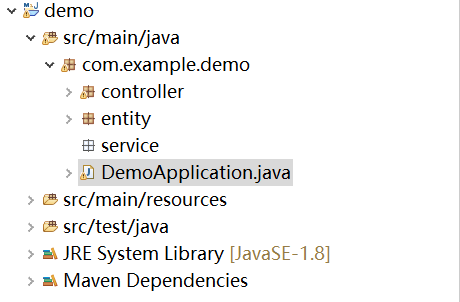
# 基础概念

## SpringBoot-web支持

spring-boot-starter-web：提供嵌入的Tomcat和SpringMVC依赖

## web工程项目结构

Spring Boot中启动类要放在root package下面



1. root package结构：com.example.demo
2. 应用启动类Application.java置于root package下，这样使用@ComponentScan注解的时候默认就扫描当前所在类的package
3. 实体（Entity）置于com.example.demo.entity包下
4. 逻辑层（Service）置于com.example.demo.service包下
5. controller层（web）置于com.example.demo.controller层包下
6. static可以用来存放静态资源
7. templates用来存放默认的模板配置路径

## SpringMVC自动配置

Spring Boot为Spring MVC提供适用于多数应用的自动配置功能。在Spring默认基础上，自动配置添加了以下特性：

1. 引入ContentNegotiatingViewResolver和BeanNameViewResolver beans。
2. 对静态资源的支持，包括对WebJars的支持。
3. 自动注册Converter，GenericConverter，Formatter beans。
4. 对HttpMessageConverters的支持。
5. 自动注册MessageCodeResolver。
6. 对静态index.html的支持。
7. 对自定义Favicon的支持。

### 3.1 自定义控制Spring MVC

如果想全面控制Spring MVC，你可以添加自己的@Configuration，并使用@EnableWebMvc对其注解。如果想保留Spring Boot MVC的特性，并只是添加其他的MVC配置(拦截器，formatters，视图控制器等)，你可以添加自己的WebMvcConfigurerAdapter类型的@Bean（不使用@EnableWebMvc注解）。

## 静态文件

1. 默认情况下，Spring Boot从classpath下一个叫/static（/public，/resources或/META-INF/resources）的文件夹或从ServletContext根目录提供静态内容。这使用了Spring MVC的ResourceHttpRequestHandler，可以通过添加自己的WebMvcConfigurerAdapter并覆写addResourceHandlers方法来改变这个行为（加载静态文件）。
2. 在一个单独的web应用中，容器默认的servlet是开启的，如果Spring决定不处理某些请求，默认的servlet作为一个回退（降级）将从ServletContext根目录加载内容。大多数时候，这不会发生（除非你修改默认的MVC配置），因为Spring总能够通过DispatcherServlet处理请求。
3. 此外，上述标准的静态资源位置有个例外情况是webjars内容。任何在/webjars/\*\*路径下的资源都将从jar文件中提供，只要它们以Webjars的格式打包。
4. 注：如果你的应用将被打包成jar，那就不要使用src/main/webapp文件夹。尽管该文件夹是一个共同的标准，但它仅在打包成war的情况下起作用，并且如果产生一个jar，多数构建工具都会静悄悄的忽略它

## 模板引擎

### 5.1 Spring Boot支持多种模版引擎包括：

1. FreeMarker

(2) Groovy

(3) Thymeleaf(官方推荐)

(4) Mustache

### 5.2 JSP技术Spring Boot官方是不推荐的

1. tomcat只支持war的打包方式，不支持可执行的jar。
2. Jetty 嵌套的容器不支持jsp
3. Undertow
4. 创建自定义error.jsp页面不会覆盖错误处理的默认视图，而应该使用自定义错误页面

### 5.3 注意：

当你使用上述模板引擎中的任何一个，它们默认的模板配置路径为：

src/main/resources/templates。当然也可以修改这个路径，具体如何修改，可在后续各模板引擎的配置属性中查询并修改。

# Thymeleaf模板引擎 -- 后端服务器渲染

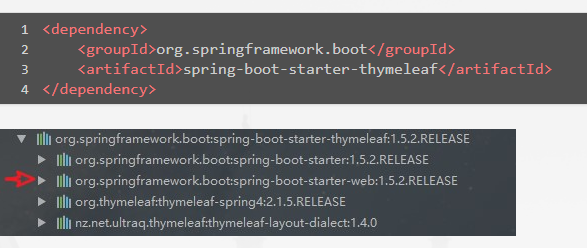
## 定义

Thymeleaf是一款用于渲染XML/XHTML/HTML5内容的模板引擎。类似JSP，Velocity，FreeMaker等，它也可以轻易的与Spring MVC等Web框架进行集成作为Web应用的模板引擎。与其它模板引擎相比，**Thymeleaf最大的特点是能够直接在浏览器中打开并正确显示模板页面，而不需要启动整个Web应用**。

