参看文献：https://blog.csdn.net/eson\_15/article/details/51718633

# 概述

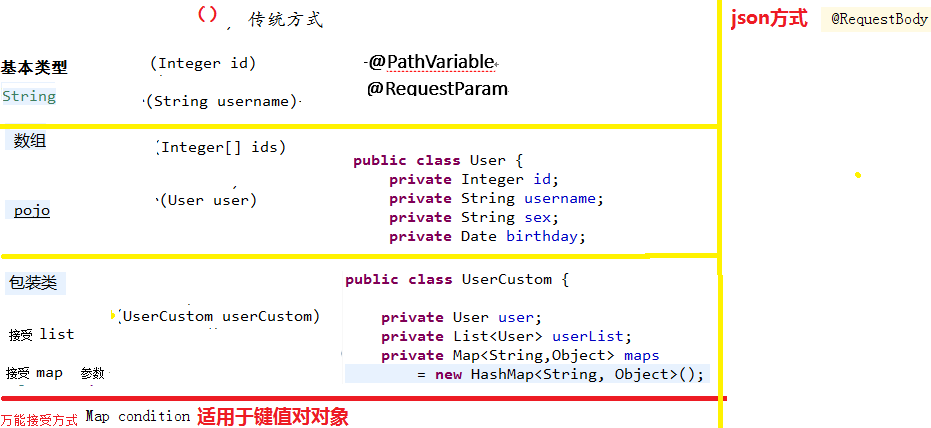
## 总述

分析接受参数类型：

基本类型，int，String等等基本类型。Pojo类型，包装类型，

Springmvc默认支持类型：HttpSession，HttpRequstServlet，Model等等。

Struts2参数：基于属性封装。Springmvc参数封装：基于方法进行封装。



记住一点，只要对应，就能接受，后台接受vo的必须比前端传回来的大

## 包装类的使用

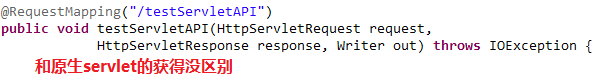
## 传统jsp



## servlet原生api（内置对象）

处理器形参中添加如下类型的参数处理适配器会默认识别并进行赋值。





# 绑定简单类型

## 原理

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。

这样，从Request取参数的方法就可以进一步简化。

## 支持的基本数据类型

参数类型推荐使用包装数据类型，因为基础数据类型不可以为null

整形：Integer、int

字符串：String

单精度：Float、float

双精度：Double、double

布尔型：Boolean、boolean

说明：对于布尔类型的参数，请求的参数值为true或false。或者1或0

请求url：

http://localhost:8080/xxx.action?id=2&status=false

处理器方法：

public String editItem(Model model,Integer id,Boolean status)

## @RequestParam

使用@RequestParam常用于处理简单类型的绑定。

value：参数名字，即入参的请求参数名字，如value=“itemId”表示请求的参数;区中的名字为itemId的参数的值将传入

required：是否必须，默认是true，表示请求中一定要有相应的参数，否则将报错;TTP Status 400 - Required Integer parameter 'XXXX' is not present

defaultValue：默认值，表示如果请求中没有同名参数时的默认值

定义如下：

@RequestMapping("/itemEdit")

**public** String queryItemById(@RequestParam(value = "itemId", required = **true**, defaultValue = "1") Integer id,

ModelMap modelMap) {

// 根据id查询商品数据

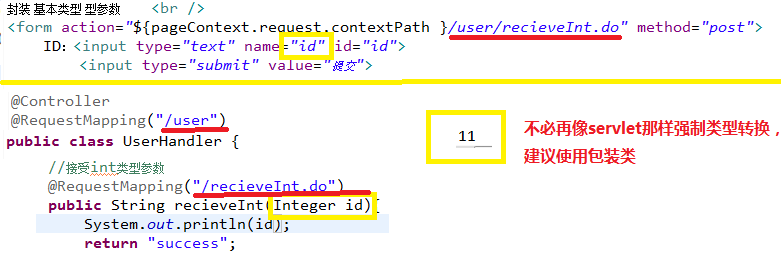
Item item = **this**.itemService.queryItemById(id);

