

## Spring Boot starter

starter介绍

starter原理

起步依赖

自动配置

基于Java代码的Bean配置

自动配置条件依赖

Bean参数获取

Bean的发现

Bean的加载

自定义starter

案例一

开发starter

第一步：初始化项目

第二步：创建配置属性类HelloConfigProperties

第三步：创建HelloService

第四步：创建自动配置类HelloServiceAutoConfiguration

第五步：在resources目录下创建META-INF/spring.factories

使用starter

第一步：在use-starter子工程中修改pom文件

第二步：创建application.yml文件添加配置

第三步：创建HelloController

第四步：启动项目

第五步：访问

第六步：更改配置

第七步：重启服务器，再次访问

案例二

开发starter

第一步：初始化项目

第二步：自定义Log注解

第三步：自定义日志拦截器LogInterceptor

第四步：创建自动配置类LogAutoConfiguration，用于自动配置拦截器、参数解析器等web组件

第五步：在spring.factories中追加LogAutoConfiguration配置

第六步：安装到本地库

使用starter

第一步：添加依赖

第二步：编写controller类

第三步：启动程序

第四步：访问服务

案例三

开发starter

第一步：初始化项目

第二步：编写IpCountService业务

第三步：编写配置类IpAutoConfiguration

第四步：在spring.factories中追加IpAutoConfiguration配置

第五步：开启定时任务功能

第六步：设置定时任务

第七步：定义属性类，加载对应属性

第八步：设置加载Properties类为bean

第九步：根据配置切换设置

第十步：使用#{beanName.attrName}读取bean的属性

第十一步：自定义拦截器IpInterceptor

第十二步: 注册拦截器IpInterceptor  
第十三步: 更改spring.factories文件  
第十四步: 自定义提示功能

使用starter

第一步: 添加依赖  
第二步: 编写配置文件  
第三步: 启动程序  
第四步: 访问服务, 查看日志  
第五步: 更改统计速度  
第六步: 重启服务, 访问服务, 查看日志  
第七步: 更改模式  
第八步: 重启服务, 访问服务, 查看日志  
第九步: 是否周期内重置数据选项  
第十步: 重启服务, 访问服务, 查看日志

---

## Spring Boot starter

Spring Boot大大简化了项目初始搭建以及开发过程, 而这些都是通过Spring Boot提供的starter来完成的

### starter介绍

spring boot 在配置上相比spring要简单许多, 其核心在于spring-boot-starter, 在使用spring boot来搭建一个项目时, 只需要引入官方提供的starter, 就可以直接使用, 免去了各种配置。starter简单来讲就是引入了一些相关依赖和一些初始化的配置。

Spring官方提供了很多starter, 第三方也可以定义starter。为了加以区分, starter从名称上进行了如下规范:

- Spring官方提供的starter名称为: spring-boot-starter-xxx, 例如Spring官方提供的spring-boot-starter-web
- 第三方提供的starter名称为: xxx-spring-boot-starter, 例如由mybatis提供的mybatis-spring-boot-starter

## starter原理

Spring Boot之所以能够帮我们简化项目的搭建和开发过程，主要是基于它提供的起步依赖和自动配置。

### 起步依赖

起步依赖，其实就是将具备某种功能的坐标打包到一起，可以简化依赖导入的过程。例如，我们导入spring-boot-starter-web这个starter，则和web开发相关的jar包都一起导入到项目中了

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

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd"
  xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
4     <!-- This module was also published with a richer model, Gradle metadata,
  -->
5     <!-- which should be used instead. Do not delete the following line which
  -->
6     <!-- is to indicate to Gradle or any Gradle module metadata file consumer
  -->
7     <!-- that they should prefer consuming it instead. -->
8     <!-- do_not_remove: published-with-gradle-metadata -->
9     <modelVersion>4.0.0</modelVersion>
10    <groupId>org.springframework.boot</groupId>
11    <artifactId>spring-boot-starter-web</artifactId>
12    <version>2.7.1</version>
13    <name>spring-boot-starter-web</name>
14    <description>Starter for building web, including RESTful, applications
  using Spring MVC. Uses Tomcat as the default embedded
  container</description>
15    <url>https://spring.io/projects/spring-boot</url>
16    <organization>
```

```

17     <name>Pivotal Software, Inc.</name>
18     <url>https://spring.io</url>
19 </organization>
20 <licenses>
21     <license>
22         <name>Apache License, Version 2.0</name>
23         <url>https://www.apache.org/licenses/LICENSE-2.0</url>
24     </license>
25 </licenses>
26 <developers>
27     <developer>
28         <name>Pivotal</name>
29         <email>info@pivotal.io</email>
30         <organization>Pivotal Software, Inc.</organization>
31         <organizationUrl>https://www.spring.io</organizationUrl>
32     </developer>
33 </developers>
34 <scm>
35     <connection>scm:git:git://github.com/spring-projects/spring-
boot.git</connection>
36     <developerConnection>scm:git:ssh://git@github.com/spring-
projects/spring-boot.git</developerConnection>
37     <url>https://github.com/spring-projects/spring-boot</url>
38 </scm>
39 <issueManagement>
40     <system>GitHub</system>
41     <url>https://github.com/spring-projects/spring-boot/issues</url>
42 </issueManagement>
43 <dependencies>
44     <dependency>
45         <groupId>org.springframework.boot</groupId>
46         <artifactId>spring-boot-starter</artifactId>
47         <version>2.7.1</version>
48         <scope>compile</scope>
49     </dependency>
50     <dependency>
51         <groupId>org.springframework.boot</groupId>
52         <artifactId>spring-boot-starter-json</artifactId>
53         <version>2.7.1</version>
54         <scope>compile</scope>
55     </dependency>
56     <dependency>
57         <groupId>org.springframework.boot</groupId>
58         <artifactId>spring-boot-starter-tomcat</artifactId>
59         <version>2.7.1</version>
60         <scope>compile</scope>
61     </dependency>
62     <dependency>
63         <groupId>org.springframework</groupId>
64         <artifactId>spring-web</artifactId>
65         <version>5.3.21</version>
66         <scope>compile</scope>
67     </dependency>
68     <dependency>
69         <groupId>org.springframework</groupId>
70         <artifactId>spring-webmvc</artifactId>
71         <version>5.3.21</version>
72         <scope>compile</scope>

```

```
73     </dependency>
74 </dependencies>
75 </project>
```

## 自动配置

自动配置，就是无须手动配置xml，自动配置并管理bean，可以简化开发过程。那么Spring Boot是如何完成自动配置的呢？

自动配置涉及到如下几个关键步骤：

- 基于Java代码的Bean配置
- 自动配置条件依赖
- Bean参数获取
- Bean的发现
- Bean的加载

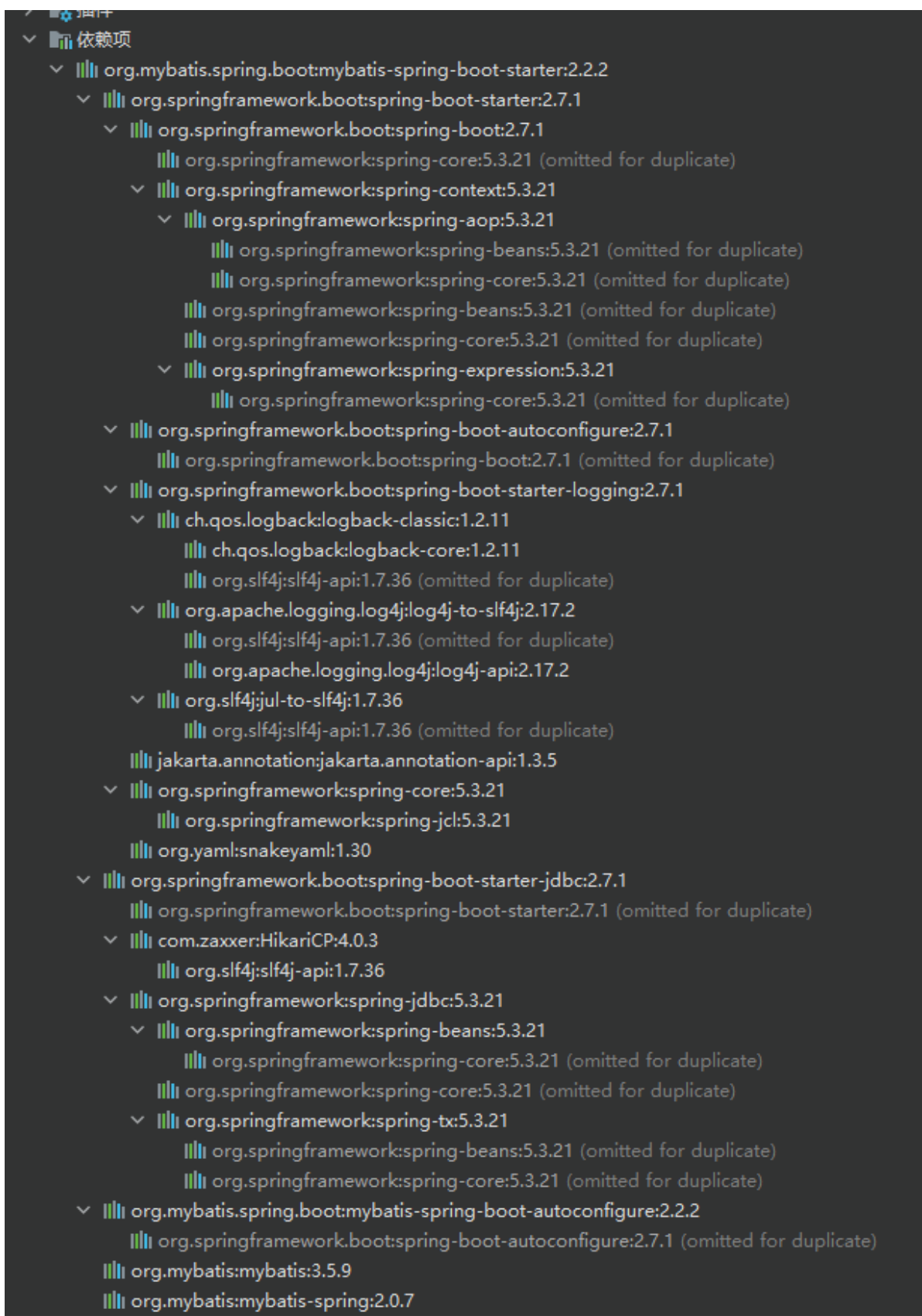
## 基于Java代码的Bean配置

当我们在项目中导入了mybatis-spring-boot-starter这个jar后，可以看到它包括了很多相关的jar包

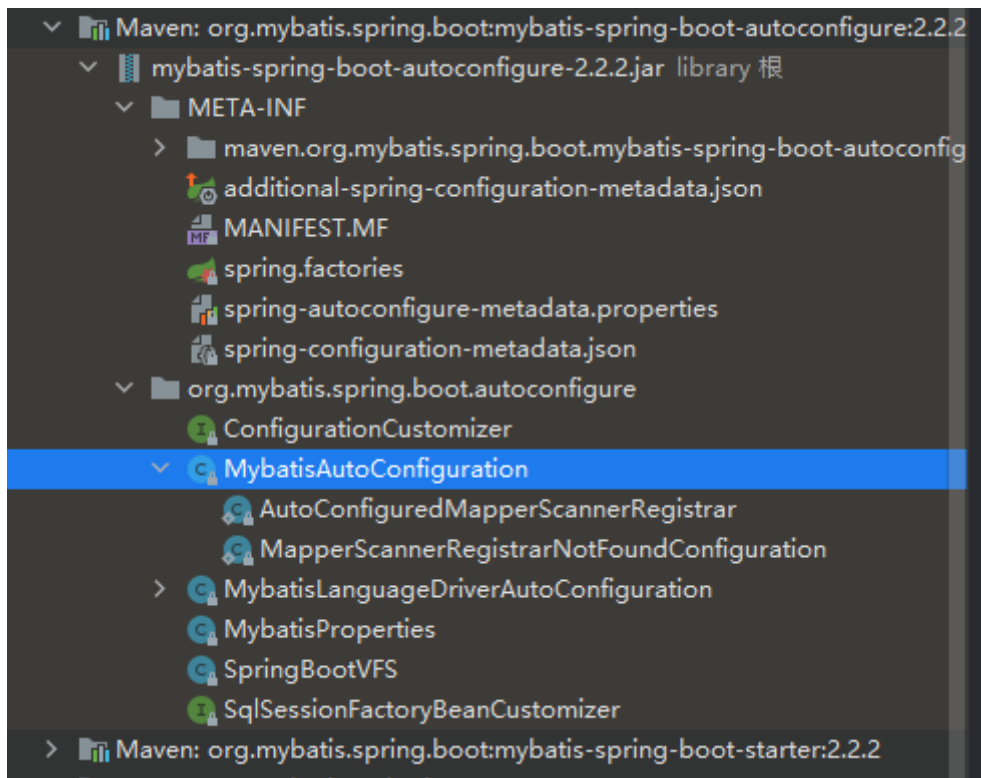
```
<dependencies>

  <!--spring-boot mybatis依赖-->
  <dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>2.2.2</version>
  </dependency>

</dependencies>
```



其中在mybatis-spring-boot-autoconfigure这个jar包中有如下一个**MybatisAutoConfiguration**自动配置类：



打开这个类，截取的关键代码如下：

```

Auto-Configuration for Mybatis. Contributes a SqlSessionFactory and a SqlSessionTemplate.
If org.mybatis.spring.annotation.MapperScan is used, or a configuration file is specified as a
property, those will be considered, otherwise this auto-configuration will attempt to register mappers
based on the interface definitions in or under the root auto-configuration package.
作者: Eddú Meléndez, Josh Long, Kazuki Shimizu, Eduardo Macarrón

@org.springframework.context.annotation.Configuration
@ConditionalOnClass({ SqlSessionFactory.class, SqlSessionFactoryBean.class })
@ConditionalOnSingleCandidate(DataSource.class)
@EnableConfigurationProperties(MybatisProperties.class)
@AutoConfigureAfter({ DataSourceAutoConfiguration.class, MybatisLanguageDriverAutoConfiguration.class })
public class MybatisAutoConfiguration implements InitializingBean {

    private static final Logger logger = LoggerFactory.getLogger(MybatisAutoConfiguration.class);

```

```

    }
}

@Bean
@ConditionalOnMissingBean
public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception {
    SqlSessionFactoryBean factory = new SqlSessionFactoryBean();
    factory.setDataSource(dataSource);
    factory.setVfs(SpringBootVFS.class);
    if (StringUtils.hasText(this.properties.getConfigLocation())) {
        factory.setConfigLocation(this.resourceLoader.getResource(this.properties
    )
    }
    applyConfiguration(factory);
}

```

```

10     }
11
12     @Bean
13     @ConditionalOnMissingBean
14     public SqlSessionFactory sqlSessionFactory(SqlSessionFactory sqlSessionFactory,
15         ExecutorType executorType = this.properties.getExecutorType();
16         if (executorType != null) {
17             return new SqlSessionFactory(sqlSessionFactory, executorType);
18         } else {
19             return new SqlSessionFactory(sqlSessionFactory);
20         }
21     }

```

**@Configuration**和**@Bean**这两个注解一起使用就可以创建一个基于java代码的配置类，可以用来替代传统的xml配置文件。

**@Configuration** 注解的类可以看作是能生产让Spring IoC容器管理的Bean实例的工厂。

**@Bean** 注解的方法返回的对象可以被注册到spring容器中。

所以上面的**MybatisAutoConfiguration**这个类，自动帮我们生成了SqlSessionFactory和SqlSessionTemplate这些Mybatis的重要实例并交给spring容器管理，从而完成bean的自动注册。

## 自动配置条件依赖

要完成自动配置是有依赖条件的。



注解	功能说明
@ConditionalOnBean	仅在当前上下文中存在某个bean时，才会实例化这个Bean
@ConditionalOnClass	某个class位于类路径上，才会实例化这个Bean
@ConditionalOnExpression	当表达式为true的时候，才会实例化这个Bean
@ConditionalOnMissingBean	仅在当前上下文中不存在某个bean时，才会实例化这个Bean
@ConditionalOnMissingClass	某个class在类路径上不存在的时候，才会实例化这个Bean
@ConditionalOnNotWebApplication	不是web应用时才会实例化这个Bean
@AutoConfigureAfter	在某个bean完成自动配置后实例化这个bean
@AutoConfigureBefore	在某个bean完成自动配置前实例化这个bean

## Bean参数获取

要完成mybatis的自动配置，需要我们在配置文件中提供数据源相关的配置参数，例如数据库驱动、连接url、数据库用户名、密码等。那么spring boot是如何读取yaml或者properties配置文件的属性来创建数据源对象的？

在我们导入mybatis-spring-boot-starter这个jar包后会传递过来一个spring-boot-autoconfigure包，在这个包中有一个自动配置类**DataSourceAutoConfiguration**

```

@AutoConfiguration(
    before = {SqlInitializationAutoConfiguration.class}
)
@ConditionalOnClass({DataSource.class, EmbeddedDatabaseType.class})
@ConditionalOnMissingBean(
    type = {"io.r2dbc.spi.ConnectionFactory"}
)
@EnableConfigurationProperties({DataSourceProperties.class})
@Import({DataSourcePoolMetadataProvidersConfiguration.class})
public class DataSourceAutoConfiguration {
    public DataSourceAutoConfiguration() {
    }
}

```

我们可以看到这个类上加入了**EnableConfigurationProperties**这个注解，继续跟踪源码到**DataSourceProperties**这个类

```
@ConfigurationProperties(
    prefix = "spring.datasource"
)
public class DataSourceProperties implements BeanClassLoaderAware, InitializingBean {
    private ClassLoader classLoader;
    private boolean generateUniqueName = true;
    private String name;
    private Class<? extends DataSource> type;
    private String driverClassName;
    private String url;
    private String username;
    private String password;
    private String jndiName;
    private EmbeddedDatabaseConnection embeddedDatabaseConnection;
    private DataSourceProperties.Xa xa = new DataSourceProperties.Xa();
    private String uniqueName;
```

可以看到这个类上加入了**ConfigurationProperties**注解，这个注解的作用就是把yml或者properties配置文件中的配置参数信息封装到**ConfigurationProperties**注解标注的bean(即**DataSourceProperties**)的相应属性上

**@EnableConfigurationProperties**注解的作用是使**@ConfigurationProperties**注解生效。

## Bean的发现

spring boot默认扫描启动类所在的包下的主类与子类的所有组件，但并没有包括依赖包中的类，那么依赖包中的bean是如何被发现和加载的？

我们需要从Spring Boot项目的启动类开始跟踪，在启动类上我们一般会加入SpringBootApplication注解

```
import ...

