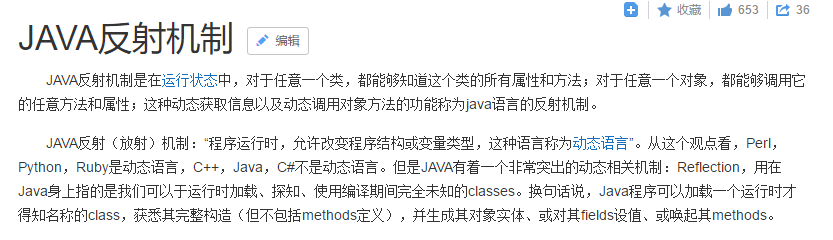
## 枚举

作用：限定取值范围

## 反射

原理：在类运行的过程中，对这个类进行操作，即jvm加载过程中



## 加载配置文件 加载器

1、类加载器

InputStream load=DBCPutils.class.getClassLoader().getResourceAsStream("xxx.properties");

返回配置文件的输入流 有些时候就需要传入加载文件后的流 没有获取字符串或其他的方法主要就是获取流

2、resourceBundle

Int num=Integer.parseInt(ResourceBundle.getBundle("xxx").getString("字段"));

不需要后缀，提供了很多方法，得到的是string，用integer的方法将字符串转成int 没有获取流的方法，各有所长

各有所长，看情况使用，需要流就用第一个，获得流就可以读文件照样可以获得字段，只是没有第二个方便；如果不需要流就用第二个’

## 形参和实参

形参就是没有赋值 只是定义了一个参数

实参就是赋了值以后 有具体的值就是实参

## WEB-INF下的资源

外部不能直接通过路径在浏览器访问，是受到保护的，只能是服务访问这个地址 如果是框架开发就是 通过action跳转才能访问，只要是服务器里面写的地址就可以，不是浏览器输入的就能访问

## Struts2的表单校验

1. 针对某个方法校验
   1. 针对方法校验指的是执行这个方法才会去执行这个校验
      1. 比如一点击注册页面 为什么他就可以执行校验呢 因为注册的时候一点提交按钮 就会去执行action中的 方法 而且会把参数都封装到请求 带过去 然后呢校验的时候自然就能获取到你提交的数据 并根据提交数据的属性名 提交的名字是页面中的名字 然后根据名字就行校验就好了
   2. 在方法所在类的包下创建一个xml配置文件（action所在包）

命名规范：action类名（方法所在类的类名）-方法对应的路径名（Struts配置文件中配置的路径名）-validator.xml

原理：Struts在执行的时候，最后加载的配置文件才是Struts.xml 之前会加载很多的配置文件 其中有个配置文件就定义了很多的拦截器，他会在加载到拦截器的配置时 去执行这些拦截器，当然只执行那些配置了并且是让他生效的拦截器，有些只是定义了，并没有引用，需要用必须自己在Struts.xml中去引用，其中就有一个拦截器是转来用来完成表单校验的（默认加载的 只要启动就会执行一大堆拦截器 多少个来着忘了，，反正他要执行）这个拦截器中就会找当前方法所在类的包路径下有没有 是根据表单校验规范命名的配置文件（因为每次执行action都是执行一个方法，所以，他直接找执行当前action的方法 所在类的包 也可以说是当前方法所在包吧 每一个请求只会执行一个方法 因为地址栏都是一个唯一的地址）然后就加载那个配置文件 因为是命名规范的配置文件才能找到，并且是制定必须在所在类的包下，所以直接加载配置文件，根据配置去找校验规则，并返回信息，完成校验

校验的是页面提交的数据，是根据页面的字段进行校验的，页面将字段都封装到请求中，拦截器就获取请求中的字段 然后进行校验 这个时候并不是封装到bean中人家还没执行action action是所以配置文件加载完毕 所以拦截器执行完才执行，也就是最后才执行的，一切都初始化，执行完才执行 校验出错就会跳转到input视图 并把错误信息封装到请求带到跳转页面 然后可以通过Struts标签获取错误信息<s:fielderror/> 一执行action，不需要你去调用这个校验器 它自动去调用并执行 在执行action

