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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.itheima</groupId>
    <artifactId>ssm_parent</artifactId>
    <packaging>pom</packaging>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <spring.version>5.0.2.RELEASE</spring.version>
        <slf4j.version>1.6.6</slf4j.version>
        <log4j.version>1.2.12</log4j.version>
        <shiro.version>1.2.3</shiro.version>
        <mysql.version>5.1.6</mysql.version>
        <mybatis.version>3.4.5</mybatis.version>
        <spring.security.version>5.0.1.RELEASE</spring.security.version>
    </properties>

    <dependencies>
        <!-- spring -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.6.8</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-support</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-orm</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>${mysql.version}</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>jstl</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>
        <!-- log start -->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>${log4j.version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
        <!-- log end -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>${mybatis.version}</version>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>1.3.0</version>
        </dependency>
        <dependency>
            <groupId>c3p0</groupId>
            <artifactId>c3p0</artifactId>
            <version>0.9.1.2</version>
            <type>jar</type>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper</artifactId>
            <version>5.1.2</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-web</artifactId>
            <version>${spring.security.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-config</artifactId>
            <version>${spring.security.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-core</artifactId>
            <version>${spring.security.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-taglibs</artifactId>
            <version>${spring.security.version}</version>
        </dependency>
    </dependencies>

    <build>
        <finalName>ssm-zero</finalName>
        <pluginManagement>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.2</version>
                    <configuration>
                        <source>1.8</source>
                        <target>1.8</target>
                        <encoding>UTF-8</encoding>
                        <showWarnings>true</showWarnings>
                    </configuration>
                </plugin>
            </plugins>
        </pluginManagement>
    </build>

</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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd">

    <!-- 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 -->
    <context:component-scan base-package="com.itheima">
        <!-- 配置要忽略的注解 -->
        <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

    <!-- 2、引入数据库连接信息 -->
    <context:property-placeholder location="classpath:jdbc.properties"/>

    <!-- 3、配置数据源 -->
    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="driverClass" value="${jdbc.driverClass}"></property>
        <property name="jdbcUrl" value="${jdbc.jdbcUrl}"></property>
        <property name="user" value="${jdbc.user}"></property>
        <property name="password" value="${jdbc.password}"></property>
    </bean>

    <!-- 4、配置sqlSessionFactory -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <!--4.1 数据源-->
        <property name="dataSource" ref="dataSource"></property>
        <!--4.2 mybatis的其他配置 -->
    </bean>

    <!--5、数据访问层接口扫描-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.itheima.dao"></property>
    </bean>

    <!--6、平台事务管理器-->
    <bean id="transactionManager" 
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource"></property>
    </bean>

    <!--7、事务增强(通知)-->
    <tx:advice id="txAdvice" transaction-manager="transactionManager">
        <tx:attributes>
            <tx:method name="find*" read-only="true"/>
            <tx:method name="*"/>
        </tx:attributes>
    </tx:advice>

    <!--8、事务的aop织入-->
    <aop:config>
        <aop:advisor advice-ref="txAdvice" pointcut="execution(* 
com.itheima.service.impl.*.*(..))"></aop:advisor>
    </aop:config>

</beans>
```

3、配置数据库连接信息jdbc.properties

```properties
jdbc.driverClass=com.mysql.jdbc.Driver
jdbc.jdbcUrl=jdbc:mysql://localhost:3306/ssm
jdbc.user=root
jdbc.password=root
```

4、配置SpringMVC的配置文件

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-mvc.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- 1、Controller组件扫描 -->
    <context:component-scan base-package="com.itheima.controller"/>

    <!-- 2、springmvc的注解驱动-->
    <mvc:annotation-driven/>

    <!-- 3、内部资源视图解析器 -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <!--资源前缀-->
        <property name="prefix" value="/pages/"></property>
        <!--资源后缀-->
        <property name="suffix" value=".jsp"></property>
    </bean>

    <!-- 4、静态资源访问-->
    <mvc:resources mapping="/css/**" location="/css/"/>
    <mvc:resources mapping="/img/**" location="/img/"/>
    <mvc:resources mapping="/plugins/**" location="/plugins/"/>
    <mvc:resources mapping="/js/**" location="/js/"/>

</beans>
```

5、配置web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
  <display-name>ssm_web</display-name>

  <!--1、Spring的监听器-->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
  </context-param>
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-
class>
  </listener>

  <!--2、SpringMVC的前端控制器 -->
  <servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <!--初始化参数指定配置文件位置-->
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:springmvc.xml</param-value>
    </init-param>
    <!--服务器启动创建servlet对象-->
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