@SpringBootApplication
public class SpringBootStarterDemoApplication
{

    public static void main(String[] args)
```

```
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(
    excludeFilters = {@Filter(
        type = FilterType.CUSTOM,
        classes = {TypeExcludeFilter.class}
    ), @Filter(
        type = FilterType.CUSTOM,
        classes = {AutoConfigurationExcludeFilter.class}
    )}
)
public @interface SpringBootApplication {
    @AliasFor(
        annotation = EnableAutoConfiguration.class
    )
    Class<?>[] exclude() default {};
```

**SpringBootConfiguration**：作用就相当于**Configuration**注解，被注解的类将成为一个bean配置类

**ComponentScan**：作用就是自动扫描并加载符合条件的组件，最终将这些bean加载到spring容器中

**EnableAutoConfiguration**：这个注解很重要，借助**@Import**的支持，收集和注册依赖包中相关的bean定义

继续跟踪**EnableAutoConfiguration**注解源码：

```

@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@AutoConfigurationPackage
@Import({AutoConfigurationImportSelector.class})
public @interface EnableAutoConfiguration {
    String ENABLED_OVERRIDE_PROPERTY = "spring.boot.enableautoconfiguration";

    Class<?>[] exclude() default {};

    String[] excludeName() default {};
}

```

@EnableAutoConfiguration注解引入了@Import这个注解。

Import：导入需要自动配置的组件

继续跟踪AutoConfigurationImportSelector类源码：

```

protected List<String> getCandidateConfigurations(AnnotationMetadata metadata, AnnotationAttributes attributes) {
    List<String> configurations = new ArrayList(SpringFactoriesLoader.loadFactoryNames(this.getSpringFactoriesLoaderFactoriesPackage(),
        ImportCandidates.load(AutoConfiguration.class, this.getBeanClassLoader()).forEach(configurations::add);
    Assert.notEmpty(configurations, "message: \"No auto configuration classes found in META-INF/spring.factories nor in ME
    return configurations;
}

```

AutoConfigurationImportSelector类的getCandidateConfigurations方法中的调用了SpringFactoriesLoader类的loadFactoryNames方法

```

public static List<String> loadFactoryNames(Class<?> factoryType, @Nullable ClassLoader classLoader) {
    ClassLoader classLoaderToUse = classLoader;
    if (classLoader == null) {
        classLoaderToUse = SpringFactoriesLoader.class.getClassLoader();
    }

    String factoryTypeName = factoryType.getName();
    return (List)loadSpringFactories(classLoaderToUse).getOrDefault(factoryTypeName, Collections.emptyList());
}

private static Map<String, List<String>> loadSpringFactories(ClassLoader classLoader) {
    Map<String, List<String>> result = (Map)cache.get(classLoader);
    if (result != null) {
        return result;
    } else {
        HashMap result = new HashMap();

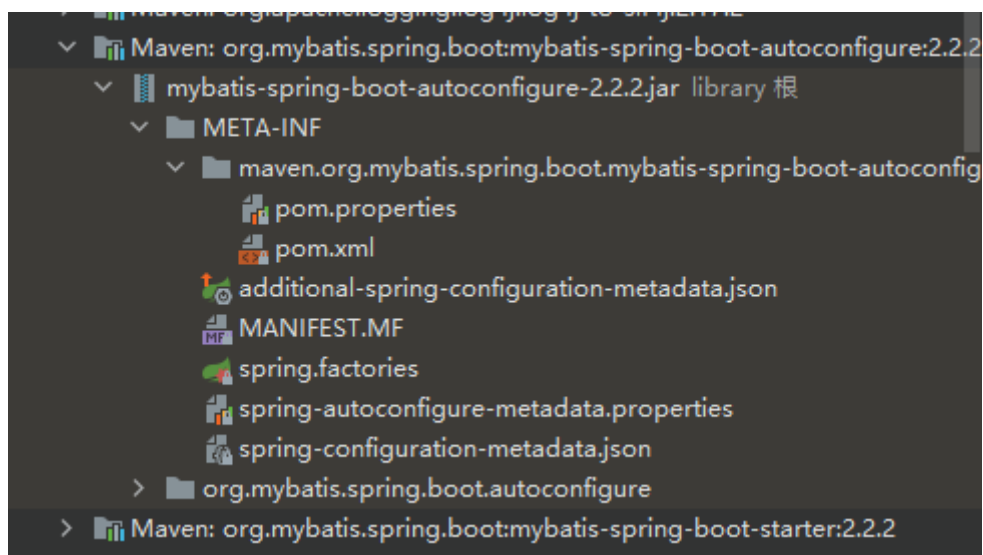
        try {
            Enumeration urls = classLoader.getResources( name: "META-INF/spring.factories");

            while(urls.hasMoreElements()) {
                URL url = (URL)urls.nextElement();
                UrlResource resource = new UrlResource(url);
                Properties properties = PropertiesLoaderUtils.loadProperties(resource);
                Iterator var6 = properties.entrySet().iterator();

                while(var6.hasNext()) {
                    Entry<?, ?> entry = (Entry)var6.next();

```

**SpringFactoriesLoader**的**loadFactoryNames**静态方法可以从所有的jar包中读取META-INF/spring.factories文件，而自动配置的类就在这个文件中进行配置：



spring.factories文件内容如下：

```
# Auto Configure
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
org.mybatis.spring.boot.autoconfigure.MybatisLanguageDriverAutoConfiguration,\
org.mybatis.spring.boot.autoconfigure.MybatisAutoConfiguration
```

```
1 # Auto Configure
2 org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
3 org.mybatis.spring.boot.autoconfigure.MybatisLanguageDriverAutoConfiguration,\
  \
4 org.mybatis.spring.boot.autoconfigure.MybatisAutoConfiguration
```

这样Spring Boot就可以加载到**MybatisAutoConfiguration**这个配置类了

## Bean的加载

在Spring Boot应用中要让一个普通类交给Spring容器管理，通常有以下方法：

- 使用 @Configuration与@Bean 注解
- 使用@Controller @Service @Repository @Component 注解标注该类并且启用 @ComponentScan自动扫描
- 使用@Import 方法

其中Spring Boot实现自动配置使用的是@Import注解这种方式，AutoConfigurationImportSelector类的selectImports方法返回一组从META-INF/spring.factories文件中读取的bean的全类名，这样Spring Boot就可以加载到这些Bean并完成实例的创建工作。

# 自定义starter

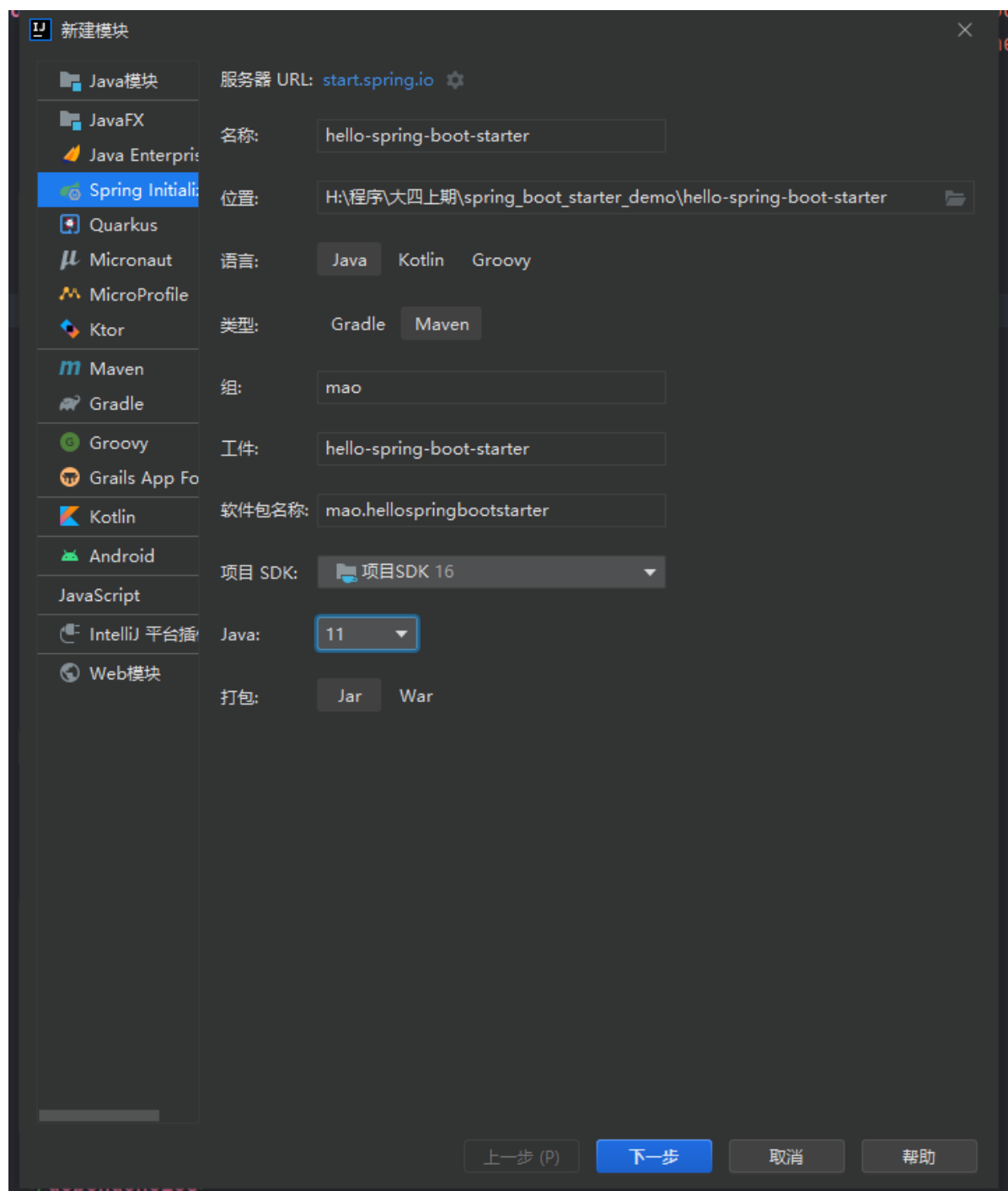
## 案例一

### 开发starter

#### 第一步：初始化项目

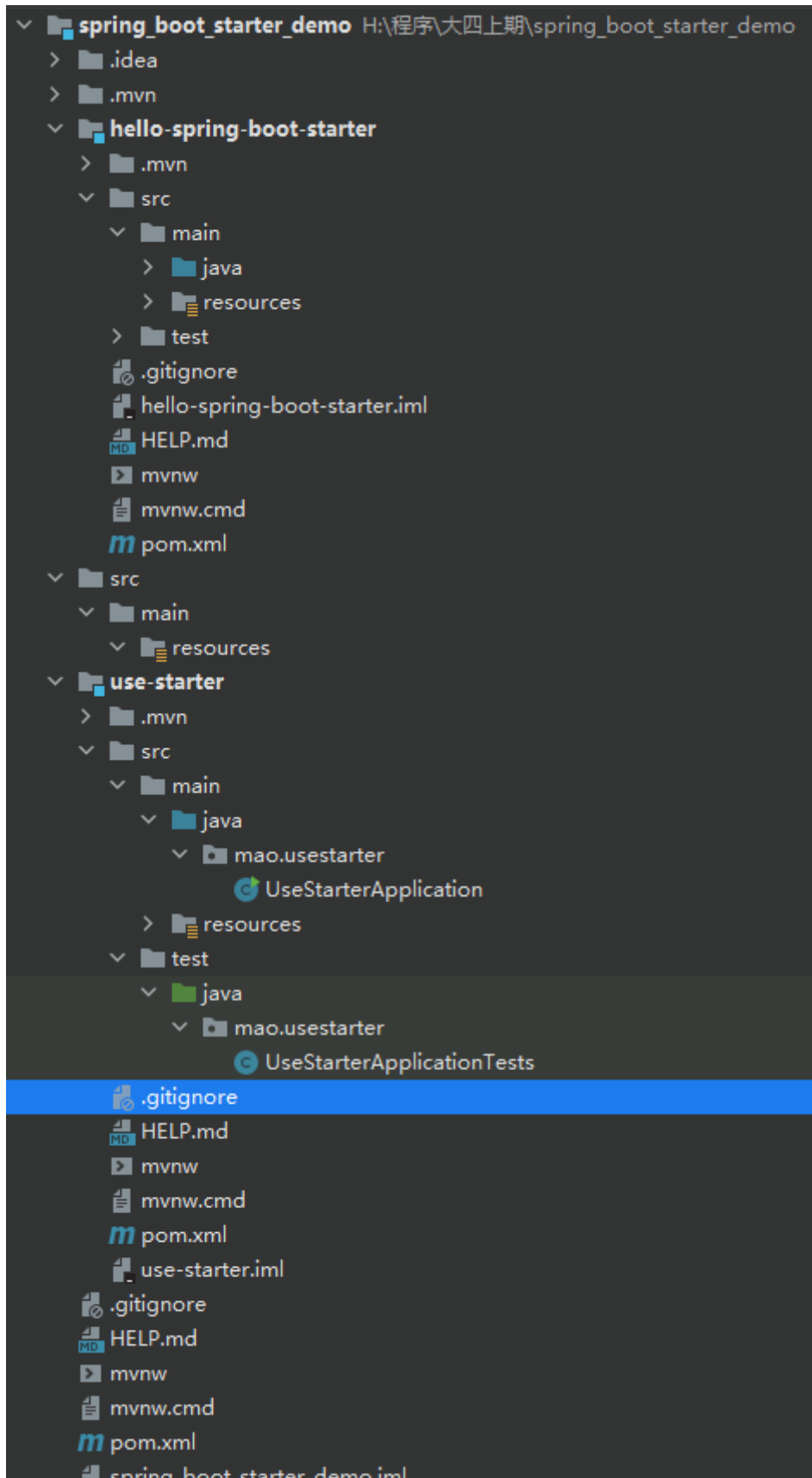
创建一个名字为spring\_boot\_starter\_demo的父工程

创建一个名字为hello-spring-boot-starter的子工程，此工程用于开发starter给使用者使用



创建一个名字为use-starter的子工程，此工程用于使用之前的starter

目录结构如下





父工程的pom文件:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5   https://maven.apache.org/xsd/maven-4.0.0.xsd">
6
7   <parent>
8     <groupId>org.springframework.boot</groupId>
9     <artifactId>spring-boot-starter-parent</artifactId>
10    <version>2.7.1</version>
11    <relativePath/>
12  </parent>
13
14
15  <groupId>mao</groupId>
16  <artifactId>spring_boot_starter_demo</artifactId>
17  <version>0.0.1</version>
18  <name>spring_boot_starter_demo</name>
19  <description>spring_boot_starter_demo</description>
20  <packaging>pom</packaging>
21
22
23  <properties>
24    <java.version>11</java.version>
25  </properties>
26
27  <modules>
28
29    <module>hello-spring-boot-starter</module>
30    <module>use-starter</module>
31
32  </modules>
33
34  <dependencies>
35
36
37  </dependencies>
38
39  <dependencyManagement>
40
41    <dependencies>
42
43
44    </dependencies>
45
46  </dependencyManagement>
47
48  <build>
```

```

49     <plugins>
50         <plugin>
51             <groupId>org.springframework.boot</groupId>
52             <artifactId>spring-boot-maven-plugin</artifactId>
53         </plugin>
54     </plugins>
55 </build>
56
57 </project>

```

hello-spring-boot-starter的pom文件:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5      https://maven.apache.org/xsd/maven-4.0.0.xsd">
6      <modelVersion>4.0.0</modelVersion>
7
8      <parent>
9          <artifactId>spring_boot_starter_demo</artifactId>
10         <groupId>mao</groupId>
11         <version>0.0.1</version>
12     </parent>
13
14     <artifactId>hello-spring-boot-starter</artifactId>
15     <version>0.0.1</version>
16     <name>hello-spring-boot-starter</name>
17     <description>hello-spring-boot-starter</description>
18
19     <properties>
20         <java.version>11</java.version>
21     </properties>
22
23     <dependencies>
24
25         <dependency>
26             <groupId>org.springframework.boot</groupId>
27             <artifactId>spring-boot-starter-web</artifactId>
28         </dependency>
29
30         <!--spring boot starter开发依赖-->
31
32         <dependency>
33             <groupId>org.springframework.boot</groupId>
34             <artifactId>spring-boot-starter</artifactId>
35         </dependency>
36
37         <dependency>
38             <groupId>org.springframework.boot</groupId>
39             <artifactId>spring-boot-autoconfigure</artifactId>
40         </dependency>

```

```

41         <groupId>org.springframework.boot</groupId>
42         <artifactId>spring-boot-configuration-processor</artifactId>
43     </dependency>
44
45 </dependencies>
46
47 <build>
48     <plugins>
49         <plugin>
50             <groupId>org.springframework.boot</groupId>
51             <artifactId>spring-boot-maven-plugin</artifactId>
52             <configuration>
53                 <skip>true</skip>
54             </configuration>
55         </plugin>
56     </plugins>
57 </build>
58
59 </project>
60

```

use-starter的pom文件:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5  https://maven.apache.org/xsd/maven-4.0.0.xsd">
6
7      <modelVersion>4.0.0</modelVersion>
8      <parent>
9
10         <artifactId>spring_boot_starter_demo</artifactId>
11         <groupId>mao</groupId>
12         <version>0.0.1</version>
13     </parent>
14
15     <artifactId>use-starter</artifactId>
16     <version>0.0.1-SNAPSHOT</version>
17     <name>use-starter</name>
18     <description>use-starter</description>
19
20     <properties>
21         <java.version>11</java.version>
22     </properties>
23
24     <dependencies>
25
26         <dependency>
27             <groupId>org.springframework.boot</groupId>
28             <artifactId>spring-boot-starter-web</artifactId>
29




```

```

30     <dependency>
31         <groupId>org.springframework.boot</groupId>
32         <artifactId>spring-boot-starter-test</artifactId>
33         <scope>test</scope>
34     </dependency>
35
36 </dependencies>
37
38 <build>
39     <plugins>
40         <plugin>
41             <groupId>org.springframework.boot</groupId>
42             <artifactId>spring-boot-maven-plugin</artifactId>
43         </plugin>
44     </plugins>
45 </build>
46
47 </project>

```

```

>  hello-spring-boot-starter
>  spring_boot_starter_demo (root)
>  use-starter