1. form表单提交数据的时候需要一个action中的一个方法来处理 ，那么Struts2的校验就是校验这个方法（针对方法校验是这样的 如果是其他的有一种配置是所有方法都校验 是配置文件所在包下的所有方法 当然一般不会这样写 只需要登录注册校验就好了）其中的name属性就是你要校验字段的字段名 校验出错就会跳转到input视图 有很多校验都跳到input是不可能的 所有要指定每个校验的input name就在校验方法上面加上一个注解就好了 校验的方法就是配置文件名字规范上有 并且校验的方法也是你表单提交处理数据的那个action的方法 跳转到那个视图并会把错误信息带过去 通过Struts2标签获取就好了
2. 带数据 都是你这个action执行以后跳转到哪个页面 那么这个页面就会得到你这个action传过去的数据 如果那个页面又去执行一个action再去跳转到一个页面 也会把数据带到跳转的页面 如果还要执行action以此类推
3. 没必要非要知道 你知道怎么用就好了 有些东西知道就会 用的多了就好了
4. 页面发送一个请求—>调用action中的方法🡪执行这个方法前执行一大堆拦截器 其中一个是校验的拦截器🡪请求中有封装的表单数据🡪校验器根据得到的表单数据的字段进行指定规则的校验🡪校验成功执行action中的方法 失败跳转到input视图并把错误信息带过去 可以通过filederror标签获取
5. 内置拦截器完成校验，拦截器是在执行某个方法之前 先进行拦截
6. 校验失败会跳转到input视图

## 指定input视图

校验出错会跳转到input视图，如登录和注册都要校验，出错都要跳到input，但是一个需要跳到登录页面，一个是注册页面，这时候一个input是不行的，所以需要指定

如：校验登录方法时出错，跳转到哪个，注册又是哪个，可以如下设置



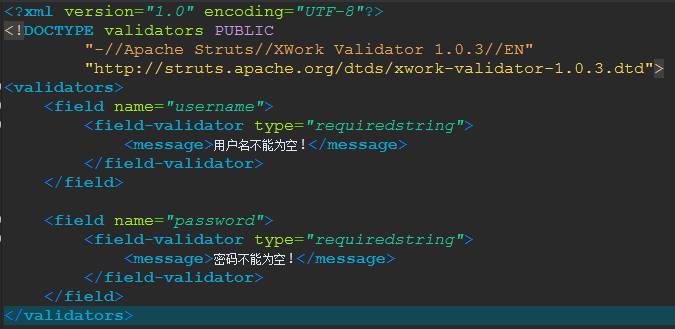
在被检验方法上，如果出错，可以添加这个注解来指定，名称随意，但是既然是input的指定，一般都是以什么Input来命名的

或许除了校验，还有其他的时候也需要跳转input，应该也是执行某个方法，或怎样就跳的吧，应该也是基于方法的，所以只要是跳input就可以在对应的方法上加指定

## Struts2的信息提示与回显

1. Struts2表单校验出错
   1. 设置信息，在校验的xml配置中，出错就会带信息

name是字段名

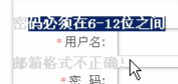


* 1. 信息显示



你也可以不指定回显哪个字段名字的错误，这里也是字段名

1. 如果不指定字段名显示的话 就会如下



所有的错误信息都是分行全部显示，如果指定，就可以将每一个信息都可以设置在任意位置，可以分开，而且样式也能更好的设置，否则就是一坨，所有错误信息全部都显示在了fielderror标签的位置

1. 在action中带信息显示
   1. 设置信息，在action中，调用父类actionsupport中的方法，因为继承了的









## AJAX通过连接发送请求是有缓存的

浏览器对同一个路径总是有缓存的，所以你异步请求 每次都是那个路径 有可能传的参数不同 也就是查询条件不同 查询不同的数据 但是浏览器有缓存 如果你第一次访问 肯定直接查 当时后面你再刷新 之类的再查询 如果你传了其他参数 不一定会查 而是查缓存中的数据 不会再去请求服务器 所以就有问题

