12天: OA项目 + JBPM框架

============================================

OA 办公自动化系统

BBS 论坛系统

CRM 客户关系管理系统

CMS 内容管理系统

# 一、什么是OA

辅助办公、提高办公效率的软件系统。

二、具体有哪些功能

具体有哪些功能有用户的需求确定。

=============================================

一、软件开发流程

1，需求分析

2，设计

3，编码

4，测试

5，交付、运维

二、每个环节具体做什么事情

1，需求分析

需求分析文档

效果页面

2，设计

概要设计文档

详细设计文档

3，编码 <━━━━━━━━━┓

按照设计文档实现功能 ┃

4，测试 ┃

测试工具、测试人员 ━━━━┫ 代码管理工具（版本管理工具） / SVN / CVS / ...

功能测试报告 ┃

性能测试报告 ┃

5，交付、运维 ┃

发现BUG ━━━━━━━━━┛

增加新功能

三、我们在哪个环节

部分设计

主要编码

部分测试

写文档

=============================================

# 一、课上要实现的功能：

1，系统管理

2，系统权限

3，内部论坛

4，JBPM + 审批流转

二、时间计划

0，准备 + 基础功能 1天

1，系统管理 2天

2，系统权限 2天

3，内部论坛 3天

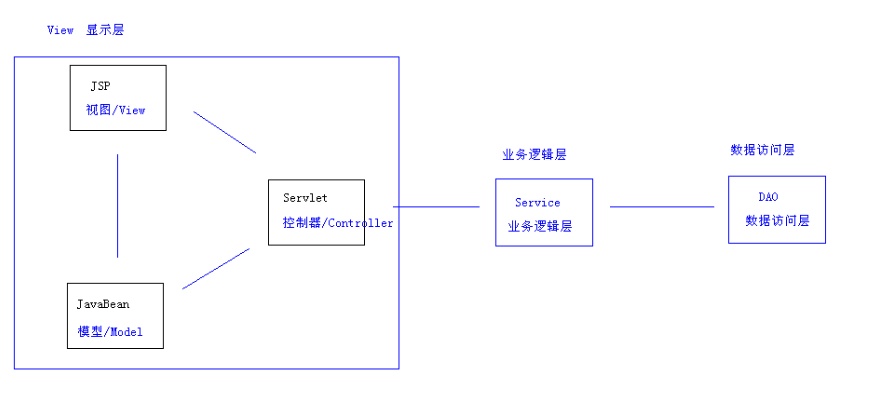
4，JBPM + 审批流转 2天 + 2天

三、整体设计

1，分层架构

View Struts2

Service JBPM

Dao Hibernate 

2，所用技术

Struts2.1 + Spring2.5 + Hibernate3.x + jQuery1.x + JBPM4.4

3，开发环境

MyEclipse 8.6 ( Eclipse / Intellj idea / NetBeans / ... )

MySQL ( Oracle / SqlServer / DB2 / ... )

IE6 ( Chrome / IE9/ IE10 / Firefox / ... )

4，代码规范

命名

注释

空行

Ctrl + Shift + F（格式化代码）

原型 + 重构

5，约定

编码使用utf-8

主键使用long型

6，项目计划

==========================================

一、创建数据库"itcastoa"：

（注意编码为utf-8）

mysql> create database itcastoa default character set utf8;

mysql> show create database itcastoa;

+----------+-------------------------------------------------------------------+

| Database | Create Database |

+----------+-------------------------------------------------------------------+

| itcastoa | CREATE DATABASE `itcastoa` /\*!40100 DEFAULT CHARACTER SET utf8 \*/ |

+----------+-------------------------------------------------------------------+

# 二、创建Web工程

1，把默认的编码设为utf-8

java类

jsp文件

xml文件

2，添加框架环境（jar包、配置文件,**全部采用手动添加的方式而不是java build bath，java build bath是引用某个地址下的jar包，并没有真正添加到自己的项目中**）

jar包：

Junit4

JSTL（2个包）

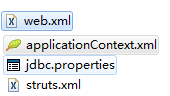


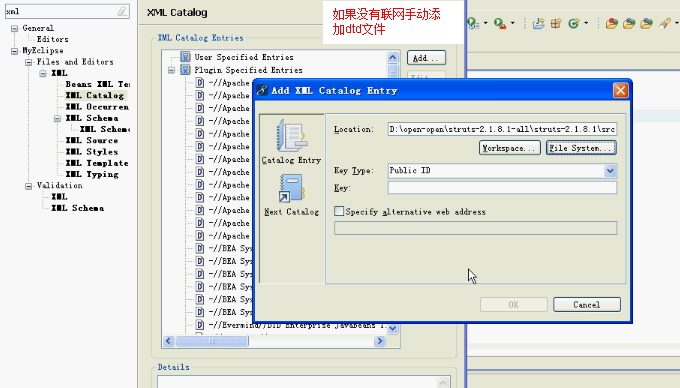
Struts2

Hibernate

Spring

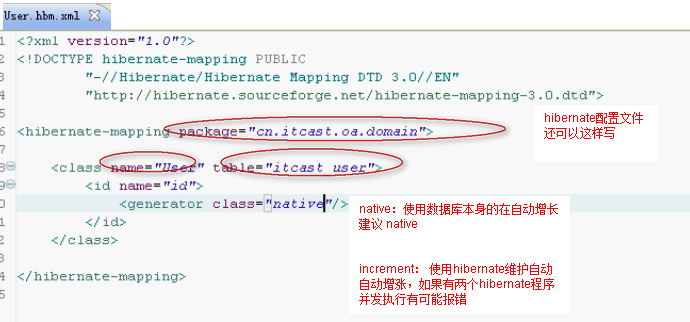
配置文件：



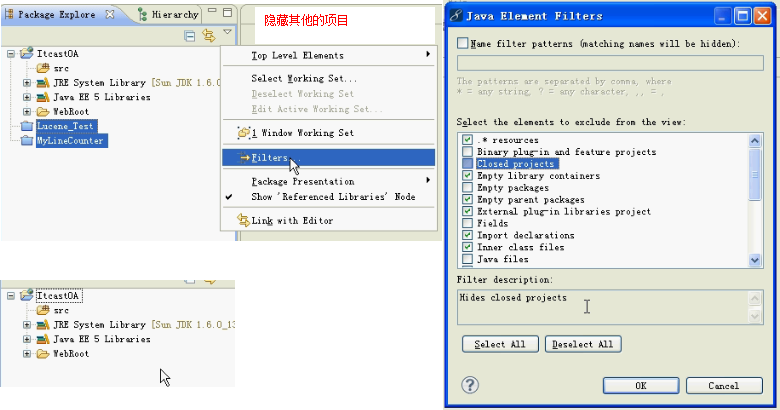












**经验：**

**Myeclipse 只改了方法体中的内容不需要重启服务器直接刷新即可**

**改了结构 类名 方法名 类中的字段 变化了需要重启服务器**

**.properties文件变成utf-8后就可以写中文了**

3，整合SSH 步骤：分两步 A整合Spring与Hibernate B 整合Spring与Struts2

a, 整合Spring与Hibernate

目的：

让Spring管理单例的SessionFactory

让Spring管理事务（采用基于注解的方式）

步骤：

...

