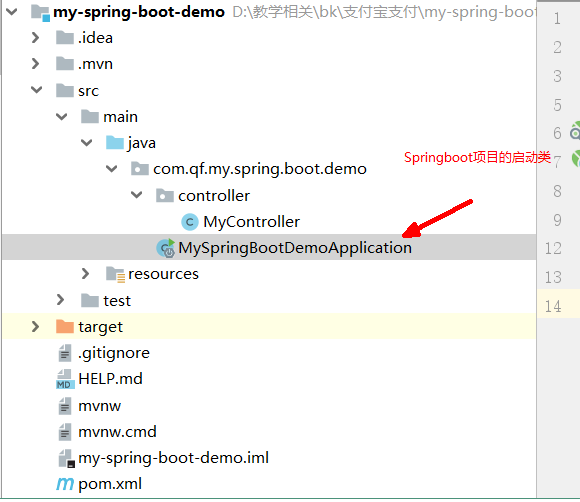
# 一、Springboot介绍

Springboot可以快速启动一个Spring项目，而且Spring项目中可以快速继承当前主流的第三方“中间件”。让启动项目变得简单，让配置变得简单。

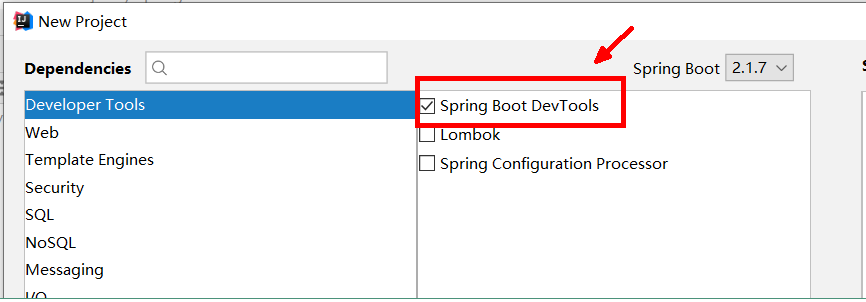
Springboot取代了SSM吗？简化了SS的配置，实际上Springboot可以理解成是封装了SS的一个工具



