

## Spring MVC6 –增加事务管理、查看购物车

### 创建工程 estore6

将 estore5 相关文件复制过来，包名称中 estore5 修改为 estore6,能正常运行，进行下一步

### 增加事务管理

### 编辑 spring 配置文件（集成 mybatis）

```
<!-- 配置数据源 -->
<bean id="dataSource" class="org.apache.tomcat.dbcp.dbcp.BasicDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
    <property name="url" value="jdbc:mysql://localhost:3306/estore"></property>
    <property name="username" value="root"></property>
    <property name="password" value=""></property>
    <property name="initialSize" value="10"></property>
    <property name="maxIdle" value="5"></property>
</bean>
<!-- 基于数据源的事务管理 -->
<bean id="txManager"
    class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"></property>
</bean>
<!-- 配置基于注解的声明式事务 -->
<tx:annotation-driven transaction-manager="txManager"
    mode="proxy" proxy-target-class="true" />
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <!-- 引用数据库连接池 -->
    <property name="dataSource" ref="dataSource" />
    <!-- 加载mybatis-config.xml 配置-->
    <property name="configLocation" value="classpath:mybatis-config.xml">
    </property>
    <!-- <property name="mapperLocations" value="classpath:com/spring/mapper/*.xml">
        <list> <value>com/spring/entity/Zp.xml</value> </list>
    </property>
    -->
</bean>
```

## 编辑 mybatis 配置文件

```
<configuration>
<!-- <properties resource="resources/db.properties" /> -->
<typeAliases>
<typeAlias type="com.estore6.domain.CartItem" alias="CartItem"></typeAlias>
<typeAlias type="com.estore6.domain.Product" alias="Product"></typeAlias>
</typeAliases>
<plugins>
<!-- com.github.pagehelper为PageHelper类所在包名 -->
<plugin interceptor="com.github.pagehelper.PageInterceptor">
<property name="supportMethodsArguments" value="true"/>
<property name="params" value="pageNum=pageNumKey;pageSize=pageSizeKey;"/>
<property name="reasonable" value="true"/>
</plugin>
</plugins>
<!-- | <environments default="development">
<environment id="development">
<transactionManager type="JDBC" />
<dataSource type="POOLED">
<property name="driver" value="${jdbc.driverClassName}" />
<property name="url" value="${jdbc.url}" />
<property name="username" value="${jdbc.username}" />
<property name="password" value="${jdbc.password}" />
</dataSource>
</environment>
</environments> -->
<mapppers>
<mapper resource="com/estore6/mapper/ProductMapper.xml" />
<mapper resource="com/estore6/mapper/CartMapper.xml" />
</mapppers>
</configuration>
```

## 声明事务管理

```
@Service
@Transactional(propagation = Propagation.REQUIRED)
public class CartServiceImpl implements CartService {
    @Autowired
    private MyDao dao;
```

## 编辑 MyDao

自动装配 spring 配置文件定义的 sqlSessionSessionFactory

```

@Autowired
private SqlSessionFactory sqlSessionFactory;

private SqlSession getConn() {
    try {
        return sqlSessionFactory.openSession();
    } catch (Exception e) {
        // TODO: handle exception
        log.fatal(e.getMessage());
        return null;
    }
}

public boolean insert(Class cls, Object obj) {
    SqlSession sqlSession = getConn();
    int n=0;
    try {
        IMapper mapper = (IMapper) sqlSession.getMapper(cls);
        n= mapper.insert(obj);
        sqlSession.commit();
    } finally {
        sqlSession.close();
    }
    return n>0;
}

public boolean delete(Class cls, Integer id) {
    SqlSession sqlSession = getConn();
    int n=0;
    try {
        IMapper mapper = (IMapper) sqlSession.getMapper(cls);
        n=mapper.delete(id);
        sqlSession.commit();
    } finally {
        sqlSession.close();
    }
    return n>0;
}

public boolean update(Class cls, Object obj) {
    SqlSession sqlSession = getConn();
    int n=0;
    try {
        IMapper mapper = (IMapper) sqlSession.getMapper(cls);
        n=mapper.update(obj);
        sqlSession.commit();
    } finally {
        sqlSession.close();
    }
    return n>0;
}