```

打包测试项目是否配置正确

```

1  [INFO] Scanning for projects...
2  [INFO] -----
3  [INFO] Reactor Build Order:
4  [INFO]
5  [INFO] spring_boot_starter_demo
6  [INFO] hello-spring-boot-starter
7  [INFO] use-starter
8  [INFO]
9  [INFO] -----< mao:spring_boot_starter_demo >-----
10 [INFO] Building spring_boot_starter_demo 0.0.1
11 [INFO] -----[ pom ]-----
12 [INFO]
13 [INFO] --- spring-boot-maven-plugin:2.7.1:repackage (repackage) @
14 [INFO] spring_boot_starter_demo ---

```

```

15 [INFO] -----< mao:hello-spring-boot-starter >-----
16 [INFO] Building hello-spring-boot-starter 0.0.1
17 [INFO] -----[ jar ]-----
18 [INFO]
19 [INFO] --- maven-resources-plugin:3.2.0:resources (default-resources) @
hello-spring-boot-starter ---
20 [INFO] Using 'UTF-8' encoding to copy filtered resources.
21 [INFO] Using 'UTF-8' encoding to copy filtered properties files.
22 [INFO] Copying 0 resource
23 [INFO] Copying 1 resource
24 [INFO]
25 [INFO] --- maven-compiler-plugin:3.10.1:compile (default-compile) @ hello-
spring-boot-starter ---
26 [INFO] Nothing to compile - all classes are up to date
27 [INFO]
28 [INFO] --- maven-resources-plugin:3.2.0:testResources (default-
testResources) @ hello-spring-boot-starter ---
29 [INFO] Using 'UTF-8' encoding to copy filtered resources.
30 [INFO] Using 'UTF-8' encoding to copy filtered properties files.
31 [INFO] skip non existing resourceDirectory H:\程序\大四上期
\spring_boot_starter_demo\hello-spring-boot-starter\src\test\resources
32 [INFO]
33 [INFO] --- maven-compiler-plugin:3.10.1:testCompile (default-testCompile) @
hello-spring-boot-starter ---
34 [INFO] No sources to compile
35 [INFO]
36 [INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ hello-spring-
boot-starter ---
37 [INFO] Tests are skipped.
38 [INFO]
39 [INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ hello-spring-boot-
starter ---
40 [INFO]
41 [INFO] --- spring-boot-maven-plugin:2.7.1:repackage (repackage) @ hello-
spring-boot-starter ---
42 [INFO]
43 [INFO] -----< mao:use-starter >-----
44 [INFO] Building use-starter 0.0.1-SNAPSHOT
45 [INFO] -----[ jar ]-----
46 [INFO]
47 [INFO] --- maven-resources-plugin:3.2.0:resources (default-resources) @ use-
starter ---
48 [INFO] Using 'UTF-8' encoding to copy filtered resources.
49 [INFO] Using 'UTF-8' encoding to copy filtered properties files.
50 [INFO] Copying 1 resource
51 [INFO] Copying 0 resource
52 [INFO]
53 [INFO] --- maven-compiler-plugin:3.10.1:compile (default-compile) @ use-
starter ---
54 [INFO] Changes detected - recompiling the module!
55 [INFO] Compiling 2 source files to H:\程序\大四上期
\spring_boot_starter_demo\use-starter\target\classes

```

```

56 [INFO]
57 [INFO] --- maven-resources-plugin:3.2.0:testResources (default-
testResources) @ use-starter ---
58 [INFO] Using 'UTF-8' encoding to copy filtered resources.
59 [INFO] Using 'UTF-8' encoding to copy filtered properties files.
60 [INFO] skip non existing resourceDirectory H:\程序\大四上期
\spring_boot_starter_demo\use-starter\src\test\resources
61 [INFO]
62 [INFO] --- maven-compiler-plugin:3.10.1:testCompile (default-testCompile) @
use-starter ---
63 [INFO] Changes detected - recompiling the module!
64 [INFO] Compiling 1 source file to H:\程序\大四上期
\spring_boot_starter_demo\use-starter\target\test-classes
65 [INFO]
66 [INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ use-starter --
-
67 [INFO] Tests are skipped.
68 [INFO]
69 [INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ use-starter ---
70 [INFO] Building jar: H:\程序\大四上期\spring_boot_starter_demo\use-
starter\target\use-starter-0.0.1-SNAPSHOT.jar
71 [INFO]
72 [INFO] --- spring-boot-maven-plugin:2.7.1:repackage (repackage) @ use-
starter ---
73 [INFO] Replacing main artifact with repackaged archive
74 [INFO] -----
---
75 [INFO] Reactor Summary:
76 [INFO]
77 [INFO] spring_boot_starter_demo 0.0.1 ..... SUCCESS [ 0.833
s]
78 [INFO] hello-spring-boot-starter 0.0.1 ..... SUCCESS [ 1.282
s]
79 [INFO] use-starter 0.0.1-SNAPSHOT ..... SUCCESS [ 1.738
s]
80 [INFO] -----
---
81 [INFO] BUILD SUCCESS
82 [INFO] -----
---
83 [INFO] Total time: 4.189 s
84 [INFO] Finished at: 2022-10-24T22:02:54+08:00
85 [INFO] -----
---

```

没有报错

## 第二步：创建配置属性类HelloConfigProperties

```
1 package mao.hellospringbootstarter.config;
2
3 import org.springframework.boot.context.properties.ConfigurationProperties;
4
5 /**
6  * Project name(项目名称): spring_boot_starter_demo
7  * Package(包名): mao.hellospringbootstarter.config
8  * Class(类名): HelloConfigProperties
9  * Author(作者): mao
10 * Author QQ: 1296193245
11 * Github: https://github.com/maomao124/
12 * Date(创建日期): 2022/10/24
13 * Time(创建时间): 20:23
14 * Version(版本): 1.0
15 * Description(描述): 无
16 */
17
18
19 @ConfigurationProperties(prefix = "hello")
20 public class HelloConfigProperties
21 {
22     /**
23      * 名字
24      */
25     private String name;
26     /**
27      * 性
28      */
29     private String sex;
30     /**
31      * 年龄
32      */
33     private String age;
34     /**
35      * 地址
36      */
37     private String address;
38
39     /**
40      * Instantiates a new Hello config properties.
41      */
42     public HelloConfigProperties()
43     {
44     }
45
46     /**
47      * Instantiates a new Hello config properties.
48      *
49      * @param name the name
50      * @param sex the sex
51      * @param age the age
52      * @param address the address
53      */
54 }
```

```

54     public HelloConfigProperties(String name, String sex, String age,
String address)
55     {
56         this.name = name;
57         this.sex = sex;
58         this.age = age;
59         this.address = address;
60     }
61
62     /**
63      * Gets name.
64      *
65      * @return the name
66      */
67     public String getName()
68     {
69         return name;
70     }
71
72     /**
73      * Sets name.
74      *
75      * @param name the name
76      */
77     public void setName(String name)
78     {
79         this.name = name;
80     }
81
82     /**
83      * Gets sex.
84      *
85      * @return the sex
86      */
87     public String getSex()
88     {
89         return sex;
90     }
91
92     /**
93      * Sets sex.
94      *
95      * @param sex the sex
96      */
97     public void setSex(String sex)
98     {
99         this.sex = sex;
100    }
101
102    /**
103     * Gets age.
104     *
105     * @return the age
106     */
107    public String getAge()
108    {
109        return age;
110    }

```



```

111
112     /**
113      * Sets age.
114      *
115      * @param age the age
116      */
117     public void setAge(String age)
118     {
119         this.age = age;
120     }
121
122     /**
123      * Gets address.
124      *
125      * @return the address
126      */
127     public String getAddress()
128     {
129         return address;
130     }
131
132     /**
133      * Sets address.
134      *
135      * @param address the address
136      */
137     public void setAddress(String address)
138     {
139         this.address = address;
140     }
141
142     @Override
143     @SuppressWarnings("all")
144     public String toString()
145     {
146         final StringBuilder stringBuilder = new StringBuilder();
147         stringBuilder.append("name: ").append(name).append('\n');
148         stringBuilder.append("sex: ").append(sex).append('\n');
149         stringBuilder.append("age: ").append(age).append('\n');
150         stringBuilder.append("address: ").append(address).append('\n');
151         return stringBuilder.toString();
152     }
153 }

```

### 第三步: 创建HelloService

```

1 package mao.hellospringbootstarter.service;
2
3 import mao.hellospringbootstarter.config.HelloConfigProperties;
4 import org.springframework.beans.factory.annotation.Autowired;

```

```

5  import org.springframework.stereotype.Service;
6
7  /**
8   * Project name(项目名称): spring_boot_starter_demo
9   * Package(包名): mao.hellospringbootstarter.service
10  * Class(类名): HelloService
11  * Author(作者): mao
12  * Author QQ: 1296193245
13  * GitHub: https://github.com/maomao124/
14  * Date(创建日期): 2022/10/24
15  * Time(创建时间): 20:33
16  * Version(版本): 1.0
17  * Description(描述): 无
18  */
19
20  //@Service
21  public class HelloService
22  {
23      /**
24       * 配置属性类
25       */
26      private final HelloConfigProperties helloConfigProperties;
27
28      @Autowired
29      public HelloService(HelloConfigProperties helloConfigProperties)
30      {
31          this.helloConfigProperties = helloConfigProperties;
32      }
33
34      /**
35       * hello
36       *
37       * @return {@link String}
38       */
39      public String hello()
40      {
41          return "你好! 我的名字是" + helloConfigProperties.getName() +
42              ",年龄是: " + helloConfigProperties.getAge() +
43              ",地址是: " + helloConfigProperties.getAddress() +
44              ",性别是: " + helloConfigProperties.getSex();
45      }
46  }
47

```

#### 第四步：创建自动配置类HelloServiceAutoConfiguration

```

1  package mao.hellospringbootstarter.config;
2
3  import mao.hellospringbootstarter.service.HelloService;
4  import org.slf4j.Logger;
5  import org.slf4j.LoggerFactory;
6  import org.springframework.beans.factory.annotation.Autowired;

```

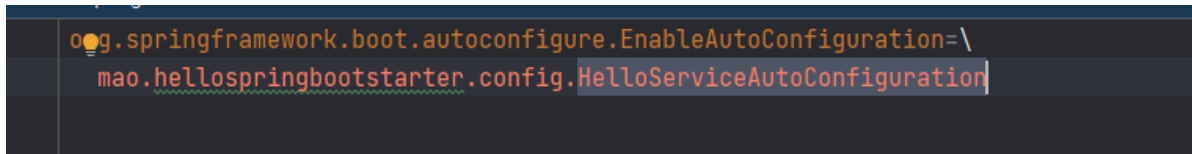
```

7  import
   org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
8  import
   org.springframework.boot.context.properties.EnableConfigurationProperties;
9  import org.springframework.context.annotation.Bean;
10 import org.springframework.context.annotation.ComponentScan;
11 import org.springframework.context.annotation.Configuration;
12
13 import javax.annotation.PostConstruct;
14
15 /**
16  * Project name(项目名称): spring_boot_starter_demo
17  * Package(包名): mao.hellospringbootstarter.config
18  * Class(类名): HelloServiceAutoConfiguration
19  * Author(作者): mao
20  * Author QQ: 1296193245
21  * GitHub: https://github.com/maomao124/
22  * Date(创建日期): 2022/10/24
23  * Time(创建时间): 20:38
24  * Version(版本): 1.0
25  * Description(描述): 无
26  */
27
28 @Configuration
29 @EnableConfigurationProperties(HelloConfigProperties.class)
30 //@ComponentScan(basePackageClasses = {HelloService.class})
31 public class HelloServiceAutoConfiguration
32 {
33
34     /**
35      * 日志
36      */
37     private static final Logger log =
38         LoggerFactory.getLogger(HelloServiceAutoConfiguration.class);
39
40     @Bean
41     @ConditionalOnMissingBean
42     public HelloService helloService(@Autowired HelloConfigProperties
43         helloConfigProperties)
44     {
45         log.info("初始化bean:HelloService");
46         return new HelloService(helloConfigProperties);
47     }
48
49     @PostConstruct
50     public void init()
51     {
52         log.info("初始化bean");
53     }
54 }

```

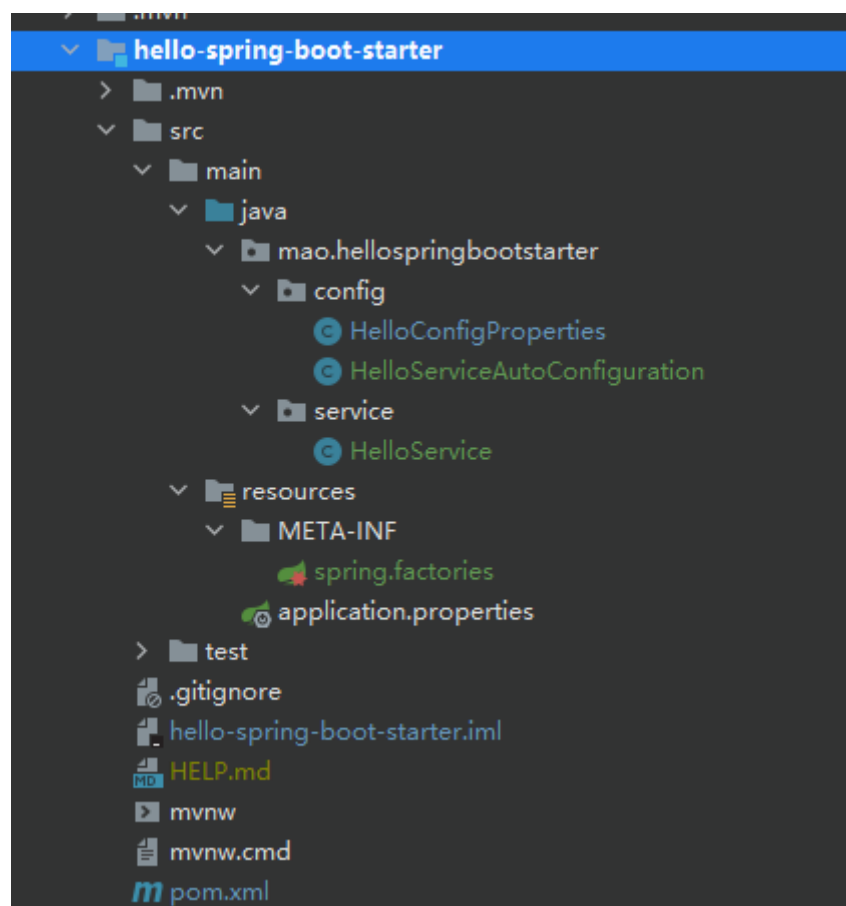
#### 第五步：在resources目录下创建META-INF/spring.factories

```
1 org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
2   mao.hellospringbootstarter.config.HelloServiceAutoConfiguration
```

A screenshot of an IDE editor showing the content of the `spring.factories` file. The text is: `org.springframework.boot.autoconfigure.EnableAutoConfiguration=\` on the first line, and `mao.hellospringbootstarter.config.HelloServiceAutoConfiguration` on the second line. The second line is highlighted with a blue selection bar.

至此starter已经开发完成了，可以将当前starter安装到本地maven仓库供其他应用来使用

结构：



## 使用starter

### 第一步：在use-starter子工程中修改pom文件

导入hello-spring-boot-starter

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5       https://maven.apache.org/xsd/maven-4.0.0.xsd">
6   <modelVersion>4.0.0</modelVersion>
7   <parent>
8     <artifactId>spring_boot_starter_demo</artifactId>
9     <groupId>mao</groupId>
10    <version>0.0.1</version>
11  </parent>
12
13  <artifactId>use-starter</artifactId>
14  <version>0.0.1-SNAPSHOT</version>
15  <name>use-starter</name>
16  <description>use-starter</description>
17
18  <properties>
19    <java.version>11</java.version>
20  </properties>
21
22
23  <dependencies>
24
25    <dependency>
26      <groupId>org.springframework.boot</groupId>
27      <artifactId>spring-boot-starter-web</artifactId>
28    </dependency>
29
30    <dependency>
31      <groupId>org.springframework.boot</groupId>
32      <artifactId>spring-boot-starter-test</artifactId>
33      <scope>test</scope>
34    </dependency>
35
36    <dependency>
37      <groupId>mao</groupId>
38      <artifactId>hello-spring-boot-starter</artifactId>
39      <version>0.0.1</version>
40    </dependency>
41
42  </dependencies>
43
44  <build>
```

```

45         <plugins>
46             <plugin>
47                 <groupId>org.springframework.boot</groupId>
48                 <artifactId>spring-boot-maven-plugin</artifactId>
49             </plugin>
50         </plugins>
51     </build>
52
53 </project>

```

## 第二步：创建application.yml文件添加配置

```

1  server:
2      port: 9090
3
4  hello:
5      name: 张三
6      sex: 男
7      age: 18
8      address: 中国

```

## 第三步：创建HelloController

```

1  package mao.usestarter.controller;
2
3  import mao.hellospringbootstarter.service.HelloService;
4  import org.springframework.beans.factory.annotation.Autowired;
5  import org.springframework.web.bind.annotation.GetMapping;
6  import org.springframework.web.bind.annotation.RestController;
7
8  /**
9   * Project name(项目名称): spring_boot_starter_demo
10  * Package(包名): mao.usestarter.controller
11  * Class(类名): HelloController
12  * Author(作者): mao
13  * Author QQ: 1296193245
14  * GitHub: https://github.com/maomao124/
15  * Date(创建日期): 2022/10/24
16  * Time(创建时间): 20:55
17  * Version(版本): 1.0
18  * Description(描述): 无
19  */

```

```

20
21 @RestController
22 public class HelloController
23 {
24     @Autowired
25     private HelloService helloService;
26
27
28     @GetMapping("/test")
29     public String test()
30     {
31         return helloService.hello();
32     }
33 }

```

#### 第四步：启动项目

```

1  .   ____          _            __ _ _
2  /\ /   _ __   ___| | _____| | | |
3  ( ( \   | '_ \ / _ \ |/ ___ \| | | |
4  \/ /   | |_) | | | | |___| | | |
5  ' /   | |_) | | | | |___| | | |
6  =====|_|=====|_|_|_\/_/_/_/_/
7  :: Spring Boot ::                (v2.7.1)
8
9  2022-10-24 22:09:13.951 INFO 12768 --- [           main]
mao.usestarter.UseStarterApplication : Starting UseStarterApplication
using Java 16.0.2 on mao with PID 12768 (H:\程序\大四上期
\spring_boot_starter_demo\use-starter\target\classes started by mao in H:\程
序\大四上期\spring_boot_starter_demo)
10 2022-10-24 22:09:13.953 INFO 12768 --- [           main]
mao.usestarter.UseStarterApplication : No active profile set, falling
back to 1 default profile: "default"
11 2022-10-24 22:09:14.641 INFO 12768 --- [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s):
8086 (http)
12 2022-10-24 22:09:14.649 INFO 12768 --- [           main]
o.apache.catalina.core.StandardService : Starting service [Tomcat]
13 2022-10-24 22:09:14.649 INFO 12768 --- [           main]
org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
Tomcat/9.0.64]
14 2022-10-24 22:09:14.723 INFO 12768 --- [           main] o.a.c.c.C.
[Tomcat].[localhost].[/] : Initializing Spring embedded
WebApplicationContext
15 2022-10-24 22:09:14.723 INFO 12768 --- [           main]
w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
initialization completed in 722 ms
16 2022-10-24 22:09:14.759 INFO 12768 --- [           main]
m.h.c.HelloServiceAutoConfiguration : 初始化bean

```

