springboot项目中定义了很多类，我们在rest返回中直接返回或者在返回对象中使用这些类，spring已经使用jackson自动帮我们完成这些的to json。但是有时候自动转的json内容太多，或者格式不符合我们的期望，因此需要调整类的to json过程，或者说希望自定义类的json过程。

解决办法：

使用@JsonIgnoreProperties、@JsonIgnore、@JsonFormat。

@JsonIgnore注解用来忽略某些字段，可以用在变量或者Getter方法上，用在Setter方法时，和变量效果一样。这个注解一般用在我们要忽略的字段上。

@JsonIgnoreProperties(ignoreUnknown = true)，将这个注解写在类上之后，就会忽略类中不存在的字段。这个注解还可以指定要忽略的字段，例如@JsonIgnoreProperties({ “password”, “secretKey” })

@JsonFormat可以帮我们完成格式转换。例如对于Date类型字段，如果不适用JsonFormat默认在rest返回的是long，如果我们使用@JsonFormat(timezone = “GMT+8”, pattern = “yyyy-MM-dd HH:mm:ss”)，就返回"2018-11-16 22:58:15"

@Scheduled(fixedRate = 5000) 五秒执行一次

@Scheduled(fixedDelay = 5000) 执行完一次之后过五秒再继续第二次

使用SpringApplication.run(Application.class)执行主函数

[@EnableScheduling](https://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/#scheduling-enable-annotation-support) ensures that a background task executor is created. Without it, nothing gets scheduled.

Spring boot 框架构建， URL 路由，参数管理，自定义配置，JPA，Thymeleaf 模板，表单处理，分页处理，开发者工具。

-------------------------------------------------

Spring boot框架构建？

框架，简化新spring应用的初始搭建以及开发过程，使用自动配置，使开发人员不再需要样板化的配置（XML Configration）。

Spring的争议：大量的XML配置，复杂的依赖管理。

Springboot： 自动配置，起步依赖，命令行界面，actuator(监控运行程序状态的组件)

解决了什么问题？搭建，配置，编码，部署，监控简单。。

能干什么？基于模板的Web Application开发，restful服务开发，微服务

Spring构建： initializr,IDEA,spring boot CLI

Spring boot 启动：

1. java类文件启动（@SpringBootApplication注解 + SpringApplication.run() + 运行main方法）
2. mvn spring-root:run启动 在项目目录下运行此命令
3. java -jar命令启动 mvn package->进入到package所在目录，然后java -jar

------------------------------------------------------------

URL路由

Restful API

@Restcontroller注解

@RequestMapping注解

@RequestMapping简写形式

@Controller注解

Restful API????例子

GET http:localhost:8080/api/vi/books 获取读书清单列表

POST http:localhost:8080/api/vi/books新增一个清单

GET 、books/{id} 获取一条读书清单

PUT /books 更新一个读书清单

DELETE books/{id} 删除一个读书清单

@RestController

@RequestMapping(value = “/say”, method=RequestMethod.GET)

不标记请求类型就是支持所有请求类型

Requestmapping的简写形式：

@PostMapping(“/say”) 代表了一个post请求

Controller注解？

返回的字符串，再去找对应的模板thymeleaf,如何找到？

模板在resource下面的template定义

所以controller用来标记返回的是view也就是个模板，类似一个HTML文档，

如果返回一个JSON，返回一个string，用restcontroller

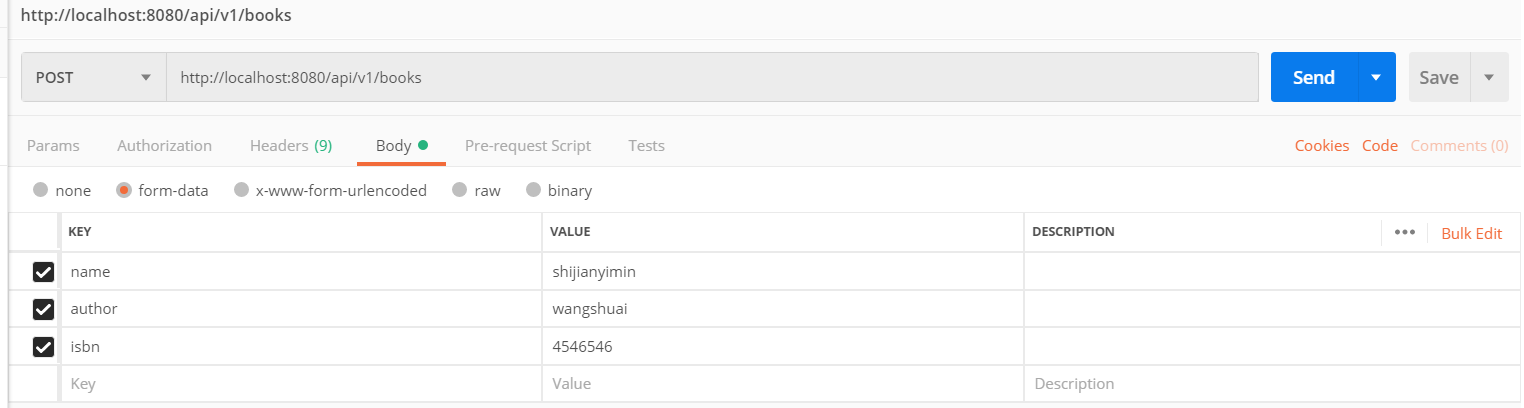
@ResponseBody标记之后，如果是字符串就返回字符串，如果是实体就返回一个JSON

参数管理 @PathVariable

@GetMapping(**"/books/{id}/{username}"**)  
**public** Object getOne(@PathVariable **long** id, @PathVariable String username){  
 System.***out***.println(id);  
 System.***out***.println(username);  
 Map<String, Object> book = **new** HashMap<>();  
 book.put(**"name"**,**"hulianwangshijieguan"**);  
 book.put(**"isbn"**,**"45646546"**);  
 book.put(**"author"**,**"wangshuai"**);  
 **return** book;  
}

参数处理 requestparam 表单接收

POST http:localhost:8080/api/vi/books新增一个清单



@PostMapping(**"/books"**)  
**public** Object post(@RequestParam(**"name"**) String name, @RequestParam(**"author"**) String author, @RequestParam(**"isbn"**) String isbn){  
 Map<String,Object> book = **new** HashMap<>();  
 book.put(**"name"**,name);  
 book.put(**"author"**, author);  
 book.put(**"isbn"**,isbn);  
 **return** book;  
}