## 特性

1. Spring MVC中@Controller中的方法可以直接返回模板名称，接下来Thymeleaf模板引擎会自动进行渲染
2. 模板中的表达式支持Spring表达式语言（Spring EL)
3. 表单支持，并兼容Spring MVC的数据绑定与验证机制
4. 国际化支持

## 引入依赖



如图所示，spring-boot-starter-thymeleaf会自动包含spring-boot-starter-web，所以我们就不需要单独引入web依赖了。

## 编写Controller

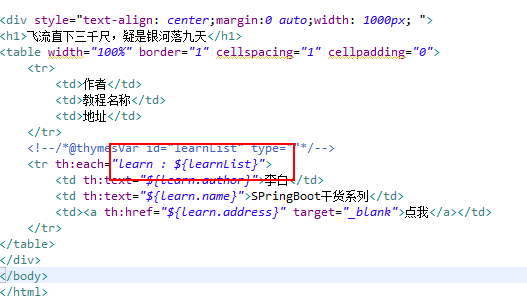


和SpringMVC一样，利用注解@Controller和注解@RequestMapping

返回ModelAndView视图

## 编写html

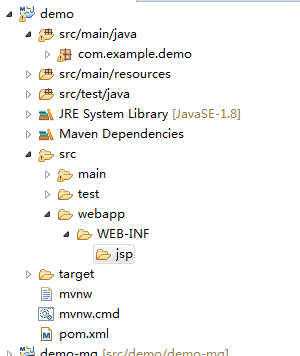
引入依赖后就在默认的模板路径src/main/resources/templates下编写模板文件即可完成。



直接从model中拿出learnList来用就行。

# JSP模板引擎 -- 后端服务器渲染

## 开发JSP的项目结构



在src目录下添加一个webapp/WEB-INF/jsp,用来存放jsp页面

## 引入依赖

### 2.1 pom.xml文件配置



依赖范围<scople>就是用来控制依赖和三种classpath(编译classpath，测试classpath、运行classpath)的关系，Maven有如下几种依赖范围：

1. compile:编译依赖范围。如果没有指定，就会默认使用该依赖范围。使用此依赖范围的Maven依赖，对于编译、测试、运行三种classpath都有效。典型的例子是spring-code,在编译、测试和运行的时候都需要使用该依赖。
2. test: 测试依赖范围。使用次依赖范围的Maven依赖，只对于测试classpath有效，在编译主代码或者运行项目的使用时将无法使用此依赖。典型的例子是Jnuit,它只有在编译测试代码及运行测试的时候才需要。
3. provided:已提供依赖范围。使用此依赖范围的Maven依赖，对于编译和测试classpath有效，但在运行时候无效。典型的例子是servlet-api,编译和测试项目的时候需要该依赖，但在运行项目的时候，由于容器以及提供，就不需要Maven重复地引入一遍。

### 2.2 application.properties

spring.mvc.view.prefix: /WEB-INF/jsp/

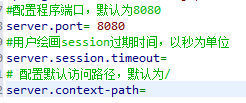
spring.mvc.view.suffix: .jsp

编写SpringMVC中的Controller -- 不变

在/WEB-INF/jsp下编写jsp文件

## 配置内嵌的tomcat服务器

### 3.1配置servlet容器



### 配置tomcat

# 配置Tomcat编码,默认为UTF-8

server.tomcat.uri-encoding=UTF-8

# 配置最大线程数

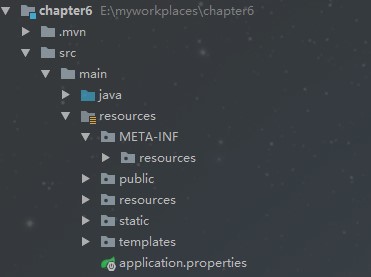
server.tomcat.max-threads=1000

## 4. 问题：



# SpringBoot的静态资源

## 项目结构



1. SpringBoot默认配置位置

classpath:/META-INF/resources

classpath:/resources

classpath:/static

classpath:/public

1. 优先级顺序为：META-INF/resources > resources > static > public
2. 修改静态资源配置

# 默认值为 /\*\*

spring.mvc.static-path-pattern=

# 默认值为

classpath:/META-INF/resources/,classpath:/resources/,classpath:/static/,classpath:/public/