```
17 2022-10-24 22:09:14.768 INFO 12768 --- [           main]
    m.h.c.HelloServiceAutoConfiguration : 初始化bean:HelloService
18 2022-10-24 22:09:14.982 INFO 12768 --- [           main]
    o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8086
    (http) with context path ''
19 2022-10-24 22:09:14.991 INFO 12768 --- [           main]
    mao.usestarter.UseStarterApplication : Started UseStarterApplication in
    1.347 seconds (JVM running for 1.808)
```

### 第五步：访问

<http://localhost:8086/test>

---

你好! 我的名字是张三,年龄是: 18,地址是: 中国,性别是: 男

### 第六步：更改配置

```
1 server:
2   port: 8086
3
4 hello:
5   name: 张三
6   sex: 男
7   age: 21
8   address: 中国
```

### 第七步：重启服务器，再次访问

---

你好! 我的名字是张三,年龄是: 21,地址是: 中国,性别是: 男



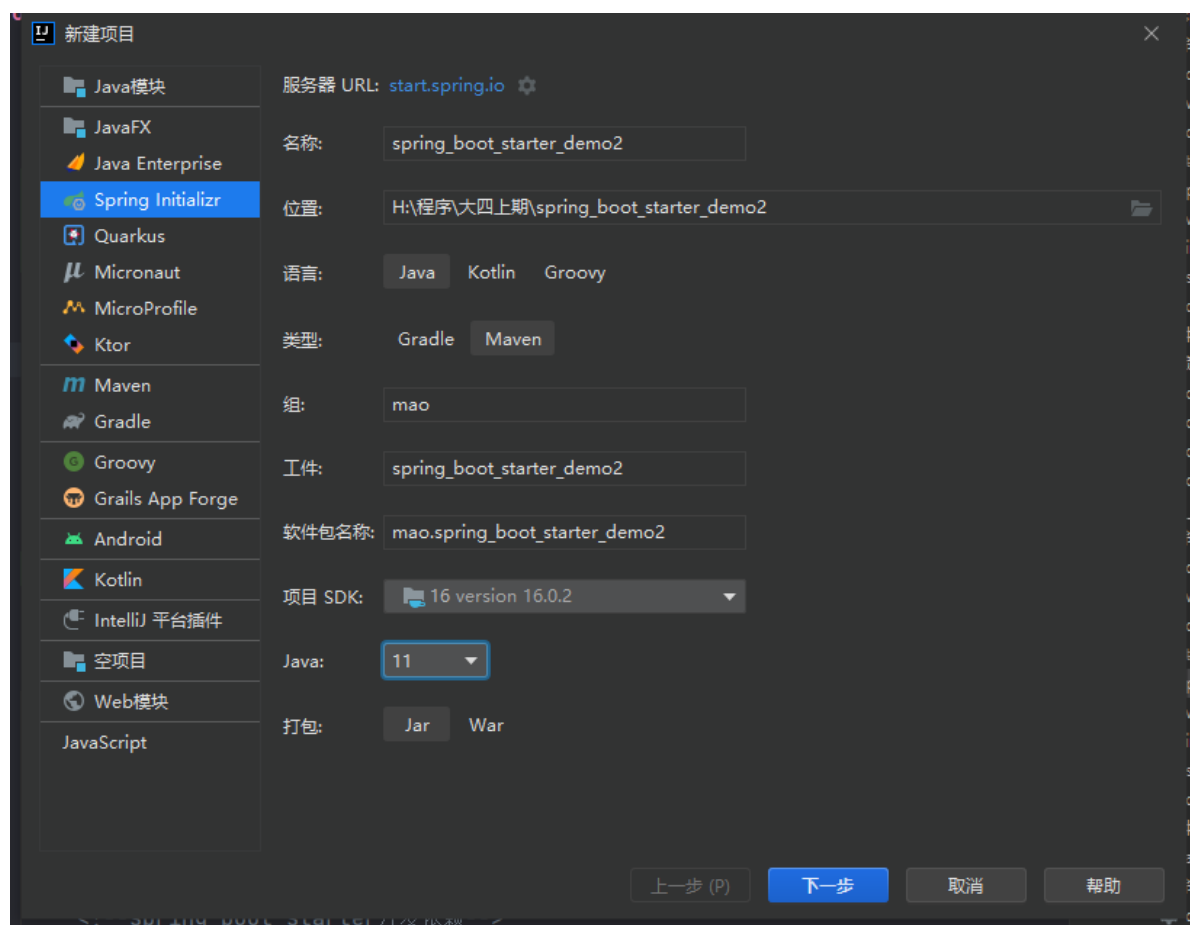
## 案例二

说明：通过自动配置来创建一个拦截器对象，通过此拦截器对象来实现记录日志功能

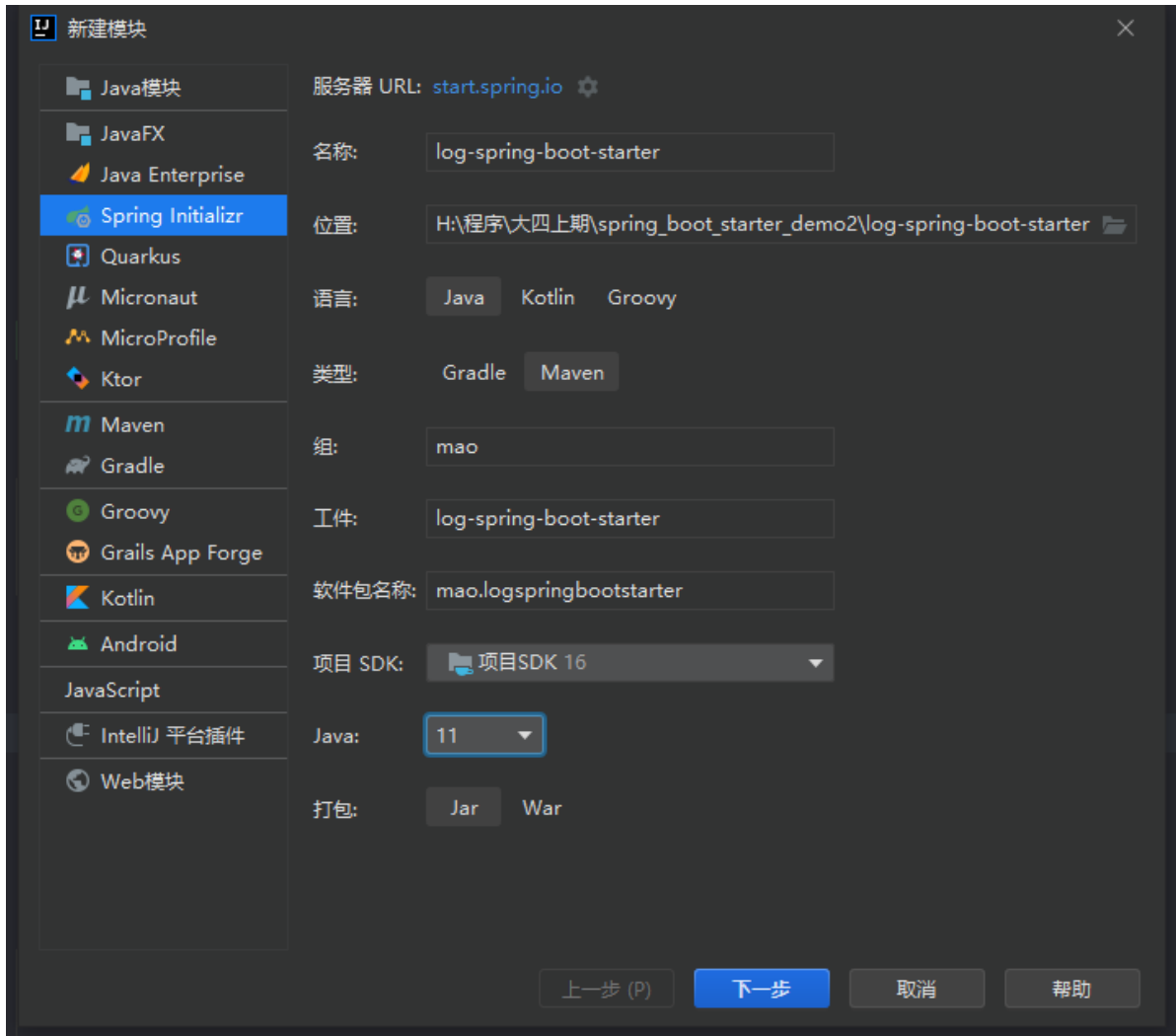
## 开发starter

### 第一步：初始化项目

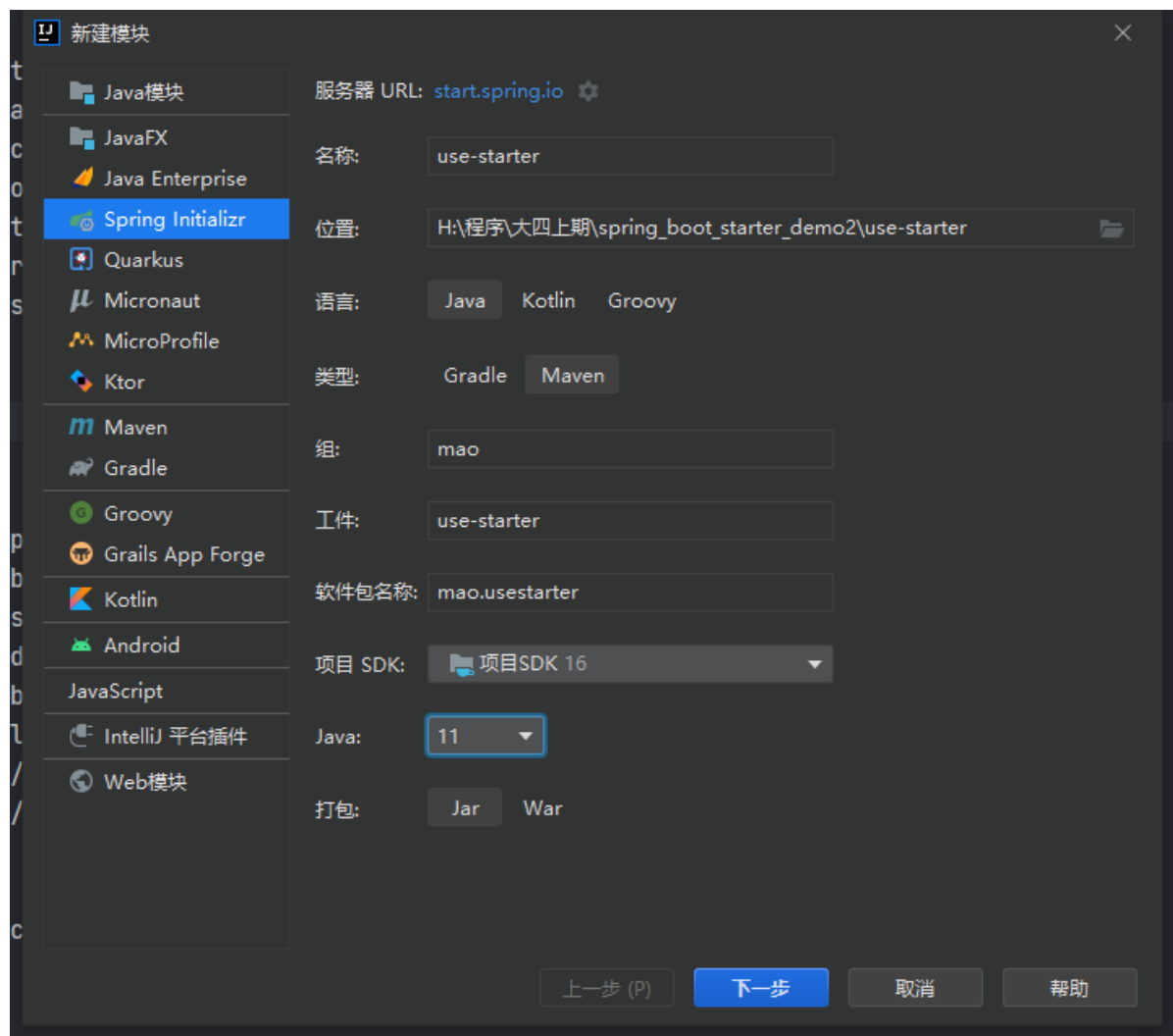
创建父工程spring\_boot\_starter\_demo2



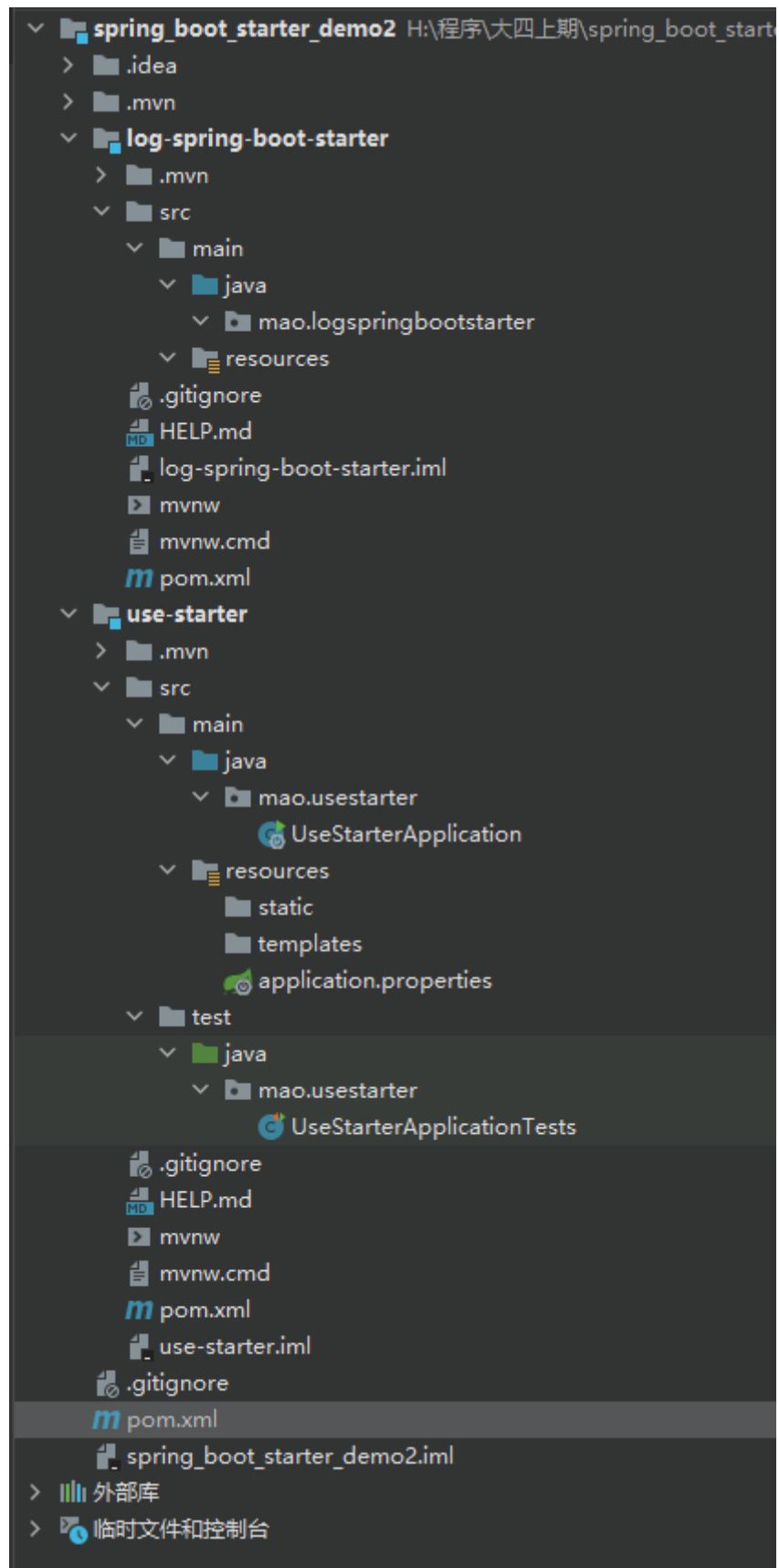
创建子工程log-spring-boot-starter



创建子工程use-starter



结构：



父工程spring\_boot\_starter\_demo2的pom文件:

```
1 <?xml version="1.0" encoding="UTF-8"?>
```

```

2 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5
6     <parent>
7         <groupId>org.springframework.boot</groupId>
8         <artifactId>spring-boot-starter-parent</artifactId>
9         <version>2.7.1</version>
10        <relativePath/>
11    </parent>
12
13    <groupId>mao</groupId>
14    <artifactId>spring_boot_starter_demo2</artifactId>
15    <version>0.0.1</version>
16    <name>spring_boot_starter_demo2</name>
17    <description>spring_boot_starter_demo2</description>
18    <packaging>pom</packaging>
19
20    <properties>
21        <java.version>11</java.version>
22    </properties>
23
24    <modules>
25
26        <module>log-spring-boot-starter</module>
27        <module>use-starter</module>
28
29    </modules>
30
31    <dependencies>
32
33
34    </dependencies>
35
36    <dependencyManagement>
37
38        <dependencies>
39
40
41        </dependencies>
42
43    </dependencyManagement>
44
45
46    <build>
47        <plugins>
48            <plugin>
49                <groupId>org.springframework.boot</groupId>
50                <artifactId>spring-boot-maven-plugin</artifactId>
51            </plugin>
52        </plugins>
53    </build>
54
55 </project>

```

子工程log-spring-boot-starter的pom文件:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5
6     <parent>
7
8         <artifactId>spring_boot_starter_demo2</artifactId>
9         <groupId>mao</groupId>
10        <version>0.0.1</version>
11
12    </parent>
13
14
15    <artifactId>log-spring-boot-starter</artifactId>
16    <version>0.0.1-SNAPSHOT</version>
17    <name>log-spring-boot-starter</name>
18    <description>log-spring-boot-starter</description>
19
20    <properties>
21        <java.version>11</java.version>
22    </properties>
23
24    <dependencies>
25
26        <dependency>
27            <groupId>org.springframework.boot</groupId>
28            <artifactId>spring-boot-starter-web</artifactId>
29        </dependency>
30
31
32        <!--spring boot starter开发依赖-->
33        <dependency>
34            <groupId>org.springframework.boot</groupId>
35            <artifactId>spring-boot-starter</artifactId>
36        </dependency>
37
38        <dependency>
39            <groupId>org.springframework.boot</groupId>
40            <artifactId>spring-boot-autoconfigure</artifactId>
41        </dependency>
42
43        <dependency>
44            <groupId>org.springframework.boot</groupId>
45            <artifactId>spring-boot-configuration-processor</artifactId>
46        </dependency>
```

```

47
48     </dependencies>
49
50     <build>
51         <plugins>
52             <plugin>
53                 <groupId>org.springframework.boot</groupId>
54                 <artifactId>spring-boot-maven-plugin</artifactId>
55                 <configuration>
56                     <skip>
57                         true
58                     </skip>
59                 </configuration>
60             </plugin>
61         </plugins>
62     </build>
63
64 </project>

