Spring Boot自定义starters

一、简介

SpringBoot 最强大的功能就是把我们常用的场景抽取成了一个个starter（场景启动器），我们通过引入springboot 为我提供的这些场景启动器，我们再进行少量的配置就能使用相应的功能。即使是这样，springboot也不能囊括我们所有的使用场景，往往我们需要自定义starter，来简化我们对springboot的使用。

二、如何自定义starter

1.实例

如何编写自动配置？

我们参照@WebMvcAutoConfiguration为例，我们看看们需要准备哪些东西，下面是WebMvcAutoConfiguration的部分代码：

```
1 @Configuration
2 @ConditionalOnWebApplication
3 @ConditionalOnClass({Servlet.class, DispatcherServlet.class, Web
MvcConfigurerAdapter.class})
4 @ConditionalOnMissingBean({WebMvcConfigurationSupport.class})
5 @AutoConfigureOrder(-2147483638)
6 @AutoConfigureAfter({DispatcherServletAutoConfiguration.class, V
alidationAutoConfiguration.class})
7 public class WebMvcAutoConfiguration {
8
9  @Import({WebMvcAutoConfiguration.EnableWebMvcConfiguration.clas
s})
```

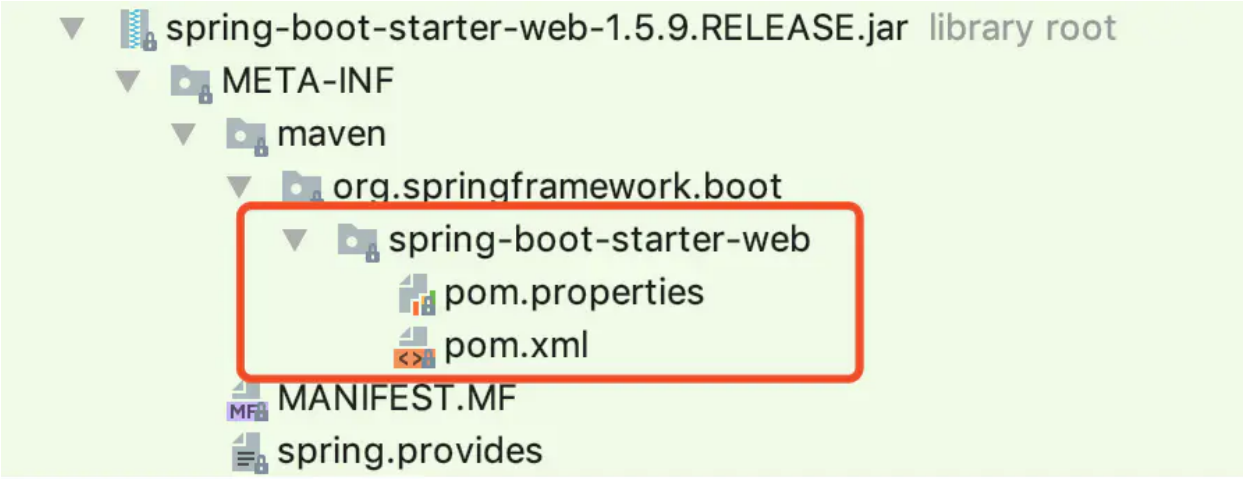
```
10  @EnableConfigurationProperties({WebMvcProperties.class, Resour
ceProperties.class})
11  public static class WebMvcAutoConfigurationAdapter extends Web
MvcConfigurerAdapter {
12
13  @Bean
14  @ConditionalOnBean({View.class})
15  @ConditionalOnMissingBean
16  public BeanNameViewResolver beanNameViewResolver() {
17  BeanNameViewResolver resolver = new BeanNameViewResolver();
18  resolver.setOrder(2147483637);
19  return resolver;
20  }
21  }
22 }
```

我们可以抽取到我们自定义starter时同样需要的一些配置。

```
1 @Configuration //指定这个类是一个配置类
2 @ConditionalOnXXX //指定条件成立的情况下自动配置类生效
3 @AutoConfigureOrder //指定自动配置类的顺序
4 @Bean //向容器中添加组件
5 @ConfigurationProperties //结合相关xxxProperties来绑定相关的配置
6 @EnableConfigurationProperties //让xxxProperties生效加入到容器中
7
8 自动配置类要能加载需要将自动配置类，配置在META-INF/spring.factories
中
9 org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
10 org.springframework.boot.autoconfigure.admin.SpringApplicationA
dminJmxAutoConfiguration,\
11 org.springframework.boot.autoconfigure.aop.AopAutoConfiguration
```