  <!--3、全局编码过滤器-->
```

```xml
  <filter>
    <filter-name>CharacterEncodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
      <param-name>encoding</param-name>
      <param-value>UTF-8</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</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

```sql
CREATE DATABASE ssm DEFAULT CHARSET utf8 COLLATE utf8_general_ci;

CREATE TABLE product(
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    productNum VARCHAR(50) NOT NULL UNIQUE,
    productName VARCHAR(50),
    cityName VARCHAR(50),
    departureTime VARCHAR(50),
    productPrice NUMERIC(8,2),
    productDesc VARCHAR(500),
    productStatus INT
)
```

其中字段描述如下：

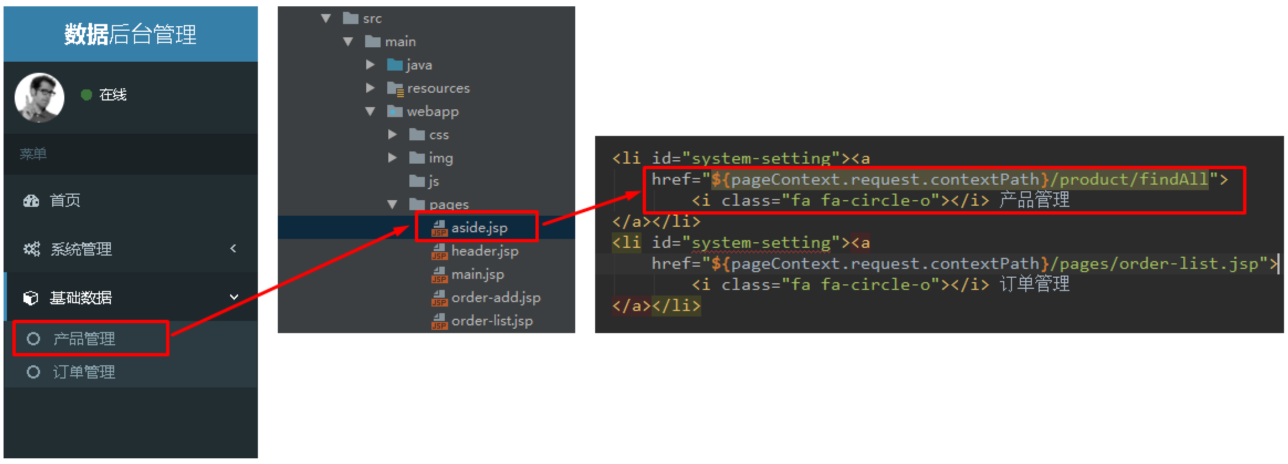
序号	字段名称	字段类型	字段描述
1	id	bigint	无意义，主键自动增长
2	productNum	varchar(50)	产品编号，唯一，不为空
3	productName	varchar(50)	产品名称（路线名称）
4	cityName	varchar(50)	出发城市
5	departureTime	varchar(50)	出发时间
6	productPrice	numeric(8,2)	产品价格
7	productDesc	varchar(500)	产品描述
8	productStatus	int	状态(0 关闭 1 开启)

1.2 创建Product实体类

```java
package com.itheima.domain;

public class Product {

    private Long id;
    private String productNum;
    private String productName;
    private String cityName;
    private String departureTime;
    private Double productPrice;
    private String productDesc;
    private Integer productStatus;

    //省略getter和setter... ...
}

```

2、查询所有产品功能

2.1 页面入口地址



```
<li id="system-setting"><a
    href="${pageContext.request.contextPath}/product/findAll">
        <i class="fa fa-circle-o"></i> 产品管理
</a></li>
<li id="system-setting"><a
    href="${pageContext.request.contextPath}/pages/order-list.jsp">
        <i class="fa fa-circle-o"></i> 订单管理
</a></li>
```

2.2 编写Controller

```java
package com.itheima.controller;

import com.itheima.domain.Product;
import com.itheima.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import java.util.List;

@Controller
@RequestMapping("/product")
public class ProductController {

    @Autowired
    private ProductService productService;

    @RequestMapping("/findAll")
    public ModelAndView findAll(){
        //查询所有商品数据
        List<Product> productList = productService.findAll();
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("productList",productList);
        modelAndView.setViewName("product-list");
        return modelAndView;
    }

}

```

2.3 编写Service

ProductService接口

```java
package com.itheima.service;

import com.itheima.domain.Product;

import java.util.List;

public interface ProductService {
    List<Product> findAll();
}

```

ProductServiceImpl接口实现

```java
package com.itheima.service.impl;

import com.itheima.dao.ProductMapper;
import com.itheima.domain.Product;
import com.itheima.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service("productService")
public class ProductServiceImpl implements ProductService{

    @Autowired
    private ProductMapper productMapper;

    @Override
    public List<Product> findAll() {
        List<Product> productList = productMapper.findAll();
        return productList;
    }

}

