的Profile配置文件，文件名分别为application-prod.properties和application-dev.properties，这两个前者表示生产环境下的配置，后者表示开发环境下的配置，如下：



application-prod.properties:

```
1 server.port=8081
```

application-dev.properties:

```
1 server.port=8080
```

然后在application.properties中进行简单配置，如下：

```
1 spring.profiles.active=dev
```

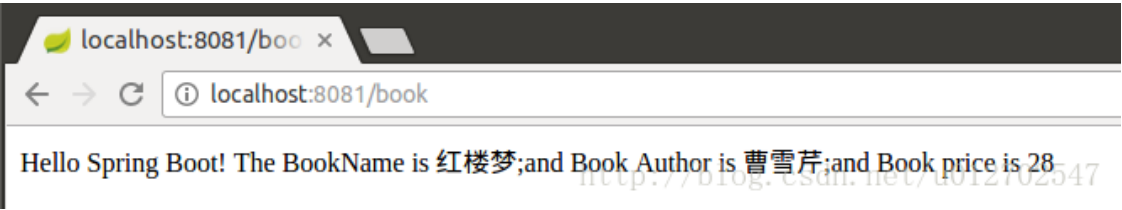
这个表示使用开发环境下的配置。然后运行项目，我们得通过8080端口才可以访问：



Hello Spring Boot! The BookName is 红楼梦;and Book Author is 曹雪芹;and Book price is 28

<http://blog.csdn.net/u012702547>

如果想换为生产环境，只需要把 `spring.profiles.active=dev` 改为 `spring.profiles.active=prod` 即可，当然访问端口这是也变为8081了，如下：



本案例下载地址：
本项目GitHub地址

以上。

参考资料：
《JavaEE开发的颠覆者 Spring Boot实战》第五章、第六章

想对作者说点什么

weixin_43627763：

[objc]

01. 感谢作者的无私分享，分享目前主流的技术干货教程：SpringBoot+SpringCloud(2 0 1 7 最新微服务系列)、Docker容器、Hadoop Spark(大数据)、RocketMq、dubbo、redis分布式、数据库性能调优、Nginx入门实战高级视频教程、SSM框架等等。
02. <https://itstorage.github.io/java/goods.html>

(3天前 #57楼)

lbh769241267：感谢分享！！！（1个月前 #56楼）

nemo0718：感谢博主，文章比较久了，注意 `server.context-path=/helloboot` 更新后写法变成了 `server.servlet.context-path=/helloboot` （1个月前 #55楼）

胡泽宽：感谢分享。（1个月前 #54楼）

趁你未老：2018最新免费Spring Cloud等多套架构师视频教程全套下载 百度云盘：<https://www.jianshu.com/p/b3392007b241> （1个月前 #53楼）

趁你未老：<https://www.jianshu.com/p/98b7335444f5> （1个月前 #52楼）

谷海涛：感谢分享，（2个月前 #51楼）