```

删除 MyBatisDAOUtil.java 文件

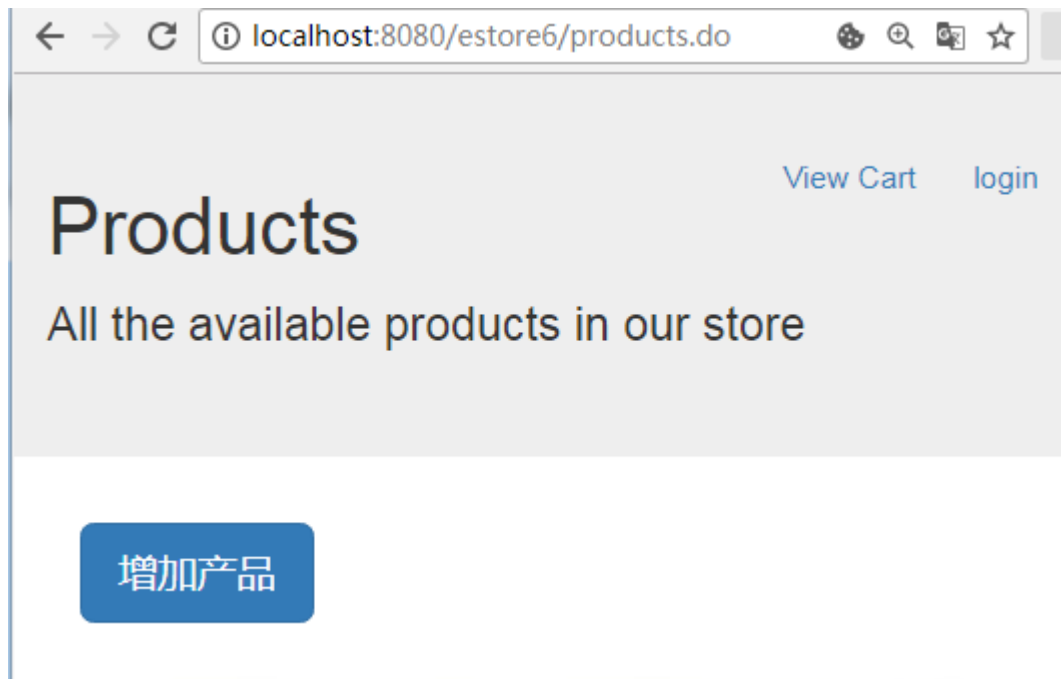
显示选购商品

编辑显示商品 products.jsp

```
<body>
  <section> <div class="jumbotron">
<c:choose>
<c:when test="${sessionScope.memlogin!=null}">
<a href="loginAction/memlogout.do" class="btn btndanger
btn-mini pull-right">logout</a>
</c:when>
<c:otherwise>
<a href="<%=basePath%>/memlogin.jsp" class="btn btndanger
btn-mini pull-right">login</a>
</c:otherwise>
</c:choose>

  <a href="javascript:viewCart()" class="btn btndanger
btn-mini pull-right">View Cart</a>

    <div class="container">
      <h1>Products</h1>
      <p>All the available products in our store</p>
    </div>
  </div>
</body>
```



## 编辑 isLogin.jsp

```
function isMemLogin(){
    <% String memLogin=(String)session.getAttribute("memlogin");

    if(memLogin==null){
    %>
        if( confirm('你还没登录，请先登录会员')) {
            window.location.href="<%=basePath %>/memlogin.jsp" ;

        }
        return false;
    <%}else{    %>
    return true;

    <%}    %>
}

function viewCart(){
    if(isMemLogin())
    {
        window.location.href="<%=basePath %>/cart.do";
    }
}
```

## cart.do 动作

```
@RequestMapping(method = RequestMethod.GET)
public String toCart() {
    return "cart";
}
```

## cart.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>Cart</title>
8 <%@ include file="head.jsp" %>

9 <%@ include file="head.jsp" %>
10 <script type="text/javascript">
11     function select(){
12         var str="";
13         $.ajax({
14             url:"cart/getCartItem.do",//请求
15             dataType:"json",//返回json类型,list的json类型是个数组
16             success:function(data){//若请求成功,则将服务器传来的信息返回给data
17                 // alert(data.length)
18                 var total=0;
19                 for(var i=0;i<data.length;i++){
20                     total=total+data[i]["totalPrice"];
21                     str+="
```

```

37=<body>
38=<section>
39=<div class="jumbotron">
40=<div class="container">
41 <h1>Cart</h1>
42 <p>All the selected products in your cart</p>
43 </div>
44 </div>
45 </section>
46=<section class="container">
47=<div>
48
49=<table class="table table-hover">
50=<thead>
51 <tr> <th>Product</th> <th>Unit price</th> <th>Qauntity</th> <th>Price</th> <th>Action</th> </tr>
52 </thead>
53
54 <tbody id="cartShow"> </tbody>
55
56=<tfoot>
57 <tr> <th></th> <th></th> <th></th> <th>Grand Total</th> <th id="grandTotal"></th> <th></th> </tr>
58 </tfoot>
59 </table>

60 |
61=<div>
62=<a href="<spring:url value="/products.do" />" class="btn btndefault">
63 <span class="glyphicon-hand-left glyphicon"></span>
64 Continue shopping
65=</a> <a href="#" class="btn btn-success pull-right"> <span
66 class="glyphicon-shopping-cart glyphicon"></span> Check out
67 </a>
68 </div>
69 </div>
70
71 </section>
72=<script type="text/javascript">
73 select();
74 </script>
75 </body>
76 </html>

```

## 实现 cart/getCartItem.do 动作

```

@Controller
@RequestMapping(value = "/cart")
public class CartController {

    @Autowired
    private CartService cartService;

    @Autowired
    private ProductService productService;