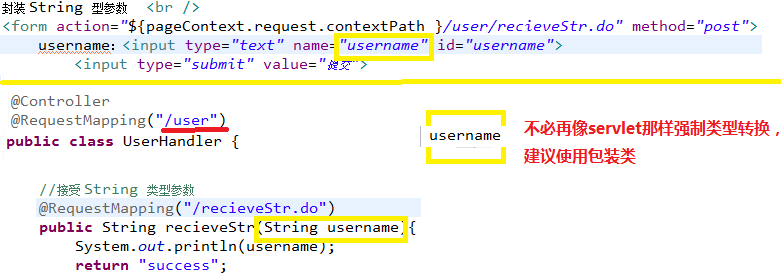
// 把商品数据放在模型中

modelMap.addAttribute("item", item);

**return** "itemEdit";

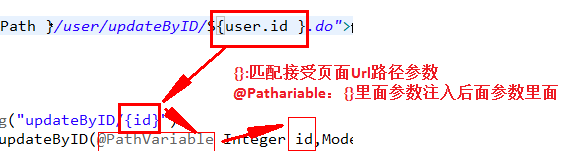
}



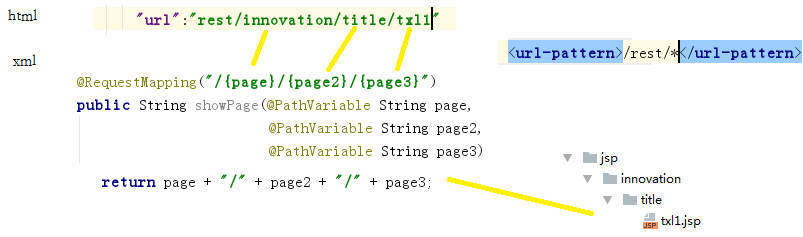


## @PathVariable

参数映射:映射请求URL传封装参数@PathVariable



参数多级传递:只能一级一级的接受



## @RequestBody

# 绑定pojo类型

## 传统

### 需求

将页面修改后的商品信息保存到数据库中。

### 需求分析

请求的url：/updateItem.action

参数：表单中的数据。

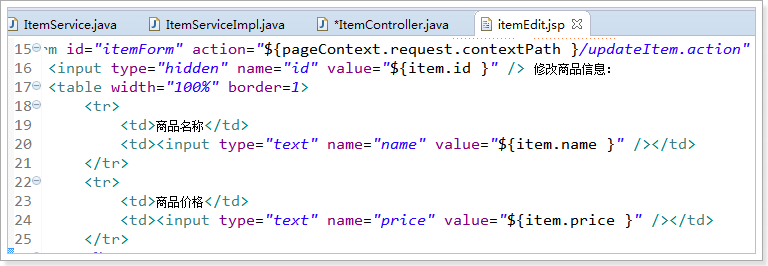
响应内容：更新成功页面

### 使用pojo接收表单数据

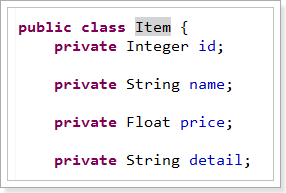
如果提交的参数很多，或者提交的表单中的内容很多的时候,可以使用简单类型接受数据,也可以使用pojo接收数据。

要求：pojo对象中的属性名和表单中input的name属性一致。

页面定义如下图：



Pojo(逆向工程生成)如下图：



请求的参数名称和pojo的属性名称一致，会自动将请求参数赋值给pojo的属性。

### ItemService接口

ItemService里编写接口方法

/\*\*

\* 根据id更新商品

\*

\* **@param** item

\*/

**void** updateItemById(Item item);

### ItemServiceImpl实现类

ItemServiceImpl里实现接口方法

使用updateByPrimaryKeySelective(item)方法，忽略空参数

@Override

**public** **void** updateItemById(Item item) {

**this**.itemMapper.updateByPrimaryKeySelective(item);

}

### ItemController

/\*\*

\* 更新商品,绑定pojo类型

\*

\* **@param** item

\* **@param** model

\* **@return**

\*/

@RequestMapping("/updateItem")