b, 整合Spring与Struts2

目的：

让Spring管理Action的实例（**多例**）。

这样就可以方便的注入依赖的对象了。

步骤：

1，Spring在Web应用中使用时要配置一个监听器

为了创建ApplicationContext容器对象 web.xml文件

2，添加struts2-spring-plugin.jar

为了让Struts2从Spring的ApplicationContext容器对象中拿Action对象

4，基本配置：

日志

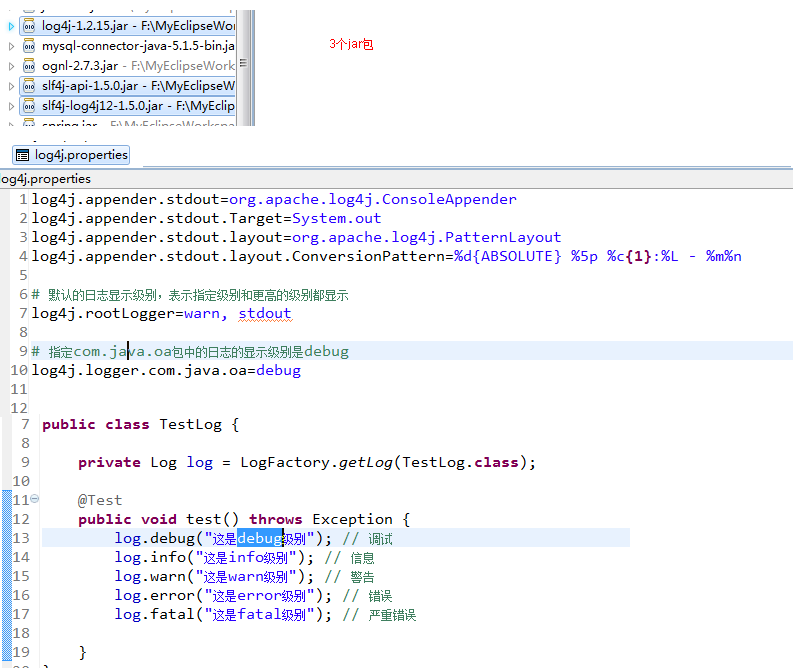
slf4j（接口）

log4j（具体的实现类） ( log4j.properties )

commons-logging（接口）

log4j（具体的实现类） ( log4j.properties )

我们使用slf4j（接口） 加 log4j（具体的实现类）



5，资源分类

源码文件夹

src

config

Test

WebRoot

css

images

js

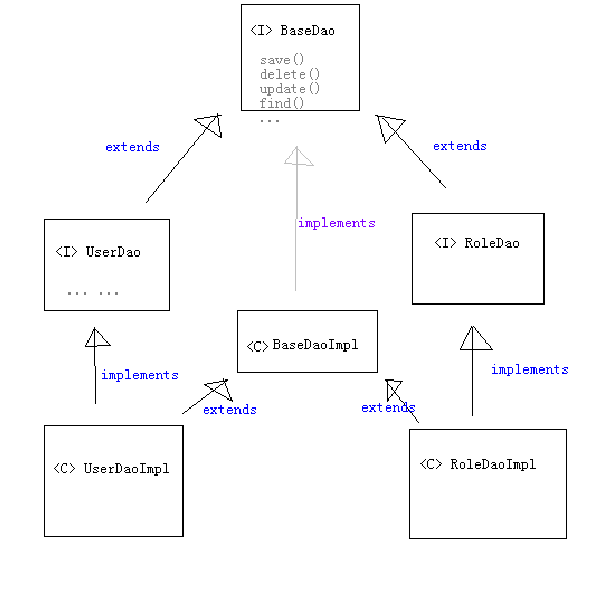
WEB-INF

jsp

包的划分

# 三、项目重构

## 1，设计 BaseDao 与 BaseDaoImpl



1，设计接口 BaseDao

1，每个实体都应有一个对应的Dao接口，封装了对这个实体的数据库操作。例

实体 Dao接口 实现类

========================================================

User --> UserDao --> UserDaoImpl

Role --> RoleDao --> RoleDaoImpl

Department --> DepartmentDao --> DepartmentDaoImpl

Article --> ArticleDao --> ArticleDaoImpl

...

2，在每个Dao接口中都应有一个基本的增删改查的方法，

但每个Dao接口中都写一遍就是重复的代码，可以把这些方法抽取到一个父接口中，定义为：

public interface BaseDao<T> {

save(entity : T) : void

delete(id : Serializable) : void

update(entity : T) :void

getById(id : Serializable) : T

getByIds(ids : Serializable[]) : List<T>

findAll() : List<T>

}

3，说明：

1，使用泛型的技术，可以在使用时更方便，如：User user = userDao.getById(1L); // 不需要强制转型

2，getById(id : Long) 与 getByIds(ids : Long[]) 不要合并为一个有可变参数的方法，

因为那个用起来就不方便了，比如要经常使用的根据一个id获取一个对象时就很不方便：

List<User> list = userDao.getByIds(1L); User user = list.size() > 0 ? list.get(0) : null。

3，以后再有Dao接口就可以继承这个 BaseDao，以拥有基本的增删改查方法，如：

UserDao extends BaseDao<User>

RoleDao extends BaseDao<Role>

2，设计实现类 BaseDaoImpl

1，每个Dao的接口还要有相应的实现类才可以，在每个DaoImpl中都要实现Dao接口中定义的所有方法，

当然也包括公共的增删改查方法（BaseDao中定义的6个基本方法）。

2，每个DaoImpl中都实现一遍公共方法显示是重复的，所以也可以抽取出来为一个父类BaseDaoImpl，

在BaseDaoImpl中实现了BaseDao接口的所有方法，我们的DaoImpl只需要继承他就可以不用重复的写公共方法的实现了。

声明如下：

public class BaseDaoImpl<T> implements BaseDao<T> {

@Resource private SessionFactory sessionFactory; // 让Spring注入sessionFactory

private Class<T> clazz; // 要想办法得到T的Class

public void save(T entity) { }

public void delete(Long id) { }

public void update(T entity) { }

public T getById(Long id) { }

public List<T> getByIds(Long[] ids) { }

public List<T> findAll() { }

}

在实现方法时，获取Session的代码为：sessionFactory.getCurrentSession()

