https://blog.csdn.net/u012702547/article/details/53750449

**初识Spring Boot框架(二)之DIY一个Spring Boot的自动配置**

2016年12月20日 14:47:30 [\_江南一点雨](https://me.csdn.net/u012702547) 阅读数：31687 标签： [spring](http://so.csdn.net/so/search/s.do?q=spring&t=blog)[SpringBoot](http://so.csdn.net/so/search/s.do?q=SpringBoot&t=blog)[SpringMVC](http://so.csdn.net/so/search/s.do?q=SpringMVC&t=blog) 更多

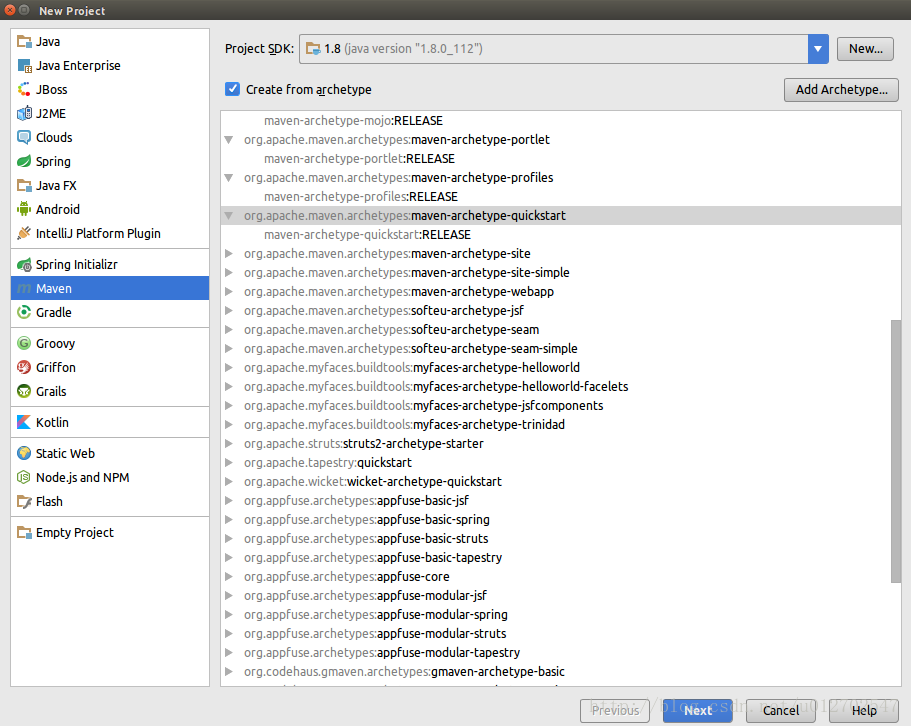
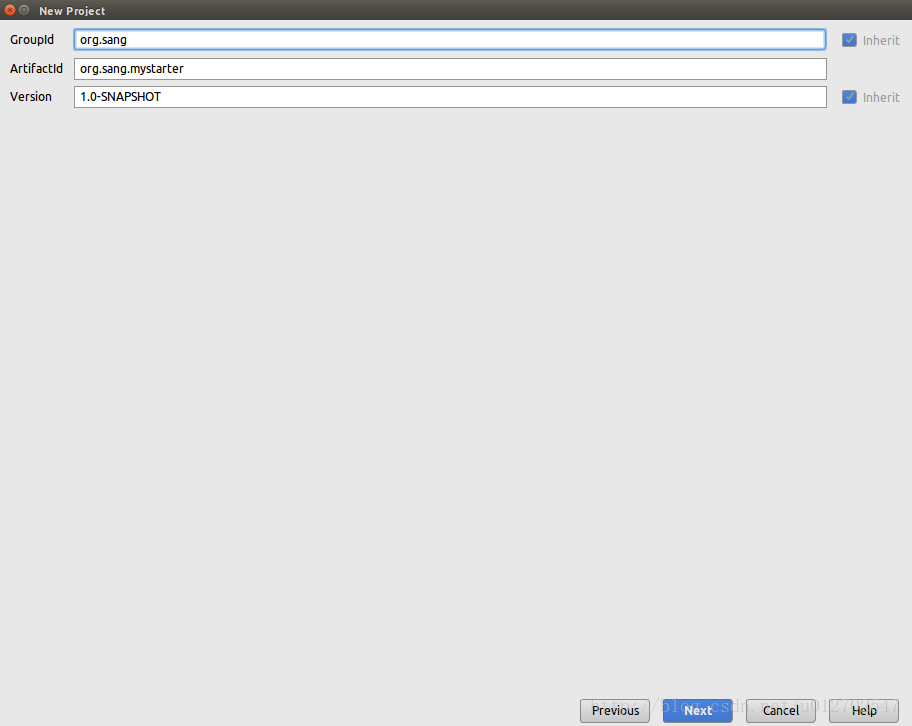
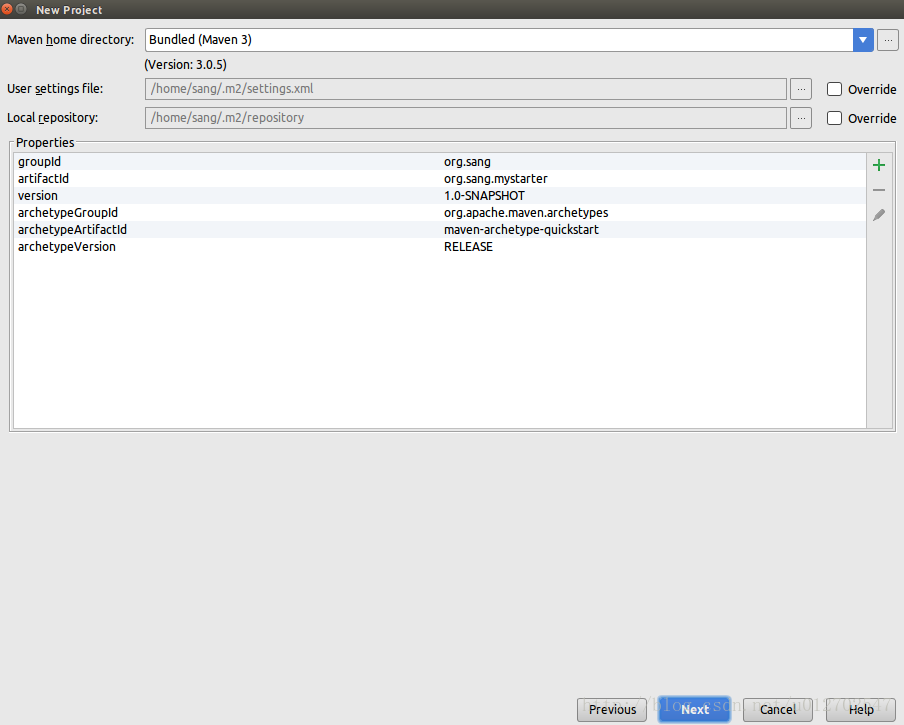
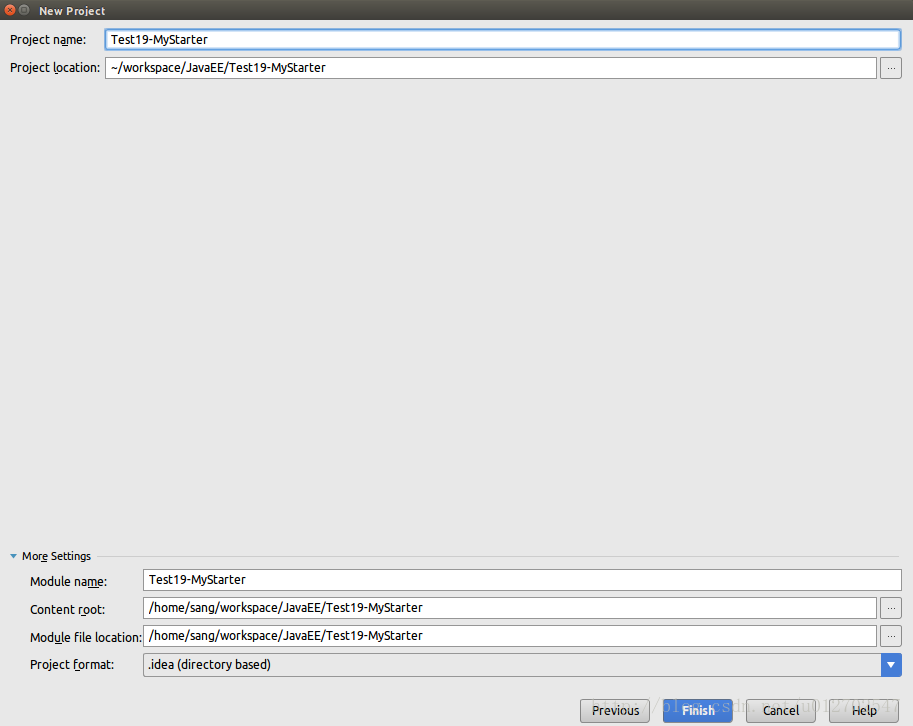
个人分类： [java](https://blog.csdn.net/u012702547/article/category/3095469)[JavaEE](https://blog.csdn.net/u012702547/article/category/6572033)

所属专栏： [关于Spring Boot](https://blog.csdn.net/column/details/13987.html)