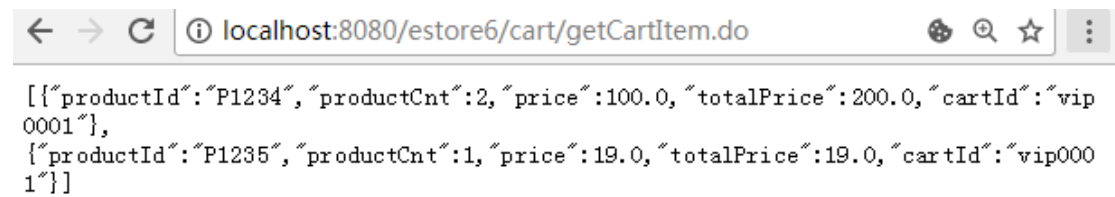
    Logger log = Logger.getLogger(CartController.class);

    @RequestMapping(method = RequestMethod.GET)
    public String toCart() {
        return "cart";
    }

    @RequestMapping(value = "/getCartItem", method = RequestMethod.GET)
    public @ResponseBody List<CartItem> getCartItem(HttpServletRequest request) {
        String cartId = (String) request.getSession().getAttribute("memlogin");
        return cartService.getAllCartItems(new CartItem(null, cartId));
    }
}

```

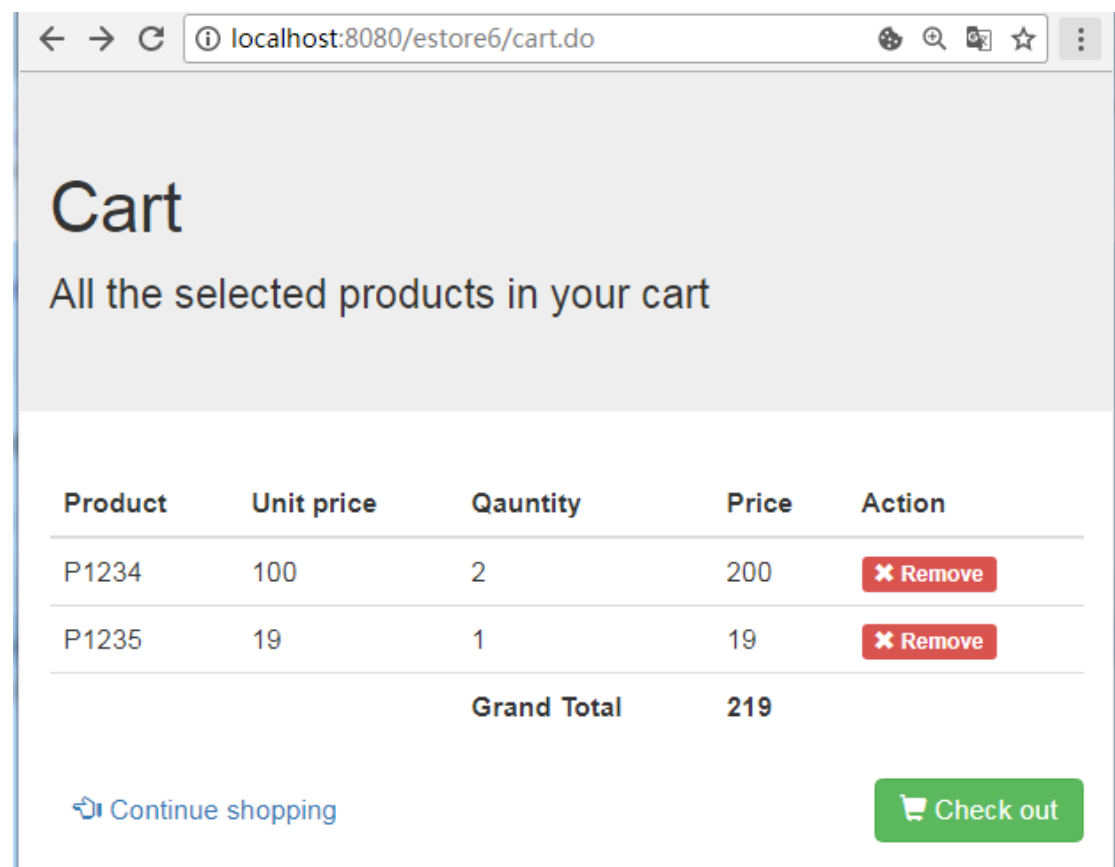
注意不需要进行 json 转换，使用 spring 自带功能（注解 `ResponseBody`）  
地址栏请求效果



## Service

