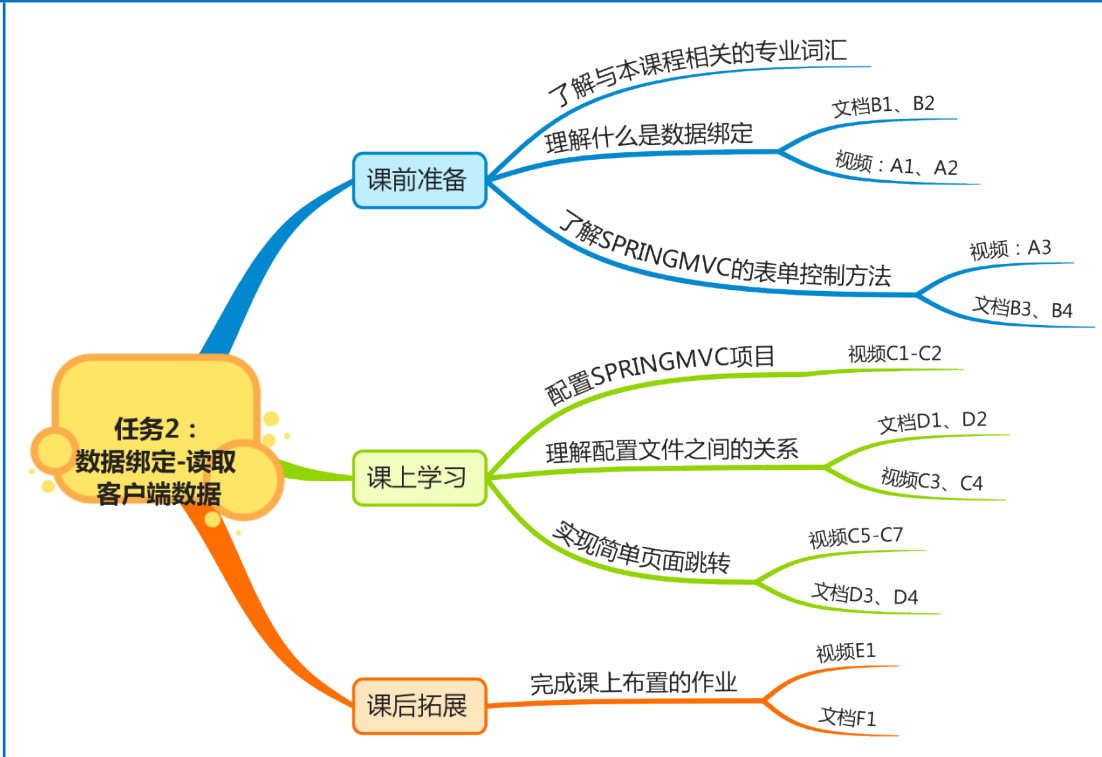
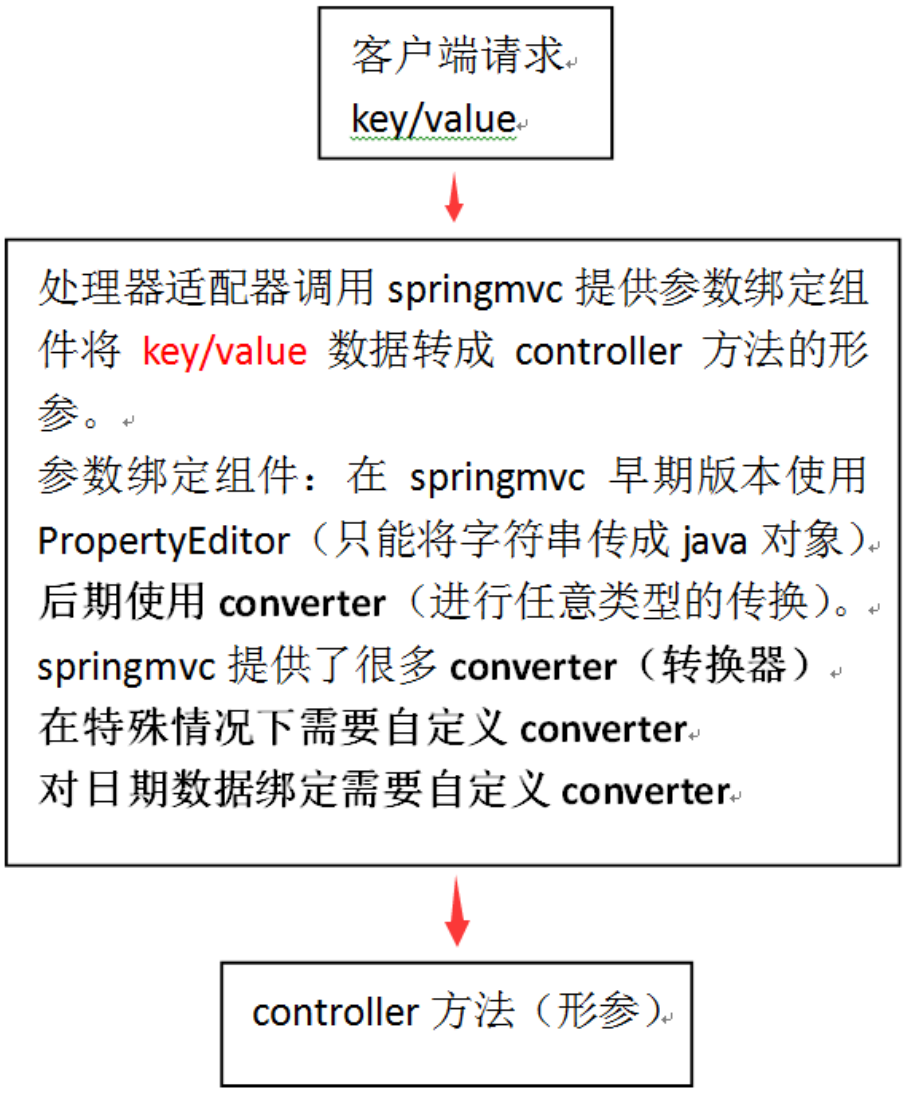
单元 4.1.2-数据绑定获取请求

