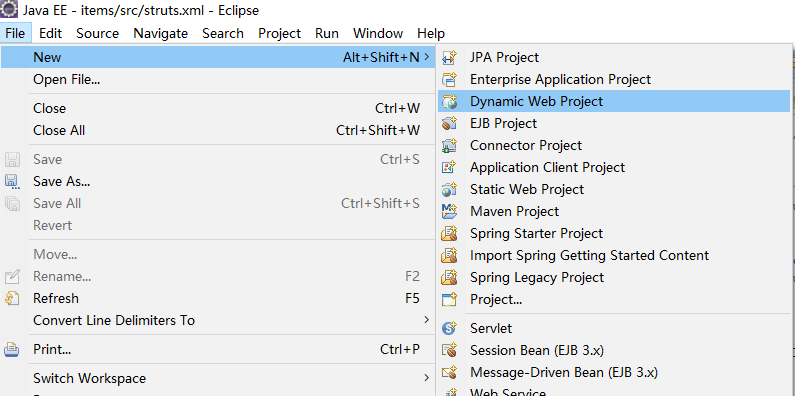
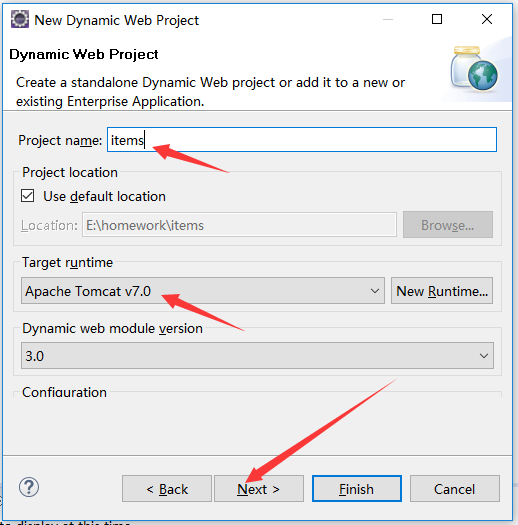
# 商品管理作业 - Step By Step

