**参数绑定**

# 本章内容

* 基本数据类型的绑定
* 包装数据类型的绑定
* POJO（实体类）类型的绑定
* 复合POJO（实体类）类型的绑定
* 数组类型的绑定
* List类型的绑定
* Map类型的绑定
* @RequestParam

# 需求

拷贝上一个案例，项目命名springmvc02\_1，在之前基础上实现商品模拟数据添加。

# 基本数据类型的绑定

PS：当请求的参数名称和处理器形参名称一致时会将请求参数与形参进行绑定

,基本数据类型参数绑定要求表单传递过来的数据不能为null，为空则抛出异常

## 创建addItems.jsp页面

<form id=*"itemForm"* action=*"*${pageContext.request.contextPath }*/items/addItems.action"* method=*"post"*>

<table width=*"100%"* border=*1*>

<tr>

<td>商品名称</td>

<td><input type=*"text"* name=*"name"* /></td>

</tr>

<tr>

<td>商品价格</td>

<td><input type=*"text"* name=*"price"*/></td>

</tr>

<tr>

<td>商品生产日期</td>

<td><input type=*"text"* name=*"createtime"* /></td>

</tr>

<tr>

<td>商品简介</td>

<td><textarea rows=*"3"* cols=*"30"* name=*"detail"*></textarea>

</td>

</tr>

<tr>

<td colspan=*"2"* align=*"center"*><input type=*"submit"* value=*"提交"* /></td>

</tr>

</table>

</form>

## 扩展ItemsController类

创建addItemsUI方法

//跳转到商品添加页面

@RequestMapping("/addItemsUI")

**public** String addItemsUI() {

**return** "items/addItems";

}

创建addItems方法

//商品添加

@RequestMapping("/addItems")

**public** String addItems(String name,**float** price,String detail,String createtime)**throws** Exception {

Items items = **new** Items();

items.setCreatetime(**new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(createtime));

items.setDetail(detail);

items.setName(name);

items.setPrice(price);

**return** "items/addItems";

}

## web.xml处理中文乱码

<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>

## tomcat测试

<http://localhost:8080/springmvc02_1/items/addItemsUI.action>

# 包装数据类型的绑定

和基本数据类型基本一样，不同之处在于，表单传递过来的数据可以为null或””，以上面代码为例，如果表单中price为””或者表单中无price这个input，那么，Controller方法参数中的price值则为null。

@RequestMapping("/addItems")

**public** String addItems(String name,**Float** price,String detail,String createtime)**throws** Exception {

//省略

}

# POJO（实体类）类型的绑定

PS: 以上面代码为例，JSP页面输入框的 name 属性值和 POJO 实体类的属性保持一致即可映射成功

## 修改ItemsController中的addItems方法

@RequestMapping("/addItems")

**public** String addItems(Items items)**throws** Exception {

System.***out***.println(items);

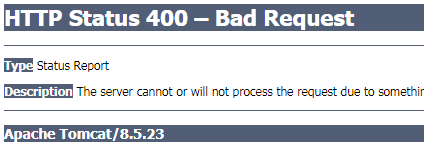
**return** "items/addItems";

}

## tomcat测试

<http://localhost:8080/springmvc02_1/items/addItemsUI.action>

提交之后jsp页面出现了异常



原因是：**Items类中的createtime属性是 Date 类型的，而我们输入的是字符串类型，故绑定不了**

解决方法：自定义Date类型的转换器

## 自定义转换器类

### 创建DateConverter

包名：com.springmvc.web.converter

//需要实现Converter接口，这里是将String类型转换成Date类型

**public** **class** DateConverter **implements** Converter<String, Date> {

@Override

**public** Date convert(String source) {

//实现将字符串转成日期类型(格式是yyyy-MM-dd HH:mm:ss)

SimpleDateFormat dateFormat = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

**try** {

**return** dateFormat.parse(source);

} **catch** (ParseException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

//如果参数绑定失败返回null

**return** **null**;

}

}

### 在 springmvc.xml 文件中配置转换器

<!-- 注解自动加载器 -->

<mvc:annotation-driven conversion-service=*"conversionService"*></mvc:annotation-driven>

<bean id=*"conversionService"* class=*"org.springframework.format.support.FormattingConversionServiceFactoryBean"*>