```
@Override
public List<CartItem> getAllCartItems(CartItem cartItem) {
    // TODO Auto-generated method stub
    return dao.getAll(CartMapper.class, cartItem);
}
```

## 查看效果





参考选购商品，实现从购物车移除商品（异步实现）

注意：删除选购商品的同时，要调整产品表的库存数量

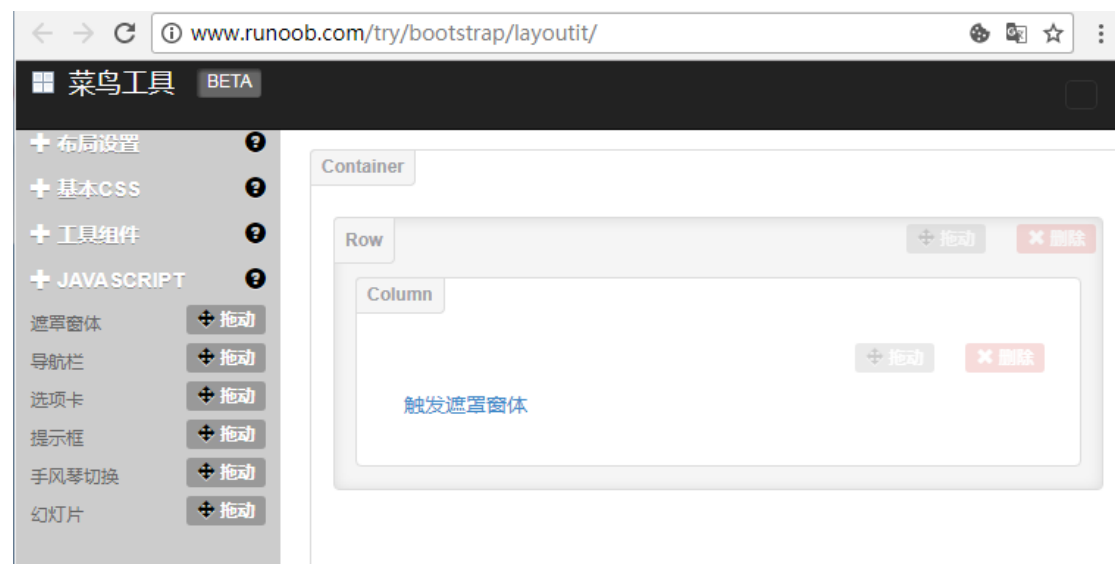
## Spring MVC7- 消息提示、数据校验

创建工程 estore7

将 estore6 相关文件复制过来，包名称中 estore6 修改为 estore7,能正常运行，进行下一步

消息提示

通过菜鸟教程中可视化布局



## 查看遮眼罩窗体代码

下载 ×

已在下面生成干净的HTML, 可以复制粘贴代码到你的body内

```
<button type="button" class="close" data-dismiss="modal" aria-hidden="true">x</button>
<h4 class="modal-title" id="myModalLabel">
    标题
</h4>
</div>
<div class="modal-body">
    内容...
</div>
<div class="modal-footer">
    <button type="button" class="btn btn-default"
data-dismiss="modal">关闭</button> <button type="button" class="btn btn-
primary">保存</button>
</div>
```

关闭

## 创建 dlg.jsp

复制代码过来，调整为提示会员登录消息框

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Products</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<%@ include file="head.jsp" %>

<script type="text/javascript">
$(function(){
    $("#btnLogin").click(
    function(){
        window.location.href="<%=basePath %>/memlogin.jsp" ;
    }
    );
});

</script>

<a id="modal-307389" href="#modal-container-307389" role="button" class="btn" style="display: none;" data-toggle="modal">触发遮罩窗体</a>
<div class="modal fade" id="modal-container-307389" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>
<h4 class="modal-title" id="myModalLabel">
提示
</h4>
<div>
<div class="modal-body">
你还没登录，请以会员身份登录
</div>
<div class="modal-footer">
<button type="button" class="btn btn-primary" id="btnLogin">确定</button>
<button type="button" class="btn btn-default" data-dismiss="modal">关闭</button>
</div>
</div>
</div>

```

## 编辑 isLogin.jsp

### 编辑前

```

function isMemLogin(){
    <%    String memLogin=(String)session.getAttribute("memlogin");

    if(memLogin==null){
    %>
        if( confirm('你还没登录，请先登录会员')) {
            window.location.href="<%=basePath %>/memlogin.jsp" ;

        }
        return false;
    <%}else{    %>
        return true;

    <%}    %>

}

```

## 编辑后

```
function isMemLogin(){  
    <%    String memLogin=(String)session.getAttribute("memlogin");  
  
    if(memLogin==null){  
        <%  
            $("#modal-307389").click()  
            return false;  
        <%>else{  
            <%>  
            return true;  
        <%>  
    }  
}
```