### 第一步，创建项目，搭建基本环境

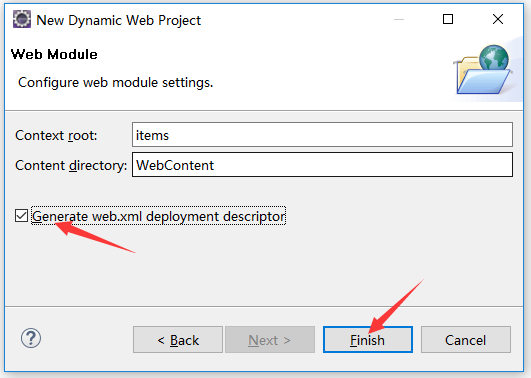
1. 新建动态 Web 工程，快捷键 Ctrl-N。



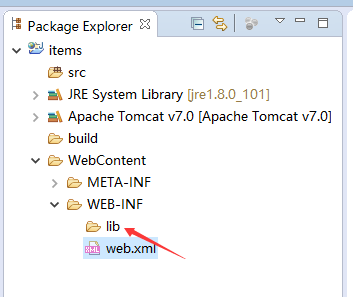
2. 填写项目名字 items ，选择你的 tom 猫。点击 Next。



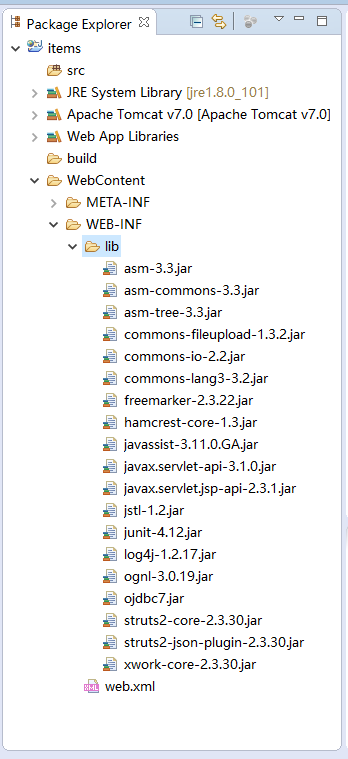
3. 注意，在新建项目的时候，把生成 web.xml 的选项勾上，如下图。

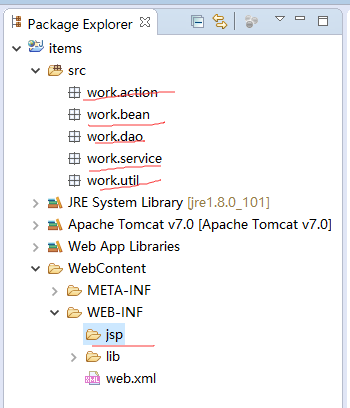


4. 这是生成的项目结构。接下来，将使用的 jar 包复制到 WebContent/WEB-INF/lib 文件夹。



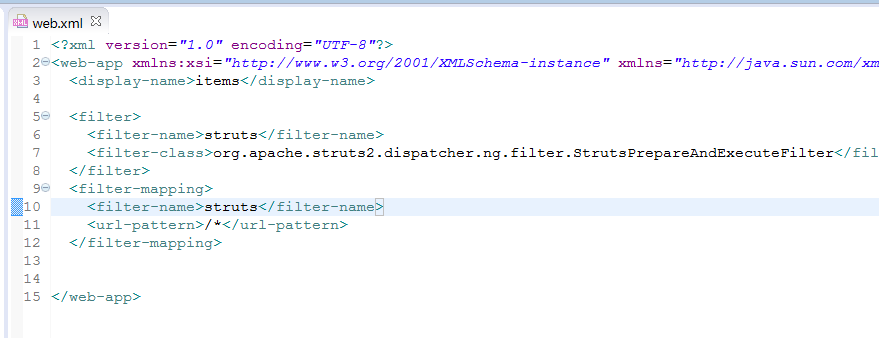
看起来是这样的：

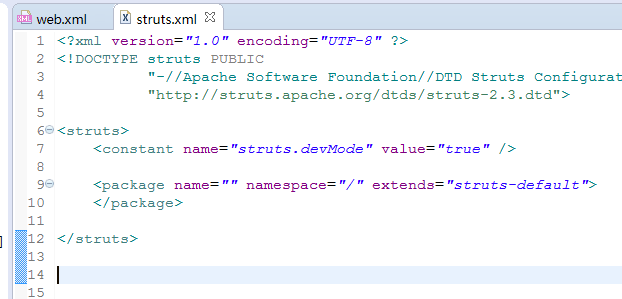
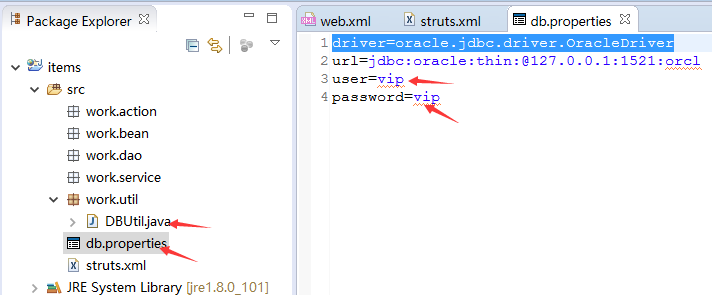


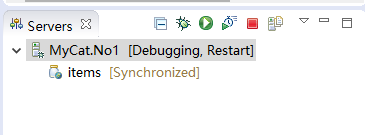
5. 在 src 目录下，创建我们要用到的package；在 WEB-INF 目录下，创建 jsp 文件夹。

### 第二步，为项目添加 struts 支持

1. 在 web.xml 中增加过滤器，如下所示。（注意，不要写错类名，不要写错对应。最好直接从现有的项目中拷贝）

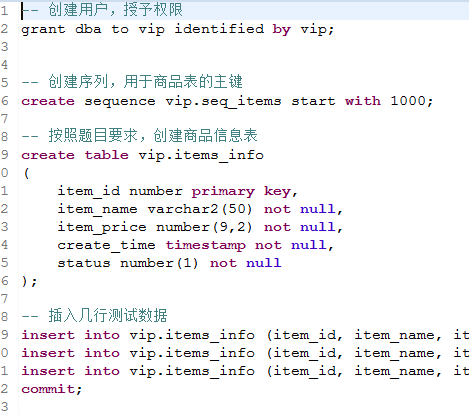


1. 在 src 目录下增加 struts.xml（不要写错头部。）  
   
2. 将我们之前写的 DBUtil.java 复制到 src/works/util 下面，将 db.properties 放在 src 下，并修改 db.properties 里面的数据库用户名、密码，要跟你自己的数据库对应。  
   
