

# ssm练习第二天

## 第一章：ssm框架整合

### 1、创建maven的工程

1. 创建ssm\_parent父工程（打包方式选择pom，必须的）
2. 创建ssm\_web子模块（打包方式是war包）
3. 创建ssm\_service子模块（打包方式是jar包）
4. 创建ssm\_dao子模块（打包方式是jar包）
5. 创建ssm\_domain子模块（打包方式是jar包）
6. 创建ssm\_utils子模块（打包方式是jar包）
7. web依赖于service，service依赖于dao，dao依赖于domain
8. 在ssm\_parent的pom.xml文件中引入坐标依赖

```
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         http://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <modelVersion>4.0.0</modelVersion>
7
8     <groupId>com.itheima</groupId>
9     <artifactId>ssm_parent</artifactId>
10    <packaging>pom</packaging>
11    <version>1.0-SNAPSHOT</version>
12
13    <properties>
14        <spring.version>5.0.2.RELEASE</spring.version>
15        <slf4j.version>1.6.6</slf4j.version>
16        <log4j.version>1.2.12</log4j.version>
17        <shiro.version>1.2.3</shiro.version>
18        <mysql.version>5.1.6</mysql.version>
19        <mybatis.version>3.4.5</mybatis.version>
20        <spring.security.version>5.0.1.RELEASE</spring.security.version>
21    </properties>
22
23    <dependencies>
24        <!-- spring -->
25        <dependency>
26            <groupId>org.aspectj</groupId>
27            <artifactId>aspectjweaver</artifactId>
28            <version>1.6.8</version>
29        </dependency>
30        <dependency>
31            <groupId>org.springframework</groupId>
```

```
31         <artifactId>spring-aop</artifactId>
32         <version>${spring.version}</version>
33     </dependency>
34     <dependency>
35         <groupId>org.springframework</groupId>
36         <artifactId>spring-context</artifactId>
37         <version>${spring.version}</version>
38     </dependency>
39     <dependency>
40         <groupId>org.springframework</groupId>
41         <artifactId>spring-context-support</artifactId>
42         <version>${spring.version}</version>
43     </dependency>
44     <dependency>
45         <groupId>org.springframework</groupId>
46         <artifactId>spring-web</artifactId>
47         <version>${spring.version}</version>
48     </dependency>
49     <dependency>
50         <groupId>org.springframework</groupId>
51         <artifactId>spring-orm</artifactId>
52         <version>${spring.version}</version>
53     </dependency>
54     <dependency>
55         <groupId>org.springframework</groupId>
56         <artifactId>spring-beans</artifactId>
57         <version>${spring.version}</version>
58     </dependency>
59     <dependency>
60         <groupId>org.springframework</groupId>
61         <artifactId>spring-core</artifactId>
62         <version>${spring.version}</version>
63     </dependency>
64     <dependency>
65         <groupId>org.springframework</groupId>
66         <artifactId>spring-test</artifactId>
67         <version>${spring.version}</version>
68     </dependency>
69     <dependency>
70         <groupId>org.springframework</groupId>
71         <artifactId>spring-webmvc</artifactId>
72         <version>${spring.version}</version>
73     </dependency>
74     <dependency>
75         <groupId>org.springframework</groupId>
76         <artifactId>spring-tx</artifactId>
77         <version>${spring.version}</version>
78     </dependency>
79     <dependency>
80         <groupId>junit</groupId>
81         <artifactId>junit</artifactId>
82         <version>4.12</version>
83         <scope>test</scope>
```

```

84     </dependency>
85     <dependency>
86         <groupId>mysql</groupId>
87         <artifactId>mysql-connector-java</artifactId>
88         <version>${mysql.version}</version>
89     </dependency>
90     <dependency>
91         <groupId>javax.servlet</groupId>
92         <artifactId>javax.servlet-api</artifactId>
93         <version>3.1.0</version>
94         <scope>provided</scope>
95     </dependency>
96     <dependency>
97         <groupId>javax.servlet.jsp</groupId>
98         <artifactId>jsp-api</artifactId>
99         <version>2.0</version>
100        <scope>provided</scope>
101    </dependency>
102    <dependency>
103        <groupId>jstl</groupId>
104        <artifactId>jstl</artifactId>
105        <version>1.2</version>
106    </dependency>
107    <!-- log start -->
108    <dependency>
109        <groupId>log4j</groupId>
110        <artifactId>log4j</artifactId>
111        <version>${log4j.version}</version>
112    </dependency>
113    <dependency>
114        <groupId>org.slf4j</groupId>
115        <artifactId>slf4j-api</artifactId>
116        <version>${slf4j.version}</version>
117    </dependency>
118    <dependency>
119        <groupId>org.slf4j</groupId>
120        <artifactId>slf4j-log4j12</artifactId>
121        <version>${slf4j.version}</version>
122    </dependency>
123    <!-- log end -->
124    <dependency>
125        <groupId>org.mybatis</groupId>
126        <artifactId>mybatis</artifactId>
127        <version>${mybatis.version}</version>
128    </dependency>
129    <dependency>
130        <groupId>org.mybatis</groupId>
131        <artifactId>mybatis-spring</artifactId>
132        <version>1.3.0</version>
133    </dependency>
134    <dependency>
135        <groupId>c3p0</groupId>
136        <artifactId>c3p0</artifactId>