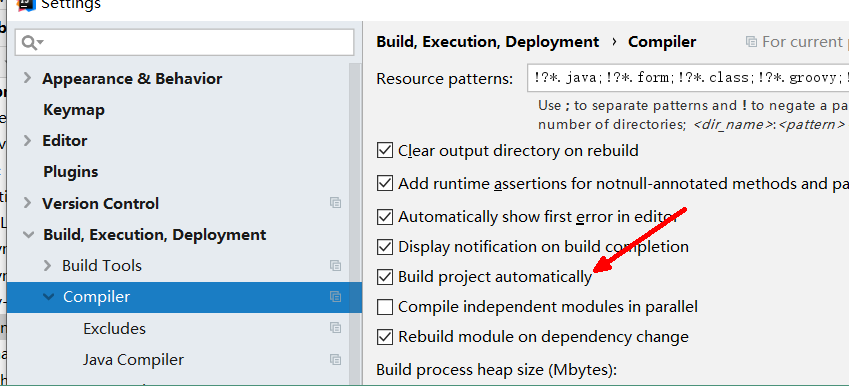
如果要运行这个Springboot项目，直接运行该类中的main即可

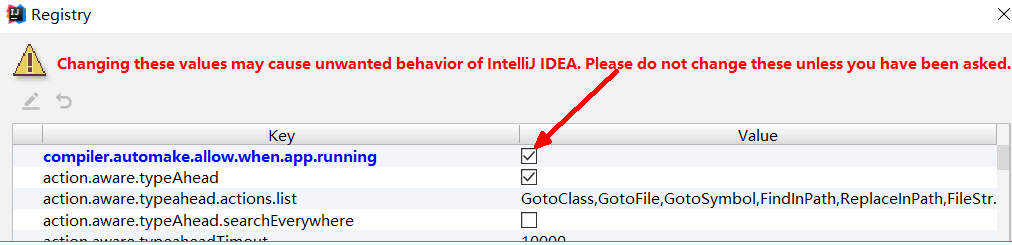
# 二、实现热部署

1.添加 springboot devtools依赖



2.对idea做配置





注意： 热部署之前，得先编译一下整个项目

# 三、关于日志



1.使用slf4j实现日志的面向接口编程

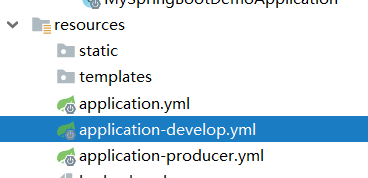
2.通过logger.debug("日志内容")来打印日志

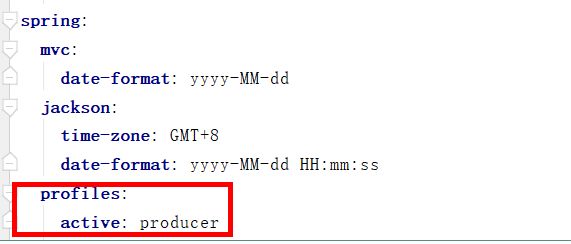
3.通过设置日志级别，来决定哪些日志会被打印。

|  |
| --- |
| @Controller @RequestMapping(**"demo"**) **public class** MyController {     *//创建一个可以在当前类中使用的日志对象，从slf4j的日志工厂获得* **private** Logger **logger** = LoggerFactory.*getLogger*(MyController.**class**);   @RequestMapping(**"showUrl"**)  @ResponseBody  **public** String showUrl(){    **logger**.debug(**"debug...."**);  **logger**.info(**"info...."**);  **logger**.warn(**"warn..."**);  **logger**.error(**"error...."**);    **return url**;  } |

# 四、配置文件在各环境之间的切换

项目开发周期中会遇到很多环境：测试环境、开发环境、生产环境。那么配置文件也需要做各种环境下的准备。





# 五、日志文件根据不同环境生成在不同文件夹内

1.将日志文件名修改成 logback-spring.xml

2.在文件中加入不同环境下的文件夹路径

<**springProfile name="develop"**>  
 <**property name="LOG\_HOME" value="d://logs//dev"** />  
</**springProfile**>  
<**springProfile name="producer"**>  
 <**property name="LOG\_HOME" value="d://logs//pro"** />  
</**springProfile**>