就那实际举例 校验用户名是否存在 第一次输入zhangsan 提示已存在 第二次输入lisi 因为你这个是访问的同一个路径浏览器有缓存 他可能不会去数据库查lisi是否有 而是浏览器直接拿缓存中的数据 把查zhangsna 的结果返回来 所以要加一个时间戳

## 一对多的理解

就是一个我自己 对应多个我自己

一个我 对应多个我有不同的值

## 会话 session 和 cookie

作用：

使web资源处理的是每个用户各自的数据

什么是会话：

会话可简单理解为：用户开一个浏览器，点击多个超链接，访问服务器多个web资源，然后关闭浏览器，整个过程称之为一个会话。会话存储在服务器或客户端（根据cookie和session决定存在哪儿）的数据，也可以叫做缓存，这是通俗易懂的叫法

为什么不用request或servletContext来进行数据的保存

因为他们的各自的周期不能解决会话问题，就是处理每个用户各自的数据，这就是会话的含义 也可以说是会话所要解决的问题 处理每个用户各自的数据，所以就引出了专门处理会话问题的技术，就是会话技术 cookie和 session，专门用来处理每个用户各自的数据，不受干扰，如果是cookie浏览器关闭会话就结束，清除cookie缓存，虽然session也是浏览器关闭就结束，默认也是关闭就清除服务器端的session数据，但是可以控制，可以让他在服务器保存任意时间然后再销毁，因为服务器只有一个，很好控制，cookie是在用户的客户端的，不好控制

Session是服务器端的技术 将会话信息存储在服务器端 服务器有一个session区（session空间） 存储一个唯一的属于用户的一个标识的id

Cookie是属于客户端的技术 将将会话信息存储在用户浏览器所在的客户端的磁盘上 是以字符串拼接的方式存的

Cookie的响应头set-cookie:product=1，浏览器的内存会保存cookie的内容，再发送请求时，会带着头cookie:product=1去访问。之后累加。

Session在服务器端开辟一块空间，分配一个唯一的标识，每次请求响应都带着唯一的标识，最终结算通过标识找到空间。

为什么session实在request中去取的？

因为session数据是每次请求服务器的时候需要的数据 肯定实在请求中的啊 你不可能说响应一个session 响应是个客户的客户要session做什么 是服务器需要session 然后做一些操作 客户得到session有什么用 所以肯定不是响应 而是请求是带一个session交给服务器 然后服务器处理数据

## Hibernate默认的事务管理

Hibernate默认是不会提交事务的 需要在配置文件中设置字段 设置自动提交为true 默认是FALSE 实际开发中 事务管理也都是Spring管理 只是测试用用 实际不会让hibernate自动提交 默认是FALSE没有错

## Web项目路径说明

Src 🡪 指的是类路径

Webroot/webcontent 🡪 项目发布后的根路径 也就是在服务器的根路径 当前应用的根路径 整个项目的根路径

## Ognl的理解

**什么是ognl**

OGNL是Object-Graph Navigation Language的缩写，它是一种功能强大的表达式语言，是一个第三方开源项目。

Struts2通过使用OGNL简单一致的表达式语法，可以存取对象的任意属性，调用对象的方法，遍历整个对象的结构图，实现字段类型转化等功能。

GNL是个第三方的开源项目，其本身是与Struts2没有任何关系的

OGNL表达式的计算需要通过一个Map进行，而该Map有一个名称叫上下文context。这个context的Map中已经存放了若干对象，这些对象分为两类：根对象与非根对象。