```

```
137         <version>0.9.1.2</version>
138         <type>jar</type>
139         <scope>compile</scope>
140     </dependency>
141     <dependency>
142         <groupId>com.github.pagehelper</groupId>
143         <artifactId>pagehelper</artifactId>
144         <version>5.1.2</version>
145     </dependency>
146
147     <dependency>
148         <groupId>org.springframework.security</groupId>
149         <artifactId>spring-security-web</artifactId>
150         <version>${spring.security.version}</version>
151     </dependency>
152     <dependency>
153         <groupId>org.springframework.security</groupId>
154         <artifactId>spring-security-config</artifactId>
155         <version>${spring.security.version}</version>
156     </dependency>
157     <dependency>
158         <groupId>org.springframework.security</groupId>
159         <artifactId>spring-security-core</artifactId>
160         <version>${spring.security.version}</version>
161     </dependency>
162     <dependency>
163         <groupId>org.springframework.security</groupId>
164         <artifactId>spring-security-taglibs</artifactId>
165         <version>${spring.security.version}</version>
166     </dependency>
167 </dependencies>
168
169 <build>
170     <finalName>ssm-zero</finalName>
171     <pluginManagement>
172         <plugins>
173             <plugin>
174                 <groupId>org.apache.maven.plugins</groupId>
175                 <artifactId>maven-compiler-plugin</artifactId>
176                 <version>3.2</version>
177                 <configuration>
178                     <source>1.8</source>
179                     <target>1.8</target>
180                     <encoding>UTF-8</encoding>
181                     <showWarnings>true</showWarnings>
182                 </configuration>
183             </plugin>
184         </plugins>
185     </pluginManagement>
186 </build>
187
188 </project>
```

9. 在ssm\_web项目中导入静态页面

10. 部署ssm\_web的项目，只要把ssm\_web项目加入到tomcat服务器中即可

## 2、配置Spring的配置文件

在ssm\_web项目中创建applicationContext.xml的配置文件，编写具体的配置信息

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:aop="http://www.springframework.org/schema/aop"
6       xmlns:tx="http://www.springframework.org/schema/tx"
7       xsi:schemaLocation="http://www.springframework.org/schema/beans
8       http://www.springframework.org/schema/beans/spring-beans.xsd
9       http://www.springframework.org/schema/context
10      http://www.springframework.org/schema/context/spring-context.xsd
11      http://www.springframework.org/schema/aop
12      http://www.springframework.org/schema/aop/spring-aop.xsd
13      http://www.springframework.org/schema/tx
14      http://www.springframework.org/schema/tx/spring-tx.xsd">
15
16     <!-- 1、开启注解扫描，要扫描的是service和dao层的注解，要忽略web层注解，因为web层让
SpringMVC框架去管理 -->
17     <context:component-scan base-package="com.itheima">
18         <!-- 配置要忽略的注解 -->
19         <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
20     </context:component-scan>
21
22     <!-- 2、引入数据库连接信息 -->
23     <context:property-placeholder location="classpath:jdbc.properties"/>
24
25     <!-- 3、配置数据源 -->
26     <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
27         <property name="driverClass" value="${jdbc.driverClass}"></property>
28         <property name="jdbcUrl" value="${jdbc.jdbcUrl}"></property>
29         <property name="user" value="${jdbc.user}"></property>
30         <property name="password" value="${jdbc.password}"></property>
31     </bean>
32
33     <!-- 4、配置sqlSessionFactory -->
34     <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
35         <!--4.1 数据源-->
36         <property name="dataSource" ref="dataSource"></property>
37         <!--4.2 mybatis的其他配置 -->
38     </bean>
39
40     <!--5、数据访问层接口扫描-->
41     <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
42         <property name="basePackage" value="com.itheima.dao"></property>
43     </bean>
44
```

```

45     <!--6、平台事务管理器-->
46     <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
47         <property name="dataSource" ref="dataSource"></property>
48     </bean>
49
50     <!--7、事务增强(通知)-->
51     <tx:advice id="txAdvice" transaction-manager="transactionManager">
52         <tx:attributes>
53             <tx:method name="find*" read-only="true"/>
54             <tx:method name="*" />
55         </tx:attributes>
56     </tx:advice>
57
58     <!--8、事务的aop织入-->
59     <aop:config>
60         <aop:advisor advice-ref="txAdvice" pointcut="execution(*
com.itheima.service.impl.*(..))"></aop:advisor>
61     </aop:config>
62
63 </beans>

```

### 3、配置数据库连接信息jdbc.properties