```

2.4 编写Mapper

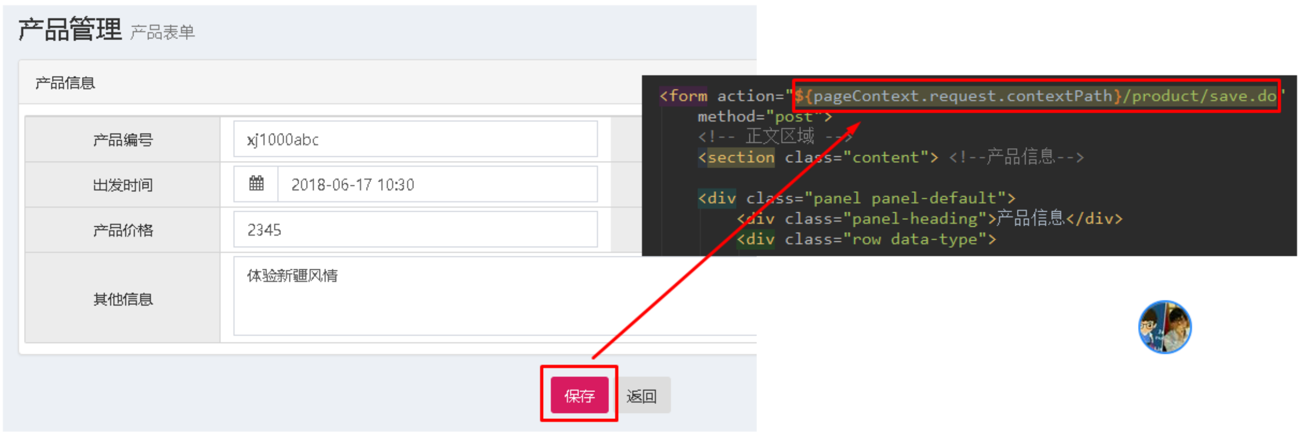
```java
package com.itheima.dao;

import com.itheima.domain.Product;
import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Select;

import java.util.List;

@Mapper
public interface ProductMapper {

    @Select("select * from product")
    List<Product> findAll();

}

```

2.5 编写页面product-list.jsp

```
<c:forEach items="${productList}" var="product">

    <tr>
        <td><input name="ids" type="checkbox"></td>
        <td>${product.id}</td>

        <td>${product.productNum}</td>
        <td>${product.productName}</td>
        <td>${product.departureTime}</td>
        <td>${product.productStatus==1?"开启":"关闭"}</td>

        <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="all-order-manage-edit.html"'>查看
</button>
        </td>
    </tr>

</c:forEach>
```

3、添加产品功能

3.1 页面入口



产品编号	xj1000abc
出发时间	2018-06-17 10:30
产品价格	2345
其他信息	体验新疆风情

```
<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

```java
@RequestMapping("/save")
public String save(Product product){
    productService.save(product);
    return "redirect:/product/findAll";
}
```

3.3 编写Service

ProductService接口

```java
void save(Product product);
```

ProductServiceImpl实现

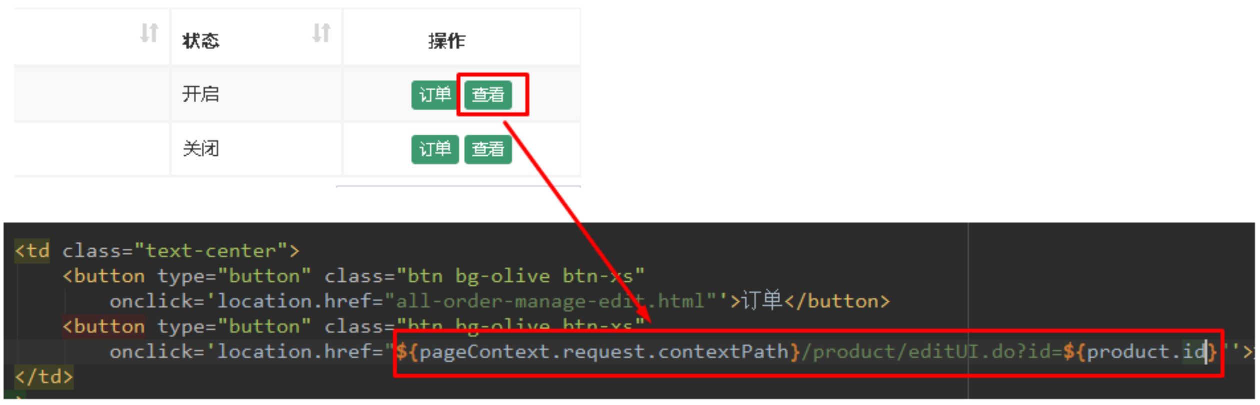
```java
@Override
public void save(Product product) {
    productMapper.save(product);
}
```

3.4 编写Mapper