3. 确保以上各步完成后，创建 server，将程序部署到上面，启动tom猫。如果console 没有输出错误信息，那么恭喜，环境已经搭建起来了。如果有错误信息，检查以上各步，排查问题。

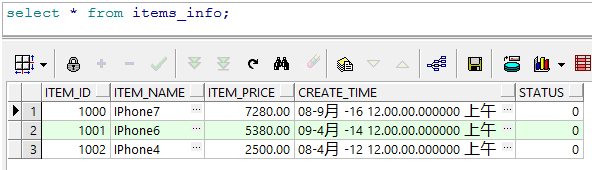


### 第三步，数据库表的创建

1. 在 Oracle 中建立用户，并创建相应的表 items\_info，插入测试数据，语句如下：

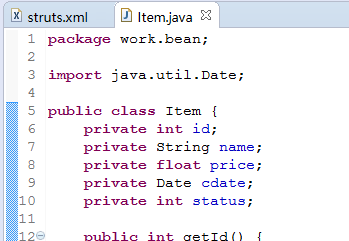


这是建完的表：



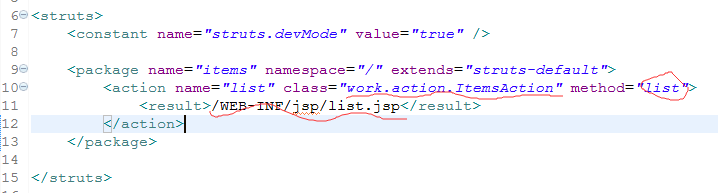
### 第四步，实现功能：显示商品列表

1. 首先，我们需要创建一个类work.bean.Item，来表示每个商品的信息。



它是一个简单的 javabean，它的结构，跟数据库基本上是对应的。

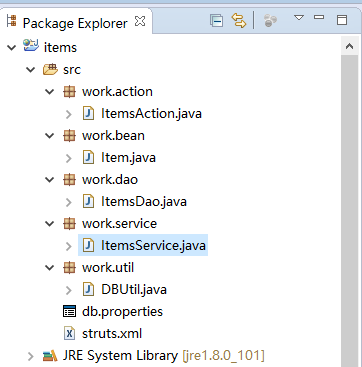
1. 在我们的 struts.xml 中，创建package，添加显示商品列表的 action：

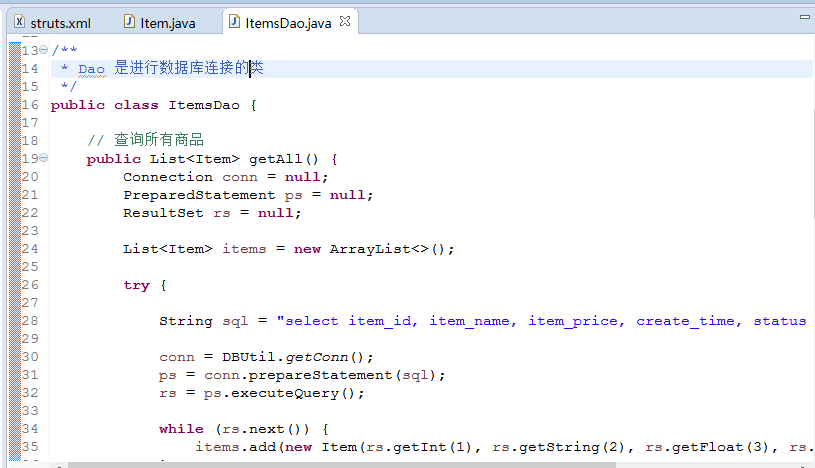


这样，我们就可以通过 localhost:8080/items/list.action 访问这个功能了。

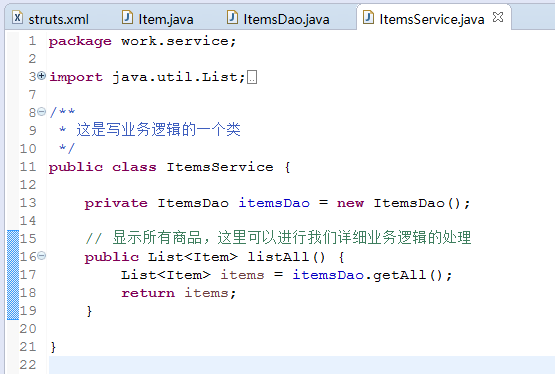
业务逻辑在 work.action.ItemsAction.list() 方法里面处理，