```

1 jdbc.driverClass=com.mysql.jdbc.Driver
2 jdbc.jdbcUrl=jdbc:mysql://localhost:3306/ssm
3 jdbc.user=root
4 jdbc.password=root

```

### 4、配置SpringMVC的配置文件

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="
6         http://www.springframework.org/schema/beans
7         http://www.springframework.org/schema/beans/spring-beans.xsd
8         http://www.springframework.org/schema/mvc
9         http://www.springframework.org/schema/mvc/spring-mvc.xsd
10        http://www.springframework.org/schema/context
11        http://www.springframework.org/schema/context/spring-context.xsd">
12
13     <!-- 1、Controller组件扫描 -->
14     <context:component-scan base-package="com.itheima.controller"/>
15
16     <!-- 2、springmvc的注解驱动-->
17     <mvc:annotation-driven/>
18
19     <!-- 3、内部资源视图解析器 -->

```

```

20     <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
21         <!--资源前缀-->
22         <property name="prefix" value="/pages/"></property>
23         <!--资源后缀-->
24         <property name="suffix" value=".jsp"></property>
25     </bean>
26
27     <!-- 4、静态资源访问-->
28     <mvc:resources mapping="/css/**" location="/css/"></mvc:resources>
29     <mvc:resources mapping="/img/**" location="/img/"></mvc:resources>
30     <mvc:resources mapping="/plugins/**" location="/plugins/"></mvc:resources>
31     <mvc:resources mapping="/js/**" location="/js/"></mvc:resources>
32
33 </beans>

```

## 5、配置web.xml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xmlns="http://java.sun.com/xml/ns/javaee"
4      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5      http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
6      <display-name>ssm_web</display-name>
7
8      <!--1、Spring的监听器-->
9      <context-param>
10         <param-name>contextConfigLocation</param-name>
11         <param-value>classpath:applicationContext.xml</param-value>
12     </context-param>
13     <listener>
14         <listener-class>org.springframework.web.context.ContextLoaderListener</listener-
15         class>
16     </listener>
17
18     <!--2、SpringMVC的前端控制器 -->
19     <servlet>
20         <servlet-name>springmvc</servlet-name>
21         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
22         <!--初始化参数指定配置文件位置-->
23         <init-param>
24             <param-name>contextConfigLocation</param-name>
25             <param-value>classpath:springmvc.xml</param-value>
26         </init-param>
27         <!--服务器启动创建servlet对象-->
28         <load-on-startup>1</load-on-startup>
29     </servlet>
30     <servlet-mapping>
31         <servlet-name>springmvc</servlet-name>
32         <url-pattern>/</url-pattern>
33     </servlet-mapping>
34
35     <!--3、全局编码过滤器-->

```

```

32     <filter>
33         <filter-name>CharacterEncodingFilter</filter-name>
34         <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
35         <init-param>
36             <param-name>encoding</param-name>
37             <param-value>UTF-8</param-value>
38         </init-param>
39     </filter>
40     <filter-mapping>
41         <filter-name>CharacterEncodingFilter</filter-name>
42         <url-pattern>/*</url-pattern>
43     </filter-mapping>
44
45     <welcome-file-list>
46         <welcome-file>index.html</welcome-file>
47         <welcome-file>index.htm</welcome-file>
48         <welcome-file>index.jsp</welcome-file>
49         <welcome-file>default.html</welcome-file>
50         <welcome-file>default.htm</welcome-file>
51         <welcome-file>default.jsp</welcome-file>
52     </welcome-file-list>
53 </web-app>

```

## 第二章：产品（旅游）模块功能实现

### 1、创建产品表和实体

#### 1.1 创建数据库和表结构