```java
@Insert("insert into product (productNum,productName,cityName,departureTime,productPrice,productDesc,productStatus) values (#{productNum},#{productName},#{cityName},#{departureTime},#{productPrice},#{productDesc},#{productStatus})")
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	订单 查看
	关闭	订单 查看

```html
<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

```java
@RequestMapping("/editUI")
public ModelAndView editUI(Long id){
    Product product = productService.findById(id);

    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("product",product);
    modelAndView.setViewName("product-update");

    return modelAndView;
}
```

4.3 编写Service

ProductService接口

```java
Product findById(Long id);
```

ProductServiceImpl实现

```java
@Override
public Product findById(Long id) {
    return productMapper.findById(id);
}
```

4.4 编写Mapper

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

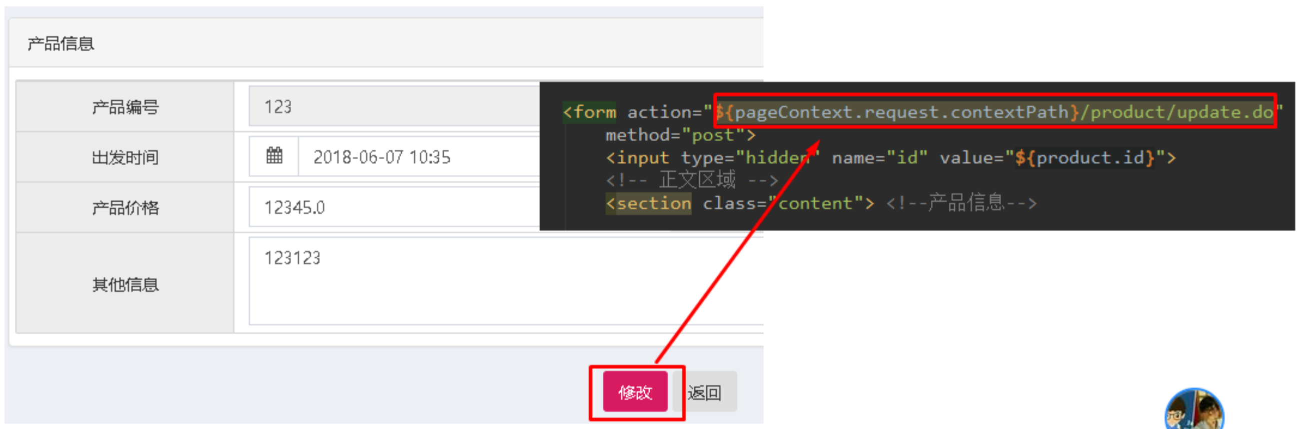
3.5 编写页面

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

        <div class="col-md-2 title">产品编号</div>
        <div class="col-md-4 data">
            <input type="text" class="form-control" name="productNum"
                   placeholder="产品编号" value="${product.productNum}"
                   readonly="readonly">
        </div>
        <div class="col-md-2 title">产品名称</div>
        <div class="col-md-4 data">
            <input type="text" class="form-control" name="productName"
                   placeholder="产品名称" value="${product.productName}">
        </div>
        <div class="col-md-2 title">出发时间</div>
        <div class="col-md-4 data">
            <div class="input-group date">
```

```
                <div class="input-group-addon">
                    <i class="fa fa-calendar"></i>
                </div>
                <input type="text" class="form-control pull-right"
                       id="datepicker-a3" name="departureTime"
                       value="${product.departureTime}">
            </div>
        </div>


        <div class="col-md-2 title">出发城市</div>
        <div class="col-md-4 data">
            <input type="text" class="form-control" name="cityName"
                   placeholder="出发城市" value="${product.cityName}">
        </div>

        <div class="col-md-2 title">产品价格</div>
        <div class="col-md-4 data">
            <input type="text" class="form-control" placeholder="产品价格"
                   name="productPrice" value="${product.productPrice}">
        </div>

        <div class="col-md-2 title">产品状态</div>
        <div class="col-md-4 data">
            <select id="status" class="form-control select2" style="width: 100%"
                    name="productStatus">
                <option value="0" selected="selected">关闭</option>
                <option value="1">开启</option>
            </select>
        </div>

        <div class="col-md-2 title rowHeight2x">其他信息</div>
        <div class="col-md-10 data rowHeight2x">
            <textarea class="form-control" rows="3" placeholder="其他信息"
                      name="productDesc">${product.productDesc}</textarea>
        </div>

    </div>
</div>

<script src="${pageContext.request.contextPath}/plugins/jQuery/jquery-2.2.3.min.js"></script>

<script>
    $("#status option[value='${product.productStatus}']").prop("selected",true);
</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	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-->
```