返回的数据，用 /WEB-INF/jsp/list.jsp 来呈现。

1. 创建 action/service/dao 类。为了写我们的 action.list()，我们需要首先完成相应的 dao 和 service 方法：  
   
2. 在 dao 类中，实现我们的 getAll() 方法：



1. 在 service 类中，实现我们的业务逻辑，写我们的 listAll() 方法：



1. 现在，轮到我们的 action.list() 了。



1. 最后，搞定用于显示的 jsp 页面：/WebContent/WEB-INF/jsp/list.jsp。



1. 重启服务器，在浏览器中查看，看到了吗？恭喜你，我们这个功能就完成了。

### 第五步，实现功能：查询上下架

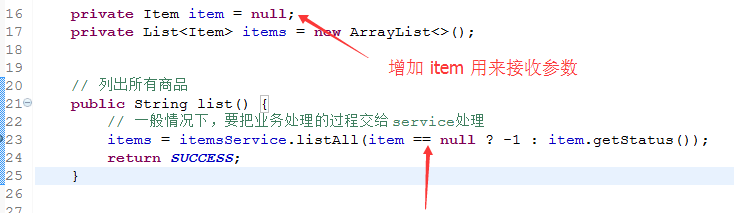
1. 就是这个，在下拉框中选择上架或下架，下面的结果集相应显示上架或下架的商品：