GET http:localhost:8080/api/vi/books 获取读书清单列表

：

@GetMapping(**"/books"**)  
**public** Object getAll(@RequestParam(**"page"**) **int** page,@RequestParam(value = **"size"**, defaultValue = **"10"**) **int** size){  
 Map<String, Object> book1 = **new** HashMap<>();  
 book1.put(**"name"**,**"hulianwangshijieguan"**);  
 book1.put(**"isbn"**,**"45646546"**);  
 book1.put(**"author"**,**"wangshuai"**);  
  
 Map<String, Object> book2 = **new** HashMap<>();  
 book2.put(**"name"**,**"hulianwangshijieguan"**);  
 book2.put(**"isbn"**,**"456465467"**);  
 book2.put(**"author"**,**"wangshuai1"**);  
 List<Map> contents = **new** ArrayList<>();  
 contents.add(book1);  
 contents.add(book2);  
 Map<String, Object> pagemap = **new** HashMap<>();  
 pagemap.put(**"page"**,page);  
 pagemap.put(**"size"**,size);  
 pagemap.put(**"content"**, contents);  
 **return** pagemap;  
}

总结：

GET http:localhost:8080/api/vi/books/{id} 一般用pathvariable

Requestparam一般用于提交一个表单，比如分页方式获取数据。<http://localhost:8080/api/v1/books?page=1>

自定义配置

配置文件，自定义属性配置，多环境配置

改端口号port，在resource文件夹下修改application.properties文件，写入：

Server.port = 8081

将公有的部分URL加入根目录下，即修改根目录，使用：

**server.servlet.context-path**=**/api**

定义日志级别，日志位置：

**logging.level.root** = **WARN  
logging.level.com.lrm** = **DEBUG  
logging.file.path**=

在properties或者yml配置文件里写入常量，例如book.name/author/isbn…

然后在controller类里增加这几个常量，并注解@Value(),即可直接使用

Controller里

@Value(**"${book.name}"**)  
**private** String **name**;  
@Value(**"${book.author}"**)  
**private** String **author**;  
@Value(**"${book.isbn}"**)  
**private** String **isbn**;

配置文件里

**book.name**=**hulianwangshijieguan  
book.author**=**wangshuai  
book.isbn**=**45646546**

自定义属性配置：

如何直接注入一个类的对象？？？ Autowired用于注入一个实体对象，

@Component标记class，使spring boot识别它，识别了才能注入此类的一个实体对象

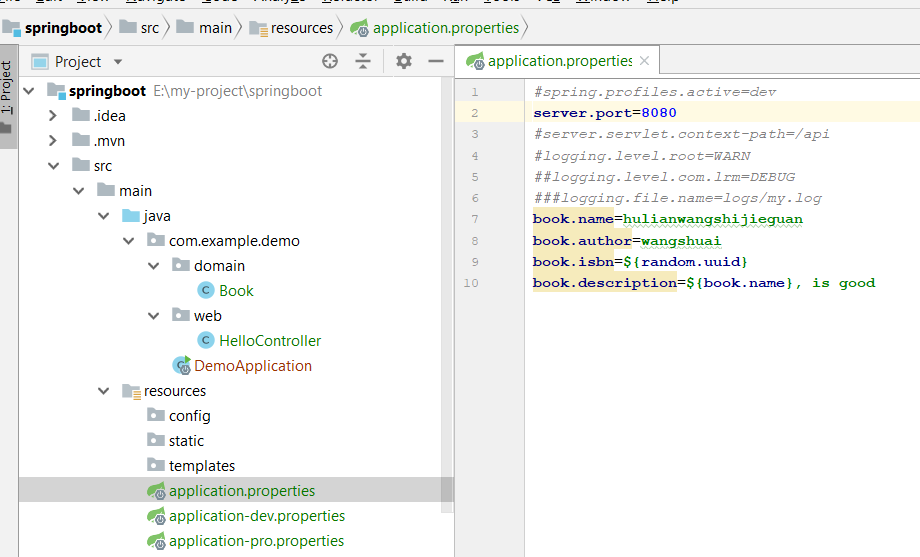
@ConfigrationProperties（prefix=?）用于表明类与配置文件中的哪个前缀匹配  
**public class** HelloController {  
 @Autowired  
 **private** Book **book**;

@GetMapping(**"/books/{id}"**)  
 **public** Object getOne(@PathVariable **long** id){**return book**;

}

@Component  
@ConfigurationProperties(prefix = **"book"**)  
**public class** Book {  
  
 **private** String **name**;  
*// @Value("${book.author}")* **private** String **author**;  
*// @Value("${book.isbn}")* **private** String **isbn**;  
 **public** Book(){  
  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public void** setAuthor(String author) {  
 **this**.**author** = author;  
 }  
  
 **public void** setIsbn(String isbn) {  
 **this**.**isbn** = isbn;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public** String getAuthor() {  
 **return author**;  
 }  
  