5.2 编写Controller

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

5.3 编写Service

ProductService接口

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

ProductServiceImpl实现

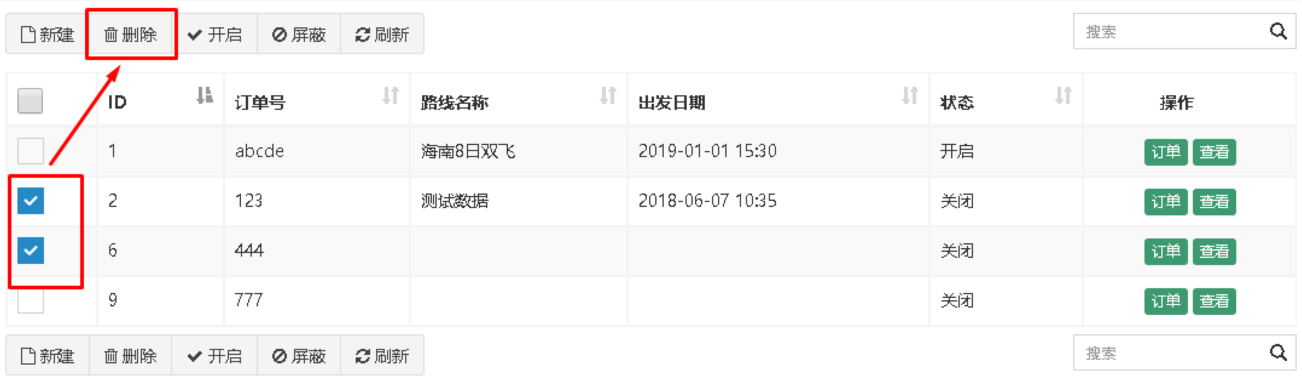
```
@Override
public void update(Product product) {
    productMapper.update(product);
}
```

5.4 编写Mapper

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

6、删除产品功能

3.1 页面代码实现



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

```
<form id="delForm" action="${pageContext.request.contextPath}/product/delSelected.do" method="post">
    //此处省略部分代码
    <tr>
        <td><input name="ids" type="checkbox" value="${product.id}"></td>
        <td>${product.id}</td>

        <td>${product.productNum}</td>
        <td>${product.productName}</td>
        <td>${product.departureTime}</td>
        <td>${product.productStatus==1?"开启":"关闭"}</td>

        <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}"'>查看</button>
        </td>
    </tr>

    //此处省略部分代码

<script
        src="${pageContext.request.contextPath}/plugins/jQuery/jquery-2.2.3.min.js"></script>

<script>
    function delSelectedProducts(){
        if(confirm("您确认要删除选中吗?")){
            $("#delForm").submit();
        }
    }
</script>
```

3.2 编写Controller

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

3.3 编写Service

ProductService接口

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

ProductServiceImpl实现

```
@Override
public void delSelected(Long[] ids) {
    if(ids!=null&&ids.length>0){
        for (Long id : ids) {
            productMapper.delete(id);
        }
    }
}
```

3.4 编写Mapper

```
@Delete("delete from product where id=#{id}")
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

```
CREATE TABLE orders(
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    orderNum VARCHAR(20) NOT NULL UNIQUE,
    orderTime VARCHAR(50),
    peopleCount INT,
    orderDesc VARCHAR(500),
    payType INT,
    orderStatus INT,
    productId BIGINT,
    FOREIGN KEY (productId) REFERENCES product(id)
)
```

其中字段描述如下：

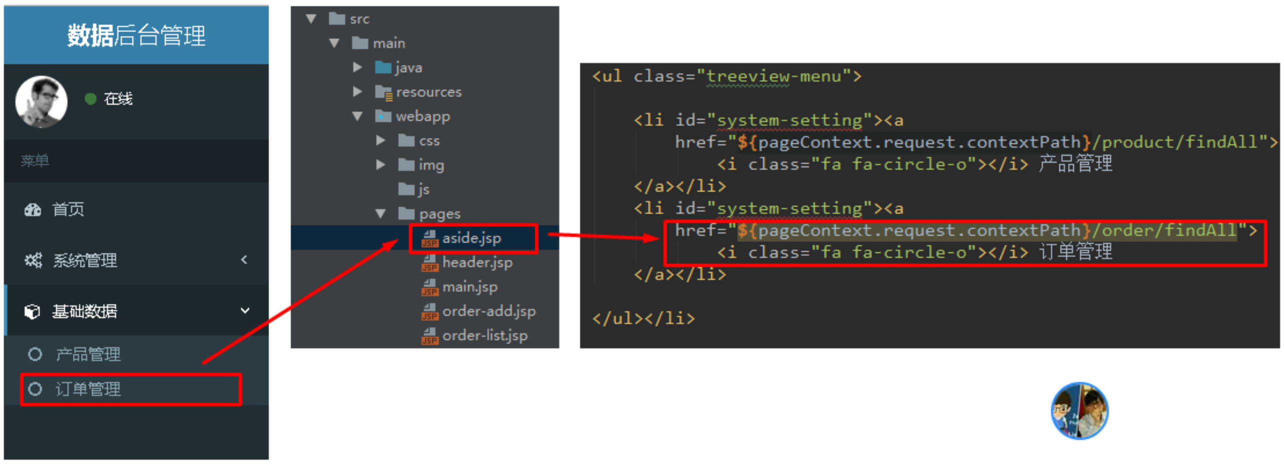
序号	字段名称	字段类型	字段描述
1	id	bigint	无意义、主键自动增长
2	orderNum	varchar(50)	订单编号 不为空 唯一
3	orderTime	varchar(50)	下单时间
4	peopleCount	int	出行人数
5	orderDesc	varchar(500)	订单描述(其它信息)
6	payType	int	支付方式(0 支付宝 1 微信 2其它)
7	orderStatus	int	订单状态(0 未支付 1 已支付)
8	productId	int	产品id 外键

1.2 创建Product实体类