**注意：因为是getCurrentSession()所以service中无论什么方法都要开启事务而且事务是否只读无所谓**



3，在获取clazz时有两种方法

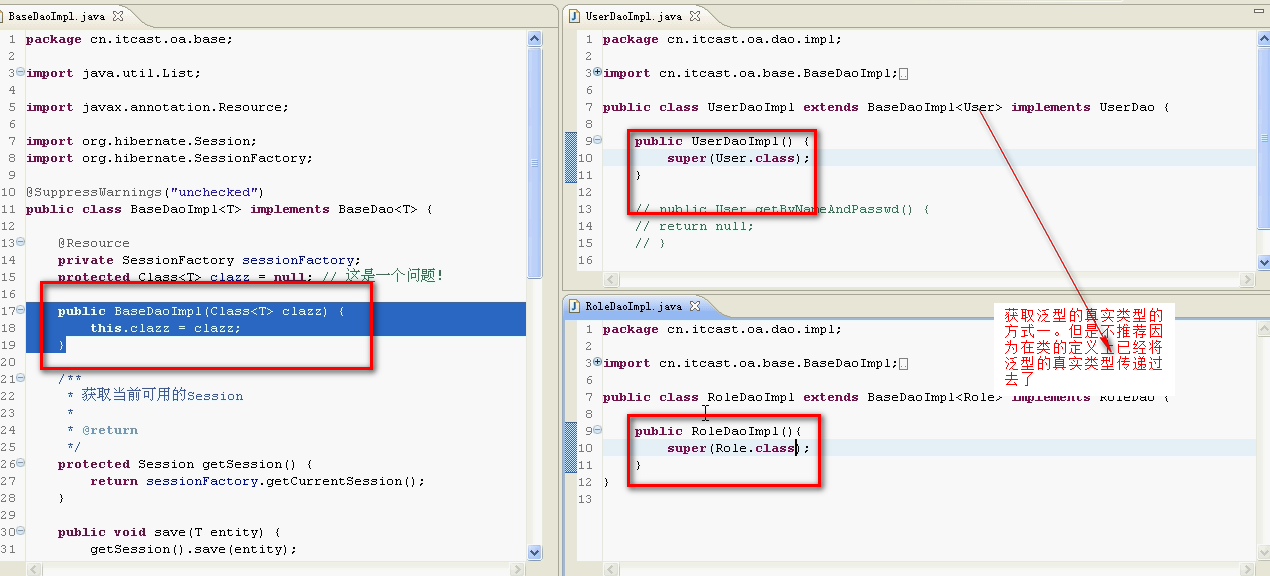
1，方法一：

1，把clazz声明成protected修饰符的，这样子类中就可以访问了，

2，在每个子类的构造方法中传递这个属性的值，如：

public RoleDaoImpl() { clazz = Role.class; }

public UserDaoImpl() { clazz = User.class; }



2，方法二：使用反射的方式：

1，在BaseDaoImpl的默认构造方法中写如下代码就可以了：

ParameterizedType pt = (ParameterizedType) this.getClass().getGenericSuperclass();

clazz = (Class) pt.getActualTypeArguments()[0];

1. 说明：BaseDaoImpl不能直接使用，只能使用他的子类，否则这段代码就无效了。

clazz.getSimpleName() 简单名称（只有类名）

clazz.getName() 全限定名称（带着包名的）

3，问题

1. 有了DaoBase与DaoImplBase，还要用UserDao、RoleDao吗？

答：要用。因为UserDao或RoleDao中的方法可以分为有公有的方法与特有的方法两部分。

公有的方法是通过继承BaseDao得到的，特有的方法要写在自己里面（BaseDao中是没有的）。

1. UserDaoImpl已经继承了BaseDaoImpl，就不实现UserDao可以吗？

答：不可以。否则UserDao userDao = new UserDaoImpl(); 就不成立。

# 系统管理模块

|  |
| --- |
| **思考步骤：**  1，充分了解需求，包括所有的细节，需要知道要做一个什么样的功能  2，设计实体/表  正向工程：设计实体、映射文件 建表  反向工程：设计表  映射文件、实体  3，分析功能  分析到每个请求的粒度。  得到的结果是我们需要处理多少种请求，其中每种请求对应一个Action方法。  4，实现功能：  1，创建Action，并定义出其中的方法。  2，实现Action方法，并创建出所用到的新的Service方法。  3，实现Service方法，并创建出所用到的新的Dao方法。  4，实现Dao方法。  5，创建并完成JSP页面。  5，测试、运行  **书写步骤：**  1、设计实体（之间的关系）写映射文件，建立数据表  2，写dao crud，  3，测试dao  4，写service crud，  5，测试service  6，action，jsp  7, struts.xml |

### 1.岗位管理

1，了解需求

2，设计实体和表

3，分析功能

每个请求对应一个方法。分析了请求之后，能确定对应的请求。

4，实现功能

1，创建action

2，实现action，创建service方法

3，实现service，创建dao方法

4，实现dao，

5，设计domain

Role

id Long

name String

description String

★写role，这里不写Serializable，有需要的时候再写。

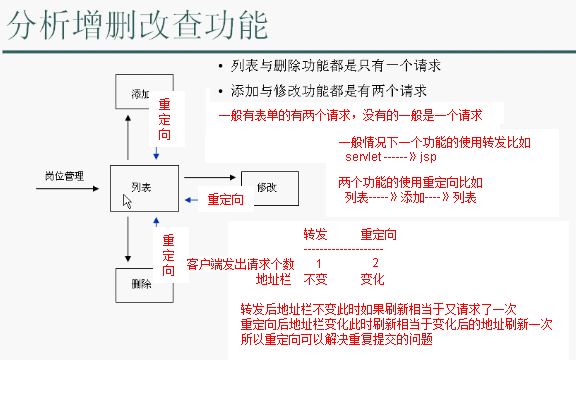
关于bean的序列化（如果该类需要网络传输就需要序列化，比如说将user类放到session中就需要实现序列化）

★写Role.hbm.xml

三个字段，写name和colum，在实际开发中，对字段长度有要求的，写length

★加入到hibernate配置文件中去

6，RoleAction



Result结果集类型：

<!-- dispatcher:默认结果类型 请求转发到一个页面 -->

<!-- redirectAction 重定向到另外一个动作 -->

★:写RoleAction：

1. 写6个方法。
2. 方法名 返回值（4个） 页面（3个）
3. ------------------------------------------------------------
4. 列表 list() list list.jsp
5. 删除 delete() toList
6. 添加页面 addUI() addUI addUI.jsp
7. 添加 add() toList
8. 修改页面 editUI() editUI editUI.jsp
9. 修改 edit() toList