课程导入		
同学们，从本单元开始， 我们将进一步学习 springMVC。springMVC 作为一个前端框架，它首先要解决数据传输的问题，在 JSP 中，我们需要用 request 对象来获取来自表单和 URL 中的信息，这种复杂的重复的操作占用了我们大量的时间，springMVC 能够快速获取来自客户端的信息，还可以自动封装成对象，为我们的开发带来了巨大的便利。		
知识目标	**能力目标**	**素质目标**
1-了解 Spring MVC 的工作原理，掌握 Spring MVC 应用的开发步骤。	1.能够绑定普通数据类型 2.能够将数据自动绑定到 JavaBean 3.能够自动绑定数据到数组、Map、List	1.培养学生的团队意识和团队协作精神，锻炼学生的沟通交流能力; 2.通过项目教学，让学生真切的体验项目分析、设计、管理及实施的全过程;
学习任务	**重点难点**	**突破方法**
用 SpringMVC 数据绑定技术实现一个比较复杂的注册功能	1.数据自动绑定到 JavaBean 2.数据绑定到数组、Map、List	采用翻转课堂、项目导入的教学模式，进行分组讨论、演示动画原理。
思维导图：		

任务 1

了解 springMVC 参数绑定的类型

我们可以回忆一下，在 struts2 中，是通过在 Action 中定义一个成员变量来接收前台传进来的参数，而在 springmvc 中，接收页面提交的数据是通过方法形参来接收的。从客户端请求的 key/value 数据，经过参数绑定，将 key/value 数据绑定到 controller 方法的形参上，然后就可以在 controller 中使用该参数了。来看一下这个过程：



所以我们知道，是 springmvc 提供了很多转换器来将页面参数绑定到 controller 方法的形参上，关于自定义 converter，我下面会提到。大概了解了该过程后，下面开始做具体的总结。

默认支持类型如下：

springmvc 中，有支持的默认类型的绑定。也就是说，直接在 controller 方法形参上定义默认类型的对象，就可以使用这些对象。

HttpServletRequest 对象

HttpServletResponse 对象

HttpSession 对象

Model/ModelMap 对象

在参数绑定过程中，如果遇到上面类型就直接进行绑定。也就是说，我们可以在 controller 的方法的形参中直接定义上面这些类型的参数，springmvc 会自动绑定。这里要说明一下的就是 Model/ModelMap 对象，Model 是一个接口，ModelMap 是一个接口实现 ，作用是将 Model 数据填充到 request 域，跟 ModelAndView 类似，关于它的使用，我在下面和简单类型参数绑定一起说。