**在Struts2中对ognl的支持**

Struts2中的值栈ValueStack是个接口，其实现类为OgnlValueStack。通过这个实现ognl的存储

**Struts2中值栈的组成**

打开OgnlValueStack类的源码可以看到，OgnlValueStack类中包含两个成员：root与context。

而原生的ognl只有一个对象这个对象里分了两个区域非根和根 而Struts2将根与非根进行了分离 成了两个对象 一个root（根）与context（非根）

**root的类型 （context是map从上图可以看出）**

属性root为CompoundRoot类型。打开CompoundRoot源码发现，该类型的本质为ArrayList。 

即CompoundRoot为增强的ArrayList，是一个栈：有截取栈元素、获取栈顶元素、弹栈、压栈操作。

ArrayList在存放数据时的特点是，ArrayList只能顺序存放数据，而无法为该数据命名。

****

存入的数据不是覆盖原有的 如已有0元素在栈顶（下标0） 再存入数据时0元素不会被覆盖而是到了下标1，新的数据到了栈顶下标为0，所以栈顶元素的下标始终为0

**获取值栈中数据的方法**

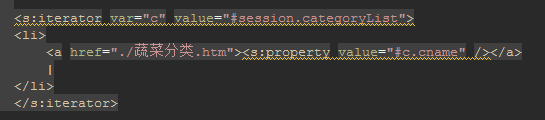
使用ognl获取 获取非根中的需要加一个# 根中的数据不需要加

## 页面获取服务器数据的方法（存我就不说了直接存就好了）

1. 存入session或者其他域对象 在页面通过在域对象中取数据 进行操作比如迭代显示之类的（通过Struts2提供的标签使用迭代器迭代） 虽然这个session也是在值栈中 但是是自动压栈的是非根中的对象需要#来取

存入session后每次访问这个数据只是第一次需要查数据库 后面都不需要了

一般存在session这些域中的数据都是一般不会变化的数据 或者比较小的 看实际情况吧 很好判断什么时候该用 但是又说不出所以然 直觉告诉我用这个就用这个

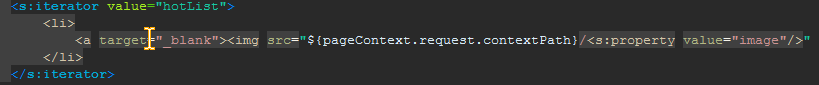


取数据 迭代时value中是ognl表达式 从非根中得到集合 获取值也是在非根中每个list在非根中，list中的元素也在非根中，list中每一个对象中的属性名，只不过是通过ognl表达式获取对象的属性

这里的list是存在值栈非根中的，并且还是在session中，所以要从session中取

1. 在类中定义一个全局变量 并提供get方法

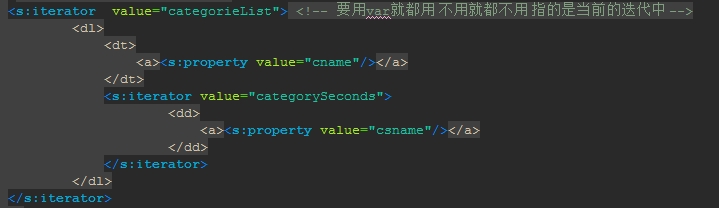
而定义全局变量的方式 每次访问这个变量都要查数据库



取数据 迭代时 value中直接是全局变量提供的get方法 也就是属性名 得到集合 获取值是list中每一个对象中的属性名

**不加var别名取数据** 直接得到list中存有对象的属性就好了 压栈的取法也和这个一样的

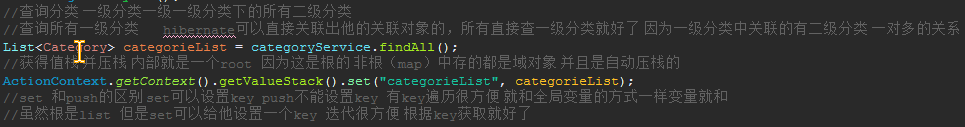




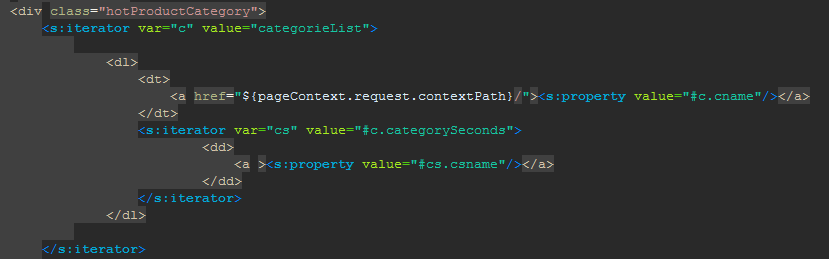
**加var别名取数据** 取的时候需要加上# 我也不知道为什么 反正从非根和加var取都要加#