```

1  CREATE DATABASE ssm DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
2
3  CREATE TABLE product(
4      id BIGINT PRIMARY KEY AUTO_INCREMENT,
5      productNum VARCHAR(50) NOT NULL UNIQUE,
6      productName VARCHAR(50),
7      cityName VARCHAR(50),
8      departureTime VARCHAR(50),
9      productPrice NUMERIC(8,2),
10     productDesc VARCHAR(500),
11     productStatus INT
12 )

```

其中字段描述如下：



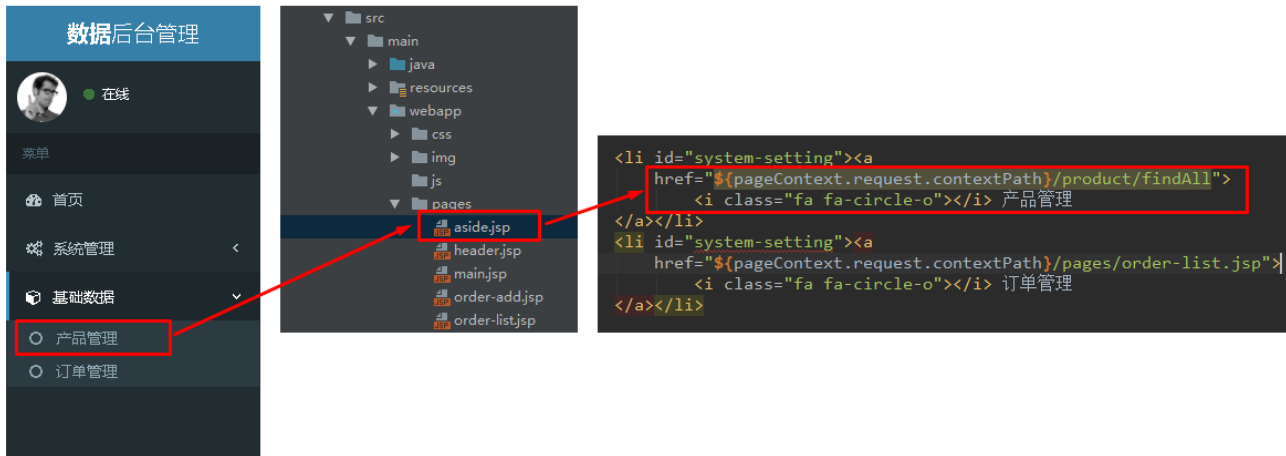
序号	字段名称	字段类型	字段描述
1	id	bigint	无意义，主键自动增长
2	productNum	varchar(50)	产品编号，唯一，不为空
3	productName	varchar(50)	产品名称（路线名称）
4	cityName	varchar(50)	出发城市
5	departureTime	varchar(50)	出发时间
6	productPrice	numeric(8,2)	产品价格
7	productDesc	varchar(500)	产品描述
8	productStatus	int	状态(0 关闭 1 开启)

## 1.2 创建Product实体类

```
1 package com.itheima.domain;
2
3 public class Product {
4
5     private Long id;
6     private String productNum;
7     private String productName;
8     private String cityName;
9     private String departureTime;
10    private Double productPrice;
11    private String productDesc;
12    private Integer productStatus;
13
14    //省略getter和setter... ...
15 }
16
```

## 2、查询所有产品功能

### 2.1 页面入口地址



## 2.2 编写Controller

```
1 package com.itheima.controller;
2
3 import com.itheima.domain.Product;
4 import com.itheima.service.ProductService;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.servlet.ModelAndView;
9
10 import java.util.List;
11
12 @Controller
13 @RequestMapping("/product")
14 public class ProductController {
15
16     @Autowired
17     private ProductService productService;
18
19     @RequestMapping("/findAll")
20     public ModelAndView findAll(){
21         //查询所有商品数据
22         List<Product> productList = productService.findAll();
23         ModelAndView modelAndView = new ModelAndView();
24         modelAndView.addObject("productList", productList);
25         modelAndView.setViewName("product-list");
26         return modelAndView;
27     }
28
29 }
30
```

## 2.3 编写Service

ProductService接口

```

1 package com.itheima.service;
2
3 import com.itheima.domain.Product;
4
5 import java.util.List;
6
7 public interface ProductService {
8     List<Product> findAll();
9 }
10

```

## ProductServiceImpl接口实现

```

1 package com.itheima.service.impl;
2
3 import com.itheima.dao.ProductMapper;
4 import com.itheima.domain.Product;
5 import com.itheima.service.ProductService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8
9 import java.util.List;
10
11 @Service("productService")
12 public class ProductServiceImpl implements ProductService{
13
14     @Autowired
15     private ProductMapper productMapper;
16
17     @Override
18     public List<Product> findAll() {
19         List<Product> productList = productMapper.findAll();
20         return productList;
21     }
22
23 }
24

```

## 2.4 编写Mapper

```

1 package com.itheima.dao;
2
3 import com.itheima.domain.Product;
4 import org.apache.ibatis.annotations.Mapper;
5 import org.apache.ibatis.annotations.Select;
6
7 import java.util.List;
8
9 @Mapper
10 public interface ProductMapper {
11

```

```

12     @Select("select * from product")
13     List<Product> findAll();
14
15 }
16

```

## 2.5 编写页面product-list.jsp