任务 2

简单数据类型的数据传递

总结这个还是以需求为例吧，这样比较容易理解，假设现在有个需求：根据商品的 id 来修改对应点商品信息。所以前台页面肯定要传进来该商品的 id，然后 springmvc 的 controller 进行处理，返回一个修改商品信息的页面。关于前台页面的东西都很简单，我就不贴代码了，主要部分截个图，具体的代码在文章最后有下载地址。

前台页面通过 url 将参数传递过来，请求的是 editItems.action。

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

下面写 controller 中的 editItems 方法：

```
@RequestMapping("/editItems")
public String editItems(Model model, Integer id) throws Exception {
    //根据 id 查询对应的 Items
    ItemsCustom itemsCustom = itemsService.findItemsById(id);

    model.addAttribute("itemsCustom", itemsCustom);

    //通过形参中的 model 将 model 数据传到页面
    //相当于 modelAndView.addObject 方法
    return "/WEB-INF/jsp/items/editItems.jsp";
}
```

这是个很简单的 demo，从上面的代码中可以看出 model 可以直接作为参数，springmvc 默认会绑定它，然后使用 model 将查询到的数据放到 request 域中，这样就可以在前台页面取出该数据了。

要注意一点的是，简单类型的绑定中，方法形参中的参数名要和前台传进来的名一样才

能完成参数的绑定。那有人要问了，如果有特殊需求（比如更好的可读性？），这里定义的参数名就是不一样，那咋整呢？有解决办法么？有！我们可以使用注解@RequestParam 对简单的类型进行参数绑定，如下：

```
@RequestMapping("/editItems")
public String editItems(Model model, @RequestParam(value="id", required=true) Integer items_id) throws Exception {
    //根据id查询对应的Items
    ItemsCustom itemsCustom = itemsService.findItemsById(items_id);
```

所以说，如果不使用@RequestParam，要求 request 传入参数名称和 controller 方法的形参名称一致，方可绑定成功。如果使用@RequestParam，不用限制 request 传入参数名称和 controller 方法的形参名称一致。通过@RequestParam 中的 required 属性指定参数是否必须要传入，如果设置为 true，没有传入参数就会报错。

任务 3

普通 POJO 类型的数据传递

再来总结下 pojo 类型的绑定，继续上面的案例，当页面展示了商品详细信息后，我做了修改，然后点击提交，后台应该将我提交的这些参数全部更新到数据库的 items 表中，也就是说，我提交的这些参数要绑定到 Items 对象或者其扩展对象中。先看一下 Items 中都有哪些属性：

```
public class Items {
    private Integer id;

    private String name;

    private Float price;

    private String pic;

    private Date createtime;

    private String detail;
```

可以看到，有各种类型的属性，当我们提交后，要将这些属性全部封装到一个 pojo 中，然后去更新数据库。

绑定很简单，对于基本类型，要求页面中 input 标签的 name 属性值和 controller 的 pojo 形参中的属性名称一致，即可将页面中数据绑定到 pojo。也就是说前台页面传进来的 name 要和要封装的 pojo 属性名一模一样，然后就可以将该 pojo 作为形参放到 controller 的方法中，如下：

```
@RequestMapping("/editItemsSubmit")
public String editItemsSubmit(HttpServletRequest request, Integer id, ItemsCustom itemsCustom) throws Exception {

    //调用service更新商品信息，页面需要将商品信息传到此方法
    itemsService.updateItems(id, itemsCustom);

//      return "redirect:queryItems.action";
//      return "forward:queryItems.action";
        return "/WEB-INF/jsp/success.jsp";
}
```

这样就能将前台表单传进来的不同属性值封装到 ItemsCustom 中了。但是运行一下就会发现报错，报错的信息是无法将 String 类型转换成 java.util.Date 类型，因为上面 Items 中有一个属性是 Date 类型的 createtime。这就需要我们自己定义转换器了，这也是这部分的重点，下面我们自己定义一个日期转换器：

```
//需要实现 Converter 接口，这里是将 String 类型转换成 Date 类型
public class CustomDateConverter implements Converter<String, Date> {


    @Override
    public Date convert(String source) {
        //实现 将日期串转成日期类型(格式是 yyyy-MM-dd HH:mm:ss)
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");

        try {
            //转成直接返回
            return simpleDateFormat.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;

    }
}
```