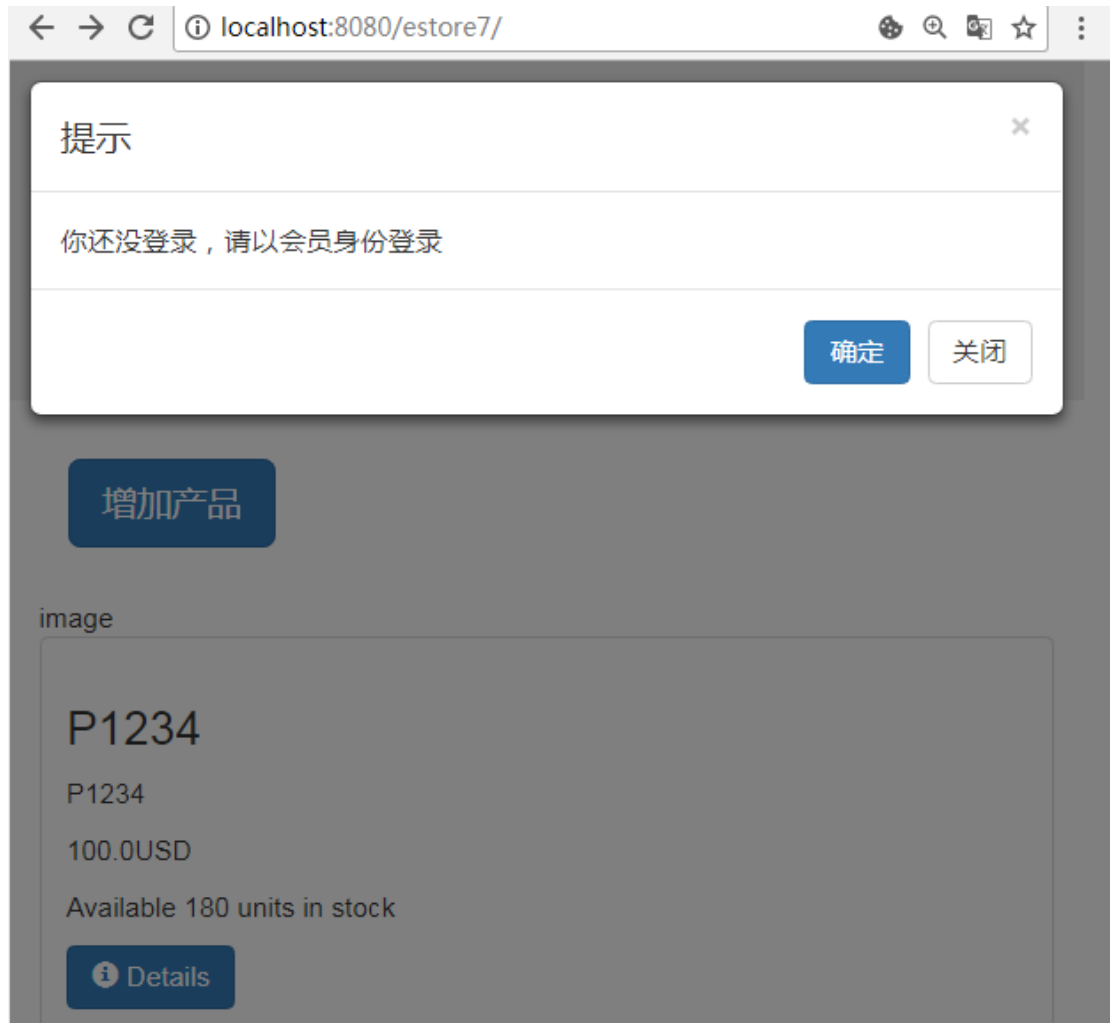
## 使用 dlg.jsp

在需要的地方，包含 dlg.jsp

```
<body>
```

```
<jsp:include page="dlg.jsp"></jsp:include>
```

第一打开浏览器，点击查看购物车效果



## 数据校验（以校验商品编码为例）

- 方法一：在前端用 js 校验（自己写函数、第三方写好的函数、正则表达式）
- 方法二：在后端 java 校验（自己写函数、第三方写好的函数、正则表达式）
- 方法三：利用校验框架(javabeen validation/ custome/validation/spring validattion)

## Java bean validation (JSR-303)

Constraint	Description	Example
<code>@AssertFalse</code>	The value of the field or property must be false.	<code>@AssertFalse</code> <code>boolean isUnsupported;</code>
<code>@AssertTrue</code>	The value of the field or property must be true.	<code>@AssertTrue</code> <code>boolean isActive;</code>
<code>@DecimalMax</code>	The value of the field or property must be a decimal value lower than or equal to the number in the value element.	<code>@DecimalMax("30.00")</code> <code>BigDecimal discount;</code>
<code>@DecimalMin</code>	The value of the field or property must be a decimal value greater than or equal to the number in the value element.	<code>@DecimalMin("5.00")</code> <code>BigDecimal discount;</code>
<code>@Digits</code>	The value of the field or property must be a number within a specified range. The <code>integer</code> element specifies the maximum integral digits for the number, and the <code>fraction</code> element specifies the maximum fractional digits for the number.	<code>@Digits(integer=6, fraction=2)</code> <code>BigDecimal price;</code>
<code>@Future</code>	The value of the field or property must be a date in the future.	<code>@Future</code> <code>Date eventDate;</code>
<code>@Max</code>	The value of the field or property must be an integer value lower than or equal to the number in the value element.	<code>@Max(10)</code> <code>int quantity;</code>
<code>@Min</code>	The value of the field or property must be an integer value greater than or equal to the number in the value element.	<code>@Min(5)</code> <code>int quantity;</code>
<code>@NotNull</code>	The value of the field or property must not be null.	<code>@NotNull</code> <code>String username;</code>
<code>@Null</code>	The value of the field or property must be null.	<code>@Null</code> <code>String unusedString;</code>
<code>@Past</code>	The value of the field or property must be a date in the past.	<code>@Past</code> <code>Date birthday;</code>