1. 手动压栈（手动的是压到根中 list）默认取栈顶元素

有二级分类的集合 获取集合就可以得到二级分类类了

取数据和全局遍历的方法一样的 因为是在根中取 所以不用加# 就和全局变量一样的



如果这里不加别名var 就不需要# 取的时候直接写list中的属性名就好了 获取别名都要加# 还有获取非根也要加

这里的list是存在根中的，而且没有存入哪个容器，所以直接从值栈取就好了 不用点什么

## 服务器获取页面数据的方法

1. 实现模型驱动 modeldriven
   1. 要求：定义一个模型实现方法getmodel，并提供一个对象的定义 需要new出来
2. 属性驱动
   1. 要求：将要接收的数据定义为字段 并提供set方法 提供setget后面对应的就是属性
3. ognl的方式
   1. 要求：在页面的标签上写上对应对象点属性 在name属性上写 order.name
      1. 因为数据传到服务器 键是通过name来获取数据的

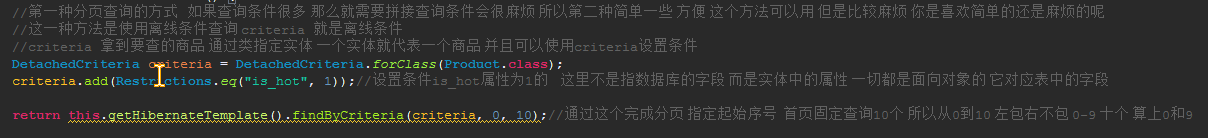
在action中提供一个对象的定义 不用new 提供getset方法，就会把数据封装到这个对象，为什么要提供get 因为拦截器需要获取这个对象 然后再反射set设置 将order中的name属性设置值

## SSH分页查询

注意：不能使用limit字段 除非你用sql语句，hql不支持的，那样要转换数据库肯定出错

只显示十个 使用分页实现(不能用sql中limit关键字 这样会导致切换数据库 报错 会有问题的 得用hibernate提供的分页查询方式完成 有两种方式） 字段is——hot 1热门 0非热门

法一、使用离线条件查询



法二、手动封装一个hibernatecallback实现类



## HQL的理解

Hql查询 from后面不是表的名字 是bean的类名 其中的条件也不是表的字段，而是对象中与表中字段映射对应的属性名，hibernate一切都是面向对象，除非你用sql方法查询，但是这样还不如不用hibernate

## List和set的区别

set 无序存放 元素不可重复

list有序存放 元素可重复

**为什么维护一对多用set而不用list？**

set元素不可重复 这是最重要的，一对多维护，多的一方确实多，但是实际中不可能出现重复的数据，这样就没有意义了，所以元素不可重复很重要，并且无序也挺不错的，数据库查询只需要查到数据就好，有没有顺序无所谓的，无序总感觉或许要好点，有序的话总感觉有限制一样，无序多爽，重要的还是不可重复，重复的数据的话，那样数据就没有意义了

因为list是有序的 所以hibernate会在表中给你添加一个字段，用于维护排序，添加了这个字段，你就需要在映射文件多配置一个东西，这不是我们所需要的

## hibernate一对多的维护

在实体类中不需要加数据表中维护的外键字段名 都是通过对象维护，在配置映射文件的时候，在关联对应的表中的外键字段名称

Hibernate会自动去找实体中维护的对象，然后得到数据库中的主外键约束，在根据你对配置的对应表中的字段，去看这个在表中是不是存在，然后再看是不是外键，是的话，就去找映射配置文件中配置的映射类中的主键，然后就会把关系维护上，反正都是通过对象维护的，配置文件中对应的外键约束字段，自动对应表中外键