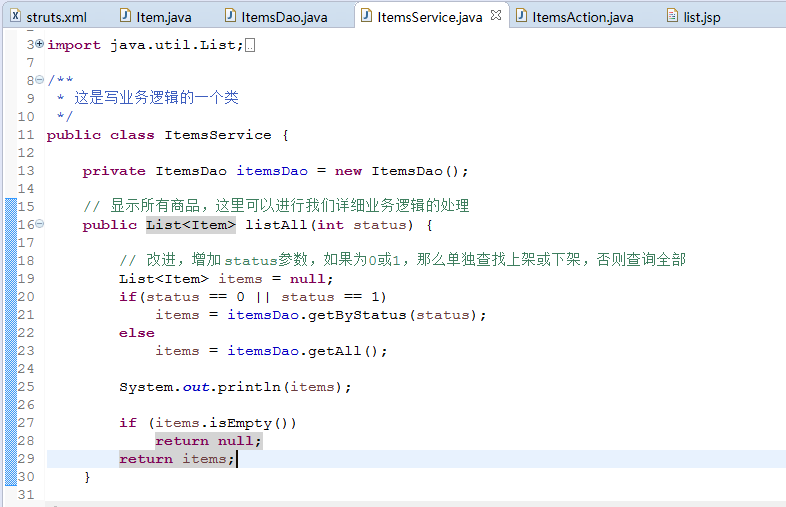
1. 在 list.jsp 中，上面下拉框的关键代码为：



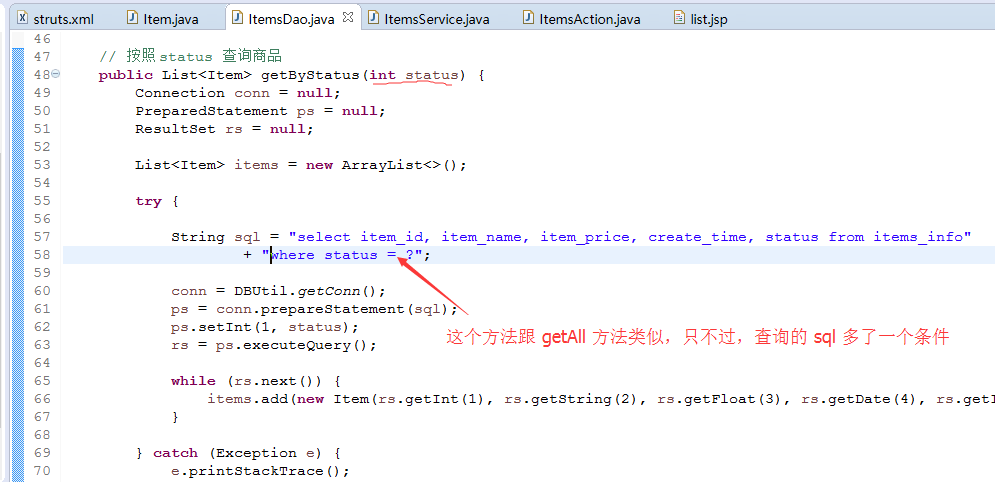
1. 我们不需要写新的 action，只需要修改之前的 action.list 就可以了。像下面所示，为service.listAll() 增加一个参数，即商品状态。如果调用的时候，传入0或1，表示我们只查询上架、下架的商品，如果传入的是其他数字，那么我们查询全部商品：



1. 当然，我们需要改进我们的 service.listAll() 方法为：

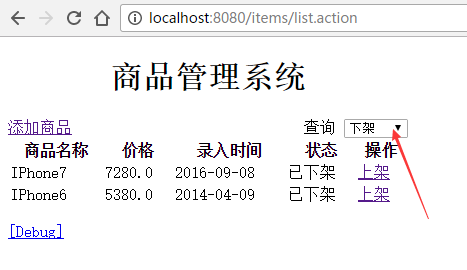


1. 对应的，在 dao 中，添加 getByStatus() 方法，用来查询上架或者下架的所有商品



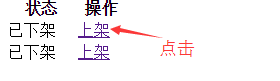
1. 在浏览器输入 localhost:8080/items/list.action，之后查询上架、下架，看一下效果。