★：写页面

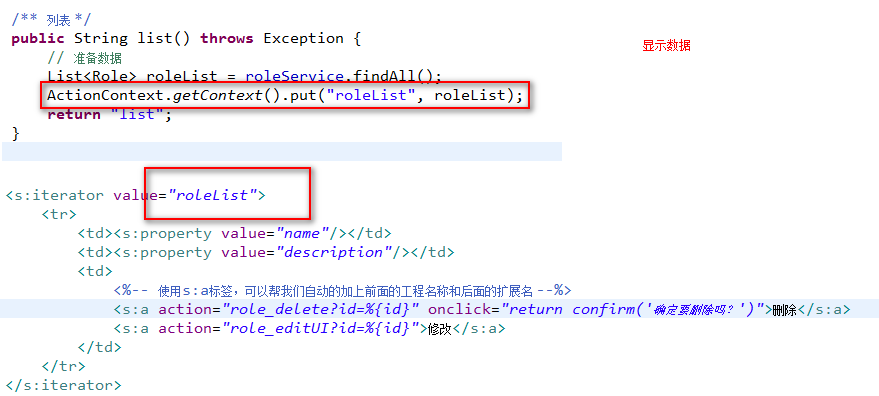
**<%--**

**使用s:a标签，可以帮我们自动的加上前面的工程名称和后面的扩展名**

**s:a中的action不能直接跟ognl表达式 action属性中的值是字符串**

**在字符串中使用ognl表达式要使用 % 号**

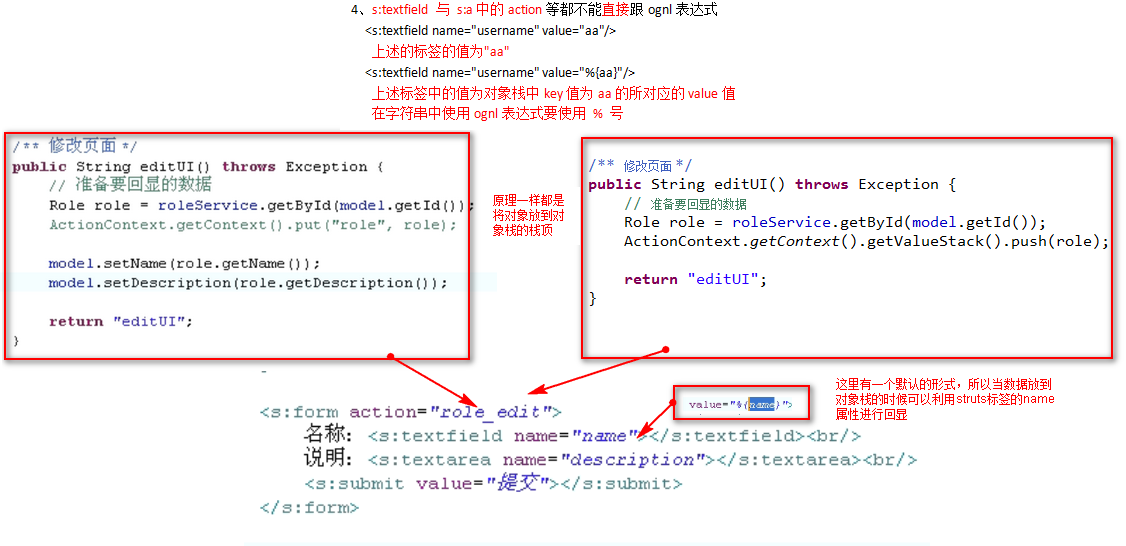
**--%>**



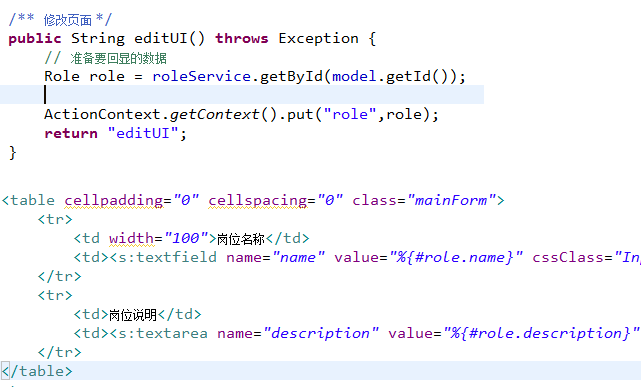
struts2数据回显

把一个对象放入到对象栈或者map栈中都可以进行回显：

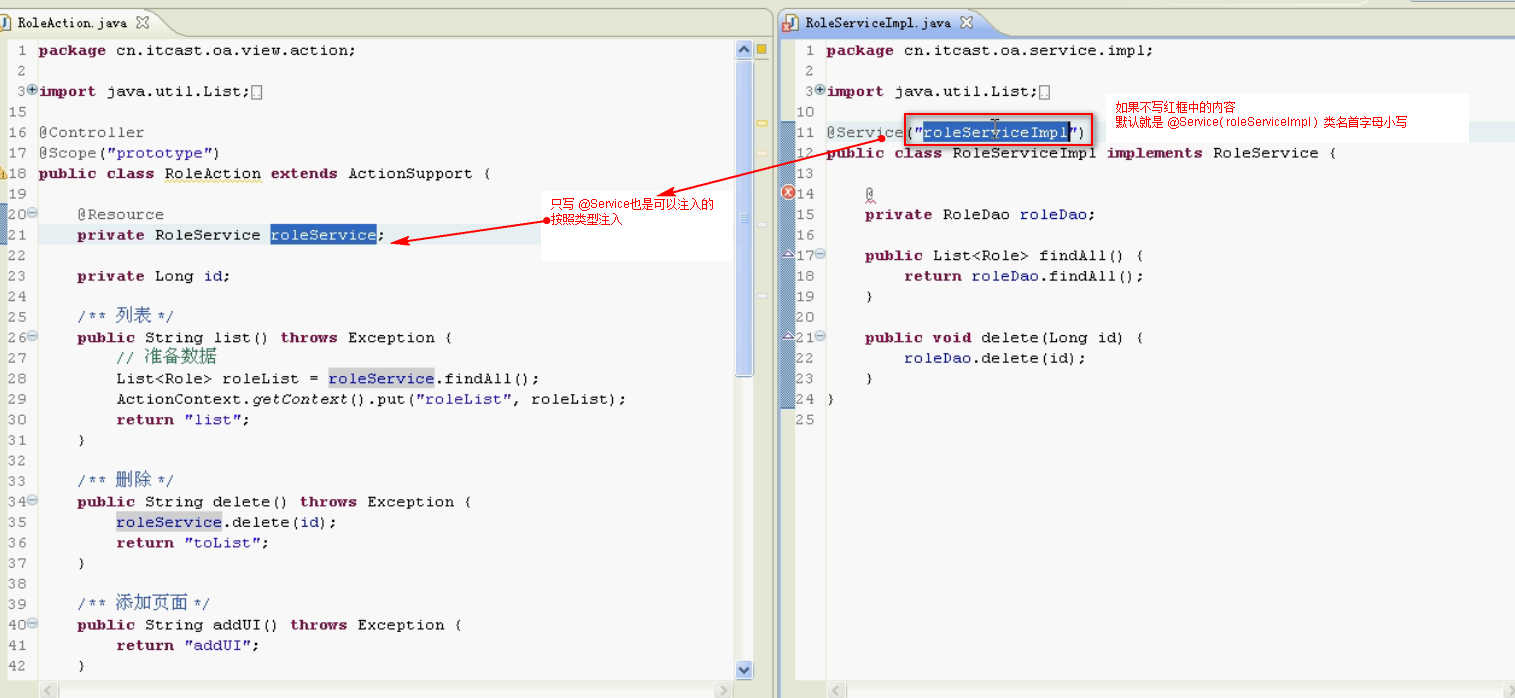
1.放在对象栈中的属性可以利用struts2标签中的name属性进行回显



2. 放在map栈中的属性可以利用struts2标签中的value属性进行回显

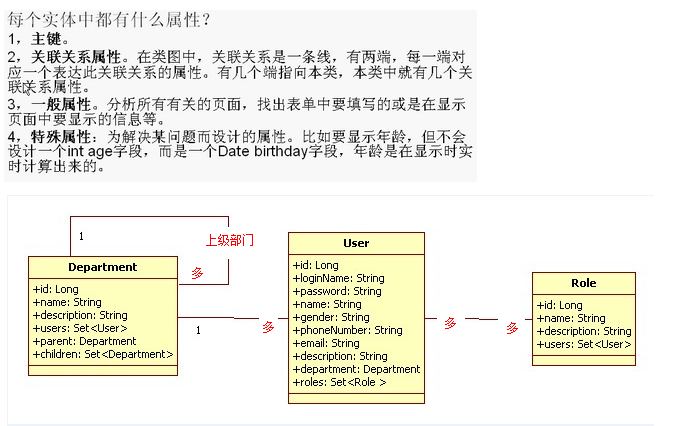






### 用户 部门 角色

0，写javabean



1，写完javabean后，写映射文件。

先写简单属性，有关联关系的属性，后面来写。

映射模板：

多对一：

<many-to-one name="" class="" column=""></many-to-one>