## jar 包

validation-api-1.0.0.GA.jar  
hibernate-validator-4.2.0.Final.jar

## 编辑 Product.java

```
@Pattern(regexp="P[0-9]{4}", message="{Pattern.Product.productId.validation}")  
private String productId;
```

## 增加消息提示（messages 相关文件）

```
Pattern.Product.productId.validation  
=   
\u4EA7\u54C1\u7F16\u53F7\u7531P\u548C4\u4E2A\u6570\u5B57\u7EC4\u6210.
```

## 增加显示消息标签（form: errors）

```
<form:form modelAttribute="newProduct" class="form-horizontal" enctype="multipart/form-data">  
<fieldset> <legend>Add new product</legend>  
<form:errors path="*" cssClass="alert alert-danger" element="div"/>  
<div class="form-group">  
<label class="control-label col-lg-2 col-lg-2" for="productId">  
<spring:message code="addProduct.form.productId.label"></spring:message>  
</label>  
<div class="col-lg-10">  
<form:input id="productId" path="productId" type="text" class="form-input-large" />  
<form:errors path="productId" cssClass="text-danger"/>  
</div>  
</div>
```

## 编辑控制器（ProductController.java）

**注意：**增加校验结果形参，产品模型后面

```
@RequestMapping(value = "/addProduct", method = RequestMethod.POST)  
public String addProduct(HttpServletRequest request, @ModelAttribute("newProduct") @Valid Product newProduct,  
    BindingResult result) {  
    if (result.hasErrors()) {  
        return "addProduct";  
    }  
}
```

## 编辑 spring 配置文件

```
<mvc:annotation-driven validator="validator"></mvc:annotation-driven>
|
<bean id="validator"
    class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean">
    <property name="validationMessageSource" ref="messageSource" />
</bean>
```

## 运行效果

localhost:8080/estore7/products/addProduct.do?language=zh\_CN

English|中文

Products

Add products

Add new product

产品编号由P和4个数字组成.

产品编号

Paaa1

产品编号由P和4个数字组成.

产品名称



## custom validation

### 定义校验注解

```
package com.estore7.validator;

import java.lang.annotation.Documented;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import static java.lang.annotation.ElementType.ANNOTATION_TYPE;
import java.lang.annotation.ElementType;
import static java.lang.annotation.ElementType.METHOD;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import javax.validation.Constraint;
import javax.validation.Payload;

@Target({ METHOD, ElementType.FIELD, ANNOTATION_TYPE })
@Retention(RUNTIME)
@Constraint(validatedBy = ProductIdValidator.class)
@Documented
public @interface ProductId {
    String message() default "{com.estore7.validator.ProductId.message}";
    Class<?>[] groups() default {};
    public abstract Class<? extends Payload>[] payload() default {};
}
```

### 实现校验

```

import javax.validation.ConstraintValidatorContext;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import com.estore7.domain.Product;
import com.estore7.service.ProductService;
@Component
public class ProductIdValidator implements ConstraintValidator<ProductId, String> {
    @Autowired
    private ProductService productService;

    public void initialize(ProductId constraintAnnotation) {
    }

    public boolean isValid(String value, ConstraintValidatorContext context) {
        Product product;
        try {
            product = productService.getProductById(value);
        } catch (Exception e) {

            return true;
        }
        if (product != null) {
            return false;
        }
        return true;
    }
}

```

## 使用校验

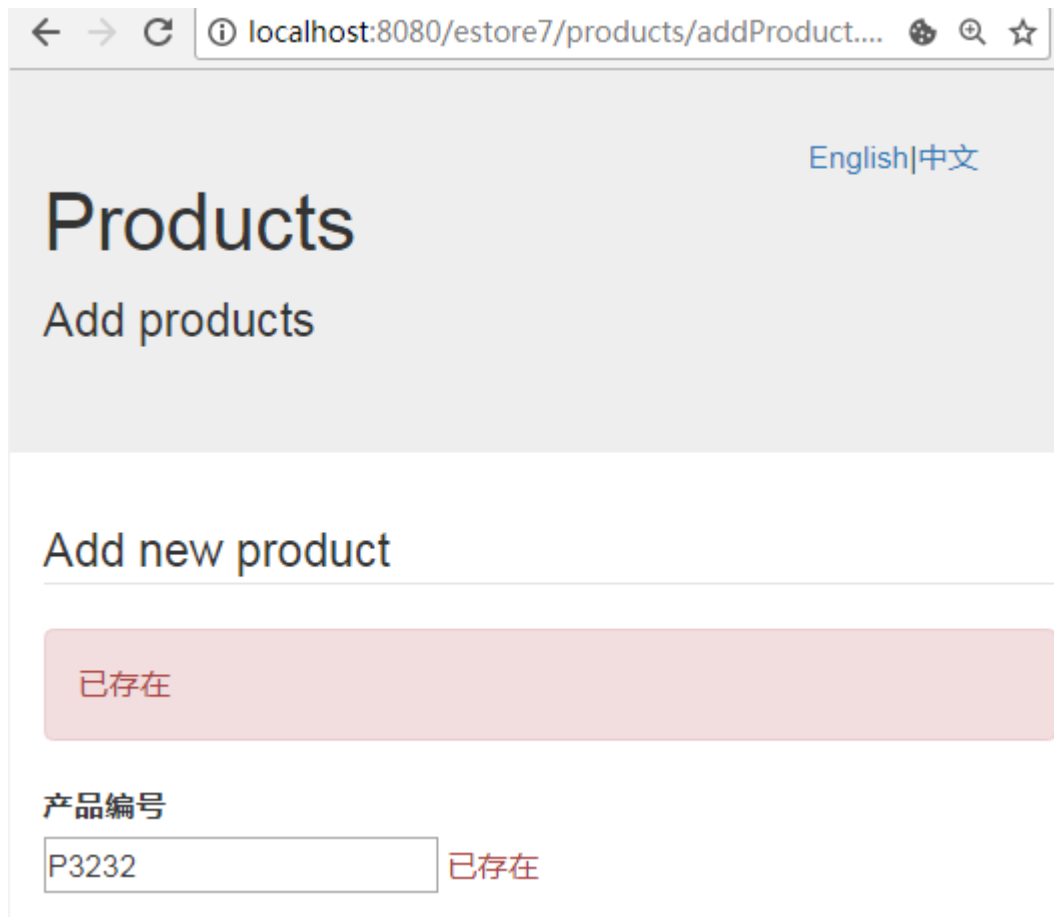
```

@Pattern(regexp="P[0-9]{4}", message="{Pattern.Product.productId.validation}")
@ProductId
private String productId;