```

1  <c:forEach items="${productList}" var="product">
2
3      <tr>
4          <td><input name="ids" type="checkbox"></td>
5          <td>${product.id}</td>
6
7          <td>${product.productNum}</td>
8          <td>${product.productName}</td>
9          <td>${product.departureTime}</td>
10         <td>${product.productStatus==1?"开启":"关闭"}</td>
11
12         <td class="text-center">
13             <button type="button" class="btn bg-olive btn-xs"
14                 onclick='location.href="all-order-manage-edit.html"'>订单
15         </button>
16         <button type="button" class="btn bg-olive btn-xs"
17             onclick='location.href="all-order-manage-edit.html"'>查看
18         </button>
19     </td>
20 </tr>
21 </c:forEach>

```

## 3、添加产品功能

### 3.1 页面入口

产品管理

产品表单

产品信息

产品编号	<input type="text" value="xj1000abc"/>	
出发时间	<input type="text" value="2018-06-17 10:30"/>	<input type="text" value="2018-06-17 10:30"/>
产品价格	<input type="text" value="2345"/>	
其他信息	<input type="text" value="体验新疆风情"/>	

保存

返回

```

<form action="${pageContext.request.contextPath}/product/save.do"
method="post">
<!-- 正文区域 -->
<section class="content"> <!--产品信息-->

<div class="panel panel-default">
<div class="panel-heading">产品信息</div>
<div class="row data-type">

```

### 3.2 编写Controller

```

1 @RequestMapping("/save")
2 public String save(Product product){
3     productService.save(product);
4     return "redirect:/product/findAll";
5 }

```

### 3.3 编写Service

ProductService接口

```

1 void save(Product product);

```

ProductServiceImpl实现

```

1 @Override
2 public void save(Product product) {
3     productMapper.save(product);
4 }

```

### 3.4 编写Mapper

```

1 @Insert("insert into product
  (productNum,productName,cityName,departureTime,productPrice,productDesc,productStatus)
  values (#{productNum},#{productName},#{cityName},#{departureTime},#{productPrice},#{
    productDesc},#{productStatus})")
2 void save(Product product);

```

## 4、修改产品功能-数据回显

### 4.1 页面入口

⇕	状态	⇕	操作
	开启		<div>订单</div> <div>查看</div>
	关闭		<div>订单</div> <div>查看</div>

```

<td class="text-center">
  <button type="button" class="btn bg-olive btn-xs"
    onclick="location.href='all-order-manage-edit.html'">订单</button>
  <button type="button" class="btn bg-olive btn-xs"
    onclick="location.href='${pageContext.request.contextPath}/product/editUI.do?id=${product.id}'">
  </td>

```



### 4.2 编写Controller

```

1  @RequestMapping("/editUI")
2  public ModelAndView editUI(Long id){
3      Product product = productService.findById(id);
4
5      ModelAndView modelAndView = new ModelAndView();
6      modelAndView.addObject("product",product);
7      modelAndView.setViewName("product-update");
8
9      return modelAndView;
10 }

```

### 4.3 编写Service

ProductService接口

```

1  Product findById(Long id);

```

ProductServiceImpl实现

```

1  @Override
2  public Product findById(Long id) {
3      return productMapper.findById(id);
4  }

```

### 4.4 编写Mapper

```

1  @Select("select * from product where id=#{id}")
2  Product findById(Long id);

```

### 3.5 编写页面

```

1  <div class="panel panel-default">
2      <div class="panel-heading">产品信息</div>
3      <div class="row data-type">
4
5          <div class="col-md-2 title">产品编号</div>
6          <div class="col-md-4 data">
7              <input type="text" class="form-control" name="productNum"
8                  placeholder="产品编号" value="${product.productNum}"
9                  readonly="readonly">
10         </div>
11         <div class="col-md-2 title">产品名称</div>
12         <div class="col-md-4 data">
13             <input type="text" class="form-control" name="productName"
14                 placeholder="产品名称" value="${product.productName}">
15         </div>
16         <div class="col-md-2 title">出发时间</div>
17         <div class="col-md-4 data">
18             <div class="input-group date">

```

```

19         <div class="input-group-addon">
20             <i class="fa fa-calendar"></i>
21         </div>
22         <input type="text" class="form-control pull-right"
23             id="datepicker-a3" name="departureTime"
24             value="${product.departureTime}">
25     </div>
26 </div>
27
28
29 <div class="col-md-2 title">出发城市</div>
30 <div class="col-md-4 data">
31     <input type="text" class="form-control" name="cityName"
32         placeholder="出发城市" value="${product.cityName}">
33 </div>
34
35 <div class="col-md-2 title">产品价格</div>
36 <div class="col-md-4 data">
37     <input type="text" class="form-control" placeholder="产品价格"
38         name="productPrice" value="${product.productPrice}">
39 </div>
40
41 <div class="col-md-2 title">产品状态</div>
42 <div class="col-md-4 data">
43     <select id="status" class="form-control select2" style="width: 100%"
44         name="productStatus">
45         <option value="0" selected="selected">关闭</option>
46         <option value="1">开启</option>
47     </select>
48 </div>
49
50 <div class="col-md-2 title rowHeight2x">其他信息</div>
51 <div class="col-md-10 data rowHeight2x">
52     <textarea class="form-control" rows="3" placeholder="其他信息"
53         name="productDesc">${product.productDesc}</textarea>
54 </div>
55
56 </div>
57 </div>
58
59 <script src="${pageContext.request.contextPath}/plugins/jquery/jquery-2.2.3.min.js">
60 </script>
61
62 <script>
63     $("#status option[value='${product.productStatus}']").prop("selected",true);
64 </script>

```

## 5、修改产品功能-数据修改


### 5.1 页面入口

产品信息	
产品编号	123
出发时间	<input type="text" value="2018-06-07 10:35"/>
产品价格	12345.0
其他信息	123123