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/53750449

在上篇博客[初识Spring Boot框架](http://blog.csdn.net/u012702547/article/details/53740047)中我们初步见识了SpringBoot的方便之处，很多小伙伴可能也会好奇这个Spring Boot是怎么实现自动配置的，那么今天我就带小伙伴我们自己来实现一个简单的Spring Boot 自动配置的案例，看看这一切到底是怎么发生的。   
假设我的需求是这样的：当我的项目中存在某个类的时候，系统自动为我配置该类的Bean，同时，我这个Bean的属性还可以在application.properties中进行配置，OK，就这样一个需求，我们来看看怎么实现。

1. 新建starter的Maven项目

我这里以IntelliJ IDEA创建Project为例，其实很简单，创建一个Maven项目，但是注意创建的时候选择quickstart，步骤如下：   
   
   
   
   
OK，这样我们就创建好一个Project了。

2.添加依赖

由于我们要使用Spring Boot的自动配置功能，所以在工程创建成功之后首先要添加Spring Boot自身的自动配置作为依赖，如下：

<dependency>

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

<artifactId>spring-boot-autoconfigure</artifactId>

<version>1.4.2.RELEASE</version>

</dependency>

* 1
* 2
* 3
* 4
* 5

3.属性配置

@ConfigurationProperties(prefix = "hello")

public class HelloServiceProperties {

private static final String MSG = "world";

private String msg = MSG;

public String getMsg() {

return msg;

}

public void setMsg(String msg) {

this.msg = msg;

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13

这里就是一个简单的类，和上篇博客中我们提到的 类型安全的属性获取是一致的，这里属性的值我们可以在application.properties中来直接设置，如果不设置的话默认为world。

4.判断依据类

public class HelloService {

private String msg;

public String sayHello(){

return "hello " + msg;

}

public String getMsg() {

return msg;

}

public void setMsg(String msg) {

this.msg = msg;

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14

我们一会的例子就是根据这个类是否存在来创建这个类的Bean，当然这个类也可以是第三方类库。

5.自动配置类

@Configuration

@EnableConfigurationProperties(HelloServiceProperties.class)

@ConditionalOnClass(HelloService.class)

@ConditionalOnProperty(prefix = "hello",value = "enable",matchIfMissing = true)

public class HelloServiceAutoConfiguration {

@Autowired

private HelloServiceProperties helloServiceProperties;

@Bean

public HelloService helloService() {

HelloService helloService = new HelloService();

helloService.setMsg(helloServiceProperties.getMsg());

return helloService;

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14

这里@ConditionalOnClass注解主要是用来判断HelloService这个类在路径中是否存在，在存在且容器中没有该类的Bean的情况下系统会自动配置需要的Bean，@ConditionalOnProperty注解则表示指定的属性要满足的条件，在helloService方法中我们则使用了HelloServiceProperties提供的参数。

6.注册配置

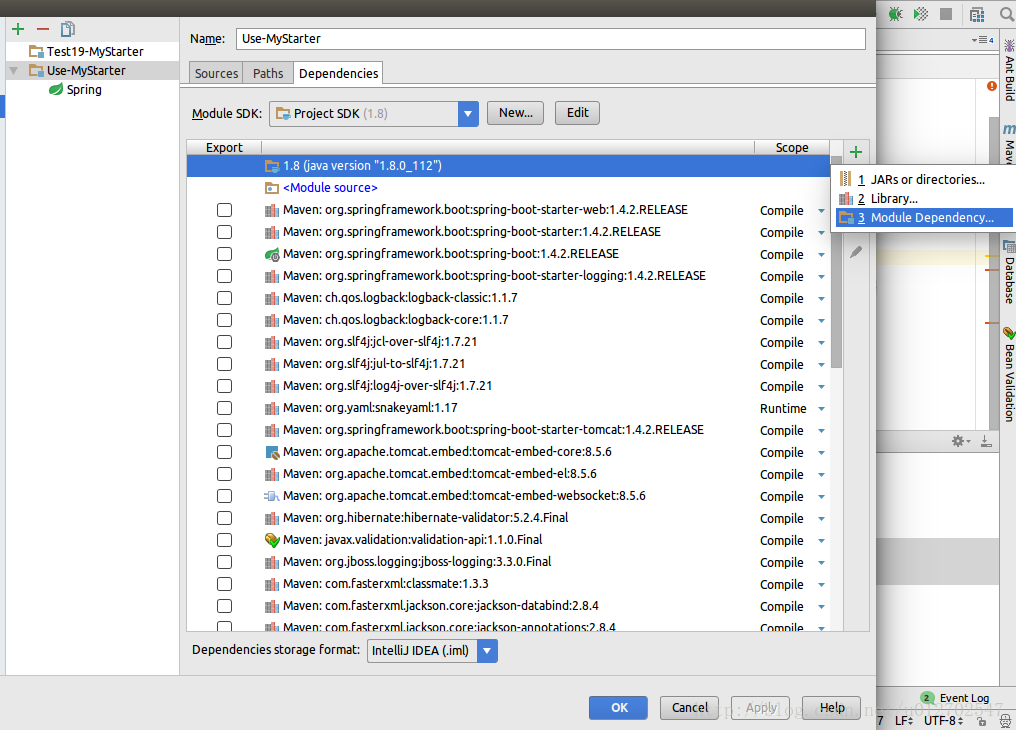
在resources目录下新建META-INF目录，然后在META-INF目录下创建spring.factories文件，文件内容如下，表示设置自动配置类的位置，若有多个配置类用”,”隔开即可。：