## Struts2当前action执行成功跳转到另一个action

1、使用的是重定向的方式 地址栏发生改变

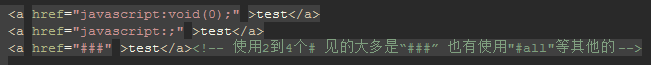


2、使用的是转发的方式 地址栏不会改变

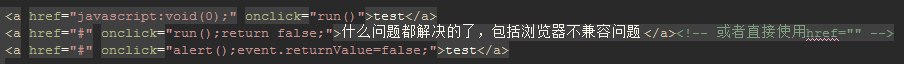
后面不能加.action 否则会报错



## 点击链接后不做任何事情



## 点击链接后，响应用户自定义的的点击事件



## Action为什么要继承ActionSupport

这个类中有很多的常量 方便使用 并且和Struts2的结合更加的好，对于Struts2内置的校验之类的也更方便 ，反正就是好处多多 基本没坏处 还有最重要的可以adderror 和message之类的

## 转发和重定向

转发只能是服务器内部的跳转 ，你要访问其他网站 其他服务器 就必须用重定向 跳转到另一个网站，转发重定向都是我响应给客户的，所以要用response中的





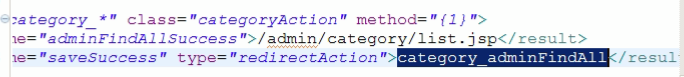
Redirect是重定向到页面（也可以到action 只是区分开了罢了）

Redirectaction 只能是重定向到一个action

## 关联数据的查询

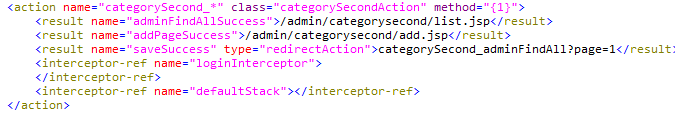
默认情况你关联的表的数据 是你用的时候才去查 但是一般你用的时候都没有session了，所以查了你 就要立即查处你关联的数据

## farm框架中页面的跳转



跳转都不是整个页面的 而是在当前的页面 之前只有一个页面 现在用的这个farm 一个界面就有三个页面 你跳转只会在当前的这个结构上跳转 当前的页面上跳转和各种操作 可以狭义的理解为 下载这个网页有有三个窗口 只刷新或者更改其中的一个窗口 其他的都不会变

## 在result配置中的路径URL也可以拼接数据的



## Hibernate删除的时候 对象不完整也可以删除的，最主要的只要你这个对应的实体对象有id就好

## 通过主键id查询数据时记得用get或load 一般都用get 因为是立即加载 不会出错

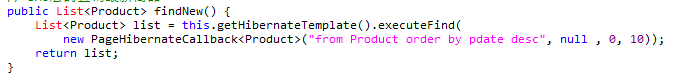
## 隐藏字段也是input只要在表单中 他也会提交到action 和普通的表单没有区别 只是在页面他没有显示罢了

## 数据库中都是用唯一的标识操作 所以要修改之类的都是用id进行操作的 要修改传传数据也是用id的

## 服务器获取数据 都是标签中的value属性的值 只有input才会自动封装数据 其他标签如select需要手动获取一下

## 级联删除 必须先查出这个对象 他就会一并查出他关联的对象 只有查了以后再删才能级联删除 因为你不查 维护的那个集合是空的，就级联不上了，维护的数据也就删不了了 删的是空数据 什么都没删咯

## 关于数据的排序的问题

1、

这里你查询后返回的是list是有序的 只要返回就是有序的 所以可以直接在hql中排序

2、

这是在映射文件的维护关系的set上的配置 因为没有单独查出这个数据 没有维护成list 所以set是无序的 就必须要在配置文件上设置他查出的set的结果数据 按什么字段排序

## Struts2的遍历的个数 记录循环的信息的属性 序号





在迭代时，加上status属性就可以了，这个是记录循环的一些信息的，比如下标 遍历的奇数个或偶数个 和遍历的个数 这个的count属性就是遍历的个数，也就是计数的 遍历的个数，第一个就遍历了一个 遍历到第二个的时候就遍历了两个了