```
<form action="${pageContext.request.contextPath}/product/update.do"
method="post">
<input type="hidden" name="id" value="${product.id}">
<!-- 正文区域 -->
<section class="content"> <!--产品信息-->
```

修改

返回



## 5.2 编写Controller

```
1 @RequestMapping("/update")
2 public String update(Product product){
3     productService.update(product);
4     return "redirect:/product/findAll";
5 }
```

## 5.3 编写Service

ProductService接口

```
1 void update(Product product);
```

ProductServiceImpl实现

```
1 @Override
2 public void update(Product product) {
3     productMapper.update(product);
4 }
```

## 5.4 编写Mapper

```
1 @Update("update product set productNum = #{productNum},productName=#{
productName},cityName=#{cityName},departureTime=#{departureTime},productPrice=#{
productPrice},productDesc=#{productDesc},productStatus=#{productStatus} where id = #
{id}")
2 void update(Product product);
```

# 6、删除产品功能

## 3.1 页面代码实现



新建

删除

开启

屏蔽

刷新

搜索

<div><div></div></div>	ID	订单号	路线名称	出发日期	状态	操作
<div><div></div></div>	1	abcde	海南8日双飞	2019-01-01 15:30	开启	<div>订单</div> <div>查看</div>
<div><div></div></div>	2	123	测试数据	2018-06-07 10:35	关闭	<div>订单</div> <div>查看</div>
<div><div></div></div>	6	444			关闭	<div>订单</div> <div>查看</div>
<div><div></div></div>	9	777			关闭	<div>订单</div> <div>查看</div>

新建

删除

开启

屏蔽

刷新

搜索

```

1  <form id="delForm" action="${pageContext.request.contextPath}/product/delSelected.do"
2  method="post">
3      //此处省略部分代码
4      <tr>
5          <td><input name="ids" type="checkbox" value="${product.id}"></td>
6          <td>${product.id}</td>
7          <td>${product.productNum}</td>
8          <td>${product.productName}</td>
9          <td>${product.departureTime}</td>
10         <td>${product.productStatus==1?"开启":"关闭"}</td>
11
12         <td class="text-center">
13             <button type="button" class="btn bg-olive btn-xs"
14                 onclick='location.href="all-order-manage-edit.html"'>订单
15             <button type="button" class="btn bg-olive btn-xs"
16                 onclick='location.href="${pageContext.request.contextPath}/product/editUI.do?
17                 id=${product.id}">查看</button>
18         </td>
19     </tr>
20     //此处省略部分代码
21
22     <script
23         src="${pageContext.request.contextPath}/plugins/jquery/jquery-2.2.3.min.js">
24     </script>
25     <script>
26         function delSelectedProducts(){
27             if(confirm("您确认要删除选中吗?")){
28                 $("#delForm").submit();
29             }
30         }
31     </script>

```

### 3.2 编写Controller

```
1 @RequestMapping("/delSelected")
2 public String delSelected(Long[] ids){
3     productService.delSelected(ids);
4     return "redirect:/product/findAll";
5 }
```

### 3.3 编写Service

ProductService接口

```
1 void delSelected(Long[] ids);
```

ProductServiceImpl实现

```
1 @Override
2 public void delSelected(Long[] ids) {
3     if(ids!=null&&ids.length>0){
4         for (Long id : ids) {
5             productMapper.delete(id);
6         }
7     }
8 }
```

### 3.4 编写Mapper

```
1 @Delete("delete from product where id=#{id}")
2 void delete(Long id);
```

## 第三章：订单模块实现

### 1、创建订单表和实体

#### 1.1 创建数据库和表结构