模式

我们参照 spring-boot-starter 我们发现其中没有代码：



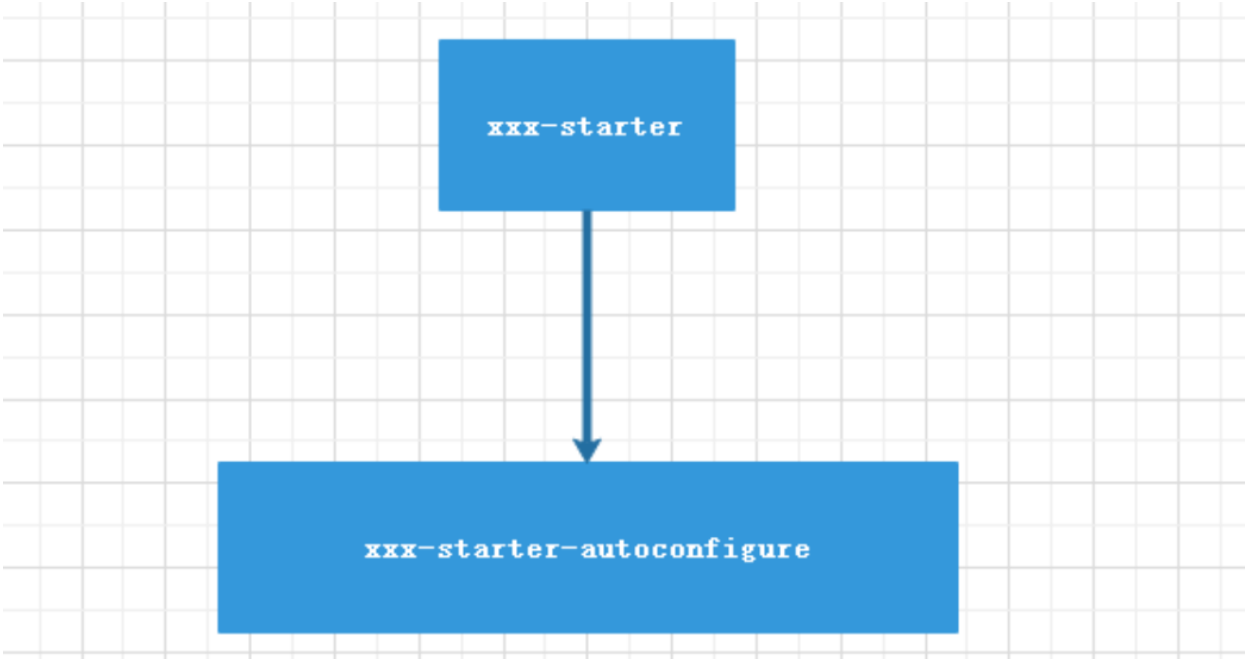
我们在看它的pom中的依赖中有个 springboot-starter

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

我们再看看 spring-boot-starter 有个 spring-boot-autoconfigure

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

关于web的一些自动配置都写在了这里， 所以我们有总结:



- 启动器（starter）是一个空的jar文件，仅仅提供辅助性依赖管理，这些依赖可能用于自动装配或其他类库。

- 需要专门写一个类似spring-boot-autoconfigure的配置模块
- 用的时候只需要引入启动器starter，就可以使用自动配置了

命名规范

官方命名空间

- 前缀：spring-boot-starter-
- 模式：spring-boot-starter-模块名
- 举例：spring-boot-starter-web、spring-boot-starter-jdbc

自定义命名空间

- 后缀：-spring-boot-starter
- 模式：模块-spring-boot-starter
- 举例：mybatis-spring-boot-starter

三、自定义starter实例

我们需要先创建一个父maven项目:springboot_custome_starter

两个Module: tulingxueyuan-spring-boot-starter 和 tulingxueyuan-spring-boot-starter-autoconfigurer

springboot_custome_starter

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="ht
tp://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://m
aven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <modules>
 <module>tulingxueyuan-spring-boot-starter</module>
 <module>tulingxueyuan-spring-boot-autoconfigure</module>
 </modules>
 <parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.3.6.RELEASE</version>
```

```
<relativePath/> <!-- lookup parent from repository -->
</parent>
<packaging>pom</packaging>
<groupId>com.tulingxueyuan.springboot</groupId>
<artifactId>springboot_custome_starter</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>springboot_custome_starter</name>
<description>SpringBoot自定义starter</description>

<properties>
<java.version>1.8</java.version>
</properties>

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

</project>
```

1. tulingxueyuan-spring-boot-starter

1.pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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://ma
ven.apache.org/xsd/maven-4.0.0.xsd">
 <parent>
 <artifactId>springboot_custome_starter</artifactId>
 <groupId>com.tulingxueyuan.springboot</groupId>
 <version>0.0.1-SNAPSHOT</version>
 </parent>
 <modelVersion>4.0.0</modelVersion>
```

```
<description>
启动器（starter）是一个空的jar文件，
仅仅提供辅助性依赖管理，
这些依赖需要自动装配或其他类库。
</description>

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

<dependencies>
<!--引入autoconfigure-->
<dependency>
<groupId>com.tulingxueyuan.springboot</groupId>
<artifactId>tulingxueyuan-spring-boot-autoconfigure</artifactId>
<version>0.0.1-SNAPSHOT</version>
</dependency>

<!--如果当前starter 还需要其他的类库就在这里引用-->
</dependencies>

</project>
```