然后我们这个功能又完成了。是不是很简单？

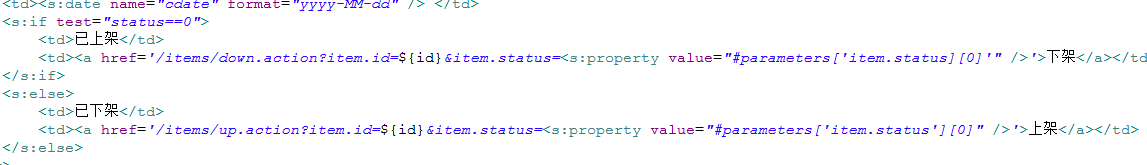


### 第六步，实现功能：将商品上架或下架

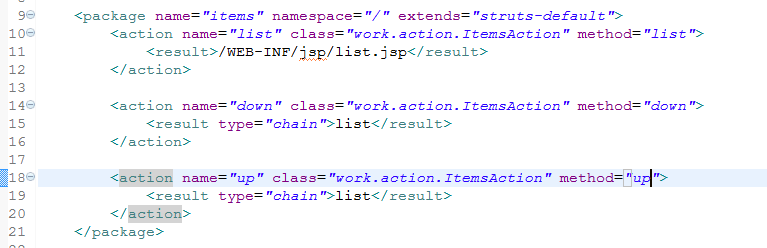
1. 在页面上，就是这里：



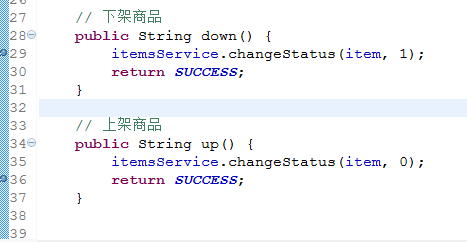
1. 在 list.jsp 中，代码是这样的：



1. 要实现功能，首先，在struts.xml 中增加 action：

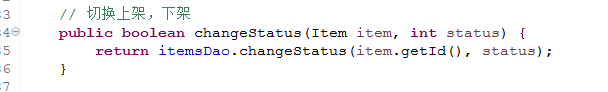


1. 实现我们的 action.down() 和 action.up() 方法：

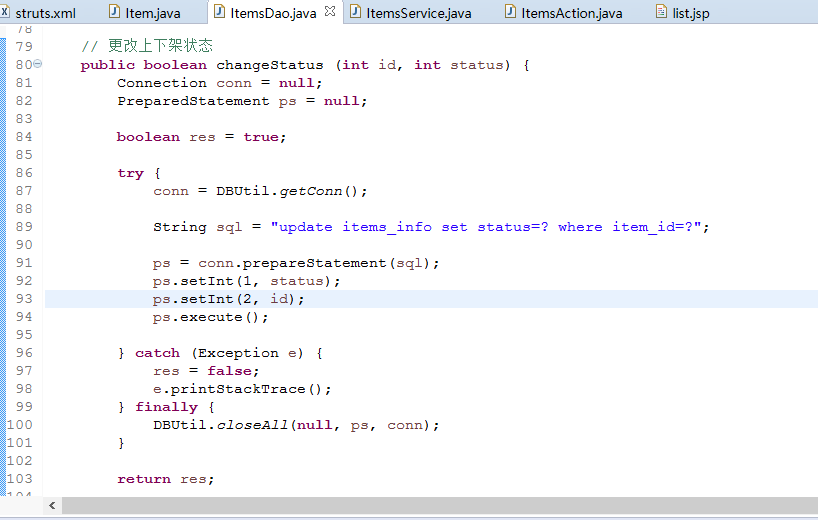


注意，两个方法都是通过调用 service.changeStatus() 实现的。因为本质上，这两个都是对商品信息表的状态进行更改，只是参数不同而已。所以，我们没必要写两个 service 方法。

1. 实现我们的 service.changeStatus() 方法：



1. 相应的 dao.changeStatus() 代码如下，是一个很简单的更新操作：

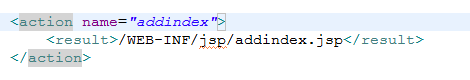


1. 完成以上代码，重新部署启动，我们的上下架功能就完成了。

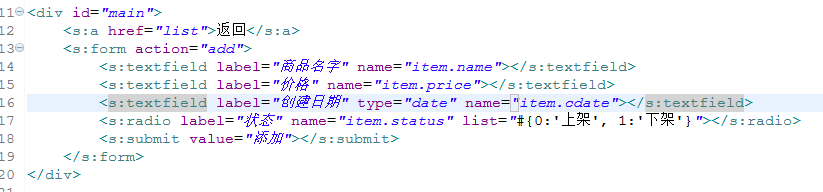
点击试试，然后下一个功能。

### 第七步，实现功能：添加新的商品

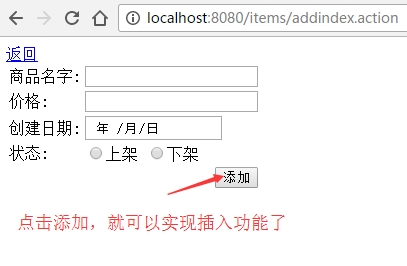
1. 首先，在 struts.xml 中增加一个 action，可以导航到增加商品的页面：



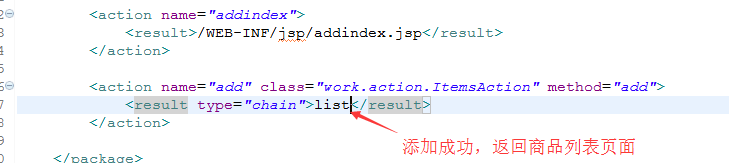
对应的 addindex.jsp，代码：



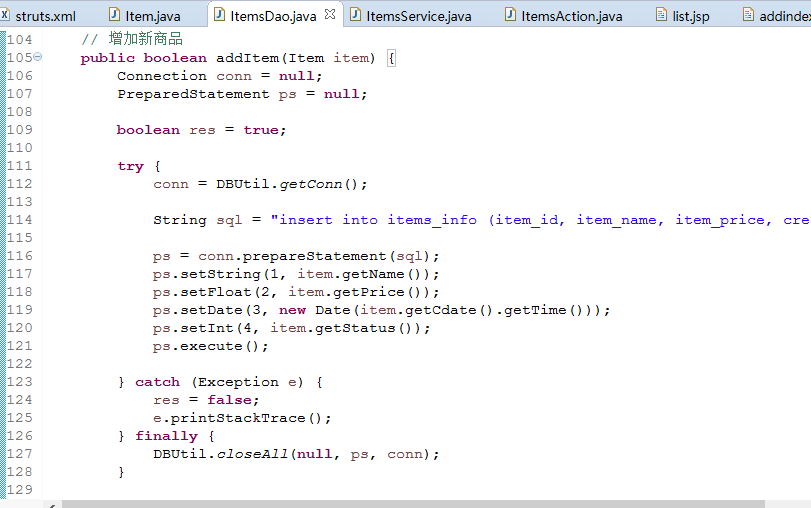
然后就可以输入 localhost:8080/items/addindex.action，访问下面页面了：