**public** String updateItem(Item item) {

// 调用服务更新商品

**this**.itemService.updateItemById(item);

// 返回逻辑视图

**return** "success";

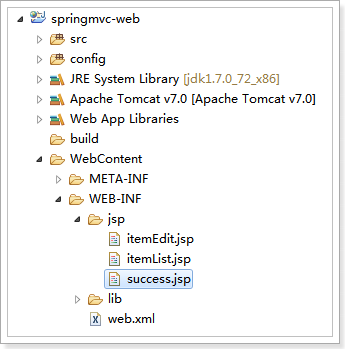
}

注意：

提交的表单中不要有日期类型的数据，否则会报400错误。如果想提交日期类型的数据需要用到后面的自定义参数绑定的内容。

### 编写success页面

如下图创建success.jsp页面



页面代码：

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<h1>商品修改成功！</h1>

</body>

</html>

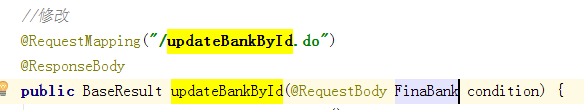
## 以json格式

### Ajax



**var** bankName = **$**(**'#bankName\_a'**).val().trim();  
**var** realName = **$**(**'#realName\_a'**).val().trim();  
**var** bankIdcard = **$**(**'#bankIdcard\_a'**).val().trim();  
**var** bankNumber = **$**(**'#bankNumber\_a'**).val().trim();  
**var** bankNameZhi = **$**(**'#bankNameZhi\_a'**).val().trim();  
**var** bankPhone = **$**(**'#bankPhone\_a'**).val().trim();  
*//进行验证  
//将数据封装成json***var** json={};  
json[**'userId'**] = **${**existUser.id**}**;  
json[**'hotelId'**] = **${**existUser.hotelId**}**;  
json[**'bankName'**] = bankName;  
json[**'realName'**] = realName;  
json[**'bankIdcard'**] = bankIdcard;  
json[**'bankNumber'**] = bankNumber;  
json[**'bankNameZhi'**] = bankNameZhi;  
json[**'bankPhone'**] = bankPhone;

### Controller接受



# 绑定包装pojo

## 传统

### 需求

使用包装的pojo接收商品信息的查询条件。

### 需求分析

包装对象定义如下：

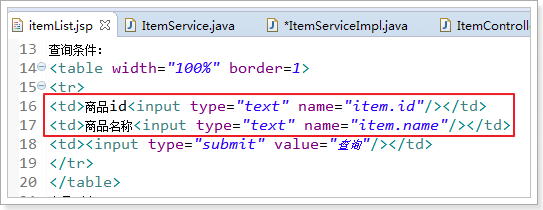
**public** **class** QueryVo {

**private** Item item;

set/get。。。

}

页面定义如下图：



### 接收查询条件

// 绑定包装数据类型

@RequestMapping("/queryItem")

**public** String queryItem(QueryVo queryVo) {

System.***out***.println(queryVo.getItem().getId());

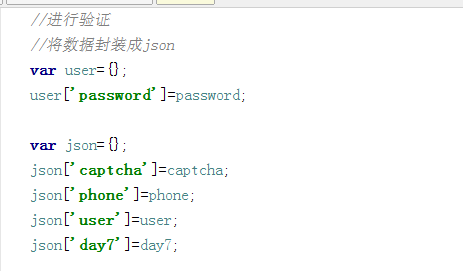
System.***out***.println(queryVo.getItem().getName());

**return** "success";

}

## 以json格式

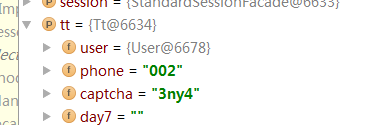
### Ajax



D:\Program Files\qq&网易云\2476955538\Image\C2C\JUU2E6$IBJQVP1WM0{BM_0O.png

### Controller接受

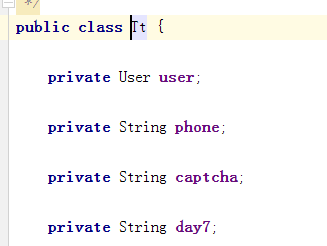
controller接受参数只能多不能少



### 获得参数

面向对象set(),get()就行

### 包装类



# Map

## 传统

### 前端



### 后台—模型驱动map



## 以json形式

### 针对

模型驱动，接受的json参数自动被map，model，modelAndView自动解析

注意接受参数的对应关系

### Ajax

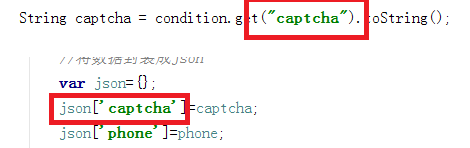


**var** phone=**$**(**'#phone'**).**val**().trim();  
**var** certificate = **$**(**'#certificate'**).**val**().trim();  
**var** beginDate=**$**(**'#beginDate'**).**val**().trim();  
**var** endDate=**$**(**'#endDate'**).**val**().trim();  
*//进行验证  
//将数据封装成json***var** json={};  
json[**'phone'**]=phone;  
json[**'certificate'**]=certificate;  
json[**'beginDate'**]=beginDate;  
json[**'endDate'**]=endDate;