 **public** String getIsbn() {  
 **return isbn**;  
 }  
}

多环境配置：



建立了开发测试生产三个环境，使用spring.propfiles.active=dev指定使用开发环境配置。

JPA数据库操作

数据持久化-> 保存到数据库

配置，基本查询，复杂查询，自定义查询，自定义更新，事务

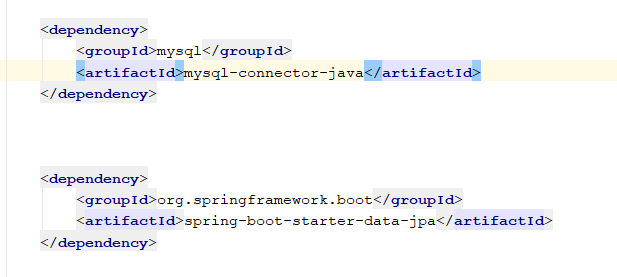
配置：

JPA ： java persistent API，在hibernate中使用。

Spring data JPA:基于JPA进一步简化了数据访问层的实现，只需要编写repository

如何使用spring data JPA？？？

引入pom文件

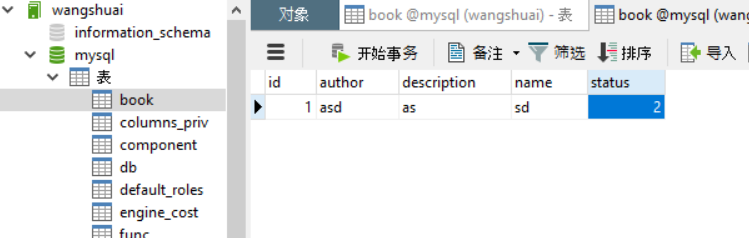


改写properties配置文件：

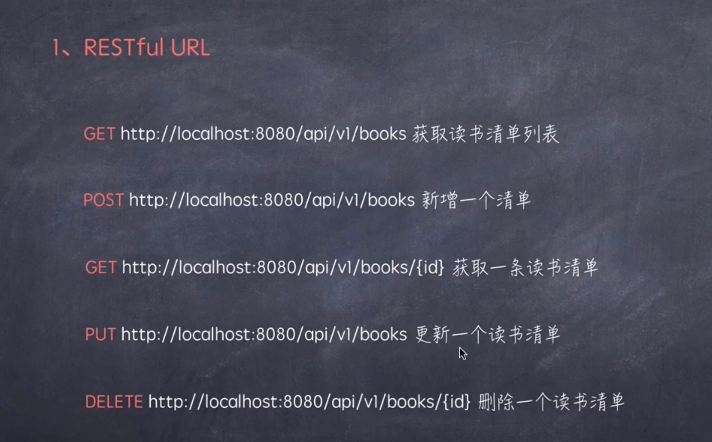
**spring.datasource.driver-class-name**=**com.mysql.cj.jdbc.Driver  
spring.datasource.url**=**jdbc:mysql://localhost:3306/mysql?useUnicode=true&characterEncoding=utf-8  
spring.datasource.username**=**root  
spring.datasource.password**=**root  
  
  
spring.jpa.hibernate.ddl-auto**=**update  
spring.jpa.show-sql**=**true**

打开navicat可视化mysql数据库

运行application



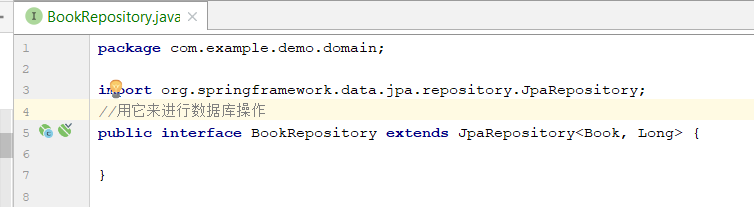
JPA基本查询：增删改查

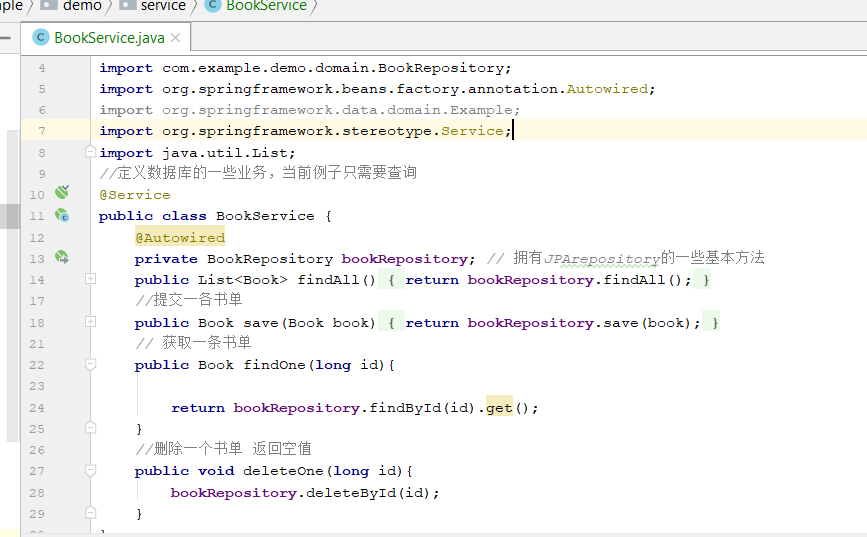


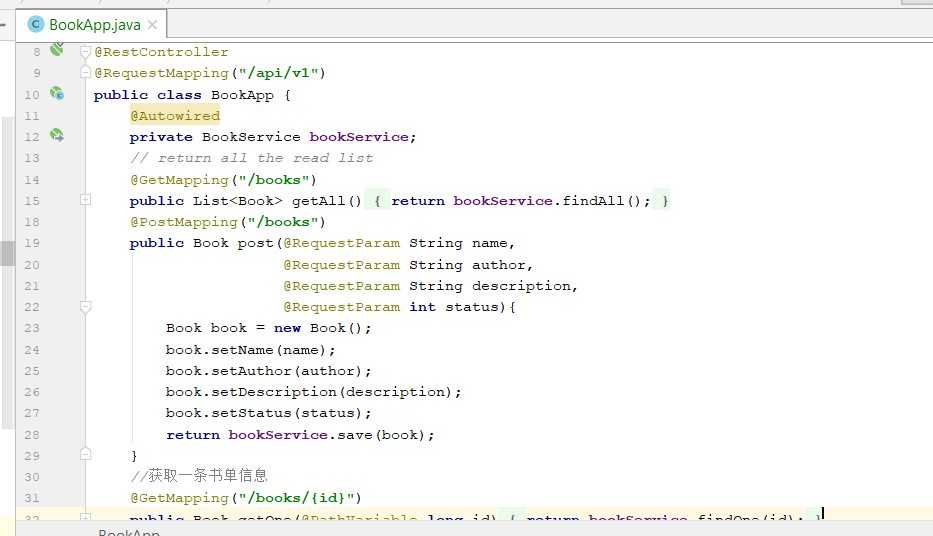
步骤：首先通过bookrepository接口继承JPARepository

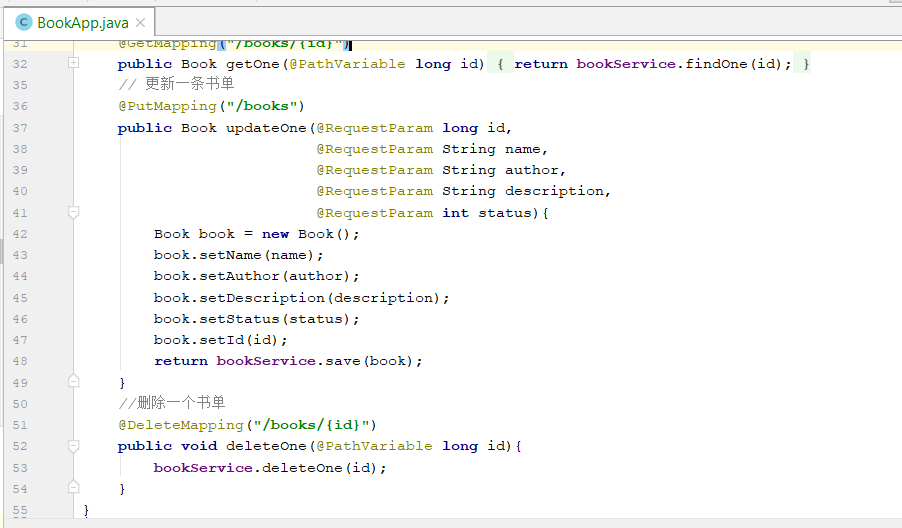
然后明确web层和service层，service层用来定义CRUD操作具体实现，通过**bookRepository，web层用来直接调用写好的方法。**