```
1 CREATE TABLE orders(
2     id BIGINT PRIMARY KEY AUTO_INCREMENT,
3     orderNum VARCHAR(20) NOT NULL UNIQUE,
4     orderTime VARCHAR(50),
5     peopleCount INT,
6     orderDesc VARCHAR(500),
7     payType INT,
8     orderStatus INT,
9     productId BIGINT,
10     FOREIGN KEY (productId) REFERENCES product(id)
11 )
```

其中字段描述如下：

序号	字段名称	字段类型	字段描述
1	id	bigint	无意义、主键自动增长
2	orderNum	varchar(50)	订单编号 不为空 唯一
3	orderTime	varchar(50)	下单时间
4	peopleCount	int	出行人数
5	orderDesc	varchar(500)	订单描述(其它信息)
6	payType	int	支付方式(0 支付宝 1 微信 2其它)
7	orderStatus	int	订单状态(0 未支付 1 已支付)
8	productId	int	产品id 外键

## 1.2 创建Product实体类

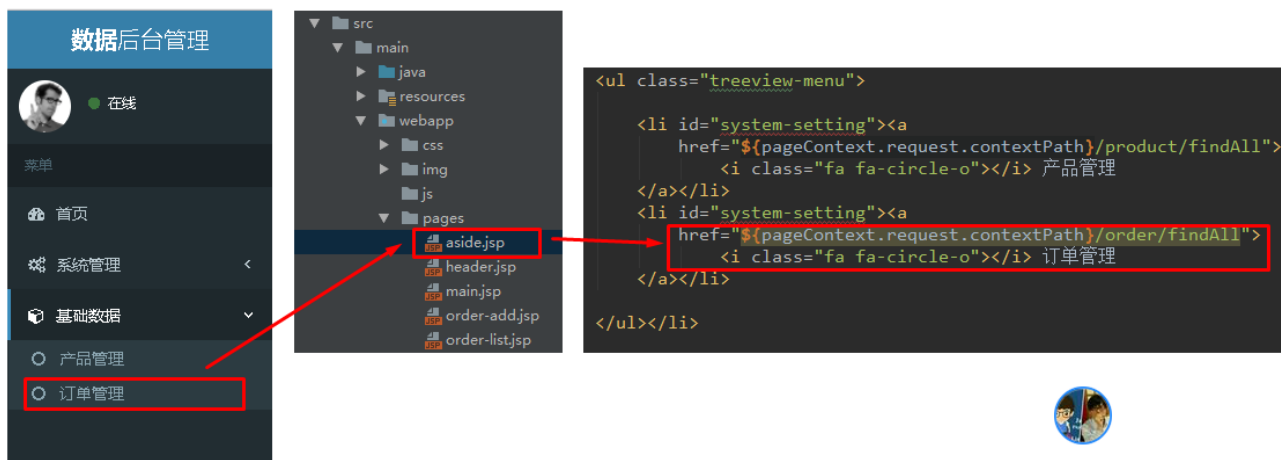
```

1 package com.itheima.domain;
2
3 public class Order {
4
5     private Long id;
6     private String orderNum;
7     private String orderTime;
8     private Integer peopleCount;
9     private String orderDesc;
10    private Integer payType;
11    private Integer orderStatus;
12
13    private Product product;//该订单属于哪一个产品
14
15    //省略getter和setter... ...
16 }
17

```

## 2、查询所有订单功能

### 2.1 页面入口地址



## 2.2 编写Controller

```
1 package com.itheima.controller;
2
3 import com.itheima.domain.Order;
4 import com.itheima.service.OrderService;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.servlet.ModelAndView;
9
10 import java.util.List;
11
12 @Controller
13 @RequestMapping("/order")
14 public class OrderController {
15
16     @Autowired
17     private OrderService orderService;
18
19     @RequestMapping("/findAll")
20     public ModelAndView findAll(){
21         List<Order> orderList = orderService.findAll();
22         ModelAndView modelAndView = new ModelAndView();
23         modelAndView.addObject("orderList",orderList);
24         modelAndView.setViewName("order-list");
25         return modelAndView;
26     }
27
28 }
29
```

## 2.3 编写Service

OrderService接口

```

1 package com.itheima.service;
2
3 import com.itheima.domain.Order;
4
5 import java.util.List;
6
7 public interface OrderService {
8     List<Order> findAll();
9 }
10

```

## OrderServiceImpl接口实现

```

1 package com.itheima.service.impl;
2
3 import com.itheima.dao.OrderMapper;
4 import com.itheima.domain.Order;
5 import com.itheima.service.OrderService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8
9 import java.util.List;
10
11 @Service("orderService")
12 public class OrderServiceImpl implements OrderService {
13
14     @Autowired
15     private OrderMapper orderMapper;
16
17     @Override
18     public List<Order> findAll() {
19         return orderMapper.findAll();
20     }
21 }
22

```

## 2.4 编写Mapper

```

1 package com.itheima.dao;
2
3 import com.itheima.domain.Order;
4 import org.apache.ibatis.annotations.*;
5 import org.apache.ibatis.mapping.FetchType;
6
7 import java.util.List;
8
9 @Mapper
10 public interface OrderMapper {
11
12     @Select("select * from orders")
13     @Results({
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