### Controller接受

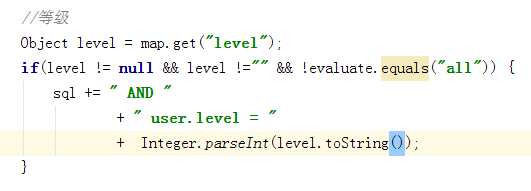


### 获得参数



此时接受到的String类型，要在dao使用什么具体类型，注意强转就行

标准点



# 绑定数组（一般拼接成字符串）

## 传统

### 需求

在商品列表页面选中多个商品，然后删除。

### 需求分析

功能要求商品列表页面中的每个商品前有一个checkbok，选中多个商品后点击删除按钮把商品id传递给Controller，根据商品id删除商品信息。

我们演示可以获取id的数组即可

### Jsp修改

修改itemList.jsp页面,增加多选框，提交url是queryItem.action

<form action=*"*${pageContext.request.contextPath }*/queryItem.action"* method=*"post"*>

查询条件：

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

<tr>

<td>商品id<input type=*"text"* name=*"item.id"* /></td>

<td>商品名称<input type=*"text"* name=*"item.name"* /></td>

<td><input type=*"submit"* value=*"查询"*/></td>

</tr>

</table>

商品列表：

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

<tr>

<td>选择</td>

<td>商品名称</td>

<td>商品价格</td>

<td>生产日期</td>

<td>商品描述</td>

<td>操作</td>

</tr>

<c:forEach items=*"*${itemList }*"* var=*"item"*>

<tr>

<td><input type=*"checkbox"* name=*"ids"* value=*"*${item.id}*"*/></td>

<td>${item.name }</td>

<td>${item.price }</td>

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

<td>${item.detail }</td>

<td><a href=*"*${pageContext.request.contextPath }*/itemEdit.action?id=*${item.id}*"*>修改</a></td>

</tr>

</c:forEach>

</table>

</form>

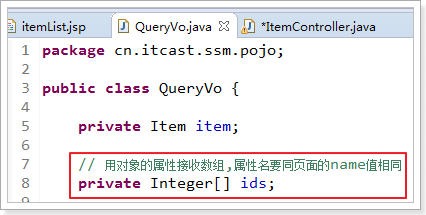
页面选中多个checkbox向controller方法传递

本身属于一个form表单，提交url是queryItem.action

### Controller

Controller方法中可以用String[]接收，或者pojo的String[]属性接收。两种方式任选其一即可。

定义QueryVo，如下图：



ItemController修改queryItem方法：

/\*\*

\* 包装类型 绑定数组类型，可以使用两种方式，pojo的属性接收，和直接接收

\*

\* **@param** queryVo

\* **@return**

\*/

@RequestMapping("queryItem")

**public** String queryItem(QueryVo queryVo, Integer[] ids) {

System.***out***.println(queryVo.getItem().getId());

System.***out***.println(queryVo.getItem().getName());

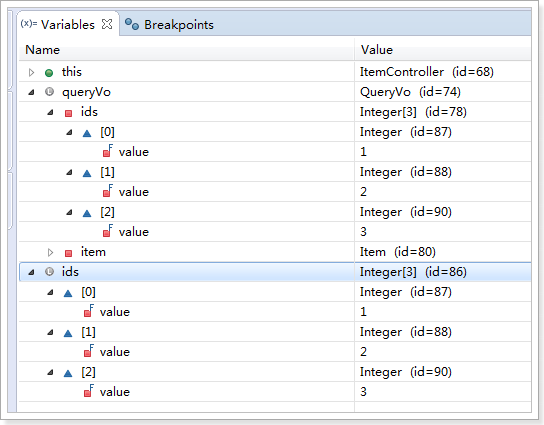
System.***out***.println(queryVo.getIds().length);