**具体代码：**

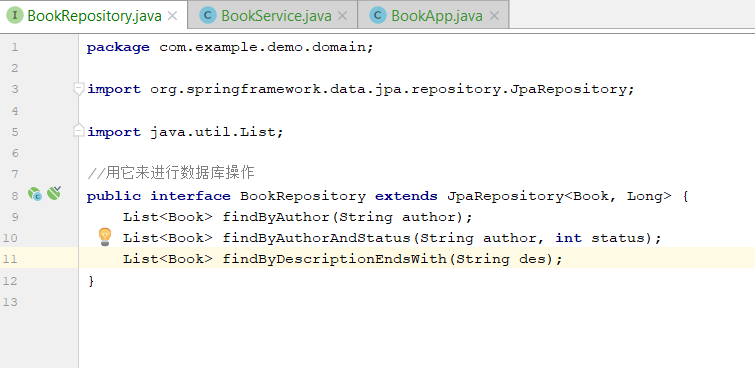


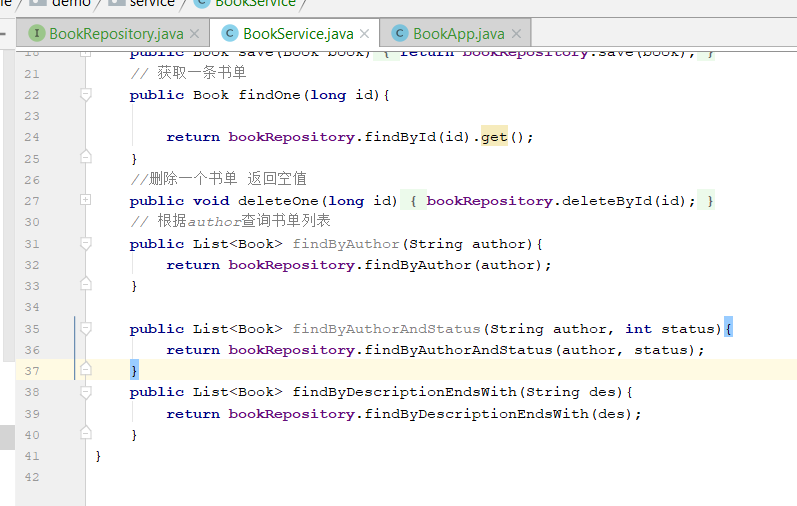


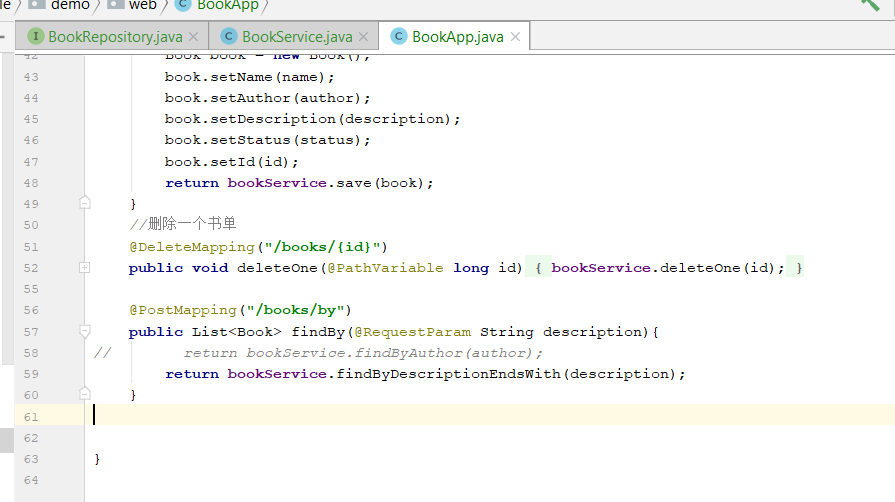




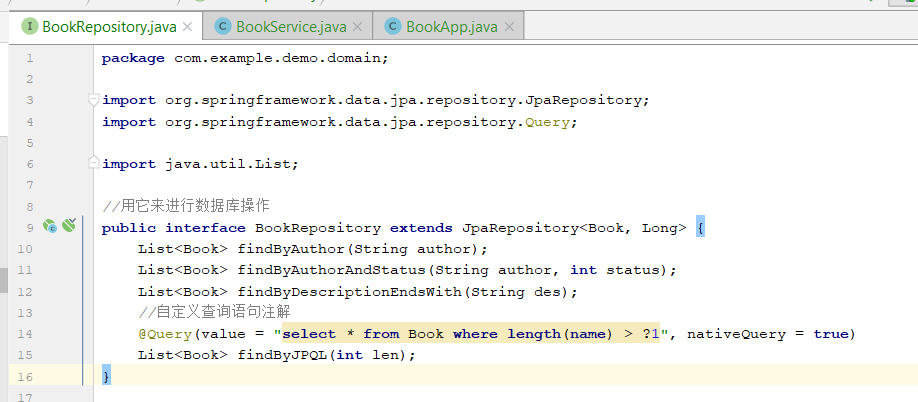
JPA复杂查询

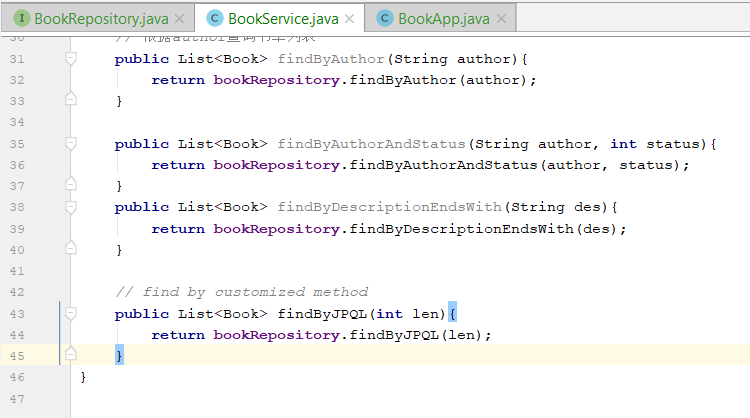


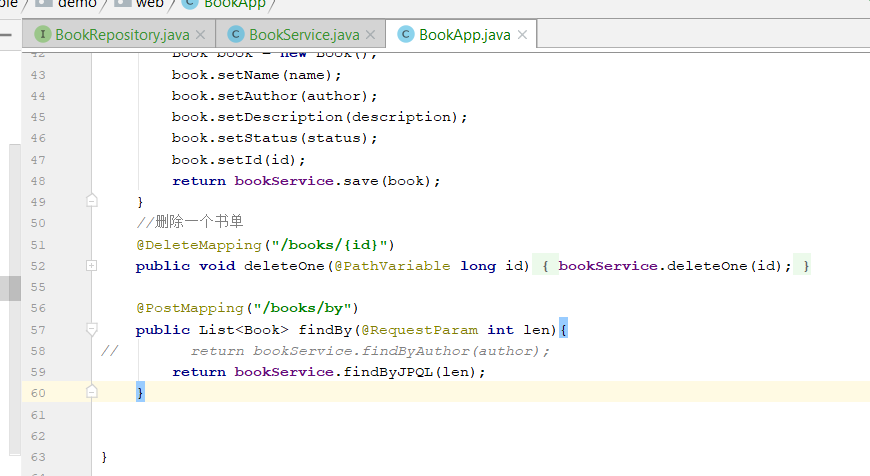




JPA自定义查询



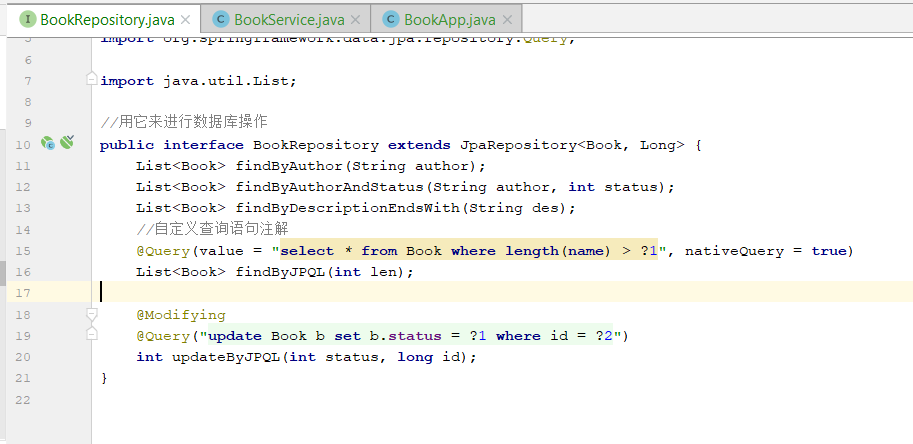


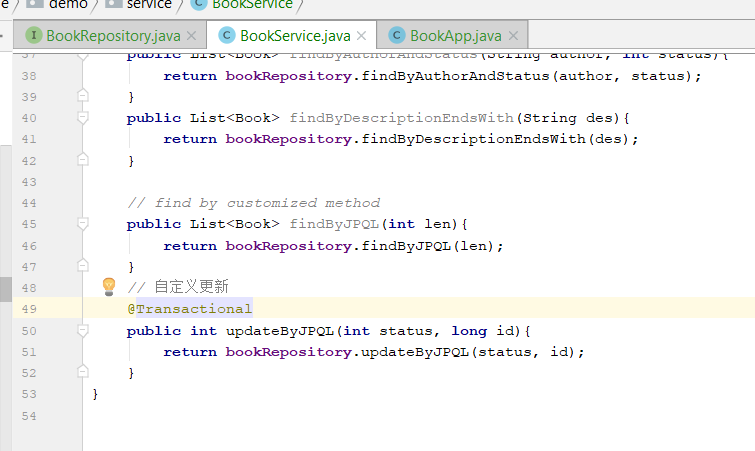


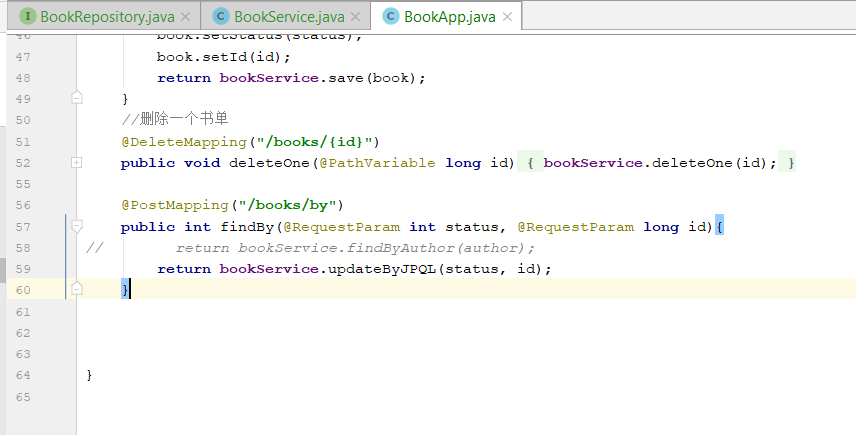
JPA自定义更新

如果只需要修改表中某一列数据，就不需要调用JPARepository里面的save方法，可以自定义更新

1. 修改ID为18的书记status为2







数据库事务

使用@Transaction在service层标记一个方法，那么方法中的所有SQL操作都在一个事务中，要么全部成功，要么全部失败。

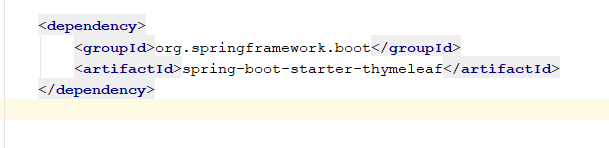
Thymeleaf模板的使用-》渲染界面展示

使用thymeleaf3,取值，静态资源处理，判断，迭代，URL

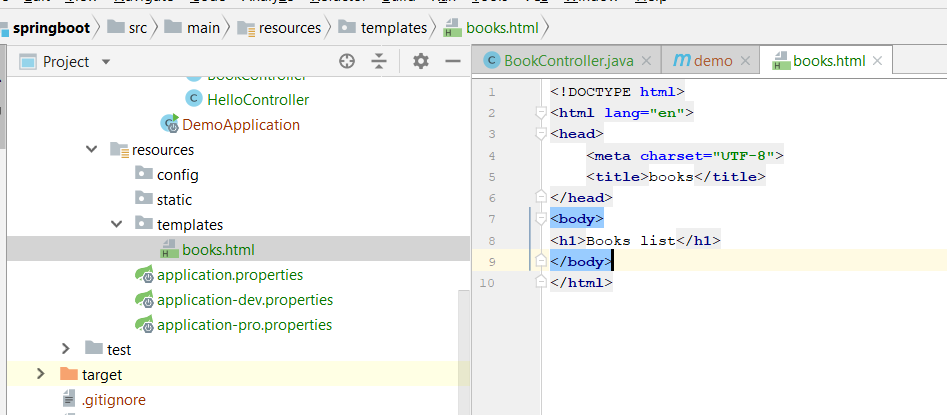