一对多：

<set name="">

<key column=""></key>

<one-to-many class=""/>

</set>

多对多：

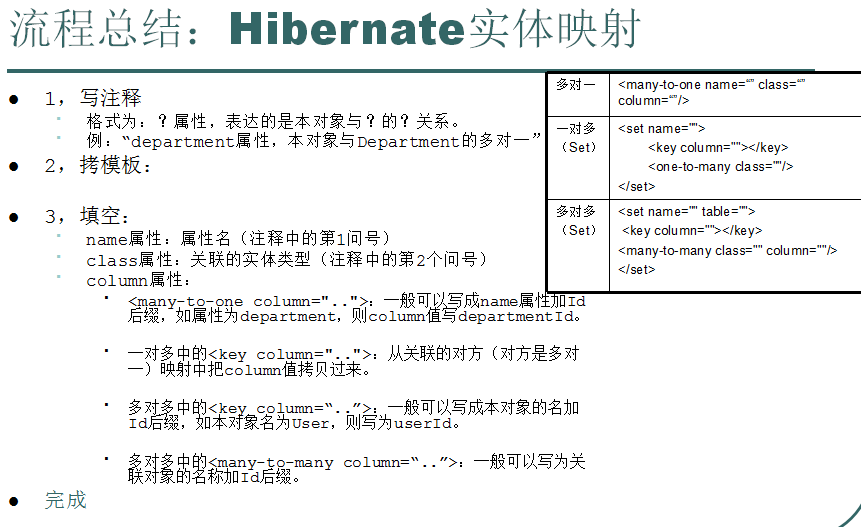
<set name="" table="">

<key column=""></key>

<many-to-many class="" column=""></many-to-many>

</set>







可以合并添加页面和修改页面

%{id==null?'add':'edit'}

### 部门的增删改查

步骤

从上往下。。。。。。依次写

1，创建Action类并在类上加注解 @Controller 和 @Scope("prototype")

2，写其中的方法CRUD的6个方法（方法名、返回值、注释）

3，创建JSP页面

4，配置struts.xml

1，创建Service接口。

2，在Action中引用Service，并写上@Resource注解

3，写Action方法的内容，并创建出所用的Service方法。

1，创建Service的实现类，并在类上写注解 @Service 和 @Transactional

2，引用Dao ...

3，实现Service方法的内容 ...