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

**return** "success";

}

效果，如下图：



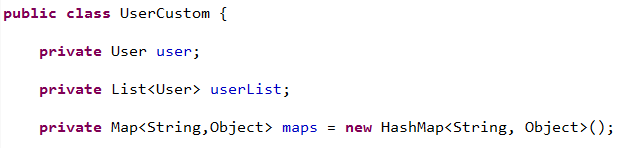
# 集合类型（这里是使用包装类）

## 将表单的数据绑定到List

### 传统

#### 简述

定义包装类





#### 需求

实现商品数据的批量修改。

#### 开发分析

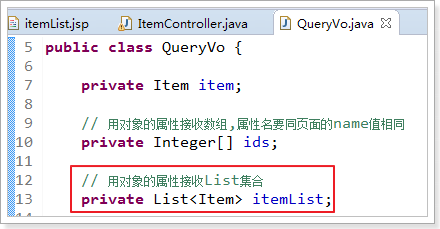
开发分析

1. 在商品列表页面中可以对商品信息进行修改。
2. 可以批量提交修改后的商品数据。

#### 定义pojo

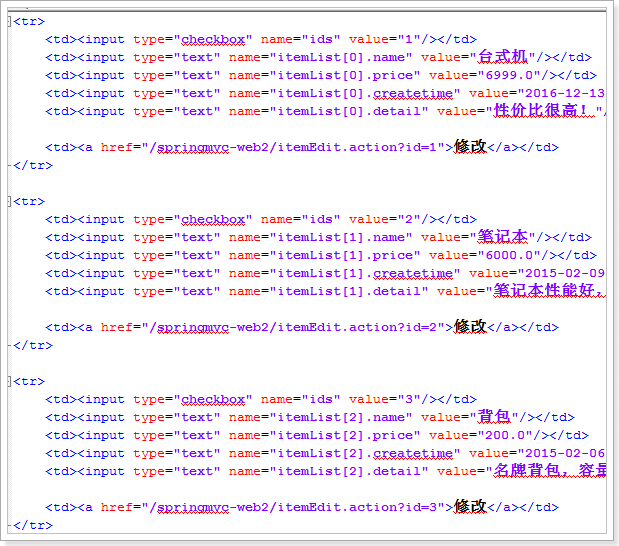
List中存放对象，并将定义的List放在包装类QueryVo中

使用包装pojo对象接收，如下图：



#### Jsp改造

前端页面应该显示的html代码，如下图：



分析发现：name属性必须是list属性名+下标+元素属性。

Jsp做如下改造：

<c:forEach items=*"*${itemList }*"* var=*"item"* varStatus=*"s"*>

<tr>

<td><input type=*"checkbox"* name=*"ids"* value=*"*${item.id}*"*/></td>

<td>

<input type=*"hidden"* name=*"itemList[*${s.index}*].id"* value=*"*${item.id }*"*/>

<input type=*"text"* name=*"itemList[*${s.index}*].name"* value=*"*${item.name }*"*/>

</td>

<td><input type=*"text"* name=*"itemList[*${s.index}*].price"* value=*"*${item.price }*"*/></td>

<td><input type=*"text"* name=*"itemList[*${s.index}*].createtime"* value=*"*<fmt:formatDate value="${item.createtime}" pattern=*"yyyy-MM-dd HH:mm:ss"*/>*"*/></td>

<td><input type=*"text"* name=*"itemList[*${s.index}*].detail"* value=*"*${item.detail }*"*/></td>

<td><a href=*"*${pageContext.request.contextPath }*/itemEdit.action?id=*${item.id}*"*>修改</a></td>

</tr>

</c:forEach>

${current} 当前这次迭代的（集合中的）项

${status.first} 判断当前项是否为集合中的第一项，返回值为true或false

${status.last} 判断当前项是否为集合中的最

varStatus属性常用参数总结下：

${status.index} 输出行号，从0开始。

${status.count} 输出行号，从1开始。

${status.后一项，返回值为true或false

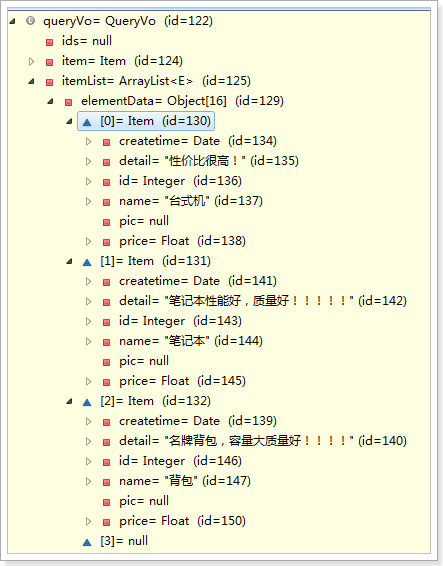
begin、end、step分别表示：起始序号，结束序号，跳跃步伐。

#### 效果

这里只演示List的绑定，能够接收到list数据。

可以拿到数据即可，不做数据库的操作。

测试效果如下图：



**注意：**接收List类型的数据必须是pojo的属性，如果方法的形参为ArrayList类型无法正确接收到数据。

## 接受map

### 传统