```

属性文件增加相关提示消息

运行效果



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/estore7/products/addProduct...'. The page has a light gray header with the text 'English|中文' on the right. Below the header, the main content area has a title 'Products' and a subtitle 'Add products'. A section titled 'Add new product' is followed by a large red error message box containing the text '已存在' (Already exists). Below this, there is a form with a label '产品编号' (Product Number) and an input field containing 'P3232'. To the right of the input field, the text '已存在' (Already exists) is displayed again.

English|中文

# Products

Add products

## Add new product

已存在

产品编号

P3232 已存在

## Spring Validation

### 创建价格校验器

```
@Component
public class UnitPriceValidator implements Validator {

    @Autowired
    private javax.validation.Validator beanValidator;

    public boolean supports(Class<?> clazz) {
        return Product.class.isAssignableFrom(clazz);
    }

    public void validate(Object target, Errors errors) {
        /*
         * 若注释这段，则之前校验无效，如下代码绑定bean validation 和 spring validation
         * Set<ConstraintViolation<Object>> constraintViolations =
         * beanValidator.validate(target); for (ConstraintViolation<Object>
         * constraintViolation : constraintViolations) { String propertyPath =
         * constraintViolation.getPropertyPath().toString(); String message =
         * constraintViolation.getMessage(); errors.rejectValue(propertyPath,
         * "", message); }
         */
        Product product = (Product) target;
        if (product.getUnitPrice() != null && product.getUnitPrice() <= 0) {
            errors.rejectValue("unitPrice", "Pattern.Product.unitPrice.validation");
        }
    }
}
```