添加上级部门

显示的时候显示上级部门

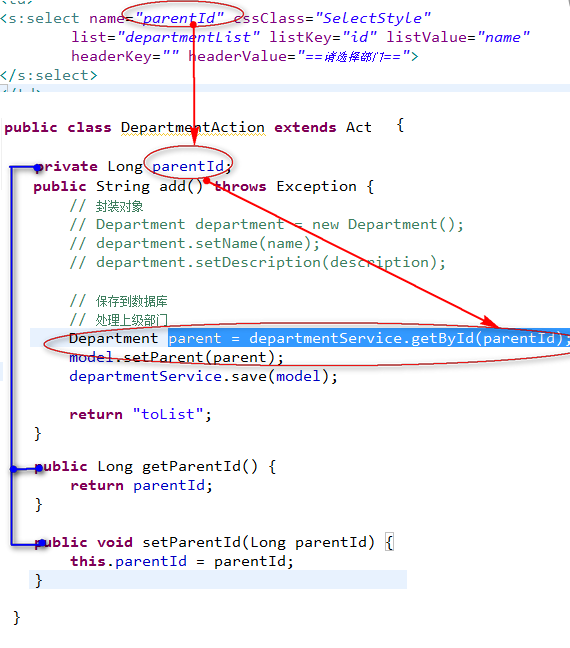
修改，添加的的时候可以选择上级部门

删除的时候本部门及其下级部门全部删除

**<s: select>的用法**



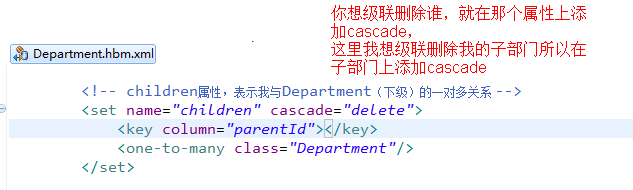
1添加操作 （在action动作类中添加parentId属性）



2修改操作



3删除操作



inverse是否维护关系（比如 删除的是关系，而不是删除基本数据，老师删除了，关联的学生的关系删除了，学生的基本数据还在）

cascade级联（比如 删除的是基本数据；老师删除了，学生的基本数据同时也删除）

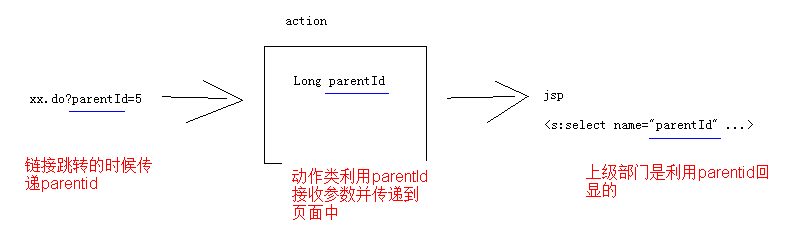
4列表显示

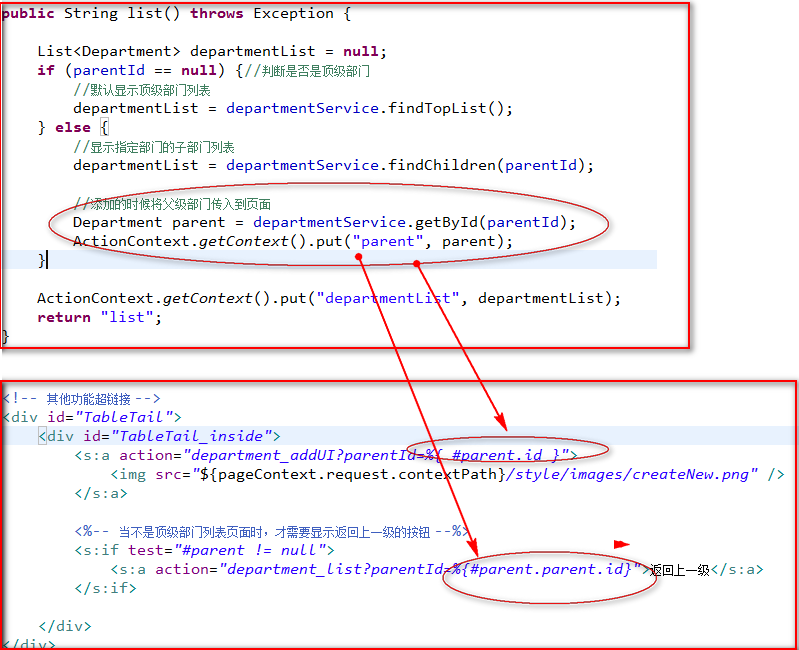


5.1

a新建的时候默认的上级部门是当前页面的上级部门

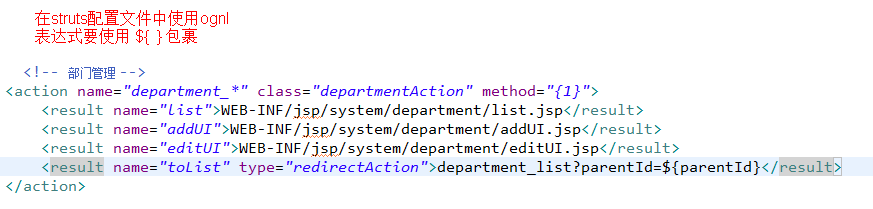
b返回上一级



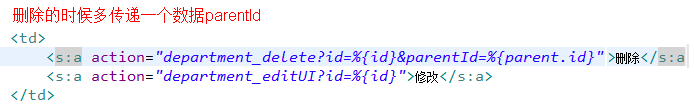


5.2添加删除修改后还回到本级目录而不是跳转到顶层

添加 修改 的时候parentid是有值的 在struts配置文件中使用ognl表达式

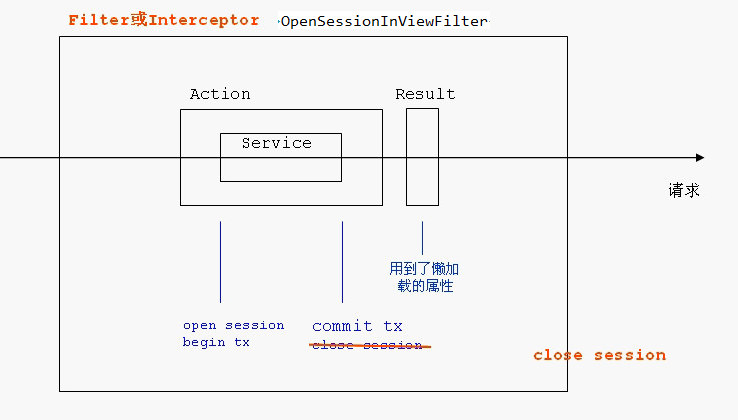


删除的时候是没有值的



6. no-session 懒加载异常



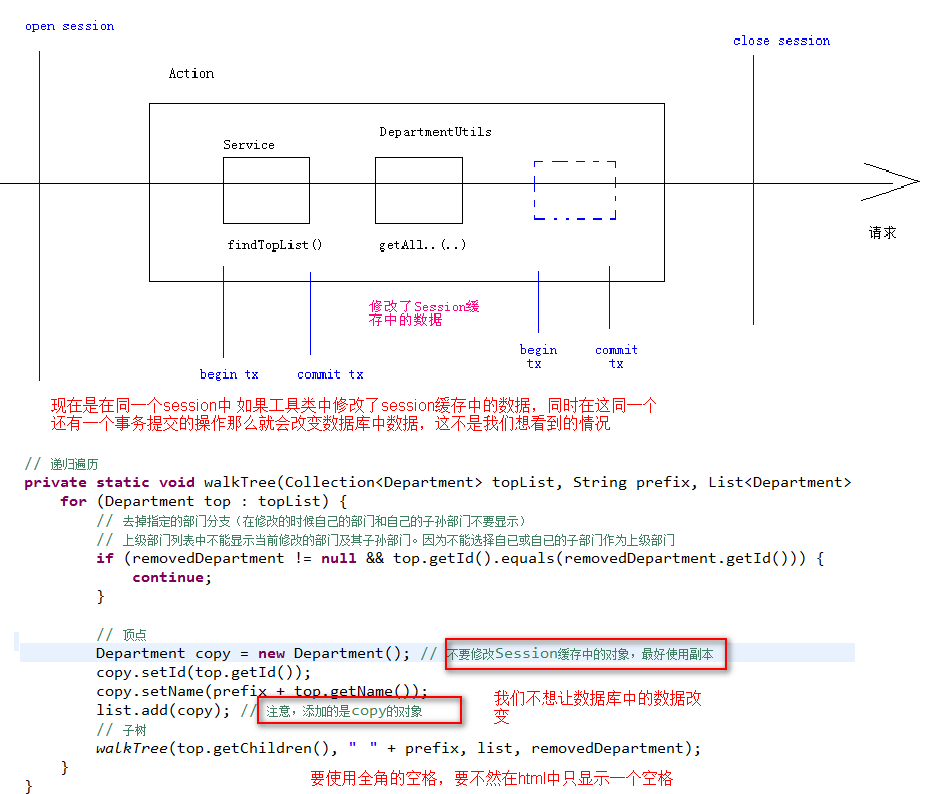


7.在添加和修改的时候对部门进行树形结构展示



DepartmentUtils.java

**注意分割符空格要使用全角空格**



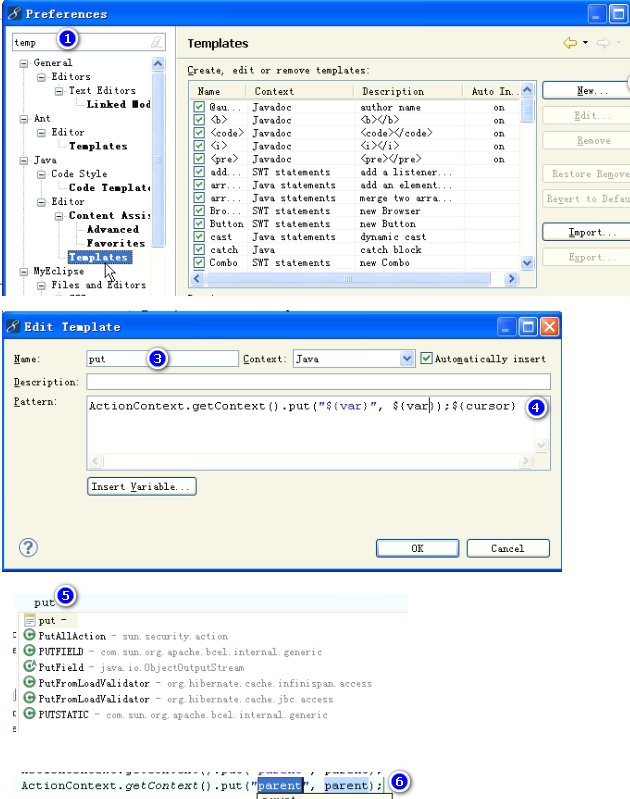
1. 修改的时候部门不能是自己或自己的下级

DepartmentUtils.java



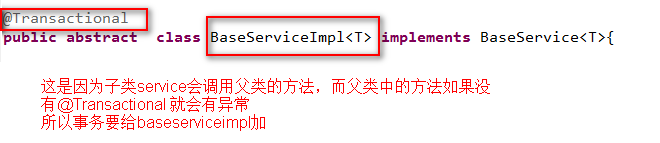