```

子工程use-starter的pom文件:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   http://maven.apache.org/xsd/maven-4.0.0.xsd">
4      <modelVersion>4.0.0</modelVersion>
5
6      <parent>
7
8          <artifactId>spring_boot_starter_demo2</artifactId>
9          <groupId>mao</groupId>
10         <version>0.0.1</version>
11
12     </parent>
13
14
15     <artifactId>use-starter</artifactId>
16     <version>0.0.1-SNAPSHOT</version>
17     <name>use-starter</name>
18     <description>use-starter</description>
19
20     <properties>
21         <java.version>11</java.version>
22     </properties>
23
24     <dependencies>
25
26         <dependency>
27             <groupId>org.springframework.boot</groupId>

```

```

28         <artifactId>spring-boot-starter-web</artifactId>
29     </dependency>
30
31     <dependency>
32         <groupId>org.springframework.boot</groupId>
33         <artifactId>spring-boot-starter-test</artifactId>
34         <scope>test</scope>
35     </dependency>
36
37 </dependencies>
38
39 <build>
40     <plugins>
41         <plugin>
42             <groupId>org.springframework.boot</groupId>
43             <artifactId>spring-boot-maven-plugin</artifactId>
44         </plugin>
45     </plugins>
46 </build>
47
48 </project>

```

## 第二步：自定义Log注解

```

1  package mao.logspringbootstarter.log;
2
3
4  import java.lang.annotation.ElementType;
5  import java.lang.annotation.Retention;
6  import java.lang.annotation.RetentionPolicy;
7  import java.lang.annotation.Target;
8
9  @Target(ElementType.METHOD) //ElementType.METHOD: 只用在方法上
10 @Retention(RetentionPolicy.RUNTIME)
11 public @interface Log
12 {
13     /**
14      * 方法的描述信息
15      *
16      * @return {@link String}
17      */
18     String desc() default "";
19 }

```



### 第三步：自定义日志拦截器LogInterceptor

```
1 package mao.logspringbootstarter.interceptor;
2
3 import mao.logspringbootstarter.log.Log;
4 import org.slf4j.Logger;
5 import org.slf4j.LoggerFactory;
6 import org.springframework.web.method.HandlerMethod;
7 import org.springframework.web.servlet.HandlerInterceptor;
8 import org.springframework.web.servlet.ModelAndView;
9 import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;
10
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13 import java.lang.reflect.Method;
14
15 /**
16  * Project name(项目名称): spring_boot_starter_demo2
17  * Package(包名): mao.logspringbootstarter.interceptor
18  * Class(类名): LogInterceptor
19  * Author(作者): mao
20  * Author QQ: 1296193245
21  * GitHub: https://github.com/maomao124/
22  * Date(创建日期): 2022/10/24
23  * Time(创建时间): 22:39
24  * Version(版本): 1.0
25  * Description(描述): 无
26  */
27
28 public class LogInterceptor implements HandlerInterceptor
29 {
30
31     private static final ThreadLocal<Long> THREAD_LOCAL = new ThreadLocal<>
32     ();
33
34     private static final Logger logger =
35     LoggerFactory.getLogger(LogInterceptor.class);
36
37     @Override
38     public boolean preHandle(HttpServletRequest request, HttpServletResponse
39     response, Object handler) throws Exception
40     {
41         if (handler instanceof HandlerMethod)
42         {
43             HandlerMethod handlerMethod = (HandlerMethod) handler;
44             Method method = handlerMethod.getMethod();
45             Log log = method.getAnnotation(Log.class);
46             if (log != null)
47             {
48                 long startTime = System.currentTimeMillis();
49                 THREAD_LOCAL.set(startTime);
50             }
51         }
52         return true;
53     }
54 }
```

```

50     }
51
52     @Override
53     public void postHandle(HttpServletRequest request, HttpServletResponse
response, Object handler, ModelAndView modelAndView)
54         throws Exception
55     {
56         if (handler instanceof HandlerMethod)
57         {
58             long endTime = System.currentTimeMillis();
59             HandlerMethod handlerMethod = (HandlerMethod) handler;
60             Method method = handlerMethod.getMethod();
61             Log log = method.getAnnotation(Log.class);
62             if (log != null)
63             {
64                 Long startTime = THREAD_LOCAL.get();
65                 long runTime = endTime - startTime;
66                 String uri = request.getRequestURI();
67                 String methodName = method.getDeclaringClass().getName() +
"." + method.getName();
68                 String desc = log.desc();
69                 logger.info("请求的url: " + uri + ", 方法名: " + methodName +
", 描述: " + desc + ", 运行时间: " + runTime + "ms");
70                 THREAD_LOCAL.remove();
71             }
72         }
73     }
74 }
75

```

#### 第四步：创建自动配置类LogAutoConfiguration，用于自动配置拦截器、参数解析器等web组件

```

1  package mao.logspringbootstarter.config;
2
3  import mao.logspringbootstarter.interceptor.LogInterceptor;
4  import org.springframework.context.annotation.Configuration;
5  import
org.springframework.web.servlet.config.annotation.InterceptorRegistry;
6  import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
7
8  /**
9   * Project name(项目名称): spring_boot_starter_demo2
10  * Package(包名): mao.logspringbootstarter.config
11  * Class(类名): LogAutoConfiguration
12  * Author(作者): mao
13  * Author QQ: 1296193245
14  * GitHub: https://github.com/maomao124/
15  * Date(创建日期): 2022/10/24
16  * Time(创建时间): 22:51
17  * Version(版本): 1.0

```

```

18  * Description(描述): 无
19  */
20
21
22  @Configuration
23  public class LogAutoConfiguration implements WebMvcConfigurer
24  {
25      @Override
26      public void addInterceptors(InterceptorRegistry registry)
27      {
28          registry.addInterceptor(new LogInterceptor());
29      }
30  }

```

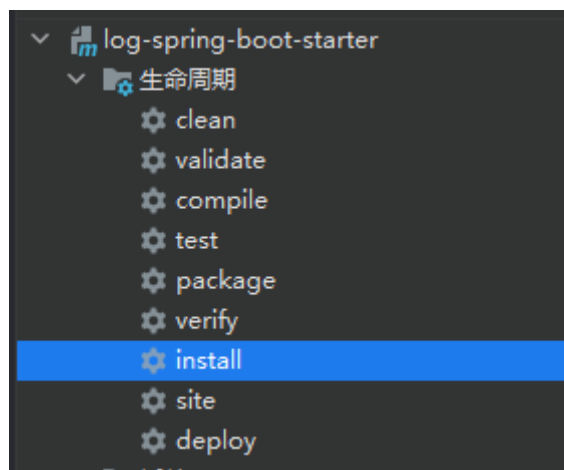
#### 第五步：在spring.factories中追加LogAutoConfiguration配置

```

1  org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
2  mao.logspringbootstarter.config.LogAutoConfiguration

```

#### 第六步：安装到本地库



```

1  [INFO] Scanning for projects...
2  [INFO]
3  [INFO] -----< mao:log-spring-boot-starter >-----
4  [INFO] Building log-spring-boot-starter 0.0.1-SNAPSHOT

```

```

5  [INFO] -----[ jar ]-----
6  [INFO]
7  [INFO] --- maven-resources-plugin:3.2.0:resources (default-resources) @ log-
   spring-boot-starter ---
8  [INFO] Using 'UTF-8' encoding to copy filtered resources.
9  [INFO] Using 'UTF-8' encoding to copy filtered properties files.
10 [INFO] Copying 0 resource
11 [INFO] Copying 1 resource
12 [INFO]
13 [INFO] --- maven-compiler-plugin:3.10.1:compile (default-compile) @ log-
   spring-boot-starter ---
14 [INFO] Changes detected - recompiling the module!
15 [INFO] Compiling 4 source files to H:\程序\大四上期
   \spring_boot_starter_demo2\log-spring-boot-starter\target\classes
16 [INFO]
17 [INFO] --- maven-resources-plugin:3.2.0:testResources (default-
   testResources) @ log-spring-boot-starter ---
18 [INFO] Using 'UTF-8' encoding to copy filtered resources.
19 [INFO] Using 'UTF-8' encoding to copy filtered properties files.
20 [INFO] skip non existing resourceDirectory H:\程序\大四上期
   \spring_boot_starter_demo2\log-spring-boot-starter\src\test\resources
21 [INFO]
22 [INFO] --- maven-compiler-plugin:3.10.1:testCompile (default-testCompile) @
   log-spring-boot-starter ---
23 [INFO] No sources to compile
24 [INFO]
25 [INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ log-spring-
   boot-starter ---
26 [INFO] Tests are skipped.
27 [INFO]
28 [INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ log-spring-boot-
   starter ---
29 [INFO] Building jar: H:\程序\大四上期\spring_boot_starter_demo2\log-spring-
   boot-starter\target\log-spring-boot-starter-0.0.1-SNAPSHOT.jar
30 [INFO]
31 [INFO] --- spring-boot-maven-plugin:2.7.1:repackage (repackage) @ log-
   spring-boot-starter ---
32 [INFO]
33 [INFO] --- maven-install-plugin:2.5.2:install (default-install) @ log-
   spring-boot-starter ---
34 [INFO] Installing H:\程序\大四上期\spring_boot_starter_demo2\log-spring-boot-
   starter\target\log-spring-boot-starter-0.0.1-SNAPSHOT.jar to
   C:\Users\mao\.m2\repository\mao\log-spring-boot-starter\0.0.1-SNAPSHOT\log-
   spring-boot-starter-0.0.1-SNAPSHOT.jar
35 [INFO] Installing H:\程序\大四上期\spring_boot_starter_demo2\log-spring-boot-
   starter\pom.xml to C:\Users\mao\.m2\repository\mao\log-spring-boot-
   starter\0.0.1-SNAPSHOT\log-spring-boot-starter-0.0.1-SNAPSHOT.pom
36 [INFO] -----
   ---
37 [INFO] BUILD SUCCESS
38 [INFO] -----
   ---
39 [INFO] Total time: 3.665 s
40 [INFO] Finished at: 2022-10-24T22:56:59+08:00
41 [INFO] -----
   ---

```

## 使用starter

### 第一步：添加依赖

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5
6     <parent>
7
8         <artifactId>spring_boot_starter_demo2</artifactId>
9         <groupId>mao</groupId>
10        <version>0.0.1</version>
11
12    </parent>
13
14
15    <artifactId>use-starter</artifactId>
16    <version>0.0.1-SNAPSHOT</version>
17    <name>use-starter</name>
18    <description>use-starter</description>
19
20    <properties>
21        <java.version>11</java.version>
22    </properties>
23
24    <dependencies>
25
26        <dependency>
27            <groupId>org.springframework.boot</groupId>
28            <artifactId>spring-boot-starter-web</artifactId>
29        </dependency>
30
31        <dependency>
32            <groupId>org.springframework.boot</groupId>
33            <artifactId>spring-boot-starter-test</artifactId>
34            <scope>test</scope>
35        </dependency>
36
37        <dependency>
```

```

38         <groupId>mao</groupId>
39         <artifactId>log-spring-boot-starter</artifactId>
40         <version>0.0.1-SNAPSHOT</version>
41     </dependency>
42
43 </dependencies>
44
45 <build>
46     <plugins>
47         <plugin>
48             <groupId>org.springframework.boot</groupId>
49             <artifactId>spring-boot-maven-plugin</artifactId>
50         </plugin>
51     </plugins>
52 </build>
53
54 </project>

```

## 第二步：编写controller类

```

1  package mao.usestarter.controller;
2
3  import mao.logspringbootstarter.log.Log;
4  import org.springframework.web.bind.annotation.GetMapping;
5  import org.springframework.web.bind.annotation.RestController;
6
7  /**
8   * Project name(项目名称): spring_boot_starter_demo2
9   * Package(包名): mao.usestarter.controller
10  * Class(类名): TestController
11  * Author(作者): mao
12  * Author QQ: 1296193245
13  * Github: https://github.com/maomao124/
14  * Date(创建日期): 2022/10/24
15  * Time(创建时间): 23:01
16  * Version(版本): 1.0
17  * Description(描述): 无
18  */
19
20 @RestController
21 public class TestController
22 {
23     /**
24      * test1
25      *
26      * @return {@link String}
27      */
28     @GetMapping("/test1")
29     @Log
30     public String test1()

```

```

31     {
32         return "1 success";
33     }
34
35     /**
36     * test2
37     *
38     * @return {@link String}
39     */
40     @GetMapping("/test2")
41     @Log
42     public String test2()
43     {
44         try
45         {
46             Thread.sleep(200);
47         }
48         catch (InterruptedException e)
49         {
50             e.printStackTrace();
51         }
52         return "2 success";
53     }
54
55     /**
56     * test3
57     *
58     * @return {@link String}
59     */
60     @GetMapping("/test3")
61     @Log
62     public String test3()
63     {
64         try
65         {
66             Thread.sleep(500);
67         }
68         catch (InterruptedException e)
69         {
70             e.printStackTrace();
71         }
72         return "3 success";
73     }
74
75     /**
76     * test4
77     *
78     * @return {@link String}
79     */
80     @GetMapping("/test4")
81     @Log
82     public String test4()
83     {
84         try
85         {
86             Thread.sleep(1000);
87         }
88         catch (InterruptedException e)

```





```

12 2022-10-24 23:23:09.216 INFO 2356 --- [          main]
   o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s):
   8080 (http)
13 2022-10-24 23:23:09.223 INFO 2356 --- [          main]
   o.apache.catalina.core.StandardService : Starting service [Tomcat]
14 2022-10-24 23:23:09.223 INFO 2356 --- [          main]
   org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
   Tomcat/9.0.64]
15 2022-10-24 23:23:09.303 INFO 2356 --- [          main] o.a.c.c.C.[Tomcat].
   [localhost].[/] : Initializing Spring embedded WebApplicationContext
16 2022-10-24 23:23:09.304 INFO 2356 --- [          main]
   w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
   initialization completed in 721 ms
17 2022-10-24 23:23:09.547 INFO 2356 --- [          main]
   o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080
   (http) with context path ''
18 2022-10-24 23:23:09.556 INFO 2356 --- [          main]
   mao.usestarter.UseStarterApplication : Started UseStarterApplication in
   1.267 seconds (JVM running for 1.74)

```

#### 第四步：访问服务

<http://localhost:8080/test1>

从test1到test6，每个资源请求3遍：

```

1 2022-10-24 23:26:03.034 INFO 8120 --- [nio-8080-exec-6]
   m.l.interceptor.LogInterceptor : 请求的url: /test1, 方法名:
   mao.usestarter.controller.TestController.test1, 描述: , 运行时间: 1ms
2 2022-10-24 23:26:04.392 INFO 8120 --- [nio-8080-exec-7]
   m.l.interceptor.LogInterceptor : 请求的url: /test1, 方法名:
   mao.usestarter.controller.TestController.test1, 描述: , 运行时间: 1ms
3 2022-10-24 23:26:05.006 INFO 8120 --- [nio-8080-exec-8]
   m.l.interceptor.LogInterceptor : 请求的url: /test1, 方法名:
   mao.usestarter.controller.TestController.test1, 描述: , 运行时间: 1ms
4 2022-10-24 23:26:07.567 INFO 8120 --- [nio-8080-exec-9]
   m.l.interceptor.LogInterceptor : 请求的url: /test2, 方法名:
   mao.usestarter.controller.TestController.test2, 描述: , 运行时间: 215ms
5 2022-10-24 23:26:08.724 INFO 8120 --- [nio-8080-exec-10]
   m.l.interceptor.LogInterceptor : 请求的url: /test2, 方法名:
   mao.usestarter.controller.TestController.test2, 描述: , 运行时间: 212ms
6 2022-10-24 23:26:09.500 INFO 8120 --- [nio-8080-exec-1]
   m.l.interceptor.LogInterceptor : 请求的url: /test2, 方法名:
   mao.usestarter.controller.TestController.test2, 描述: , 运行时间: 211ms
7 2022-10-24 23:26:12.514 INFO 8120 --- [nio-8080-exec-3]
   m.l.interceptor.LogInterceptor : 请求的url: /test3, 方法名:
   mao.usestarter.controller.TestController.test3, 描述: , 运行时间: 510ms
8 2022-10-24 23:26:13.597 INFO 8120 --- [nio-8080-exec-2]
   m.l.interceptor.LogInterceptor : 请求的url: /test3, 方法名:
   mao.usestarter.controller.TestController.test3, 描述: , 运行时间: 503ms

```

```
9 2022-10-24 23:26:14.892 INFO 8120 --- [nio-8080-exec-4]
   m.l.interceptor.LogInterceptor          : 请求的url: /test3, 方法名:
   mao.usestarter.controller.TestController.test3, 描述: , 运行时间: 514ms
10 2022-10-24 23:26:17.936 INFO 8120 --- [nio-8080-exec-5]
   m.l.interceptor.LogInterceptor          : 请求的url: /test4, 方法名:
   mao.usestarter.controller.TestController.test4, 描述: , 运行时间: 1011ms
11 2022-10-24 23:26:20.464 INFO 8120 --- [nio-8080-exec-6]
   m.l.interceptor.LogInterceptor          : 请求的url: /test4, 方法名:
   mao.usestarter.controller.TestController.test4, 描述: , 运行时间: 1007ms
12 2022-10-24 23:26:22.922 INFO 8120 --- [nio-8080-exec-7]
   m.l.interceptor.LogInterceptor          : 请求的url: /test4, 方法名:
   mao.usestarter.controller.TestController.test4, 描述: , 运行时间: 1009ms
13 2022-10-24 23:26:25.366 INFO 8120 --- [nio-8080-exec-8]
   m.l.interceptor.LogInterceptor          : 请求的url: /test5, 方法名:
   mao.usestarter.controller.TestController.test5, 描述: 这是test5方法的描述信息, 运
   行时间: 1ms
14 2022-10-24 23:26:26.557 INFO 8120 --- [nio-8080-exec-9]
   m.l.interceptor.LogInterceptor          : 请求的url: /test5, 方法名:
   mao.usestarter.controller.TestController.test5, 描述: 这是test5方法的描述信息, 运
   行时间: 1ms
15 2022-10-24 23:26:27.289 INFO 8120 --- [nio-8080-exec-10]
   m.l.interceptor.LogInterceptor          : 请求的url: /test5, 方法名:
   mao.usestarter.controller.TestController.test5, 描述: 这是test5方法的描述信息, 运
   行时间: 1ms
```