### 设置校验器

```
@Controller
@RequestMapping("/products")
public class ProductController {

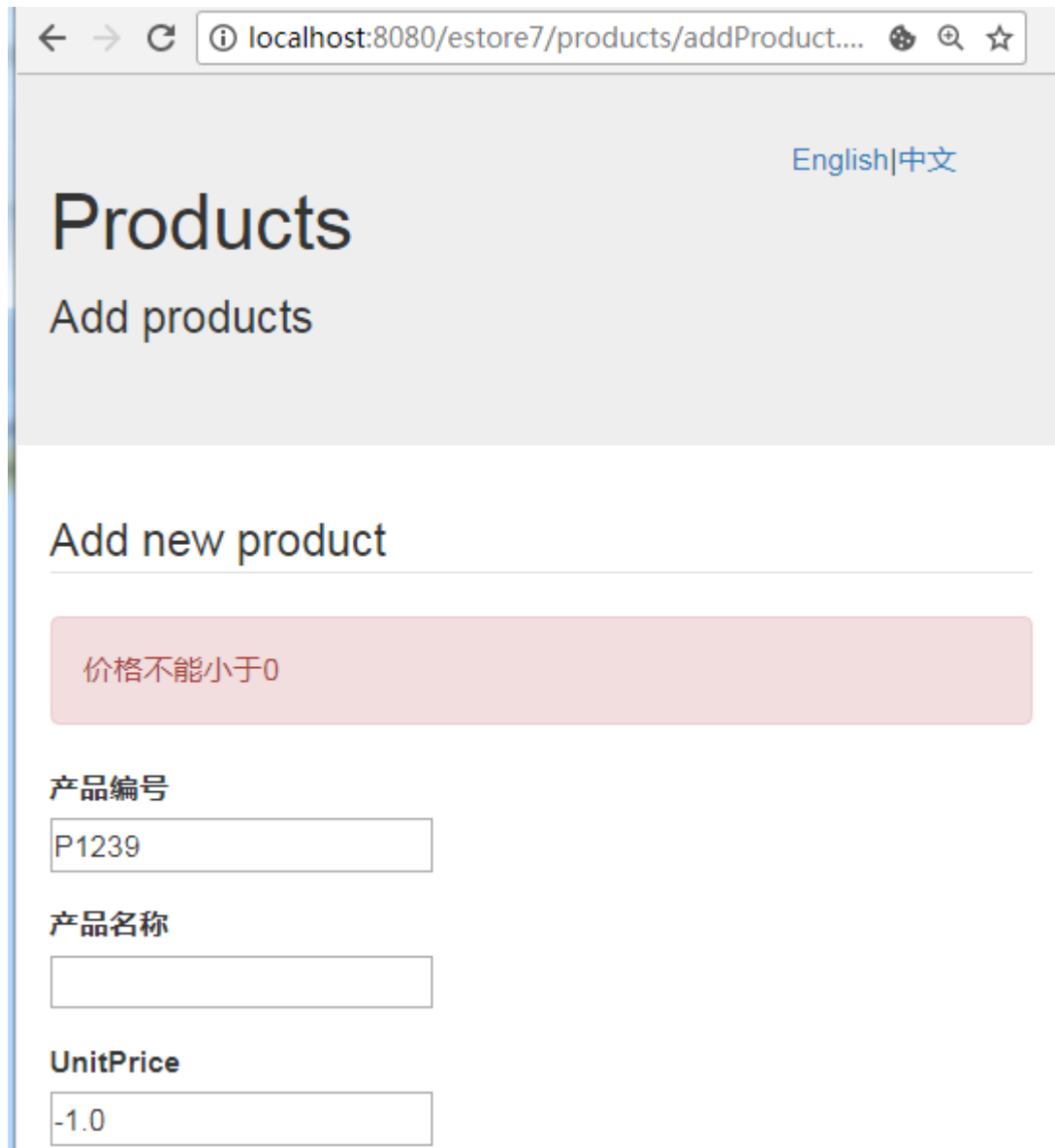
    Logger log = Logger.getLogger(ProductController.class);

    @Autowired
    private UnitPriceValidator unitsInStockValidator;

    @InitBinder
    public void initBinder(WebDataBinder binder) {
        binder.setValidator(unitsInStockValidator);
    }

    @Autowired
    private ProductService productService;
```

运行效果



The screenshot shows a web browser window with the address bar displaying `localhost:8080/estore7/products/addProduct...`. The page has a header with a language switcher set to "English|中文". The main heading is "Products" with a sub-heading "Add products". Below this is a section titled "Add new product" which contains a form. A red error message "价格不能小于0" (Price cannot be less than 0) is displayed above the form fields. The form has three fields: "产品编号" (Product ID) with the value "P1239", "产品名称" (Product Name) which is empty, and "UnitPrice" with the value "-1.0".

English|中文

# Products

Add products

## Add new product

价格不能小于0

产品编号

产品名称

UnitPrice

# Products

Add products

## Add new product

已存在  
在1和500之间  
价格不能小于0

产品编号

已存在

产品名称

UnitPrice

## 绑定多个 Spring 校验器

### 1 1. Create a class called ProductValidator

```
public class ProductValidator implements Validator{

    @Autowired
    private javax.validation.Validator beanValidator;

    private Set<Validator> springValidators;

    public ProductValidator() {
        springValidators = new HashSet<Validator>();
    }

    public void setSpringValidators(Set<Validator>springValidators) {
        this.springValidators = springValidators;
    }

    public boolean supports(Class<?> clazz) {
        return Product.class.isAssignableFrom(clazz);
    }

    public void validate(Object target, Errors errors) {
        Set<ConstraintViolation<Object>> constraintViolations
        =beanValidator.validate(target);

        for (ConstraintViolation<Object> constraintViolation
        :constraintViolations) {
            String propertyPath
        =constraintViolation.getPropertyPath().toString();
            String message = constraintViolation.getMessage();
            errors.rejectValue(propertyPath, "", message);
        }

        for(Validator validator: springValidators) {
            validator.validate(target, errors);
        }
    }
}
```

---

## 2 配置校验器（ref 可以引用多个校验器）

2. Now, open the web application context configuration file `DispatcherServlet-context.xml` and add the following bean definition to it:

```
<bean
id="productValidator"class="com.packt.webstore.validator.ProductValidator">
  <property name = "springValidators">
    <set>
      <ref bean = "unitsInStockValidator"/>
    </set>
  </property>
</bean>
```

3. Create one more bean definition for the `UnitsInStockValidator` class, as follows, and save `DispatcherServlet-context.xml`:

```
<bean id="unitsInStockValidator"
class="com.packt.webstore.validator.UnitsInStockValidator"/>
```

## 3 将如下校验器修改为 `ProductValidator` 即可

```
@Controller
@RequestMapping("/products")
public class ProductController {

    Logger log = Logger.getLogger(ProductController.class);

    @Autowired
    private UnitPriceValidator unitsInStockValidator;

    @InitBinder
    public void initBinder(WebDataBinder binder) {
        binder.setValidator(unitsInStockValidator);
    }

    @Autowired
    private ProductService productService;
```