### 一些改进

1.自定义代码模板快捷键



1. BaseAction.java

1. BaseService.java BaseServiceImpl.java



1. Jsp页面合并

Common.jsp

### 小知识点

@Transactional注解，当写到类上时：

1，对本类中的公共方法有效。

2，对子类中的公共方法有效（即这个注解可以被继承）

3，对父类中的方法无效！！！

注解是可以继承的，所以必须而且一定要在基类中添加@Transactional

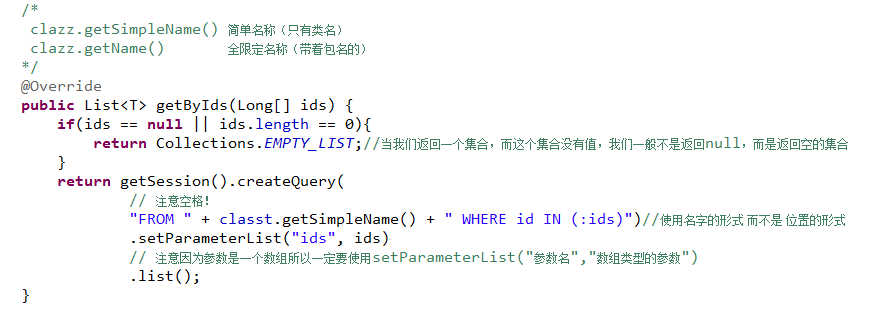
子类service会调用父类的方法，父类中的方法必须有@Transactional

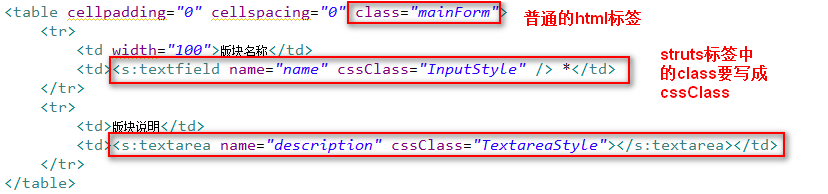
EL表达式的查找顺序：

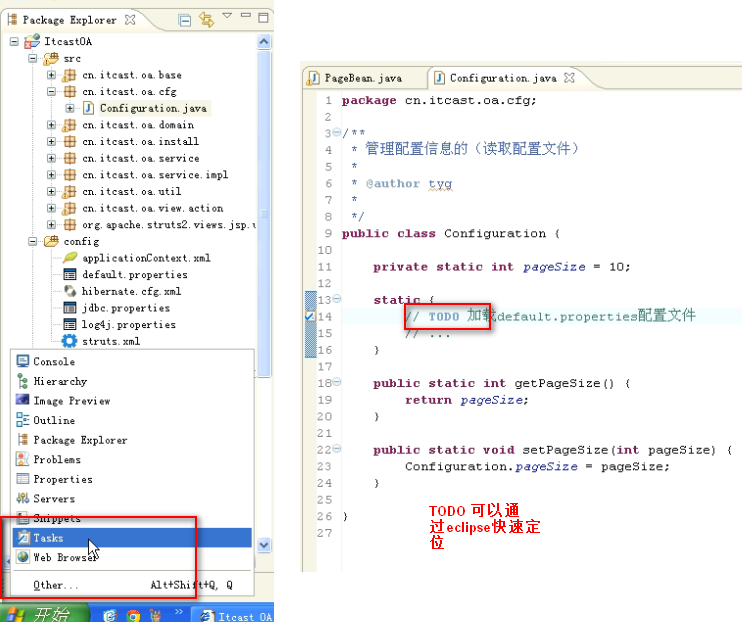
1, 原始的顺序：page, request, session, application

2, 在Struts2中：page, request, ValueStack, session, application

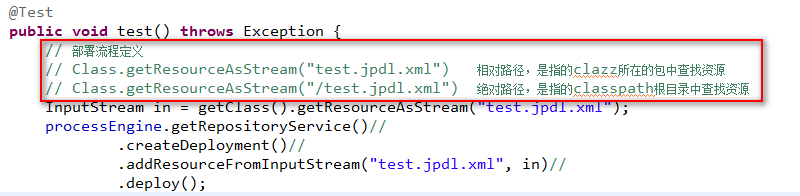
当我们查询出一个集合的时候如果集合中没有数据那么不要反返回空值，要返回空的集合











### 用户管理



MD5加密

commons-codec-1.3.jar

在userserviceImpl中重写save方法

|  |
| --- |
| @Override  **public** **void** save(User user) {  // 默认密码是1234  String md5 = DigestUtils.*md5Hex*("1234"); // 密码要使用MD5摘要  //用到的包 commons-codec-1.3.jar  user.setPassword(md5);  // 保存到数据库  **this**.userDao.save(user);  } |