<property name=*"converters"*>

<!-- 自定义转换器的类名 -->

<bean class=*"com.springmvc.web.converter.DateConverter"*></bean>

</property>

</bean>

## 重新tomcat测试

<http://localhost:8080/springmvc02_1/items/addItemsUI.action>

# 复合POJO（实体类）类型的绑定

## 创建ItemsVO

**public** **class** ItemsVO {

**private** String tel;

**private** Items items;//此处，引用了items对象属性

}

## 修改ItemsController中的addItems方法

@RequestMapping("/addItems") //【复合POJO（实体类）类型的绑定】

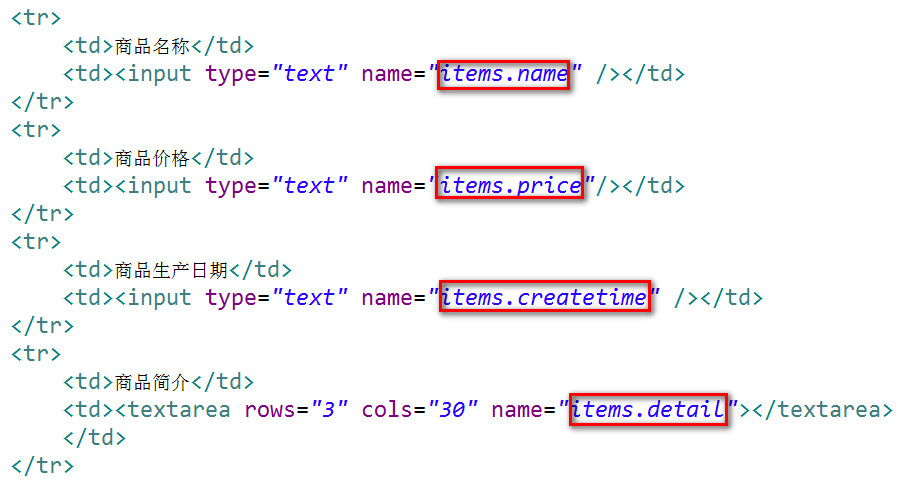
**public** String addItems(ItemsVO itemsVO)**throws** Exception {

System.***out***.println(itemsVO);

**return** "items/addItems";

}

## 修改addItems.jsp页面



**PS:**ItemsVO中有个items对象属性，在表单中，需要使用**“对象类型的属性.属性名”**来命

名input的name

## tomcat测试

<http://localhost:8080/springmvc02_1/items/addItemsUI.action>

# 数组类型的绑定

## addItems.jsp

<input type=*"hidden"* name=*"id"* value=*"1"*>

<input type=*"hidden"* name=*"id"* value=*"2"*>

<input type=*"hidden"* name=*"id"* value=*"3"*>

## 修改ItemsController中的addItems方法

@RequestMapping("/addItems") //【数组类型的绑定】

**public** String addItems(Integer[] id)**throws** Exception {

System.***out***.println(id.length);

**return** "items/addItems";

}

## tomcat测试

<http://localhost:8080/springmvc02_1/items/addItemsUI.action>

# List类型的绑定

## 修改ItemsVO.java

**public** **class** ItemsVO {

**private** String tel;

**private** Items items;//此处，引用了items对象属性

**private** List<Items> itemsList;//集合对象属性

}

## 创建editItems.jsp

<form id=*"itemForm"* action=*"*${pageContext.request.contextPath }*/items/addItems.action"* method=*"post"*>

<c:forEach items=*"*${itemsList}*"* var=*"items"* varStatus=*"status"*>

<table width=*"100%"* border=*1*>

<tr>

<td>商品名称</td>

<td><input type=*"text"* name=*"itemsList[*${status.index}*].name"* value=*"*${items.name }*"* /></td>

</tr>

<tr>

<td>商品价格</td>

<td><input type=*"text"* name=*"itemsList[*${status.index}*].price"* value=*"*${items.price }*"* /></td>

</tr>

<tr>

<td>商品生产日期</td>

<td><input type=*"text"* name=*"itemsList[*${status.index}*].createtime"*

value=*"*<fmt:formatDate value="${items.createtime}" pattern=*"yyyy-MM-dd HH:mm:ss"*/>*"* /></td>

</tr>

<tr>

<td>商品简介</td>

<td><textarea rows=*"3"* cols=*"30"* name=*"itemsList[*${status.index}*].detail"*>${items.detail }</textarea>

</td>

</tr>

</c:forEach>

<tr>

<td colspan=*"2"* align=*"center"*><input type=*"submit"* id=*"btn"* value=*"提交"* /></td>

</tr></form>

## 扩展ItemsController.java

**创建showAllItemsList方法,构建模拟数据**

@RequestMapping("/showAllItemsList")

**public** String showAllItemsList(Model model,ItemsVO itemsVO)**throws** Exception {

//商品列表

List<Items> itemsList = **new** ArrayList<Items>();

Items items\_1 = **new** Items();

items\_1.setName("联想笔记本");

items\_1.setPrice(6000f);

items\_1.setCreatetime(**new** Date());

items\_1.setDetail("ThinkPad T430 联想笔记本电脑！");

Items items\_2 = **new** Items();

items\_2.setName("苹果手机");

items\_2.setPrice(5000f);

items\_2.setCreatetime(**new** Date());

items\_2.setDetail("iphone6苹果手机！");

itemsList.add(items\_1);

itemsList.add(items\_2);

//填充数据到request

model.addAttribute("itemsList", itemsList);

**return** "items/editItems";

}

**创建addItems方法**

@RequestMapping("/addItems") //【List类型的绑定】

**public** String addItems(ItemsVO itemsVO)**throws** Exception {

System.***out***.println(itemsVO.getItemsList());

**return** **null**;

}

## tomcat测试

http://localhost:8080/springmvc02\_1/items/showAllItemsList.action

# Map类型的绑定

略……

# @RequestParam

@RequestParam是传递参数的.

@RequestParam用于将请求参数区数据映射到功能处理方法的参数上

## RequestParam 4个参数

1、String name ：参数名

2、String value ：即入参的请求参数名，如userName表示请求的参数区中的名字为userName的参数的值将传入

3、boolean required：是否必须，默认是true，表示请求中一定要有相应的参数，否则将报404错误码

4、String defaultValue：默认值

## 示例

public String queryUserName

(@RequestParam(value="userName" ,required =false, defaultValue="zhang" ) String uname)

userName要与jsp页面input对应的name值一样

uname属于形参，可以任意