> 如果使用spring Initializr创建的需要删除 启动类、resources下的文件，test文件。

2. tulingxueyuan-spring-boot-starter-autoconfigurer

1. pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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">
 <parent>
 <artifactId>springboot_custome_starter</artifactId>
 <groupId>com.tulingxueyuan.springboot</groupId>
```

```
 <version>0.0.1-SNAPSHOT</version>
 </parent>
  <modelVersion>4.0.0</modelVersion>

 <artifactId>tulingxueyuan-spring-boot-autoconfigure</artifactId>
 <dependencies>
 <dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-web</artifactId>
 </dependency>
 <!---导入配置文件处理器，配置文件进行绑定就会有提示-->
 <dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-configuration-processor</artifactId>
 <optional>true</optional>
 </dependency>
 </dependencies>


</project>

```

2. HelloProperties

```
package com.starter.tulingxueyuan;

import org.springframework.boot.context.properties.ConfigurationProperties;

/***
 * @Author 徐庶 QQ:1092002729
 * @Slogan 致敬大师，致敬未来的你
 */
@ConfigurationProperties("tuling.hello")
public class HelloProperties {
```

```
    private String name;

    public String getName() {
    return name;
    }

    public void setName(String name) {
    this.name = name;
    }
}

```

3. IndexController

```
package com.starter.tulingxueyuan;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/***
 * @Author 徐庶 QQ:1092002729
 * @Slogan 致敬大师，致敬未来的你
 */
@RestController
public class IndexController {

    HelloProperties helloProperties;

    public IndexController(HelloProperties helloProperties) {
    this.helloProperties=helloProperties;
    }

    @RequestMapping("/")
    public String index(){
    return helloProperties.getName()+"欢迎您";
```

```
    }

}

```

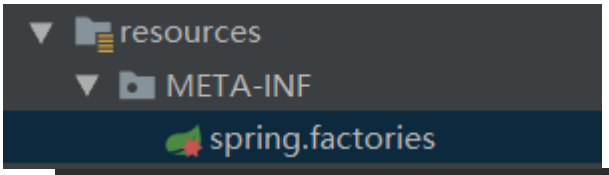
4. HelloAutoConfitguration

```
package com.starter.tulingxueyuan;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.condition.ConditionalOnProperty;
import org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/***
 * @Author 徐庶 QQ:1092002729
 * @Slogan 致敬大师，致敬未来的你
 *
 * 给web应用自动添加一个首页
 */
@Configuration
@ConditionalOnProperty(value = "tuling.hello.name")
@EnableConfigurationProperties(HelloProperties.class)
public class HelloAutoConfitguration {

    @Autowired
    HelloProperties helloProperties;

    @Bean
    public IndexController indexController(){
    return new IndexController(helloProperties);
    }
}
```

```
28
```

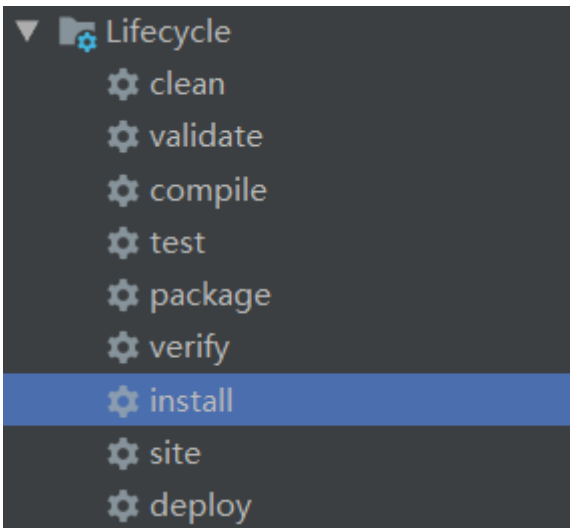
5. spring.factories

在 resources 下创建文件夹 META-INF 并在 META-INF 下创建文件 spring.factories，内容如下：




```
1
2 org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
3  com.starter.tulingxueyuan.HelloAutoConfitguration
```

到这儿，我们的配置自定义的starter就写完了，我们hello-spring-boot-starter-autoconfigurer、hello-spring-boot-starter 安装成本地jar包。



三、测试自定义starter

我们创建个Module: 12_springboot_starter，来测试系我们写的stater。

1. pom.xml

```
1 <dependency>
2  <groupId>com.tulingxueyuan.springboot</groupId>
3  <artifactId>tulingxueyuan-spring-boot-starter</artifactId>
4  <version>0.0.1-SNAPSHOT</version>
5 </dependency>
6
7
```

2.浏览

http://localhost:8080/

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Dec 22 20:02:17 CST 2020
There was an unexpected error (type=Not Found, status=404).

由于在自动配置上设置了

```
1 @ConditionalOnProperty(value = "tuling.hello.name")
```

但我们还没有配置。so.....

3. application.properties

```
1 tuling.hello.name="图灵学院"
```

再次访问：http://localhost:8080/

"图灵学院"欢迎您