## 案例三

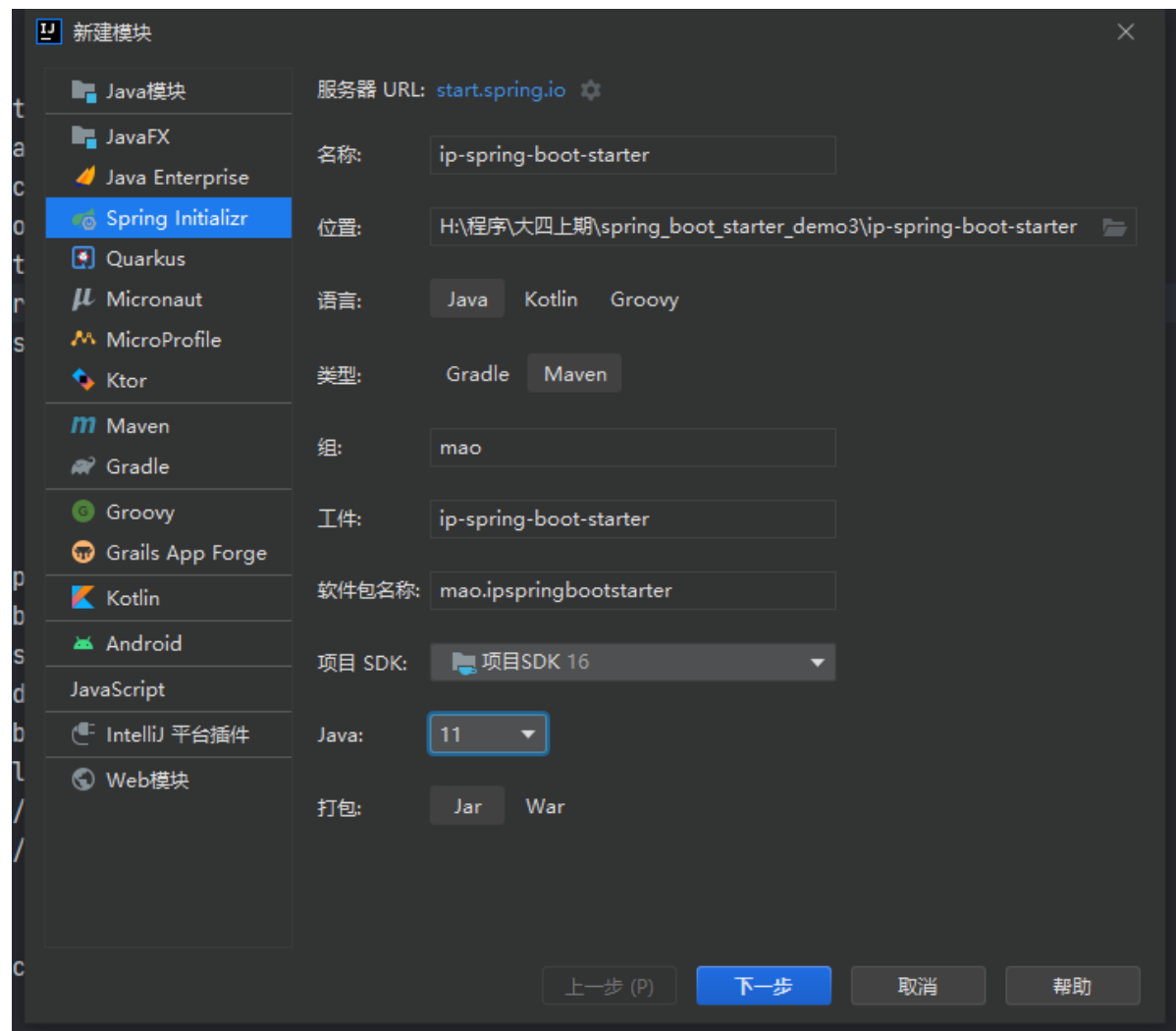
记录系统访客独立IP访问次数

## 开发starter

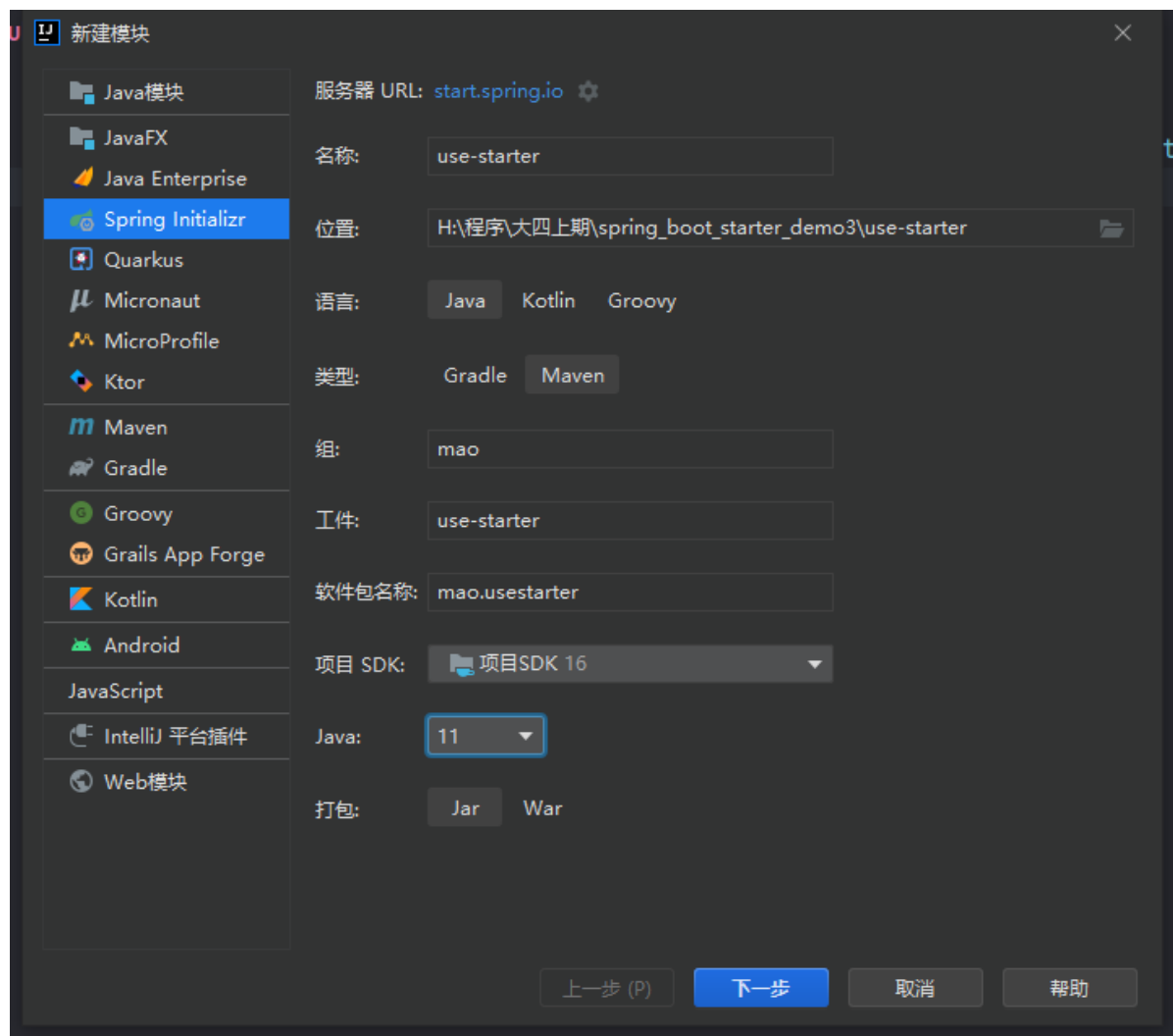
### 第一步：初始化项目

创建父工程spring\_boot\_starter\_demo3

创建子工程ip-spring-boot-starter



创建子工程use-starter



父工程的pom文件:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5     https://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <modelVersion>4.0.0</modelVersion>
7     <parent>
8         <groupId>org.springframework.boot</groupId>
9         <artifactId>spring-boot-starter-parent</artifactId>
10        <version>2.7.1</version>
11        <relativePath/>
12    </parent>
13    <groupId>mao</groupId>
14    <artifactId>spring_boot_starter_demo3</artifactId>
15    <version>0.0.1-SNAPSHOT</version>
16    <name>spring_boot_starter_demo3</name>
17    <description>spring_boot_starter_demo3</description>
```

```

18     <packaging>pom</packaging>
19
20     <properties>
21         <java.version>11</java.version>
22     </properties>
23
24
25     <modules>
26
27         <module>ip-spring-boot-starter</module>
28         <module>use-starter</module>
29
30     </modules>
31
32
33     <dependencies>
34
35     </dependencies>
36
37     <dependencyManagement>
38
39         <dependencies>
40
41         </dependencies>
42
43     </dependencyManagement>
44
45     <build>
46         <plugins>
47             <plugin>
48                 <groupId>org.springframework.boot</groupId>
49                 <artifactId>spring-boot-maven-plugin</artifactId>
50             </plugin>
51         </plugins>
52     </build>
53
54 </project>

```

创建子工程ip-spring-boot-starter的pom文件:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5  https://maven.apache.org/xsd/maven-4.0.0.xsd">
6      <modelVersion>4.0.0</modelVersion>
7
8      <parent>
9          <artifactId>spring_boot_starter_demo3</artifactId>
10         <groupId>mao</groupId>
11         <version>0.0.1-SNAPSHOT</version>
12     </parent>

```

```

11
12     <artifactId>ip-spring-boot-starter</artifactId>
13     <version>0.0.1-SNAPSHOT</version>
14     <name>ip-spring-boot-starter</name>
15     <description>ip-spring-boot-starter</description>
16
17     <properties>
18
19     </properties>
20
21     <dependencies>
22
23         <dependency>
24             <groupId>org.springframework.boot</groupId>
25             <artifactId>spring-boot-starter-web</artifactId>
26         </dependency>
27
28         <!--spring boot starter开发依赖-->
29         <dependency>
30             <groupId>org.springframework.boot</groupId>
31             <artifactId>spring-boot-starter</artifactId>
32         </dependency>
33
34         <dependency>
35             <groupId>org.springframework.boot</groupId>
36             <artifactId>spring-boot-autoconfigure</artifactId>
37         </dependency>
38
39         <dependency>
40             <groupId>org.springframework.boot</groupId>
41             <artifactId>spring-boot-configuration-processor</artifactId>
42         </dependency>
43
44     </dependencies>
45
46     <build>
47         <plugins>
48             <plugin>
49                 <groupId>org.springframework.boot</groupId>
50                 <artifactId>spring-boot-maven-plugin</artifactId>
51                 <configuration>
52                     <skip>true</skip>
53                 </configuration>
54             </plugin>
55         </plugins>
56     </build>
57
58 </project>
59

```

创建子工程use-starter的pom文件：

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5
6     <parent>
7         <artifactId>spring_boot_starter_demo3</artifactId>
8         <groupId>mao</groupId>
9         <version>0.0.1-SNAPSHOT</version>
10    </parent>
11
12
13    <artifactId>use-starter</artifactId>
14    <version>0.0.1-SNAPSHOT</version>
15    <name>use-starter</name>
16    <description>use-starter</description>
17
18    <properties>
19
20    </properties>
21
22    <dependencies>
23
24        <dependency>
25            <groupId>org.springframework.boot</groupId>
26            <artifactId>spring-boot-starter-web</artifactId>
27        </dependency>
28
29        <dependency>
30            <groupId>org.springframework.boot</groupId>
31            <artifactId>spring-boot-starter-test</artifactId>
32            <scope>test</scope>
33        </dependency>
34
35    </dependencies>
36
37    <build>
38        <plugins>
39            <plugin>
40                <groupId>org.springframework.boot</groupId>
41                <artifactId>spring-boot-maven-plugin</artifactId>
42            </plugin>
43        </plugins>
44    </build>
45
46 </project>

```

## 第二步：编写IpCountService业务

```
1 package mao.ipspringbootstarter.service;
2
3 import org.slf4j.Logger;
4 import org.slf4j.LoggerFactory;
5 import org.springframework.beans.factory.annotation.Autowired;
6
7 import javax.servlet.http.HttpServletRequest;
8 import java.util.Map;
9 import java.util.concurrent.ConcurrentHashMap;
10
11 /**
12  * Project name(项目名称): spring_boot_starter_demo3
13  * Package(包名): mao.ipspringbootstarter.service
14  * Class(类名): IpCountService
15  * Author(作者): mao
16  * Author QQ: 1296193245
17  * GitHub: https://github.com/maomao124/
18  * Date(创建日期): 2022/10/25
19  * Time(创建时间): 13:40
20  * Version(版本): 1.0
21  * Description(描述): 无
22  */
23
24 public class IpCountService
25 {
26     private final Map<String, Integer> ipCountMap = new ConcurrentHashMap<>
27     ();
28
29     private static final Logger log =
30     LoggerFactory.getLogger(IpCountService.class);
31
32     @Autowired
33     private HttpServletRequest request;
34
35     /**
36      * 记录某个IP访问该网站的次数
37      */
38     public void count()
39     {
40         String ipAddress = request.getRemoteAddr();
41         if (ipCountMap.containsKey(ipAddress))
42         {
43             synchronized (ipAddress.intern())
44             {
45                 ipCountMap.put(ipAddress, ipCountMap.get(ipAddress) + 1);
46             }
47         }
48         else
49         {
50             ipCountMap.put(ipAddress, 1);
51         }
52         log.debug("IP:" + ipAddress);
53     }
54 }
```



### 第三步：编写配置类IpAutoConfiguration

```

1 package mao.ipspringbootstarter.config;
2
3 import mao.ipspringbootstarter.service.IpCountService;
4 import org.springframework.context.annotation.Bean;
5 import org.springframework.context.annotation.Configuration;
6
7 /**
8  * Project name(项目名称): spring_boot_starter_demo3
9  * Package(包名): mao.ipspringbootstarter.config
10  * Class(类名): IpAutoConfiguration
11  * Author(作者): mao
12  * Author QQ: 1296193245
13  * GitHub: https://github.com/maomao124/
14  * Date(创建日期): 2022/10/25
15  * Time(创建时间): 13:46
16  * Version(版本): 1.0
17  * Description(描述): 无
18  */
19
20 @Configuration
21 public class IpAutoConfiguration
22 {
23     @Bean
24     public IpCountService ipCountService()
25     {
26         return new IpCountService();
27     }
28 }

```

### 第四步：在spring.factories中追加IpAutoConfiguration配置

```

1 org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
2     mao.ipspringbootstarter.config.IpAutoConfiguration

```

## 第五步：开启定时任务功能

```
1 package mao.ipspringbootstarter.config;
2
3 import mao.ipspringbootstarter.service.IpCountService;
4 import
5 org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.context.annotation.Configuration;
8 import org.springframework.scheduling.annotation.EnableScheduling;
9
10 /**
11  * Project name(项目名称): spring_boot_starter_demo3
12  * Package(包名): mao.ipspringbootstarter.config
13  * Class(类名): IpAutoConfiguration
14  * Author(作者): mao
15  * Author QQ: 1296193245
16  * GitHub: https://github.com/maomao124/
17  * Date(创建日期): 2022/10/25
18  * Time(创建时间): 13:46
19  * Version(版本): 1.0
20  * Description(描述): 无
21  */
22 @EnableScheduling
23 @Configuration
24 public class IpAutoConfiguration
25 {
26     @Bean
27     @ConditionalOnMissingBean
28     public IpCountService ipCountService()
29     {
30         return new IpCountService();
31     }
32 }
```

## 第六步：设置定时任务

```
1 package mao.ipspringbootstarter.service;
2
3 import org.slf4j.Logger;
4 import org.slf4j.LoggerFactory;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.scheduling.annotation.Scheduled;
7
8 import javax.servlet.http.HttpServletRequest;
9 import java.util.Map;
10 import java.util.concurrent.ConcurrentHashMap;
```

```

11
12 /**
13  * Project name(项目名称): spring_boot_starter_demo3
14  * Package(包名): mao.ipspringbootstarter.service
15  * Class(类名): IpCountService
16  * Author(作者): mao
17  * Author QQ: 1296193245
18  * Github: https://github.com/maomao124/
19  * Date(创建日期): 2022/10/25
20  * Time(创建时间): 13:40
21  * Version(版本): 1.0
22  * Description(描述): 无
23  */
24
25 public class IpCountService
26 {
27     private final Map<String, Integer> ipCountMap = new ConcurrentHashMap<>
28 ();
29
30     private static final Logger log =
31 LoggerFactory.getLogger(IpCountService.class);
32
33     @Autowired
34     private HttpServletRequest request;
35
36     /**
37      * 记录某个IP访问该网站的次数
38      */
39     public void count()
40     {
41         String ipAddress = request.getRemoteAddr();
42         if (ipCountMap.containsKey(ipAddress))
43         {
44             synchronized (ipAddress.intern())
45             {
46                 ipCountMap.put(ipAddress, ipCountMap.get(ipAddress) + 1);
47             }
48         }
49         else
50         {
51             ipCountMap.put(ipAddress, 1);
52         }
53         log.debug("IP:" + ipAddress);
54     }
55
56     /**
57      * 定时打印一个表格
58      */
59     @Scheduled(cron = "0/10 * * * * ?")
60     public void print()
61     {
62         StringBuilder stringBuilder = new StringBuilder(" IP访问监控\n");
63         stringBuilder.append("+-----ip-address-----+---num---+\n");
64
65         for (Map.Entry<String, Integer> info : ipCountMap.entrySet())
66         {
67             String key = info.getKey();

```

```

67         Integer count = info.getValue();
68         String lineInfo = String.format("|%18s |%6d |", key, count);
69         stringBuilder.append(lineInfo).append("\n");
70     }
71     stringBuilder.append("+-----+-----+");
72     log.info(stringBuilder.toString());
73 }
74 }

```

## 第七步：定义属性类，加载对应属性