回忆-》》》 controller关键字标记了之后，会自动去找对应名字的模板

如何使用？

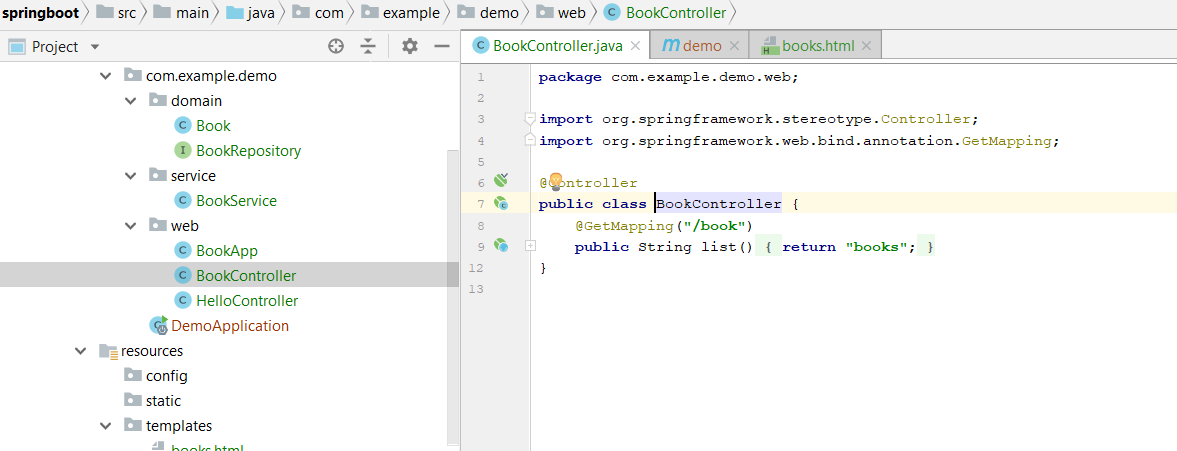
Pom引入



创建template模板



建立bookcontroller并且附加@Controller关键词



就会返回对应的books模板

Spring boot2默认支持thymeleaf3不需要再手动引入

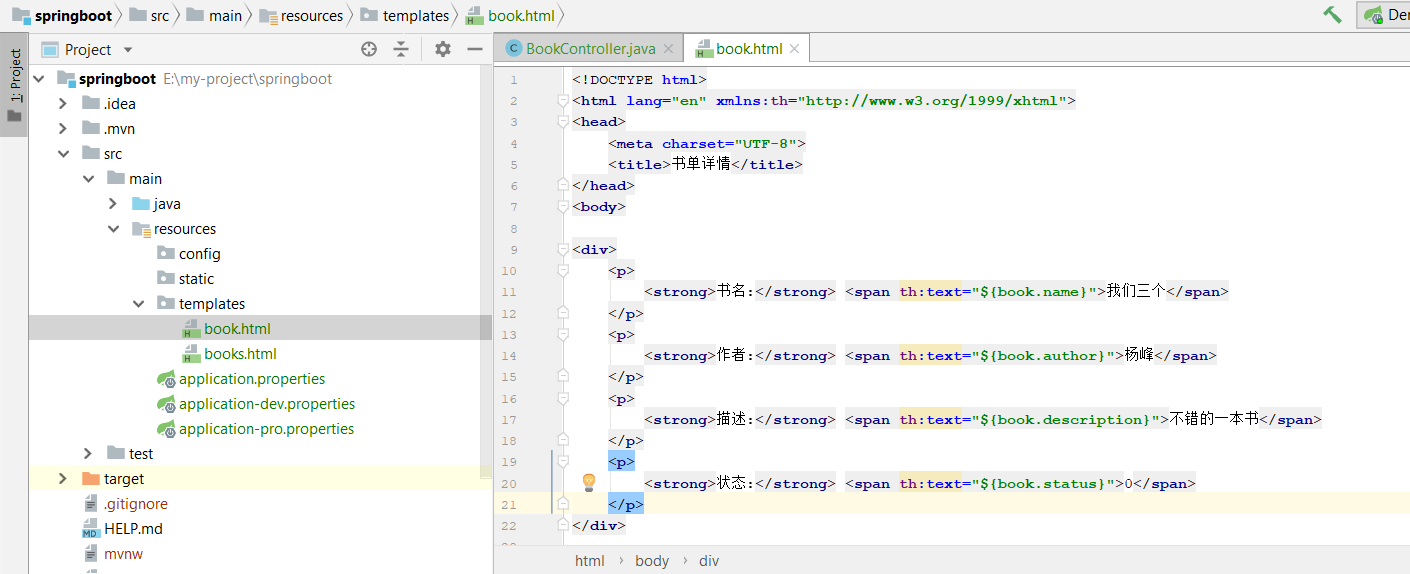
Thymeleaf取值

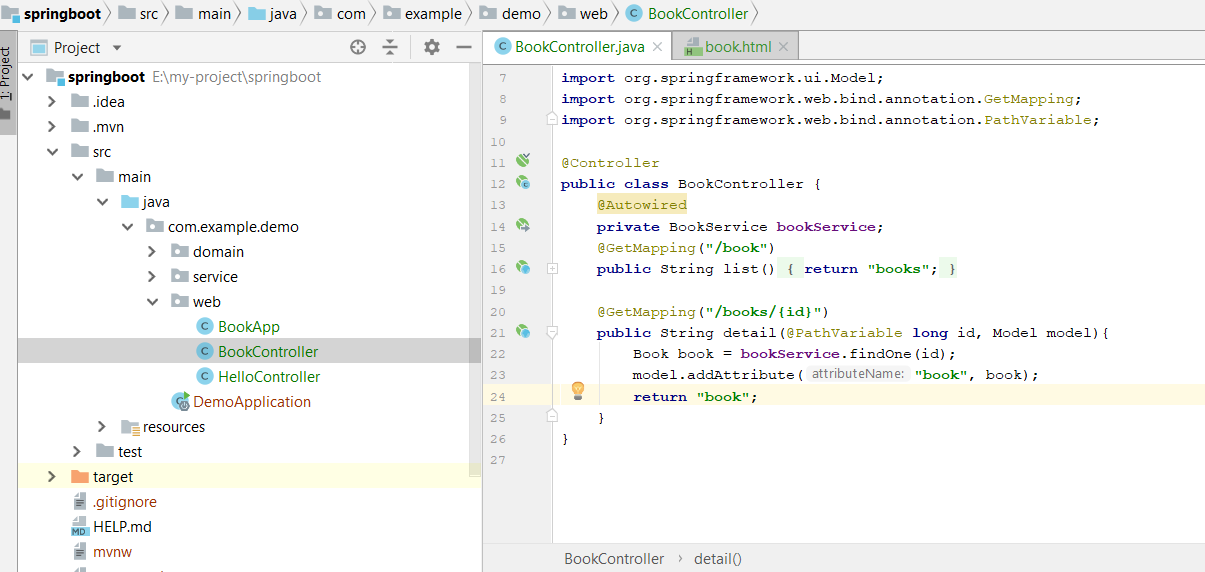
如何使用thymeleaf3？？？

前后端分离，JSP没有前后端分离

如何使用spring boot实现简单的书单管理？？？

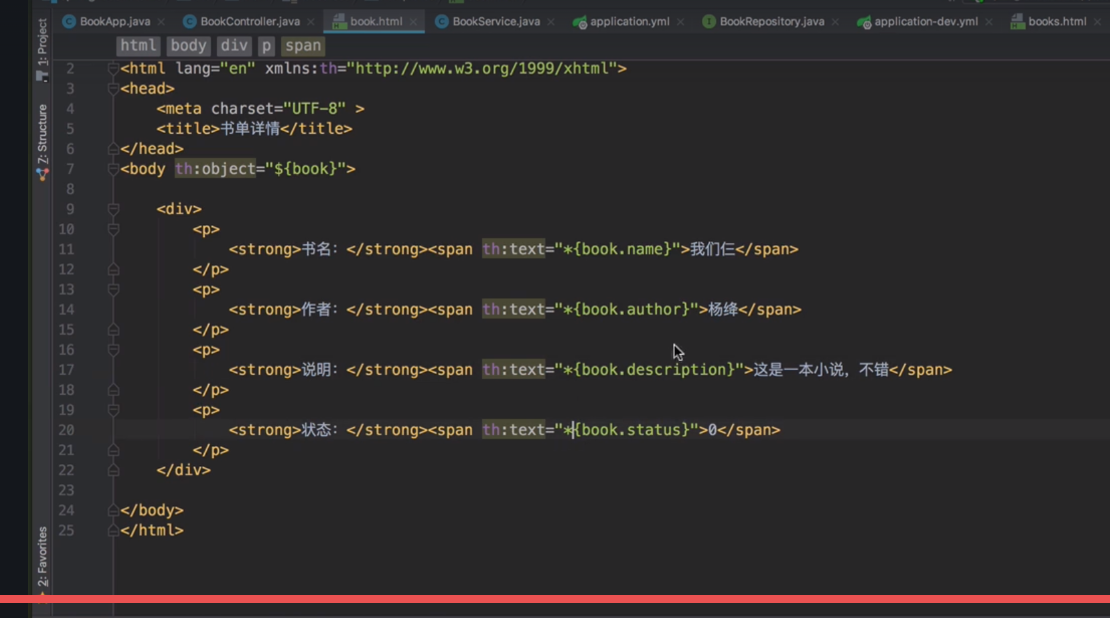
如何从数据库取得值，然后渲染到页面上？？？





通过th:text=”${book.name}”

另一种方式



都可以

每次都可以省略写book了

Thymeleaf静态资源处理

首先把静态资源放在正确的位置，static文件夹下。

HTML文件里面引入bootstrap，使用相对路径。

**th:href="@{/css/bootstrap.min.css}"**

**th:src="@{/js/bootstrap.min.js}"**

<!DOCTYPE **html**>  
<**html lang="en" xmlns:th="http://www.w3.org/1999/xhtml"**>  
<**head**>  