14         @Result(id = true,property = "id",column = "id"),
15         @Result(
16             property = "product",
17             column = "productId",
18             one = @One(select =
"com.itheima.dao.ProductMapper.findById",fetchType = FetchType.EAGER)
19         )
20     })
21     List<Order> findAll();
22 }
23

```

## 2.5 编写页面order-list.jsp

```

1  <c:forEach items="${orderList}" var="order">
2
3      <tr>
4          <td><input name="ids" type="checkbox"></td>
5          <td>${order.id}</td>
6          <td>${order.orderNum}</td>
7          <td>${order.product.productName}</td>
8          <td>${order.orderTime}</td>
9          <td>${order.peopleCount}</td>
10         <td>${order.orderStatus==1?"已支付":"未支付"}</td>
11
12         <td class="text-center">
13             <button type="button" class="btn bg-olive btn-xs"
14
15                 onclick='location.href="${pageContext.request.contextPath}/pages/order-show.jsp"'>订
单</button>
16
17                 <button type="button" class="btn bg-olive btn-xs"
18
19                     onclick='location.href="${pageContext.request.contextPath}/pages/order-show.jsp"'>查
看</button>
20             </td>
21         </tr>
22     </c:forEach>

```

## 3、添加订单功能-添加订单页面数据准备

### 3.1 页面入口

新增订单页面需要准备产品的数据列表

订单管理全部订单

首页 > 订单管理 > 订单编辑

订单信息

订单编号	<input type="text" value="订单编号"/>	下单时间	<input type="text" value="日期"/>
出行人数	<input type="text" value="出行人数"/>	支付方式	支付宝
订单状态	未支付	选择产品	<div>选择产品</div>
订单描述	<div></div>		

保存

返回

订单管理全部订单

列表

当点击新建按钮时 请求Controller准备产品数据列表

新建

删除

开启

屏蔽

刷新

	ID	订单号	路线名称	下单时间	出行人数	支付状态
<input type="checkbox"/>	2	abc100	测试数据	2018-12-12 12:12:12	20	已支付

新建

删除

开启

屏蔽

刷新

总共2 页, 共14 条数据。 每页 10 条

首页

上一页

1

2

3

```
<!--工具栏-->
<div class="pull-left">
  <div class="form-group form-inline">
    <div class="btn-group">
      <button type="button" class="btn btn-default" title="新建"
        onclick='location.href='${pageContext.request.contextPath}/order/addOrderUI.do'>
        <i class="fa fa-file-o"></i> 新建
      </button>
      <button type="button" class="btn btn-default" title="删除"
        onclick='confirm("你确认要删除吗?")'>
        <i class="fa fa-trash-o"></i> 删除
      </button>
```

## 3.2 编写Controller

```

1  @Autowired
2  private ProductService productService;
3
4  @RequestMapping("/addOrderUI")
5  public ModelAndView addOrderUI(){
6      List<Product> productList = productService.findAll();
7      ModelAndView modelAndView = new ModelAndView();
8      modelAndView.addObject("productList", productList);
9      modelAndView.setViewName("order-add");
10     return modelAndView;
11 }

```

### 3.3 编写页面order-add.jsp

```

1  <div class="col-md-4 data">
2      <select class="form-control select2" style="width: 100%"
3          name="product.id">
4
5          <c:forEach items="${ productList }" var="p">
6              <option value="${ p.id }" >${ p.productName }</option>
7          </c:forEach>
8
9      </select>
10 </div>

```

## 3、添加订单功能-保存订单数据

### 3.1 页面入口地址

订单信息	
订单编号	haohao123
出行人数	100
订单状态	已支付
订单描述	订单已经完成

```

<form action="${pageContext.request.contextPath}/order/save" method="post">
<div class="panel panel-default">
<div class="panel-heading">订单信息</div>
<div class="row data-type">
<div class="col-md-2 title">订单编号</div>
<div class="col-md-4 data">

```

保存
返回

### 2.2 编写Controller

```

1  @RequestMapping("/save")
2  public String save(Order order){
3      orderService.save(order);
4      return "redirect:/order/findAll";
5  }

```



## 2.3 编写Service

OrderService接口

```
1 void save(Order order);
```

OrderServiceImpl接口实现

```
1 @Override
2 public void save(Order order) {
3     orderMapper.save(order);
4 }
```

## 2.4 编写Mapper

```
1 @Insert("insert into orders
  (orderNum,orderTime,peopleCount,orderDesc,payType,orderStatus,productId) values (#
  {orderNum},{orderTime},{peopleCount},{orderDesc},{payType},{orderStatus},{
  {product.id}})")
2 void save(Order order);
```