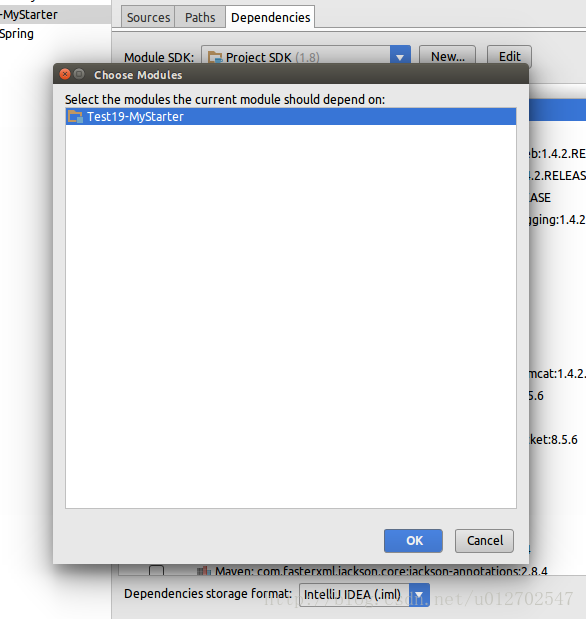
org.springframework.boot.autoconfigure.EnableAutoConfiguration=org.sang.HelloServiceAutoConfiguration

* 1

正常情况下我们按步骤一创建出来的Project是没有resources这个文件夹的，在IntelliJ IDEA这个工具中，我们需要先创建一个directory，然后将之设置为resources root即可，设置方式如下：选中resources目录右键单击–》Mark Directory as–》Resource Root。

7.在项目中使用

OK，经过上面几个步骤我们的自动配置功能就算死写好了，接下来我们要在自己的项目中来使用它了，正常情况下做好自动配置类之后我们需要将之上传到   
maven 私服上然后直接在项目中引用，也可以将之安装到本地库，这里我选择后者的一个简化方式，直接创建Module来引用这个自动配置Module（如果小伙伴是用STS或者MyEclipse之类的开发工具直接创建新工程然后添加引用类库即可）。   
创建Module方式很简单，选中刚才的Project右键单击，New一个Module，这个Module是一个SpringBoot项目（不清楚怎么创建SpringBoot项目可以查看这篇文章[初识Spring Boot框架](http://blog.csdn.net/u012702547/article/details/53740047)），创建成功之后选择新建的Module按F4打开Module设置，然后选择右边的加号添加依赖，如下：   




然后在新建Module的pom.xml文件中添加如下依赖：

<dependency>

<groupId>org.sang</groupId>

<artifactId>org.sang.mystarter</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

* 1
* 2
* 3
* 4
* 5

实际上就是我们刚刚创建的自动配置文件的坐标。然后新Module的入口类中添加如下代码：

@Autowired

private HelloService helloService;

@RequestMapping("/")

public String index(){

return helloService.sayHello();

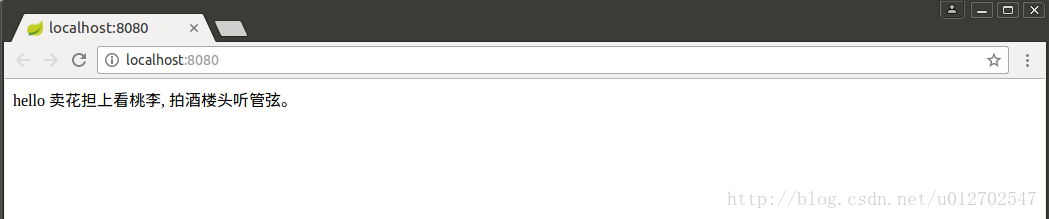
}

* 1
* 2
* 3
* 4
* 5
* 6
* 7

使用刚刚创建的HelloService，如果需要我们可以在application.properties中添加如下配置：

hello.msg = 卖花担上看桃李, 拍酒楼头听管弦。

* 1

运行Module，然后在浏览器中访问，结果如下：   


.

OK，这就是一个简单的自定义自动配置类。

本案例下载地址：

[本案例GitHub地址](https://github.com/lenve/JavaEETest/tree/master/Test19-MyStarter)

以上。

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