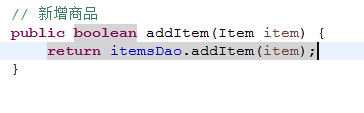
1. 上面的页面有了，我们现在开始“添加”按钮的功能实现。如果添加成功，我们要返回 list 页面，而且弹出一个添加成功的提示。
2. 需要首先在 struts.xml 中增加 action：



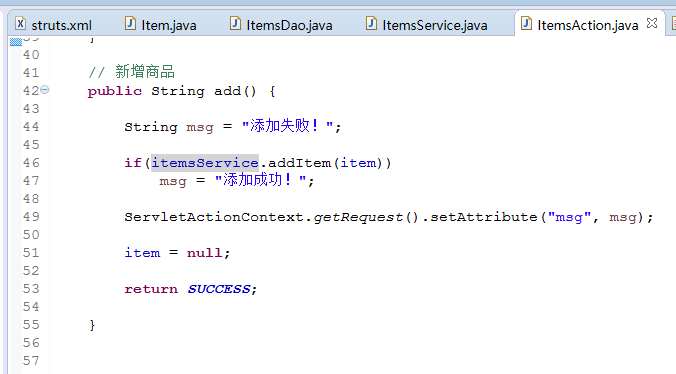
1. 实现我们的 action.add() 方法，当然，先完成我们的 dao 和 service:



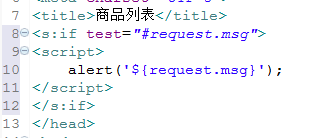
1. 这是我们的 service.addItem():



1. 然后，就是我们 action.add() 方法了，直接调用上面的 service.addItem()：



1. 最后的最后，对我们的 list.jsp 稍作修改，如果这个 jsp 是从添加商品返回的话，需要有弹出窗口。



因为在 action.add() 方法里，我们设置了 request.setAttribute(“msg”)，所以可以在这里取出来。判断如果非空，那么显示 alert 窗口

1. 这样，BINGO。该有的功能基本都有了。这就是我们的页面：

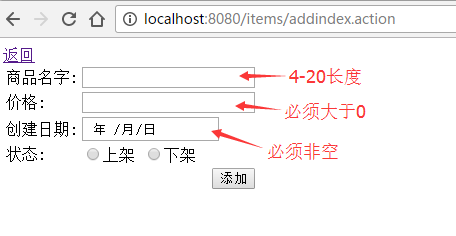


=============== 我是漂亮的分界线 ================

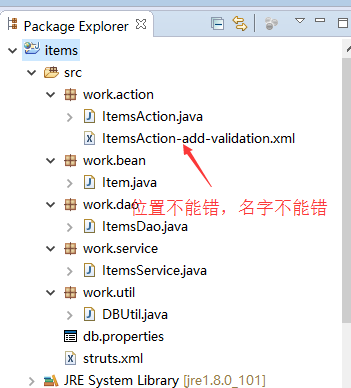
**上面的是最基本的功能实现，无论如何，必须要搞定。不会就问。搞懂后，再精进其他。**