```java
package com.itheima.domain;

public class Order {

    private Long id;
    private String orderNum;
    private String orderTime;
    private Integer peopleCount;
    private String orderDesc;
    private Integer payType;
    private Integer orderStatus;

    private Product product;//该订单属于哪一个产品

    //省略getter和setter... ...
}

```

2、查询所有订单功能

2.1 页面入口地址



```
<ul class="treeview-menu">

    <li id="system-setting"><a
        href="${pageContext.request.contextPath}/product/findAll">
            <i class="fa fa-circle-o"></i> 产品管理
    </a></li>
    <li id="system-setting"><a
        href="${pageContext.request.contextPath}/order/findAll">
            <i class="fa fa-circle-o"></i> 订单管理
    </a></li>

</ul></li>
```

2.2 编写Controller

```java
package com.itheima.controller;

import com.itheima.domain.Order;
import com.itheima.service.OrderService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import java.util.List;

@Controller
@RequestMapping("/order")
public class OrderController {

    @Autowired
    private OrderService orderService;

    @RequestMapping("/findAll")
    public ModelAndView findAll(){
        List<Order> orderList = orderService.findAll();
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("orderList",orderList);
        modelAndView.setViewName("order-list");
        return  modelAndView;
    }

}

```

2.3 编写Service

OrderService接口

```java
package com.itheima.service;

import com.itheima.domain.Order;

import java.util.List;

public interface OrderService {
    List<Order> findAll();
}

```

OrderServiceImpl接口实现

```java
package com.itheima.service.impl;

import com.itheima.dao.OrderMapper;
import com.itheima.domain.Order;
import com.itheima.service.OrderService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service("orderService")
public class OrderServiceImpl implements OrderService {

    @Autowired
    private OrderMapper orderMapper;

    @Override
    public List<Order> findAll() {
        return orderMapper.findAll();
    }
}

```

2.4 编写Mapper

```java
package com.itheima.dao;

import com.itheima.domain.Order;
import org.apache.ibatis.annotations.*;
import org.apache.ibatis.mapping.FetchType;

import java.util.List;

@Mapper
public interface OrderMapper {

    @Select("select * from orders")
    @Results({
            @Result(id = true,property = "id",column = "id"),
            @Result(
                    property = "product",
                    column = "productId",
                    one = @One(select = "com.itheima.dao.ProductMapper.findById",fetchType = FetchType.EAGER)
            )
    })
    List<Order> findAll();
}

```

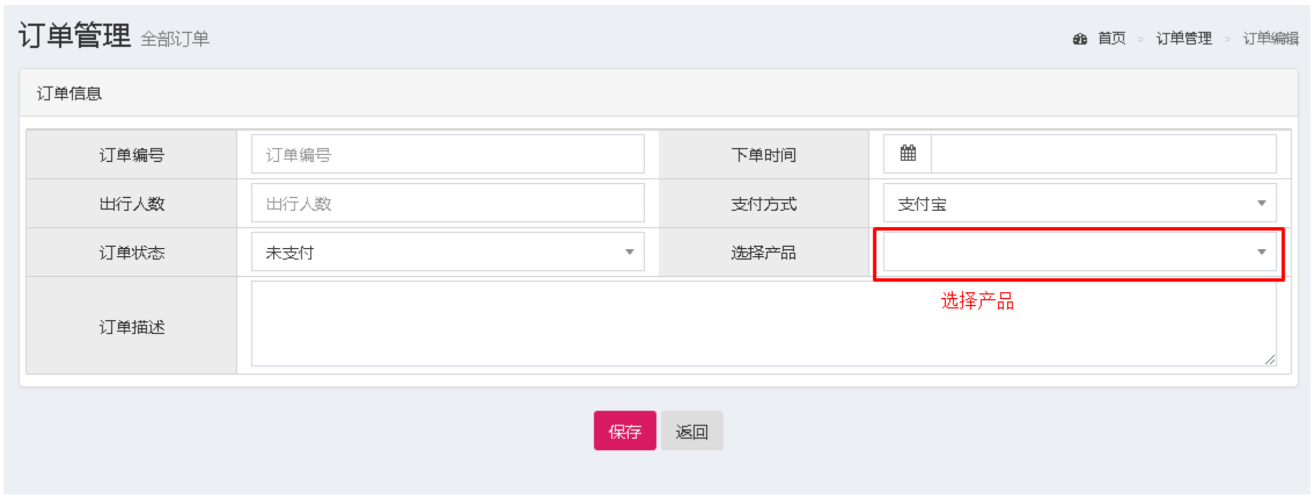
2.5 编写页面order-list.jsp