# 六、在SpringBoot中配置Filter

关键不是Filter怎么配，而是这种注解的方式如何替代XML的方式

@Configuration @Bean

## 1.创建Filter类并重写方法

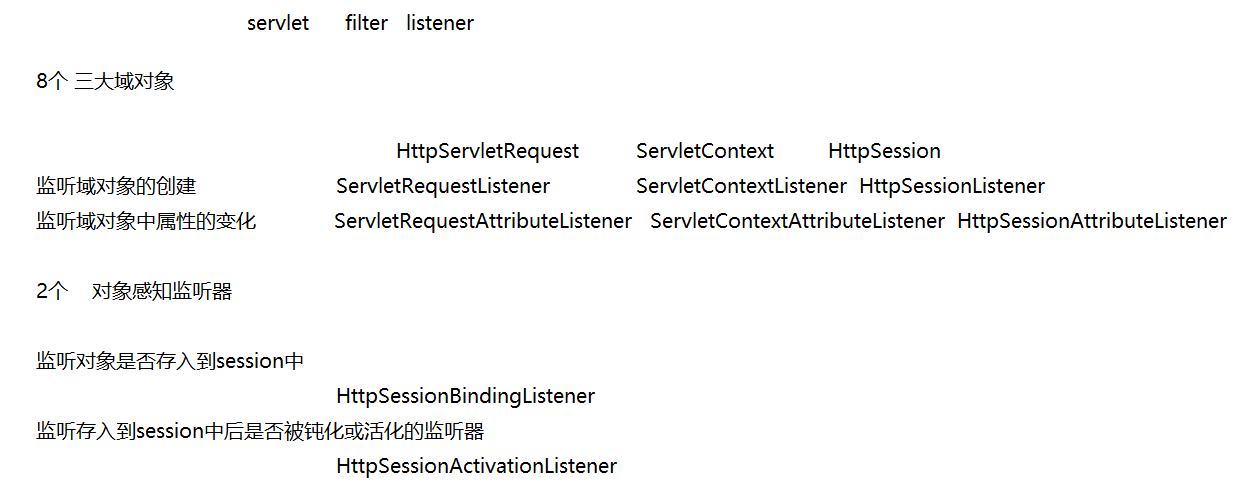
## 2.编写配置类并编写配置Filter的方法（替代XML）

|  |
| --- |
| @Configuration **public class** WebConfig {    @Bean  **public** FilterRegistrationBean getFilter(){  FilterRegistrationBean bean = **new** FilterRegistrationBean();  MyFilter myFilter = **new** MyFilter();  bean.setFilter(myFilter);  bean.setName(**"myfilter"**);  bean.addUrlPatterns(**"/\*"**);  **return** bean;  }  } |

# 七、在SpringBoot中配置Listener

## 1.选择你要创建的监听器

一共有8个，选一个。



## 2.通过注解的方式配置监听器。

|  |
| --- |
| @Configuration **public class** WebConfig {   @Autowired  **private** MyFilter **myFilter**;     @Autowired  **private** MyListener **myListener**;       @Bean  **public** ServletListenerRegistrationBean getListener(){  ServletListenerRegistrationBean bean = **new** ServletListenerRegistrationBean();  bean.setListener(**myListener**);  **return** bean;  } |

# 八、模板引擎技术——Thymeleaf

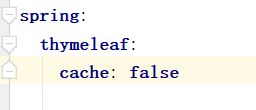
1.jsp在后端技术中做视图层的局限

jsp是必须运行在servlet容器内的， 比如tomcat，jetty等容器内。

2. 模板引擎技术

Thymeleaf，可以将html(模板)+数据 整合展示，当没有数据时，模板只会展示原静态的数据。当有数据时就会展示数据的部分，那么他与jsp的特点就在于，没有数据时，依然能展示。而不需要提供Servlet容器。

注意，可以关闭thymeleaf的缓存

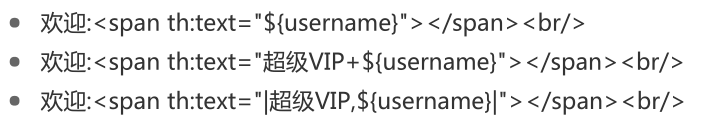


3.Thymeleaf

th是必须作为标签的属性而存在。

th:text

el表达式和文字结合使用可以使用+，或者||



th:if

满足if的条件，内容被显示

不满足，则不显示。

<**span th:if="${student.age}>=18"**>请上楼</**span**>  
<**span th:if="${student.age}<18"**>也上来</**span**>

th:each

# 九、SpringBoot整合MyBatis

1.添加依赖

|  |
| --- |
| <**dependency**>  <**groupId**>org.mybatis.spring.boot</**groupId**>  <**artifactId**>mybatis-spring-boot-starter</**artifactId**>  <**version**>1.3.2</**version**> </**dependency**> <**dependency**>  <**groupId**>mysql</**groupId**>  <**artifactId**>mysql-connector-java</**artifactId**> </**dependency**> |

2.编写配置文件中数据库信息

注意url后面要跟着时区

|  |
| --- |
| **spring**:  **datasource**:  **driver-class-name**: com.mysql.jdbc.Driver  **username**: root  **password**: 123456  **url**: jdbc:mysql://localhost:3306/db\_1904?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai |

3.编写entity、mapper、 xml

4.在SpringBoot启动（入口）程序上打上注解：

@MapperScan("mapper文件存放的包的路径")

|  |
| --- |
| @SpringBootApplication @MapperScan(**"my.spring.boot.thymeleaf.demo.mapper"**) **public class** MySpringBootThymeleafDemoApplication {  **public static void** main(String[] args) {  SpringApplication.*run*(MySpringBootThymeleafDemoApplication.**class**, args);  } } |

5.写ssm业务

6.需要在pom中指明打包xml文件的路径

|  |
| --- |
| <**build**>   <**resources**>  <**resource**>  <**directory**>src/main/java</**directory**>  <**includes**>  <**include**>\*\*/\*.xml</**include**>  </**includes**>  </**resource**>  </**resources**>  </build> |