定义好了转换器后，需要在 springmvc.xml 中进行如下配置：

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


<!-- 自定义参数绑定 -->
<bean id="conversionService" class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
    <!-- 转换器 -->
    <property name="converters">
        <list>
            <!-- 日期类型转换 -->
            <bean class="ssm.controller.converter.CustomDateConverter"/>
        </list>                                                    刚刚定义的
    </property>
</bean>
```

现在就可以了，springmvc 就能根据这个转换器将 String 类型正确转换成 Date 类型，然后封装到 ItemsCustom 中去了。

这里说一个小小的插曲：修改商品详细信息后提交，可能会有中文乱码问题，表达提交都是 post 方式，springmvc 中关于 post 方式的中文乱码问题可以在 web.xml 中配置一个过滤器来解决，如下：

```
<!-- 过滤器解决post乱码问题 -->
<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>
```

任务 4

包装 POJO 类型的数据传递

这个包装类型 pojo 与上面普通的 pojo 有啥区别呢？包装类型 pojo 指的是 pojo 中有另一个也是 pojo 的属性，即 pojo 套 pojo，为什么会设计这种 pojo 呢？在前面的博文中我也有提到，这种组合的设计方法对于后期程序的扩展很有用，比如复杂的查询条件就需要包装到这种包装类型中。

那么该如何绑定呢？有两个思路：

在形参中添加 HttpServletRequest request 参数，通过 request 接收查询条件参数。

在形参中让包装类型的 pojo 接收查询条件参数。

第一种方式就跟原始 servlet 差不多，这里使用第二种方法，我们传进来一个包装类型的 pojo。看一下这个包装类型的 pojo：

```
public class ItemsQueryVo {

    //原始的商品信息
    private Items items;

    //为了系统 可扩展性，对原始生成的po进行扩展
    private ItemsCustom itemsCustom;
```

这个包装 pojo 中还有一个 ItemsCustom 类，这个类继承了 Items 类，并且用来扩展与 Items 相关的 User 对象中的相关信息。所以这个 ItemsCustom 中有 name 属性，假如我们要想将前台传进来的 name 属性封装到 ItemsCustom 中的 name 属性中，该如何传入呢？这就是包装类型的 pojo 参数绑定问题。

很简单，在前台我们可以通过这种方式来传：

```
<td>商品名称：<input name="itemsCustom.name" /></td>
```

然后 controller 中方法的形参传入包装类型的 pojo，即 ItemsQueryVo，打个断点，即可查看值有没有传进来。如下：

```
@RequestMapping("/editItemsAllSubmit")
public String editItemsAllSubmit(Model model, ItemsQueryVo itemsQueryVo) throws Exception {

    //打个断点，即可查看itemsQueryVo中itemsCustom是否已经接收到了前面传来的name
    List<ItemsCustom> itemsList = itemsService.findItemsList(itemsQueryVo);
    model.addAttribute("itemsList", itemsList);
    return "/WEB-INF/jsp/items/itemsList.jsp";
}
```

这样就能根据用户传进来的参数，进行复制的查询操作了。

任务 5

数组类型的数据绑定

数组的绑定指的是前台传来多个同一类型的数据，我们在 controller 中使用数组形参来接收前台传来的数据。还是以案例来驱动这部分内容，比如现在我们要批量删除商品，那么我们需要勾选好几个商品，这些商品都有 id 号，在 controller 中，我们需要将这些 id 号全部获取并放到一个数组中，然后再根据数组中的 id 号挨个删除数据库中对应的项。那么该如何绑定呢？其实也很简单，如下：

controller 的方法定义为：

```
@RequestMapping("/deleteItems")
public String deleteItems(Integer[] items_id) throws Exception {

    //这里就不删了，因为数据我们后面还需要，在return前打个断点，看一下items_id中的值即可
    return "/WEB-INF/jsp/success.jsp";
}
```

对应前台传入的参数为：