```jsp
<c:forEach items="${orderList}" var="order">

    <tr>
        <td><input name="ids" type="checkbox"></td>
        <td>${order.id}</td>
        <td>${order.orderNum}</td>
        <td>${order.product.productName}</td>
        <td>${order.orderTime}</td>
        <td>${order.peopleCount}</td>
        <td>${order.orderStatus==1?"已支付":"未支付"}</td>

        <td class="text-center">
            <button type="button" class="btn bg-olive btn-xs"
                onclick='location.href="${pageContext.request.contextPath}/pages/order-show.jsp"'>订单</button>
            <button type="button" class="btn bg-olive btn-xs"
                onclick='location.href="${pageContext.request.contextPath}/pages/order-show.jsp"'>查看</button>
        </td>
    </tr>

</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

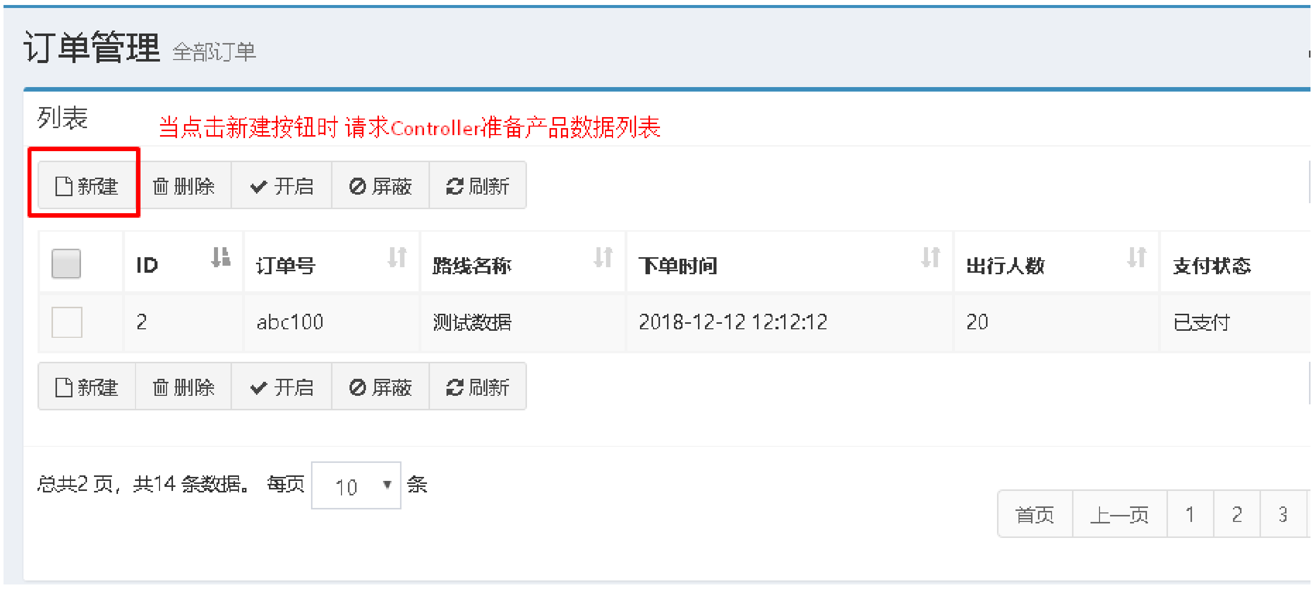


订单管理 全部订单

首页 > 订单管理 > 订单编辑

订单信息

订单编号	订单编号	下单时间	
出行人数	出行人数	支付方式	支付宝
订单状态	未支付	选择产品	
订单描述			

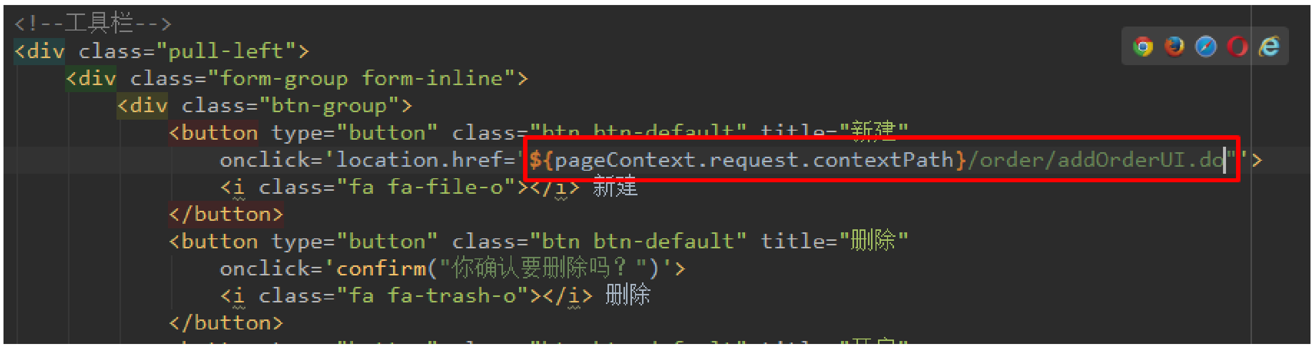


订单管理 全部订单

列表 当点击新建按钮时 请求Controller准备产品数据列表

	ID	订单号	路线名称	下单时间	出行人数	支付状态
	2	abc100	测试数据	2018-12-12 12:12:12	20	已支付

总共2 页，共14 条数据。 每页 10 条



```
<!--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

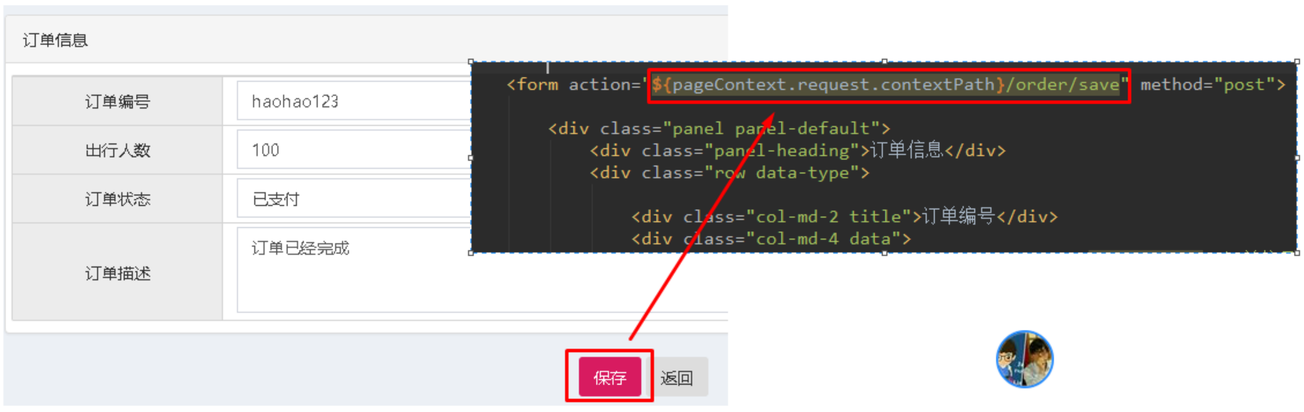
```java
@Autowired
private ProductService productService;

@RequestMapping("/addOrderUI")
public ModelAndView addOrderUI(){
    List<Product> productList = productService.findAll();
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("productList",productList);
    modelAndView.setViewName("order-add");
    return  modelAndView;
}
```

3.3 编写页面order-add.jsp

```html
<div class="col-md-4 data">
    <select class="form-control select2" style="width: 100%"
            name="product.id">

        <c:forEach items="${ productList }" var="p">
            <option value="${ p.id }" >${ p.productName }</option>
        </c:forEach>

    </select>
</div>
```

3、添加订单功能-保存订单数据

3.1 页面入口地址



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

```html
<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">
```

2.2 编写Controller

```java
@RequestMapping("/save")
public String save(Order order){
    orderService.save(order);
    return  "redirect:/order/findAll";
}
```

2.3 编写Service

OrderService接口

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

OrderServiceImpl接口实现

```java
@Override
public void save(Order order) {
    orderMapper.save(order);
}
```

2.4 编写Mapper

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