## HTML的表格

<tr></tr> 行 row rowspan 行合并

<td></td> 列/存储格 column/cell colspan列合并

## 页面传值都是通过标签上的name和value实现的 服务器获取都是通过name获取的 否则获取不到



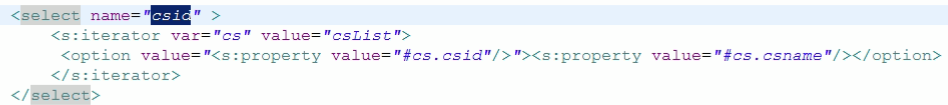
## 因为你修改后台的时候要选择二级分类 而二级分类需要数据库才能得到

所以就要在跳转到那个修改页面的同时 把数据查出来 准确的说是在跳转之前查到 跳转后使用，选择以后传给服务器id就好了 通过name value获取

这个是下拉列表的迭代 跌打其中一个option你展开就是所有的数据

只要一提交 服务器就通过name获取value

页面 action接收的属性名 接收的字段名 必须保持一致



## Struts2的文件上传

1. form标签上加上这个 我也不太确定加不加 是不是对的 method必须是post

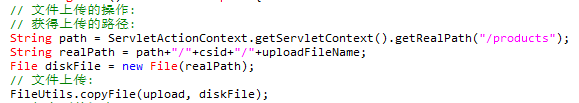
因为你不可能把图片信息拼接到url后面 这是不现实的



2、需要在action中提供三个属性（上传文件的名字 必须和input的name相同）



提供set方法

3、

获得的是绝对路径 和服务器保存的路径不同的 因为这个文件的路径是具体的路径 保存的路径是相对于服务器的 是固定的 而上传路径不同环境都不同，比如说盘符 这是最典型的

只要提供的属性是上面的那个 他就会在你提交的时候 自动将数据封装到字段中

Ps:上传时 文件名是中文没有问题的 因为post方式hibernate是处理了乱码的（反正有一个框架是把post的乱码处理了的）但是你显示的时候有问题 因为img标签的src属性显示中文会有问题的

## 将日期类型转换成数据库的格式再存入



但是视频中直接setordertime(new Date());也能成功的存入数据库的 而我必须要重新格式化才行

视频中就是这样的 直接也能成功保存到服务器 不知道我为什么不行

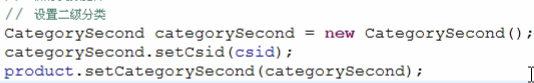


## 向一级分类表中存csid（二级分类的id）

二级分类是多的 一级分类是一的 一般外键的维护都是二级分类中维护 所以二级分类中的csid是外键 一级分类中的csid是主键

两张表通过csid关联，要在二级分类的表中插入一条数据，插入csid时，不能是设置csid，因为没有这个方法 是通过对象关联的，你要设置关联对象中的csid 其他的都不用设置就可以了，就相当于你设置了外键

New一个空对象（关联的那个对象） 只设置他的csid 然后二级分类再设置这个对象 就相当于设置了一个外键csid了 那个对象其他的属性都可以不用设置 设置也不会出错 但是不会用 这不是无用功吗 他只需要一个关联的外键设置就好了



## 查询订单的时候同时查询订单项

需要在订单中将维护的外键将lazy设为FALSE 查订单立即查询订单项 否则用的时候查询的话那时候就没有session了 默认是true你不用就不显示 这是懒加载 用的时候就没有session了就查不了数据库了