=============== 我是漂亮的分界线 ================

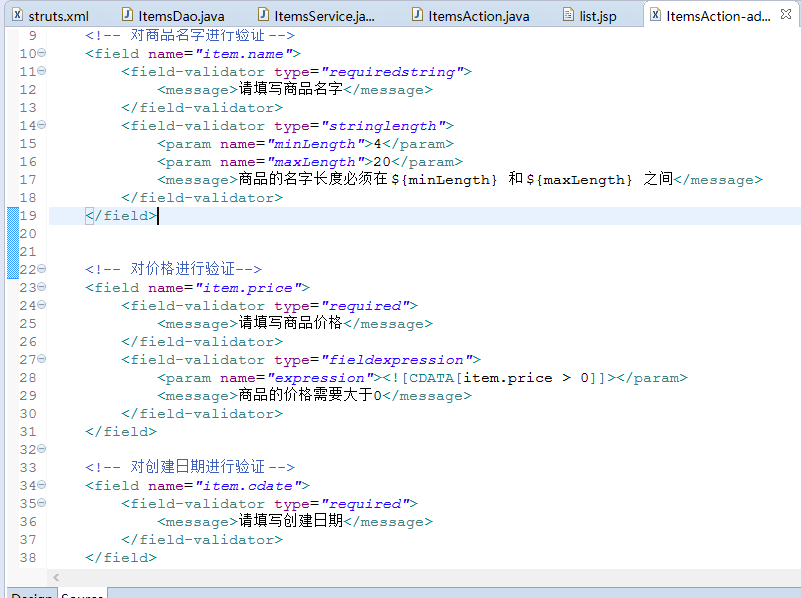
### 第八步，更多：为添加商品的请求增加表单验证。



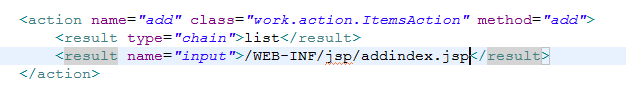
1. 创建 ItemsAction-add-validation.xml:



1. 编辑上面的验证文件，为每个字段增加相关验证：



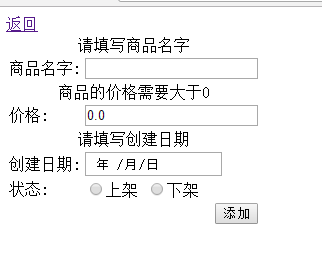
1. 在 struts.xml 中，为 add.action 增加 input，指向出错后返回的页面：



1. 必须要确保我们的 action 继承了 validatable 接口，或者继承 ActionSupport 类，否则验证不会生效：



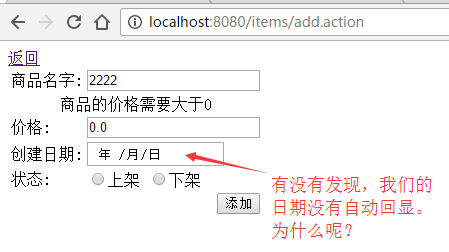
1. 好了，试一下，是不是这种效果：



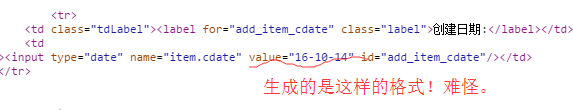
1. 现在，你可以尝试添加更多更严谨的验证。也可以为我们已经实现的其他的功能中的表单添加相关验证。

验证很重要，是保护我们网站安全重要的手段之一。

### 第九步，更多：自定义类型转换



看一下源码：

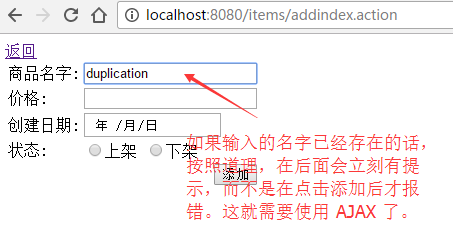


解决方案：

通过自定义类型转换，告诉 struts2，在自动对日期转型转化为字符串的时候，格式为 yyyyMMdd！

请大家实现。

### 第十步，更多：AJAX 的使用



### 尾声

**还有很多工作没做。**

比如：

1. 页面端需要用 js 进行初步验证。
2. 我们的页面要美化，相关的 css 渲染。
3. 如果有登录功能的话，需要有完善的**权限控制**。怎么通过拦截器实现。
4. 所有的表单提交，要避免出现**重复提交**。
5. 需要为我们的程序实现完善的**异常处理**机制。
6. 添加**国际化**资源文件，可以在不同国家地区显示不同语言。
7. 文件的**上传下载**功能。Result type=”stream” 的使用。
8. 其他。