```

1  package mao.ipspringbootstarter.config;
2
3  import org.springframework.boot.context.properties.ConfigurationProperties;
4  import org.springframework.stereotype.Component;
5
6
7  /**
8   * Project name(项目名称): spring_boot_starter_demo3
9   * Package(包名): mao.ipspringbootstarter.config
10  * Class(类名): IpConfigurationProperties
11  * Author(作者): mao
12  * Author QQ: 1296193245
13  * Github: https://github.com/maomao124/
14  * Date(创建日期): 2022/10/25
15  * Time(创建时间): 13:59
16  * Version(版本): 1.0
17  * Description(描述): 无
18  */
19
20
21  @Component("ipConfigurationProperties")
22  @ConfigurationProperties(prefix = "tools.ip")
23  public class IpConfigurationProperties
24  {
25
26      /**
27       * 日志显示周期
28       */
29      private long cycle = 10L;
30
31
32      /**
33       * 是否周期内重置数据
34       */
35      private Boolean cycleReset = false;
36
37
38      /**
39       * 日志输出模式 detail:明细模式 simple:极简模式

```

```

40     */
41     private String mode = LogModel.DETAIL.value;
42
43
44     /**
45      * The enum Log model.
46      */
47     public enum LogModel
48     {
49         /**
50          * Detail log model.
51          */
52         DETAIL("detail"),
53         /**
54          * Simple log model.
55          */
56         SIMPLE("simple");
57
58         private String value;
59
60         LogModel(String value)
61         {
62             this.value = value;
63         }
64
65         /**
66          * Gets value.
67          *
68          * @return the value
69          */
70         public String getValue()
71         {
72             return value;
73         }
74     }
75
76
77     /**
78      * Instantiates a new Ip configuration properties.
79      */
80     public IpConfigurationProperties()
81     {
82
83     }
84
85     /**
86      * Instantiates a new Ip configuration properties.
87      *
88      * @param cycle      the cycle
89      * @param cycleReset the cycle reset
90      * @param mode        the mode
91      */
92     public IpConfigurationProperties(long cycle, Boolean cycleReset, String
mode)
93     {
94         this.cycle = cycle;
95         this.cycleReset = cycleReset;
96         this.mode = mode;

```

```

97     }
98
99     /**
100      * Gets cycle.
101      *
102      * @return the cycle
103      */
104     public long getCycle()
105     {
106         return cycle;
107     }
108
109     /**
110      * Sets cycle.
111      *
112      * @param cycle the cycle
113      */
114     public void setCycle(long cycle)
115     {
116         this.cycle = cycle;
117     }
118
119     /**
120      * Gets cycle reset.
121      *
122      * @return the cycle reset
123      */
124     public Boolean getCycleReset()
125     {
126         return cycleReset;
127     }
128
129     /**
130      * Sets cycle reset.
131      *
132      * @param cycleReset the cycle reset
133      */
134     public void setCycleReset(Boolean cycleReset)
135     {
136         this.cycleReset = cycleReset;
137     }
138
139     /**
140      * Gets mode.
141      *
142      * @return the mode
143      */
144     public String getMode()
145     {
146         return mode;
147     }
148
149     /**
150      * Sets mode.
151      *
152      * @param mode the mode
153      */
154     public void setMode(String mode)

```

```

155     {
156         this.mode = mode;
157     }
158
159     @Override
160     @SuppressWarnings("all")
161     public String toString()
162     {
163         final StringBuilder stringBuilder = new StringBuilder();
164         stringBuilder.append("cycle: ").append(cycle).append('\n');
165
166         stringBuilder.append("cycleReset: ").append(cycleReset).append('\n');
167         stringBuilder.append("mode: ").append(mode).append('\n');
168         return stringBuilder.toString();
169     }
170 }

```

## 第八步：设置加载Properties类为bean

```

1  package mao.ipspringbootstarter.config;
2
3  import mao.ipspringbootstarter.service.IpCountService;
4  import
5  org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
6  import
7  org.springframework.boot.context.properties.EnableConfigurationProperties;
8  import org.springframework.context.annotation.Bean;
9  import org.springframework.context.annotation.Configuration;
10 import org.springframework.context.annotation.Import;
11 import org.springframework.scheduling.annotation.EnableScheduling;
12
13 /**
14  * Project name(项目名称): spring_boot_starter_demo3
15  * Package(包名): mao.ipspringbootstarter.config
16  * Class(类名): IpAutoConfiguration
17  * Author(作者): mao
18  * Author QQ: 1296193245
19  * GitHub: https://github.com/maomao124/
20  * Date(创建日期): 2022/10/25
21  * Time(创建时间): 13:46
22  * Version(版本): 1.0
23  * Description(描述): 无
24  */
25
26 @EnableScheduling
27 @Configuration
28 @Import(IpConfigurationProperties.class)
29 @EnableConfigurationProperties(IpConfigurationProperties.class)
30 public class IpAutoConfiguration

```

```

29 {
30     @Bean
31     @ConditionalOnMissingBean
32     public IpCountService ipCountService()
33     {
34         return new IpCountService();
35     }
36 }
37

```

### 第九步：根据配置切换设置

```

1  package mao.ipspringbootstarter.service;
2
3  import mao.ipspringbootstarter.config.IpConfigurationProperties;
4  import org.slf4j.Logger;
5  import org.slf4j.LoggerFactory;
6  import org.springframework.beans.factory.annotation.Autowired;
7  import org.springframework.scheduling.annotation.Scheduled;
8
9  import javax.servlet.http.HttpServletRequest;
10 import java.util.Map;
11 import java.util.concurrent.ConcurrentHashMap;
12
13 /**
14  * Project name(项目名称): spring_boot_starter_demo3
15  * Package(包名): mao.ipspringbootstarter.service
16  * Class(类名): IpCountService
17  * Author(作者): mao
18  * Author QQ: 1296193245
19  * Github: https://github.com/maomao124/
20  * Date(创建日期): 2022/10/25
21  * Time(创建时间): 13:40
22  * Version(版本): 1.0
23  * Description(描述): 无
24  */
25
26 public class IpCountService
27 {
28     private final Map<String, Integer> ipCountMap = new ConcurrentHashMap<>
29     ();
30
31     private static final Logger log =
32     LoggerFactory.getLogger(IpCountService.class);
33
34     @Autowired
35     private IpConfigurationProperties ipConfigurationProperties;
36
37     @Autowired
38     private HttpServletRequest request;
39

```



```

37
38     /**
39      * 记录某个IP访问该网站的次数
40      */
41     public void count()
42     {
43         String ipAddress = request.getRemoteAddr();
44         if (ipCountMap.containsKey(ipAddress))
45         {
46             synchronized (ipAddress.intern())
47             {
48                 ipCountMap.put(ipAddress, ipCountMap.get(ipAddress) + 1);
49             }
50         }
51         else
52         {
53             ipCountMap.put(ipAddress, 1);
54         }
55         log.debug("IP:" + ipAddress);
56     }
57
58
59     /**
60      * 定时打印一个表格
61      */
62     @Scheduled(cron = "0/10 * * * * ?")
63     public void print()
64     {
65
66         //模式切换
67         if
68         (ipConfigurationProperties.getMode().equals(IpConfigurationProperties.LogModel.DETAIL.getValue()))
69         {
70             //明细模式
71             detailPrint();
72         }
73         else if
74         (ipConfigurationProperties.getMode().equals(IpConfigurationProperties.LogModel.SIMPLE.getValue()))
75         {
76             //极简模式
77             simplePrint();
78         }
79
80         //周期内重置数据
81         if (ipConfigurationProperties.getCycleReset())
82         {
83             ipCountMap.clear();
84         }
85
86     /**
87      * 更详细的输出
88      */
89     private void detailPrint()
90     {

```

```

91     StringBuilder stringBuilder = new StringBuilder(" IP访问监控\n");
92     stringBuilder.append("+-----ip-address-----+---num---+\n");
93
94     for (Map.Entry<String, Integer> info : ipCountMap.entrySet())
95     {
96         String key = info.getKey();
97         Integer count = info.getValue();
98         String lineInfo = String.format("|%18s |%6d |", key, count);
99         stringBuilder.append(lineInfo).append("\n");
100    }
101    stringBuilder.append("+-----+-----+");
102    log.info(stringBuilder.toString());
103 }
104
105
106 /**
107  * 简单的输出
108  */
109 private void simplePrint()
110 {
111     StringBuilder stringBuilder = new StringBuilder(" IP访问监控\n");
112     stringBuilder.append("+-----ip-address-----+---num---+\n");
113
114     for (Map.Entry<String, Integer> info : ipCountMap.entrySet())
115     {
116         String key = info.getKey();
117         String lineInfo = String.format("|%18s |", key);
118         stringBuilder.append(lineInfo).append("\n");
119     }
120     stringBuilder.append("+-----+-----+");
121     log.info(stringBuilder.toString());
122 }
123
124
125 }

```

## 第十步：使用#{beanName.attrName}读取bean的属性

```

1  package mao.ipspringbootstarter.service;
2
3  import mao.ipspringbootstarter.config IpConfigurationProperties;
4  import org.slf4j.Logger;
5  import org.slf4j.LoggerFactory;
6  import org.springframework.beans.factory.annotation.Autowired;
7  import org.springframework.scheduling.annotation.Scheduled;
8
9  import javax.servlet.http.HttpServletRequest;
10 import java.util.Map;
11 import java.util.concurrent.ConcurrentHashMap;
12

```

```

13  /**
14   * Project name(项目名称): spring_boot_starter_demo3
15   * Package(包名): mao.ipspringbootstarter.service
16   * Class(类名): IpCountService
17   * Author(作者): mao
18   * Author QQ: 1296193245
19   * GitHub: https://github.com/maomao124/
20   * Date(创建日期): 2022/10/25
21   * Time(创建时间): 13:40
22   * Version(版本): 1.0
23   * Description(描述): 无
24   */
25
26  public class IpCountService
27  {
28      private final Map<String, Integer> ipCountMap = new ConcurrentHashMap<>
29      ();
30
31      private static final Logger log =
32      LoggerFactory.getLogger(IpCountService.class);
33
34      @Autowired
35      private IpConfigurationProperties ipConfigurationProperties;
36
37      @Autowired
38      private HttpServletRequest request;
39
40      /**
41       * 记录某个IP访问该网站的次数
42       */
43      public void count()
44      {
45          String ipAddress = request.getRemoteAddr();
46          if (ipCountMap.containsKey(ipAddress))
47          {
48              synchronized (ipAddress.intern())
49              {
50                  ipCountMap.put(ipAddress, ipCountMap.get(ipAddress) + 1);
51              }
52          }
53          else
54          {
55              ipCountMap.put(ipAddress, 1);
56          }
57          log.debug("IP:" + ipAddress);
58      }
59
60      /**
61       * 定时打印一个表格
62       */
63      @Scheduled(cron = "0/#{ipConfigurationProperties.cycle} * * * * ?")
64      public void print()
65      {
66          //模式切换

```

```

67         if
        (ipConfigurationProperties.getMode().equals(IpConfigurationProperties.LogModel.DETAIL.getValue()))
68         {
69             //明细模式
70             detailPrint();
71         }
72         else if
        (ipConfigurationProperties.getMode().equals(IpConfigurationProperties.LogModel.SIMPLE.getValue()))
73         {
74             //极简模式
75             simplePrint();
76         }
77
78         //周期内重置数据
79         if (ipConfigurationProperties.getCycleReset())
80         {
81             ipCountMap.clear();
82         }
83     }
84
85
86     /**
87     * 更详细的输出
88     */
89     private void detailPrint()
90     {
91         StringBuilder stringBuilder = new StringBuilder(" IP访问监控\n");
92         stringBuilder.append("+-----ip-address-----+---num--+ \n");
93
94         for (Map.Entry<String, Integer> info : ipCountMap.entrySet())
95         {
96             String key = info.getKey();
97             Integer count = info.getValue();
98             String lineInfo = String.format("|%18s |%6d |", key, count);
99             stringBuilder.append(lineInfo).append("\n");
100         }
101         stringBuilder.append("+-----+-----+");
102         log.info(stringBuilder.toString());
103     }
104
105
106     /**
107     * 简单的输出
108     */
109     private void simplePrint()
110     {
111         StringBuilder stringBuilder = new StringBuilder(" IP访问监控\n");
112         stringBuilder.append("+-----ip-address-----+ \n");
113
114         for (Map.Entry<String, Integer> info : ipCountMap.entrySet())
115         {
116             String key = info.getKey();
117             String lineInfo = String.format("|%18s |", key);
118             stringBuilder.append(lineInfo).append("\n");
119         }
120         stringBuilder.append("+-----+-----+");

```

```

121         log.info(stringBuilder.toString());
122     }
123
124
125 }

```

## 第十一步：自定义拦截器IpInterceptor

```

1  package mao.ipspringbootstarter.interceptor;
2
3  import mao.ipspringbootstarter.service.IpCountService;
4  import org.springframework.beans.factory.annotation.Autowired;
5  import org.springframework.web.servlet.HandlerInterceptor;
6  import org.springframework.web.servlet.ModelAndView;
7
8  import javax.servlet.http.HttpServletRequest;
9  import javax.servlet.http.HttpServletResponse;
10
11  /**
12   * Project name(项目名称): spring_boot_starter_demo3
13   * Package(包名): mao.ipspringbootstarter.interceptor
14   * Class(类名): IpInterceptor
15   * Author(作者): mao
16   * Author QQ: 1296193245
17   * Github: https://github.com/maomao124/
18   * Date(创建日期): 2022/10/25
19   * Time(创建时间): 14:19
20   * Version(版本): 1.0
21   * Description(描述): 无
22   */
23
24  public class IpInterceptor implements HandlerInterceptor
25  {
26
27      @Autowired
28      private IpCountService ipCountService;
29
30      @Override
31      public boolean preHandle(HttpServletRequest request, HttpServletResponse
response, Object handler) throws Exception
32      {
33          ipCountService.count();
34          return true;
35      }
36  }

```

## 第十二步：注册拦截器IpInterceptor

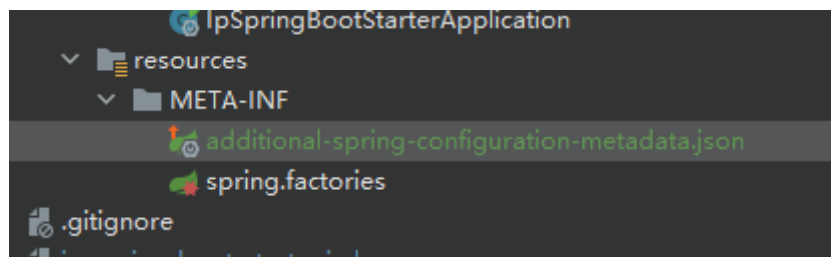
```
1 package mao.ipspringbootstarter.config;
2
3 import mao.ipspringbootstarter.interceptor.IpInterceptor;
4 import org.springframework.context.annotation.Bean;
5 import org.springframework.context.annotation.Configuration;
6 import
7 org.springframework.web.servlet.config.annotation.InterceptorRegistry;
8 import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
9
10 /**
11  * Project name(项目名称): spring_boot_starter_demo3
12  * Package(包名): mao.ipspringbootstarter.config
13  * Class(类名): SpringMvcConfig
14  * Author(作者): mao
15  * Author QQ: 1296193245
16  * GitHub: https://github.com/maomao124/
17  * Date(创建日期): 2022/10/25
18  * Time(创建时间): 14:22
19  * Version(版本): 1.0
20  * Description(描述): 无
21  */
22 @Configuration
23 public class SpringMvcConfig implements WebMvcConfigurer
24 {
25     @Bean
26     public IpInterceptor ipInterceptor()
27     {
28         return new IpInterceptor();
29     }
30
31     @Override
32     public void addInterceptors(InterceptorRegistry registry)
33     {
34         //proxyBeanMethods默认为true，不是直接new，是从容器里拿
35
36         registry.addInterceptor(ipInterceptor()).excludePathPatterns("/error");
37     }
38
39
40 }
41
```

### 第十三步：更改spring.factories文件

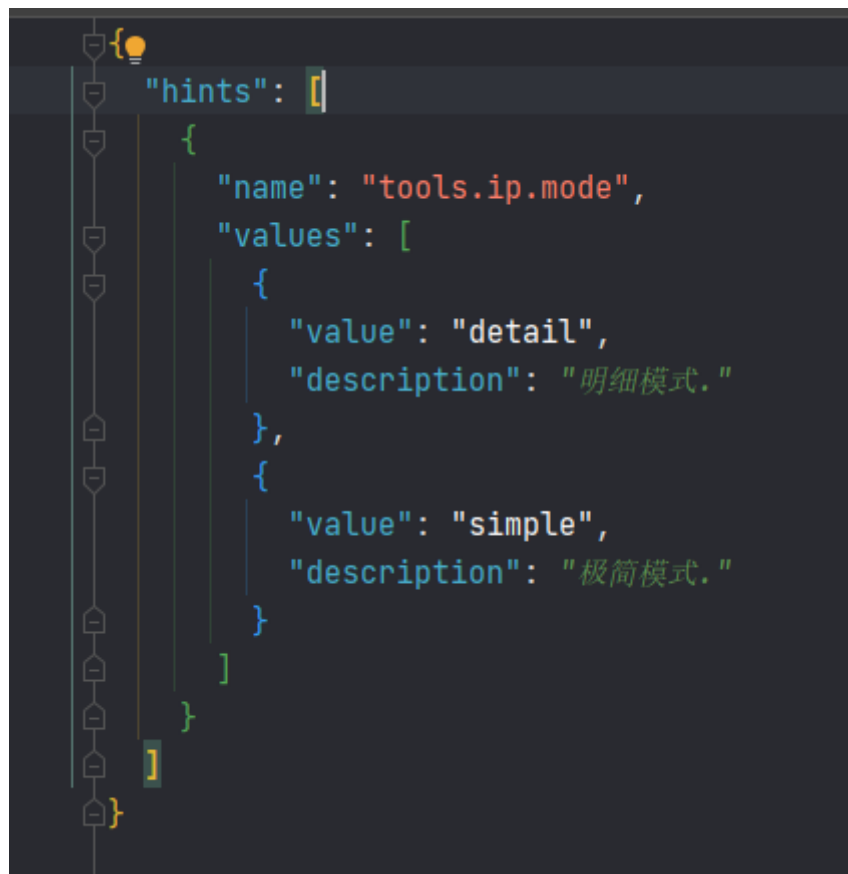
```
1 org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
2   mao.ipspringbootstarter.config.IpAutoConfiguration,\
3   mao.ipspringbootstarter.config.SpringMvcConfig
```

### 第十四步：自定义提示功能

在resources/META-INF目录下创建additional-spring-configuration-metadata.json