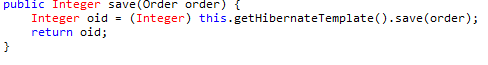
## 全局结果页面的配置

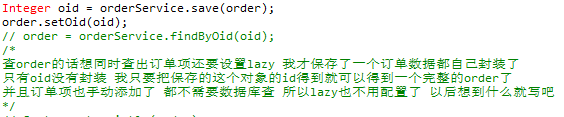


## 得到级联或者关联的对象 需要用用标签属性进行排序 当然如果维护的list就不用 但是list不常用的 或者你得到里面的再封装成list 何必这么麻烦呢 直接用标签的属性就好了

## 页面获取数据必须用order点什么的 不能直接写属性 除非已经有遍历了，因为万一有多个对象呢 属性重名怎么办，你至少得指定是哪个对象的属性吧

## 属性都设置了 只缺自动生成的id 可以在保存的时候会返回当前保存对象生成的id，保存时会返回当前保存的id





## Desc是倒序排序



从高到低

## New一个数组

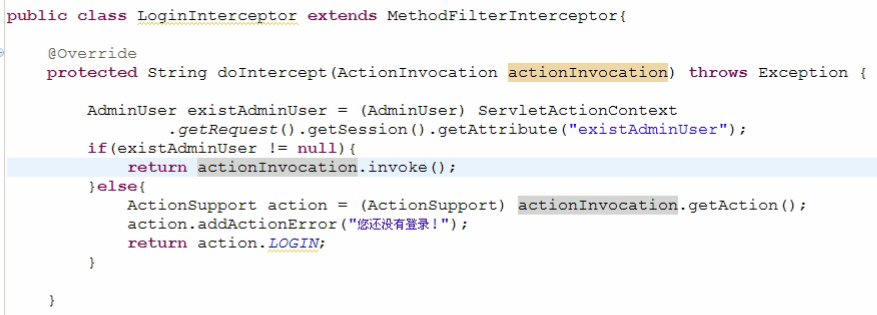


## 返回到这个页面并且还有传分页的数据 并且不能让他一直停留在更改的页面 就需要重定向

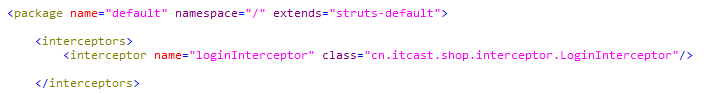


## Struts2的拦截器

1、建一个类继承拦截器的实现类（这里是通过方法拦截）



2、申明拦截器需要在最上面 因为这个是要最先执行的

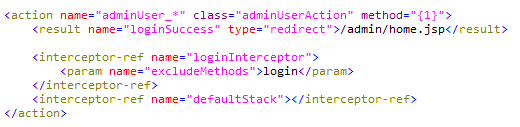


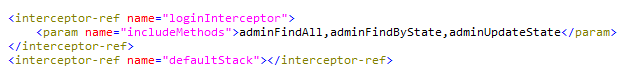
3、引栈 也就是拦截器

通过方法拦截继承的那个类中有set方法 可以设置属性拦截哪些 不拦截哪些一般设置其中一种就好了 在引栈的时候可以设置param 多个加逗号

Ref不一定必须是栈名 还可以是什么的拦截器的名称 只不过这是单个的 多个就定义栈再引栈就好了

这个实在action中添加的 如果在申明的时候就引就是所有的方法 action都生效 在单独的action中添加就只对那个action中的方法拦截 再设置方法 否则的话在申明的时候就引设置方法很麻烦

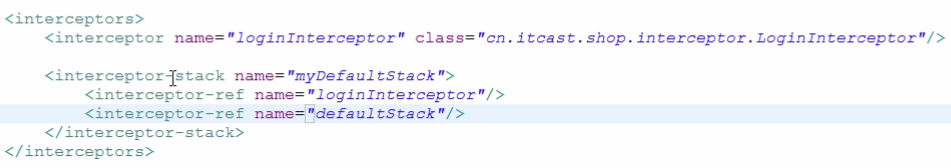




默认的18个拦截器也必须要引入

**作用**：后台除了登录的方法 其他的方法全部拦截 如果不拦截的话 可以直接访问那些后台的action这是我不想要的 必须要登录以后才能访问

申明的时候就定义栈 对所以方法 所以action有效 当然你配置了拦截哪些方法 或不拦截哪些方法就不一样了



这也只是什么或定义 要生效还是需要在action中引的

这样就好了

但是你要设置拦截的方法的话 要在申明的时候写一大堆

**注意：**拦截器只能拦截Struts2中的action的方法 如果是访问jsp 要在访问jsp的时候也拦截需要过滤器才能实现

过滤器是拦截jsp的 拦截器是拦截action的

## 在标签上或图片上使用js提交表单