spring.resources.static-locations=这里设置要指向的路径，多个使用英文逗号隔开

# 自定义SpringBoot中SpringMVC的配置

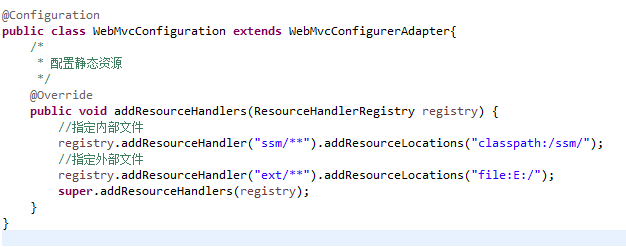
## 方式

如果Spring Boot提供的Sping MVC不符合要求，则可以通过一个配置类（注解有@Configuration的类）加上@EnableWebMvc注解来实现完全自己控制的MVC配置。

这个WebMvcConfigurerAdapter这个类，重写这个类中的方法可以让我们增加额外的配置。

## 自定义资源映射addResourceHandlers -- 指定访问地址以及实际的位置

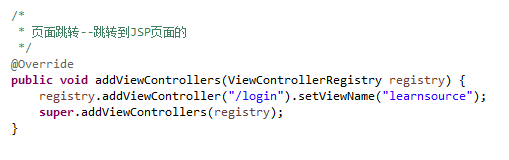
1. 自定义静态资源映射目录的话，只需重写addResourceHandlers方法即可。



## 页面跳转addViewControllers -- 跳转的路径主要看页面引擎

1. 重写WebMvcConfigurerAdapter中的addViewControllers方法就可以来跳转页面，而不用非得写@Controller
2. 值的指出的是，在这里重写addViewControllers方法，并不会覆盖

WebMvcAutoConfiguration中的addViewControllers（在此方法中，Spring Boot将“/”映射至index.html），这也就意味着我们自己的配置和Spring Boot的自动配置同时有效.



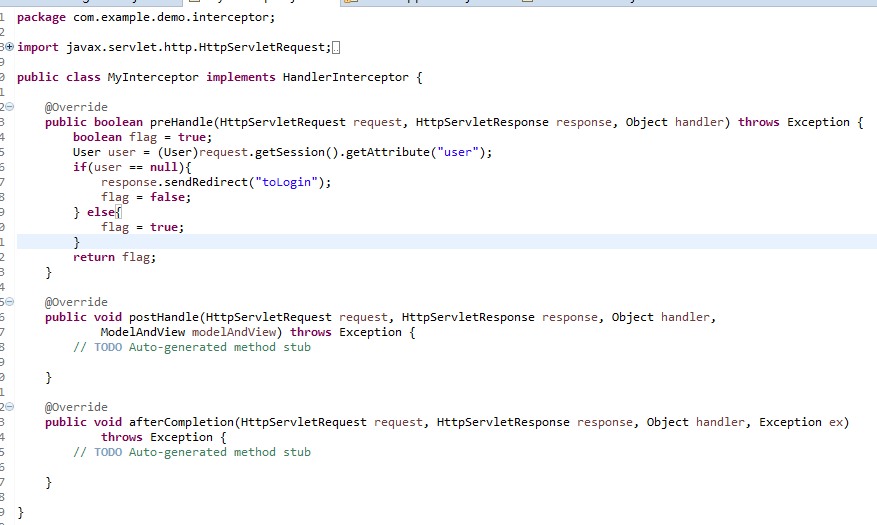
## 拦截器 -- 验证登录

### 步骤：

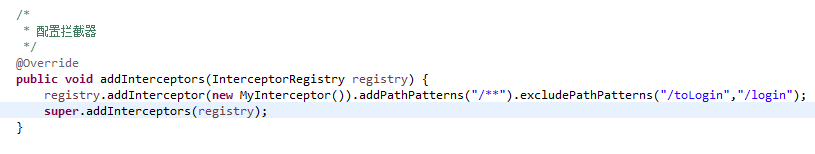
1. 创建我们自己的拦截器类并实现 HandlerInterceptor 接口
2. 其实重写WebMvcConfigurerAdapter中的addInterceptors方法把自定义的拦截器类添加进来即可

### 实例代码：

1. 创建拦截器类



1. 添加拦截器类



访问除了tologin和login两个路径，其他路径都会经过拦截器。