 <**title**>书单详情</**title**>  
 <**meta charset="UTF-8"**>  
 <**meta http-equiv="X-UA-Compatible" content="IE=edge"**>  
 <**meta name="viewport" content="width=device-width, initial-scale=1"**>  
  
 *<!-- Bootstrap -->* <**link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" href="../static/css/bootstrap.min.css"**>  
  
</**head**>  
<**body**>  
  
<**div class = "container" style="max-width**: 700**px**;**"**>  
 <**h2 class="page-header" style="margin-top**: 50**px**;**"**>详情</**h2**>  
 <**div class = "well"**>  
 <**p**>  
 <**strong**>书名:</**strong**> <**span th:text="${book.name}"**>我们三个</**span**>  
 </**p**>  
 <**p**>  
 <**strong**>作者:</**strong**> <**span th:text="${book.author}"**>杨峰</**span**>  
 </**p**>  
 <**p**>  
 <**strong**>描述:</**strong**> <**span th:text="${book.description}"**>不错的一本书</**span**>  
 </**p**>  
 <**p**>  
 <**strong**>状态:</**strong**> <**span th:text="${book.status}"**>0</**span**>  
 </**p**>  
 </**div**>  
</**div**>  
*<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->*<**script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"**></**script**>  
<**script th:src="@{/js/bootstrap.min.js}" src="../static/js/bootstrap.min.js"**></**script**>  
  
</**body**>  
</**html**>

Thymeleaf 判断

使用switch

<**p th:switch="${book.status}"**>  
 <**strong**>状态:</**strong**>  
  
<**span th:case="0"**>响度</**span**>  
<**span th:case="1"**>在读</**span**>  
<**span th:case="2"**>已读</**span**>  
  
  
  