```
1 {
2   "hints": [
3     {
4       "name": "tools.ip.mode",
5       "values": [
6         {
7           "value": "detail",
8           "description": "明细模式."
9         },
10        {
11          "value": "simple",
12          "description": "极简模式."
13        }
14      ]
15    }
16  ]
17 }
```



## 使用starter

### 第一步：添加依赖

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5       https://maven.apache.org/xsd/maven-4.0.0.xsd">
6   <modelVersion>4.0.0</modelVersion>
7
8   <parent>
9     <artifactId>spring_boot_starter_demo3</artifactId>
10    <groupId>mao</groupId>
11    <version>0.0.1-SNAPSHOT</version>
12  </parent>
```

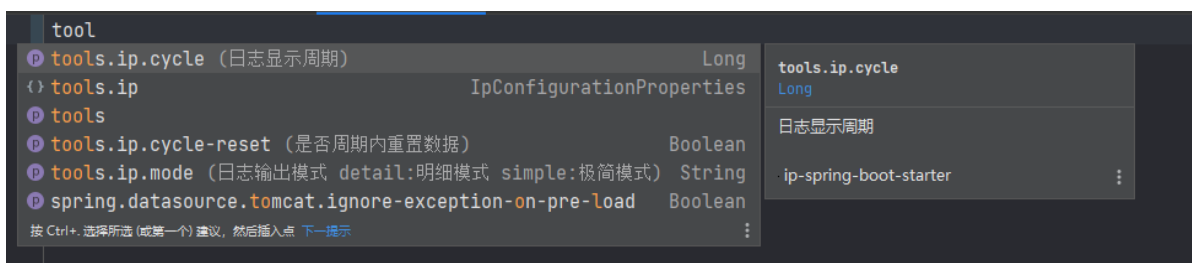


```

12
13     <artifactId>use-starter</artifactId>
14     <version>0.0.1-SNAPSHOT</version>
15     <name>use-starter</name>
16     <description>use-starter</description>
17
18     <properties>
19
20     </properties>
21
22     <dependencies>
23
24         <dependency>
25             <groupId>org.springframework.boot</groupId>
26             <artifactId>spring-boot-starter-web</artifactId>
27         </dependency>
28
29         <dependency>
30             <groupId>org.springframework.boot</groupId>
31             <artifactId>spring-boot-starter-test</artifactId>
32             <scope>test</scope>
33         </dependency>
34
35         <dependency>
36             <groupId>mao</groupId>
37             <artifactId>ip-spring-boot-starter</artifactId>
38             <version>0.0.1-SNAPSHOT</version>
39         </dependency>
40
41     </dependencies>
42
43     <build>
44         <plugins>
45             <plugin>
46                 <groupId>org.springframework.boot</groupId>
47                 <artifactId>spring-boot-maven-plugin</artifactId>
48             </plugin>
49         </plugins>
50     </build>
51
52 </project>

```

## 第二步：编写配置文件





```

14 2022-10-25 15:19:07.628 INFO 13892 --- [          main]
    org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
    Tomcat/9.0.64]
15 2022-10-25 15:19:07.701 INFO 13892 --- [          main] o.a.c.c.C.
    [Tomcat].[localhost].[/]           : Initializing Spring embedded
    WebApplicationContext
16 2022-10-25 15:19:07.702 INFO 13892 --- [          main]
    w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
    initialization completed in 713 ms
17 2022-10-25 15:19:07.983 INFO 13892 --- [          main]
    o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080
    (http) with context path ''
18 2022-10-25 15:19:07.994 INFO 13892 --- [          main]
    mao.usestarter.UseStarterApplication : Started UseStarterApplication in
    1.362 seconds (JVM running for 1.835)
19 2022-10-25 15:19:10.011 INFO 13892 --- [   scheduling-1]
    m.i.service.IpCountService          : IP访问监控
20 +-----ip-address-----+--num--+
21 +-----+-----+
22 2022-10-25 15:19:15.007 INFO 13892 --- [   scheduling-1]
    m.i.service.IpCountService          : IP访问监控
23 +-----ip-address-----+--num--+
24 +-----+-----+
25 2022-10-25 15:19:20.006 INFO 13892 --- [   scheduling-1]
    m.i.service.IpCountService          : IP访问监控
26 +-----ip-address-----+--num--+
27 +-----+-----+
28 2022-10-25 15:19:25.004 INFO 13892 --- [   scheduling-1]
    m.i.service.IpCountService          : IP访问监控
29 +-----ip-address-----+--num--+
30 +-----+-----+

```

#### 第四步：访问服务，查看日志

```

1 +-----ip-address-----+--num--+
2 | 0:0:0:0:0:0:1 | 4 |
3 +-----+-----+
4 2022-10-25 15:20:15.011 INFO 13892 --- [   scheduling-1]
    m.i.service.IpCountService          : IP访问监控
5 +-----ip-address-----+--num--+
6 | 0:0:0:0:0:0:1 | 18 |
7 +-----+-----+
8 2022-10-25 15:20:20.007 INFO 13892 --- [   scheduling-1]
    m.i.service.IpCountService          : IP访问监控
9 +-----ip-address-----+--num--+
10 | 0:0:0:0:0:0:1 | 44 |
11 +-----+-----+
12 2022-10-25 15:20:25.017 INFO 13892 --- [   scheduling-1]
    m.i.service.IpCountService          : IP访问监控
13 +-----ip-address-----+--num--+
14 | 0:0:0:0:0:0:1 | 72 |
15 +-----+-----+

```

```

16 2022-10-25 15:20:30.002 INFO 13892 --- [ scheduling-1]
   m.i.service.IpCountService           : IP访问监控
17 +-----ip-address-----+--num--+
18 | 0:0:0:0:0:0:1 | 72 |
19 +-----+-----+
20 2022-10-25 15:20:35.010 INFO 13892 --- [ scheduling-1]
   m.i.service.IpCountService           : IP访问监控
21 +-----ip-address-----+--num--+
22 | 0:0:0:0:0:0:1 | 80 |
23 +-----+-----+
24 2022-10-25 15:20:40.006 INFO 13892 --- [ scheduling-1]
   m.i.service.IpCountService           : IP访问监控
25 +-----ip-address-----+--num--+
26 | 0:0:0:0:0:0:1 | 80 |
27 +-----+-----+

```

### 第五步：更改统计速度

```

1 tools:
2   ip:
3     cycle: 1
4     mode: detail
5     cycle-reset: false

```

### 第六步：重启服务，访问服务，查看日志

```

1 +-----ip-address-----+--num--+
2 +-----+-----+
3 2022-10-25 15:23:20.007 INFO 4616 --- [ scheduling-1]
   m.i.service.IpCountService           : IP访问监控
4 +-----ip-address-----+--num--+
5 +-----+-----+
6 2022-10-25 15:23:21.011 INFO 4616 --- [ scheduling-1]
   m.i.service.IpCountService           : IP访问监控
7 +-----ip-address-----+--num--+
8 +-----+-----+
9 2022-10-25 15:23:21.735 INFO 4616 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].
   [localhost].[/] : Initializing Spring DispatcherServlet
   'dispatcherServlet'
10 2022-10-25 15:23:21.735 INFO 4616 --- [nio-8080-exec-1]
   o.s.web.servlet.DispatcherServlet    : Initializing Servlet
   'dispatcherServlet'

```

```

11 2022-10-25 15:23:21.736 INFO 4616 --- [nio-8080-exec-1]
    o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
12 2022-10-25 15:23:22.006 INFO 4616 --- [ scheduling-1]
    m.i.service.IpCountService : IP访问监控
13 +-----ip-address-----+--num--+
14 | 0:0:0:0:0:0:1 | 4 |
15 +-----+-----+
16 2022-10-25 15:23:23.014 INFO 4616 --- [ scheduling-1]
    m.i.service.IpCountService : IP访问监控
17 +-----ip-address-----+--num--+
18 | 0:0:0:0:0:0:1 | 10 |
19 +-----+-----+
20 2022-10-25 15:23:24.005 INFO 4616 --- [ scheduling-1]
    m.i.service.IpCountService : IP访问监控
21 +-----ip-address-----+--num--+
22 | 0:0:0:0:0:0:1 | 22 |
23 +-----+-----+
24 2022-10-25 15:23:25.015 INFO 4616 --- [ scheduling-1]
    m.i.service.IpCountService : IP访问监控
25 +-----ip-address-----+--num--+
26 | 0:0:0:0:0:0:1 | 30 |
27 +-----+-----+
28 2022-10-25 15:23:26.010 INFO 4616 --- [ scheduling-1]
    m.i.service.IpCountService : IP访问监控
29 +-----ip-address-----+--num--+
30 | 0:0:0:0:0:0:1 | 40 |
31 +-----+-----+
32 2022-10-25 15:23:27.005 INFO 4616 --- [ scheduling-1]
    m.i.service.IpCountService : IP访问监控
33 +-----ip-address-----+--num--+
34 | 0:0:0:0:0:0:1 | 44 |
35 +-----+-----+
36 2022-10-25 15:23:28.003 INFO 4616 --- [ scheduling-1]
    m.i.service.IpCountService : IP访问监控
37 +-----ip-address-----+--num--+
38 | 0:0:0:0:0:0:1 | 46 |
39 +-----+-----+
40 2022-10-25 15:23:29.013 INFO 4616 --- [ scheduling-1]
    m.i.service.IpCountService : IP访问监控
41 +-----ip-address-----+--num--+
42 | 0:0:0:0:0:0:1 | 46 |
43 +-----+-----+
44 2022-10-25 15:23:30.011 INFO 4616 --- [ scheduling-1]
    m.i.service.IpCountService : IP访问监控
45 +-----ip-address-----+--num--+
46 | 0:0:0:0:0:0:1 | 46 |
47 +-----+-----+

```

## 第七步：更改模式

```
1 tools:
2   ip:
3     cycle: 3
4     mode: simple
5     cycle-reset: false
```

## 第八步：重启服务，访问服务，查看日志

```
1 2022-10-25 15:26:21.006 INFO 17512 --- [ scheduling-1]
m.i.service.IpCountService : IP访问监控
2 +-----ip-address-----+
3 | 142.93.242.135 |
4 | 0:0:0:0:0:0:1 |
5 +-----+-----+
6 2022-10-25 15:26:24.007 INFO 17512 --- [ scheduling-1]
m.i.service.IpCountService : IP访问监控
7 +-----ip-address-----+
8 | 142.93.242.135 |
9 | 0:0:0:0:0:0:1 |
10 +-----+-----+
11 2022-10-25 15:26:27.007 INFO 17512 --- [ scheduling-1]
m.i.service.IpCountService : IP访问监控
12 +-----ip-address-----+
13 | 142.93.242.135 |
14 | 0:0:0:0:0:0:1 |
15 +-----+-----+
16 2022-10-25 15:26:30.007 INFO 17512 --- [ scheduling-1]
m.i.service.IpCountService : IP访问监控
17 +-----ip-address-----+
18 | 142.93.242.135 |
19 | 0:0:0:0:0:0:1 |
20 +-----+-----+
21 2022-10-25 15:26:33.007 INFO 17512 --- [ scheduling-1]
m.i.service.IpCountService : IP访问监控
22 +-----ip-address-----+
23 | 142.93.242.135 |
24 | 0:0:0:0:0:0:1 |
25 +-----+-----+
26 2022-10-25 15:26:36.014 INFO 17512 --- [ scheduling-1]
m.i.service.IpCountService : IP访问监控
27 +-----ip-address-----+
28 | 142.93.242.135 |
29 | 0:0:0:0:0:0:1 |
30 +-----+-----+
31 2022-10-25 15:26:39.010 INFO 17512 --- [ scheduling-1]
m.i.service.IpCountService : IP访问监控
```

```

32 +-----ip-address-----+
33 | 142.93.242.135 |
34 | 0:0:0:0:0:0:1 |
35 +-----+-----+
36 2022-10-25 15:26:42.002 INFO 17512 --- [ scheduling-1]
   m.i.service.IpCountService : IP访问监控
37 +-----ip-address-----+
38 | 142.93.242.135 |
39 | 0:0:0:0:0:0:1 |
40 +-----+-----+
41 2022-10-25 15:26:45.002 INFO 17512 --- [ scheduling-1]
   m.i.service.IpCountService : IP访问监控
42 +-----ip-address-----+
43 | 142.93.242.135 |
44 | 0:0:0:0:0:0:1 |
45 +-----+-----+
46 2022-10-25 15:26:48.012 INFO 17512 --- [ scheduling-1]
   m.i.service.IpCountService : IP访问监控
47 +-----ip-address-----+
48 | 142.93.242.135 |
49 | 0:0:0:0:0:0:1 |
50 +-----+-----+
51 2022-10-25 15:26:51.009 INFO 17512 --- [ scheduling-1]
   m.i.service.IpCountService : IP访问监控
52 +-----ip-address-----+
53 | 142.93.242.135 |
54 | 0:0:0:0:0:0:1 |
55 +-----+-----+
56 2022-10-25 15:26:54.002 INFO 17512 --- [ scheduling-1]
   m.i.service.IpCountService : IP访问监控
57 +-----ip-address-----+
58 | 142.93.242.135 |
59 | 0:0:0:0:0:0:1 |
60 +-----+-----+

```

## 第九步：是否周期内重置数据选项

```

1 tools:
2   ip:
3     cycle: 3
4     mode: detail
5     cycle-reset: true

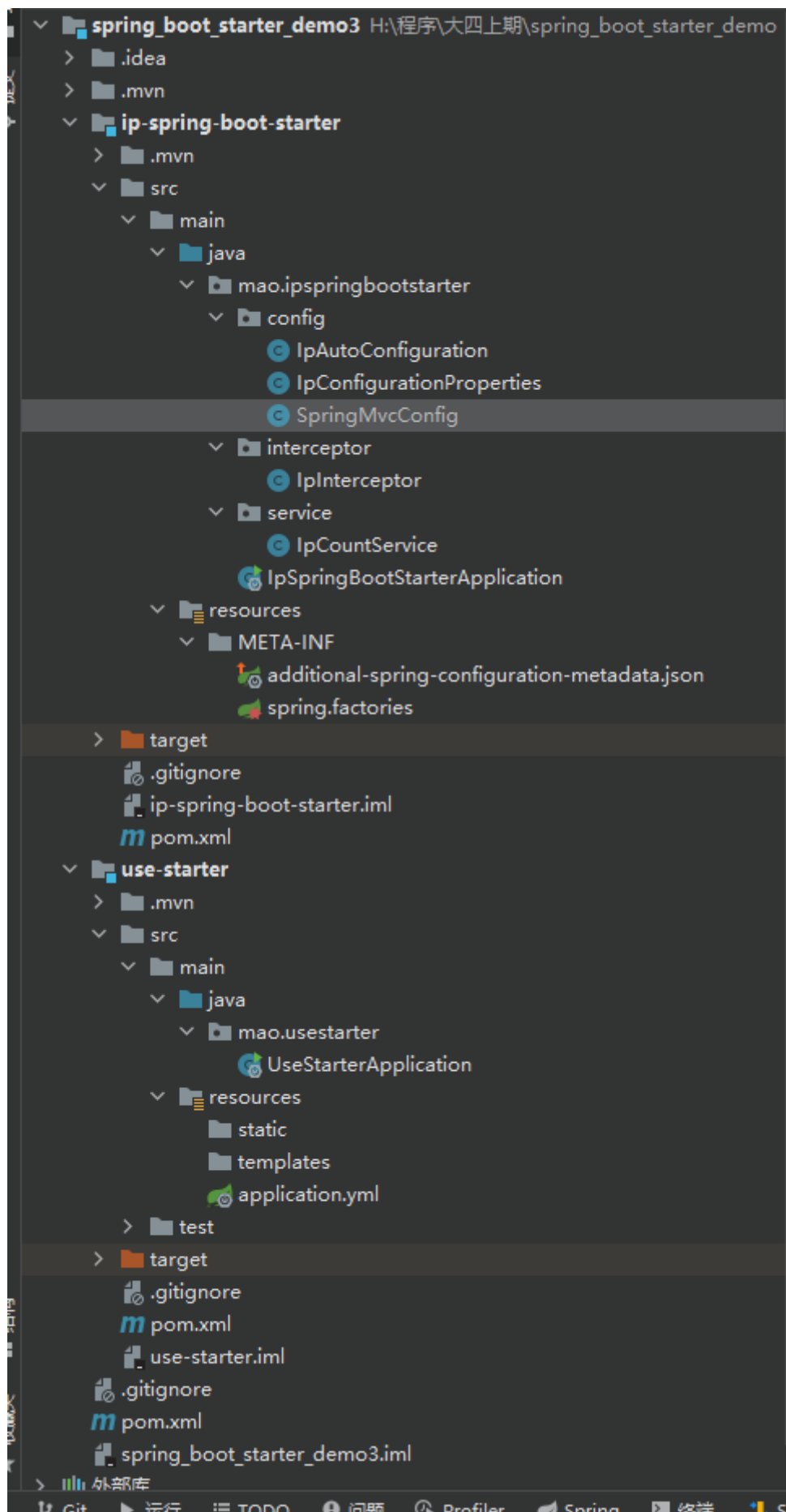
```

## 第十步：重启服务，访问服务，查看日志

```
1 +-----ip-address-----+--num--+
2 | 0:0:0:0:0:0:0:1 | 22 |
3 +-----+-----+
4 2022-10-25 15:29:24.004 INFO 7680 --- [ scheduling-1]
  m.i.service.IpCountService : IP访问监控
5 +-----ip-address-----+--num--+
6 | 0:0:0:0:0:0:0:1 | 12 |
7 +-----+-----+
8 2022-10-25 15:29:27.014 INFO 7680 --- [ scheduling-1]
  m.i.service.IpCountService : IP访问监控
9 +-----ip-address-----+--num--+
10 | 0:0:0:0:0:0:0:1 | 4 |
11 +-----+-----+
12 2022-10-25 15:29:30.006 INFO 7680 --- [ scheduling-1]
   m.i.service.IpCountService : IP访问监控
13 +-----ip-address-----+--num--+
14 | 0:0:0:0:0:0:0:1 | 6 |
15 +-----+-----+
16 2022-10-25 15:29:33.004 INFO 7680 --- [ scheduling-1]
   m.i.service.IpCountService : IP访问监控
17 +-----ip-address-----+--num--+
18 | 0:0:0:0:0:0:0:1 | 8 |
19 +-----+-----+
20 2022-10-25 15:29:36.011 INFO 7680 --- [ scheduling-1]
   m.i.service.IpCountService : IP访问监控
21 +-----ip-address-----+--num--+
22 +-----+-----+
```

项目结构：





---

end

---

by mao

2022 10 25

---