```
<c:forEach items="${itemsList }" var="item">
    <tr>
        <td><input type="checkbox" name="items_id" value="${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>
```

这样就能将前台传入的多个 id 绑定到数组中，然后我们就可以从数组中拿出要删除的商品的 id 了。

任务 6

List 类型的数据绑定

通常在需要批量提交数据时，将提交的数据绑定到 list<pojo>中，比如：成绩录入（录入多门课成绩，批量提交），在这里我们假设有需求：批量商品修改，在页面输入多个商品信息，将多个商品信息提交到 controller 方法中，即一次性更新多个商品信息。

所以思路是在扩展类 ItemsQueryVo 中新添加一个 List<ItemsCustom>，然后将不同商品的信息都存到这个 List 中，所以修改如下：

```
public class ItemsQueryVo {

    //原始的商品信息
    private Items items;

    //为了系统 可扩展性，对原始生成的po进行扩展
    private ItemsCustom itemsCustom;

    private List<ItemsCustom> itemsList;
```

controller 方法的定义：

1、进入批量商品修改页面

2、批量修改商品提交

所以 controller 中应该有两个方法，如下：

```
// 批量修改商品页面，将商品信息查询出来，在页面中可以编辑商品信息
@RequestMapping("/editItemsQuery") 用来通过浏览器url输入的
public ModelAndView editItemsQuery(HttpServletRequest request,
        ItemsQueryVo itemsQueryVo) throws Exception {

    // 调用service查找 数据库，查询商品列表
    List<ItemsCustom> itemsList = itemsService.findItemsList(itemsQueryVo);

    // 返回ModelAndView
    ModelAndView modelAndView = new ModelAndView();
    // 相当于request的setAttribut，在jsp页面中通过itemsList取数据
    modelAndView.addObject("itemsList", itemsList);          跳转到显示所有商品的页面

    modelAndView.setViewName("/WEB-INF/jsp/items/editItemsQuery.jsp");

    return modelAndView;

}

@RequestMapping("/editItemsQueryResult")  更新完后跳转的url
public String editItemsQueryResult(ItemsQueryVo itemsQueryVo) throws Exception {
    //下面打个断点，进来后看看itemsQueryVo中的List<ItemsCustom>属性有没有正确接收参数
    return "/WEB-INF/jsp/success.jsp";
}
```

前台 jsp 页面中是如何传入参数的呢？这是我们所关心的问题，因为后台形参中接收数据用的就是包装类 ItemsQueryVo。看下面：

```
<c:forEach items="${itemsList }" var="item" varStatus="status">
<tr>    包装类中的list属性名        下标        对应list中的pojo中的属性名

    <td><input name="itemsList[${status.index }].name" value="${item.name }"/></td>
    <td><input name="itemsList[${status.index }].price" value="${item.price }"/></td>
    <td><input name="itemsList[${status.index }].createtime" value="<fmt:formatDate value
    <td><input name="itemsList[${status.index }].detail" value="${item.detail }"/></td>
```

所以我们知道了，前台是通过类似于 list[i].name 这种形式来传递参数的。list[i]表示第 i 个 ItemsCustom，然后 list[i].属性 表示第 i 个 ItemsCustom 中对应的属性。

任务 7	**Map 类型的数据绑定**
	
	Map 的绑定其实和 List 的绑定是差不多的，首先也是在包装的 pojo 中新添加一个 Map 类型的属性，如（我随便举个例子，跟本例无关了）
	`Public class QueryVo {`
	`private Map<String, Student> itemInfo = new HashMap<String, Student>();`
	`//get/set 方法..`
	`}`
	
	关键是前台传参的时候和 List 不太一样，Map 是这样传的，比如：
	`<tr>`
	`<td>学生信息：</td>`
	`<td>`
	`姓名：<input type="text" name="itemInfo['name']"/>`
	`年龄：<input type="text" name="itemInfo['price']"/>`
	`.. .. ..`
	`</td>`
	`</tr>`
	
	我们可以看到，Map 的参数绑定传来的是 Map 中的 key，然后 value 会自动绑定到 Map 中的那个对象的属性中。在 controller 中的方法里，形参就直接使用 QueryVo 接收即可，也很简单。
	关于 springmvc 的参数绑定基本就总结到这了，其实原理都差不多，只是针对于不同的类型，绑定的方式有些区别而已，多想想多写写，基本就能掌握这些了。