</**p**>

另一个场景：

想读的时候，给出提示

已读或者在读：给出出一个绿色的提示，不错 开始行动了

使用bootstrap模板的判断

<**div class="alert alert-success" th:unless="\*{status == 0}"**>  
 <**strong**>不错</**strong**>，你已经开始行动了  
</**div**>  
<**div class="alert alert-warning" th:if="\*{status == 0}"**>  
 <**strong**>注意</**strong**>，你还没开始行动  
</**div**>

Switch可以判断多个元素，也可以直接使用if,unless判断

Thymeleaf迭代

获取书单的列表，把书单的所有信息在一个页面显示出来

将需要循环的标签放在需要循环的元素上面，

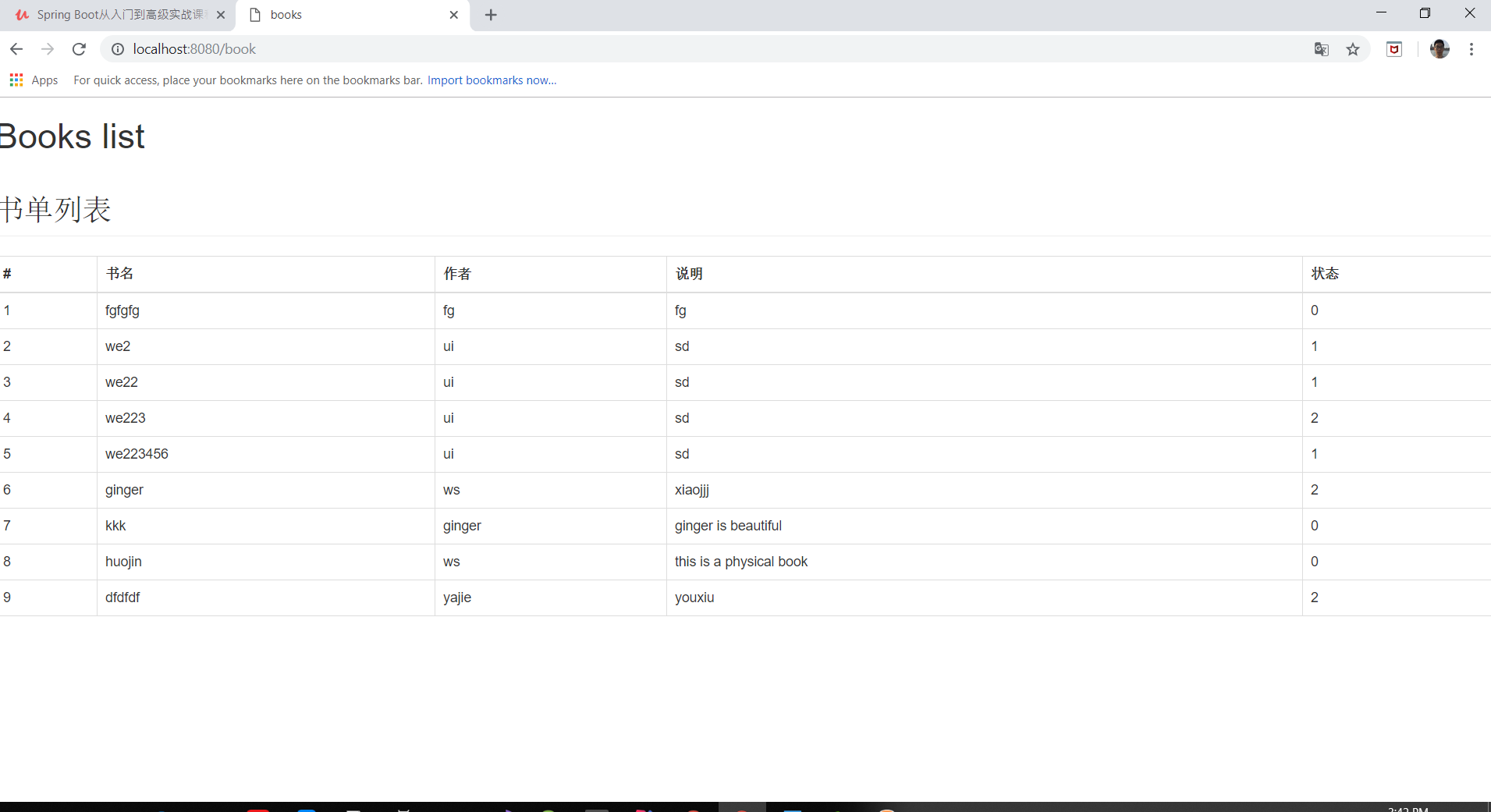
比如要把所有书单显示出来，循环书单，将每一本书放到每一行里面。

所以在<tr></tr>里循环

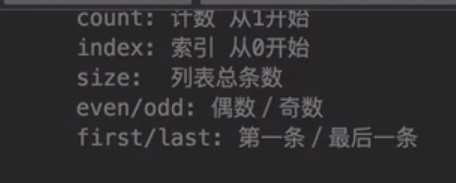
<**tr th:each="book,iterStat:${books}"**>

接着使用$取出对应的属性

<**td th:text="${iterStat.count}"**>1</**td**>  
<**td th:text="${book.name}"**>书名</**td**>  
<**td th:text="${book.author}"**>作者</**td**>  
<**td th:text="${book.description}"**>说明</**td**>  
<**td th:text="${book.status}"**>状态</**td**>



iterStat的属性如下：



THymeleaf URL使用

点击书单列表如何能跳转到详情页面？？？

*<!--<td><a href="#" th:text="${book.name}" th:href="@{/books/{id}(id=${book.id})}">书名</a></td>-->*<**td**><**a href="#" th:text="${book.name}" th:href="@{'/books/' + ${book.id}}"**>书名</**a**></**td**>

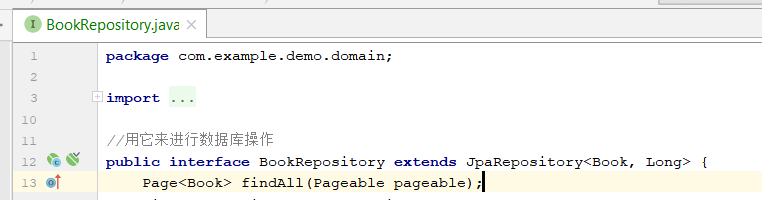
表单处理

分页处理：

分页查询方法：

目前数据全部在第一页渲染出来，所以需要分页获取

分页查询



自动重启：

File settings->compiler->build automatically

Maintenance->allow app auto running