注解：只有事务的注解可以被继承，像 Controller resource 等等其他注解不可以被继承

### 权限

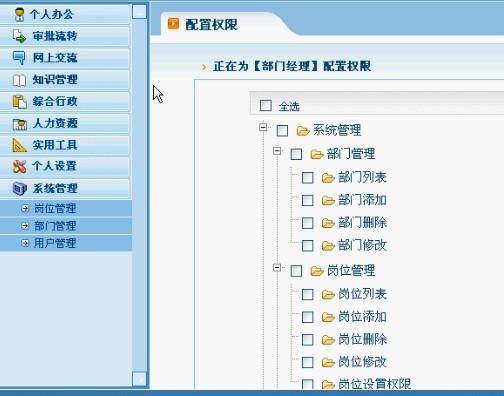
系统中的权限基本数据是安装系统时就有的，超级管理员也不能添加修改。

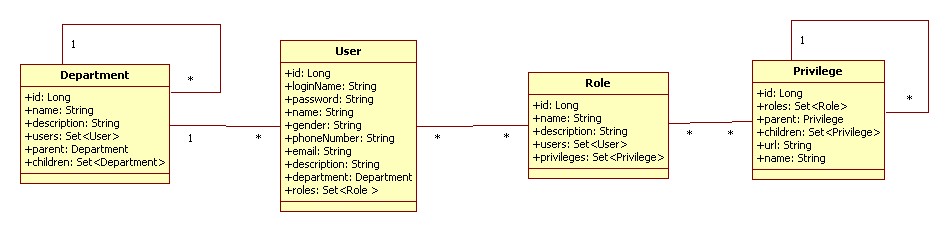
在Web应用中，控制权限就是控制URL的访问。

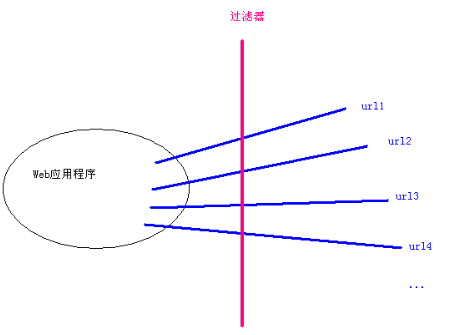
超级管理员：登录名为admin，有所有的权限，他的权限不需要分配，也不能取消。

**用户与角色的关系：多对多 一个用户可以有多个角色，一个角色可以分配给多个用户**

**角色与权限的关系：多对多 一个角色可以有多个权限，一个权限可以分配给多个角色**

****

****



技术点：使用过滤器拦截处理

初始化数据（安装）

权限数据。

超级管理员。

分配权限

给角色分配权限。

用户的权限就是用户所有角色的权限。

使用权限

1，登录、注销、主页面

2，左侧的菜单是根据权限显示的。

3，右侧页面中的链接是根据权限显示的。

4，拦截每一个action请求，验证用户是否有权限访问。

#### 1，需求分析

1， 开始的登录，和权限有关，登录完之后，看到左侧有菜单，根据权限，显示你可以操作的菜单。

拦截每一个action请求，查看是否有权限访问。

2，分配权限

给角色分配权限

用户的权限就是用户所有的角色的权限

3，初始化数据（安装）

权限数据

超级管理员

#### 2，步骤

1，设计实体。

2，写Privilege的javabean

3，Role中增加一个属性。

4，增加一个新的映射文件。修改Role中新添加的映射文件

5，一般会有一个init.sql文本文件。我们可以利用框架，写一个安装程序。

执行后会安装程序和插入数据

install包中Install.java 初始化数据



1. 权限分配 （显示树状结构）

在RoleAction中

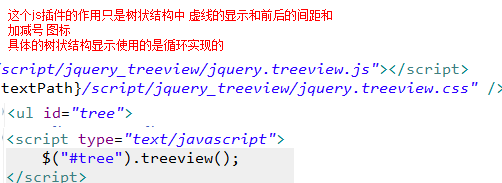
@Resource(name="privilegeService")

**private** PrivilegeService privilegeService;

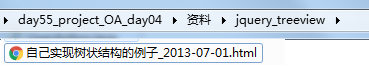
**private** Long[] privilegeIds;



|  |
| --- |
| 使用Struts2的标签显示复选框，有回显功能，但不方便修改显示效果。  <s:checkboxlist name="privilegeIds" list="privilegeList" listKey="id" listValue="name" >  </s:checkboxlist>    自己使用循环显示复选框，需要自己实现回显功能  <s:iterator value="privilegeList">  <input type="checkbox" name="privilegeIds" value="${id}" id="cb\_${id}"  <s:property value="%{id in privilegeIds ? 'checked' : ''}"/>  />  <label for="cb\_${id}">${name}</label>  </s:iterator> |

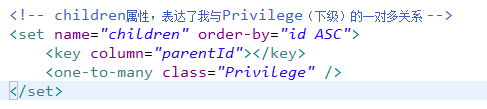


具体查看：





排序问题使用hibernate的配置解决



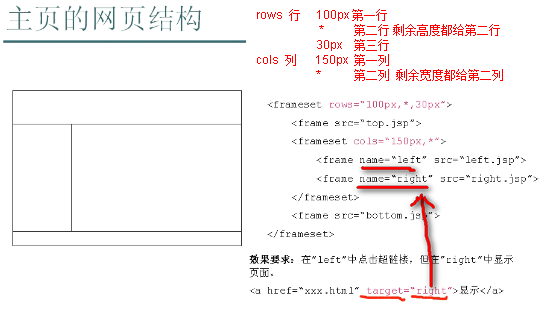
#### 3，登录注销步骤

说明：

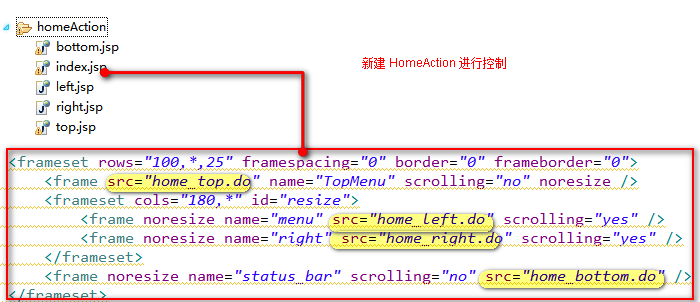
1.登录： 用户名 密码找到user 保存到session 跳转到**主页面** ，如果找不到user“用户名或密码不正确” 密码不回显，用户名回显

2.注销 ：session中清除user

**3.主页面：**



target=“\_self” 在自己的frame中显示



A Top页面中显示 登录用户 和 注销功能

B Left 页面显示 当前用户的权限列表（只需要显示第一级和第二级，第三级是具体的操作不需要显示）

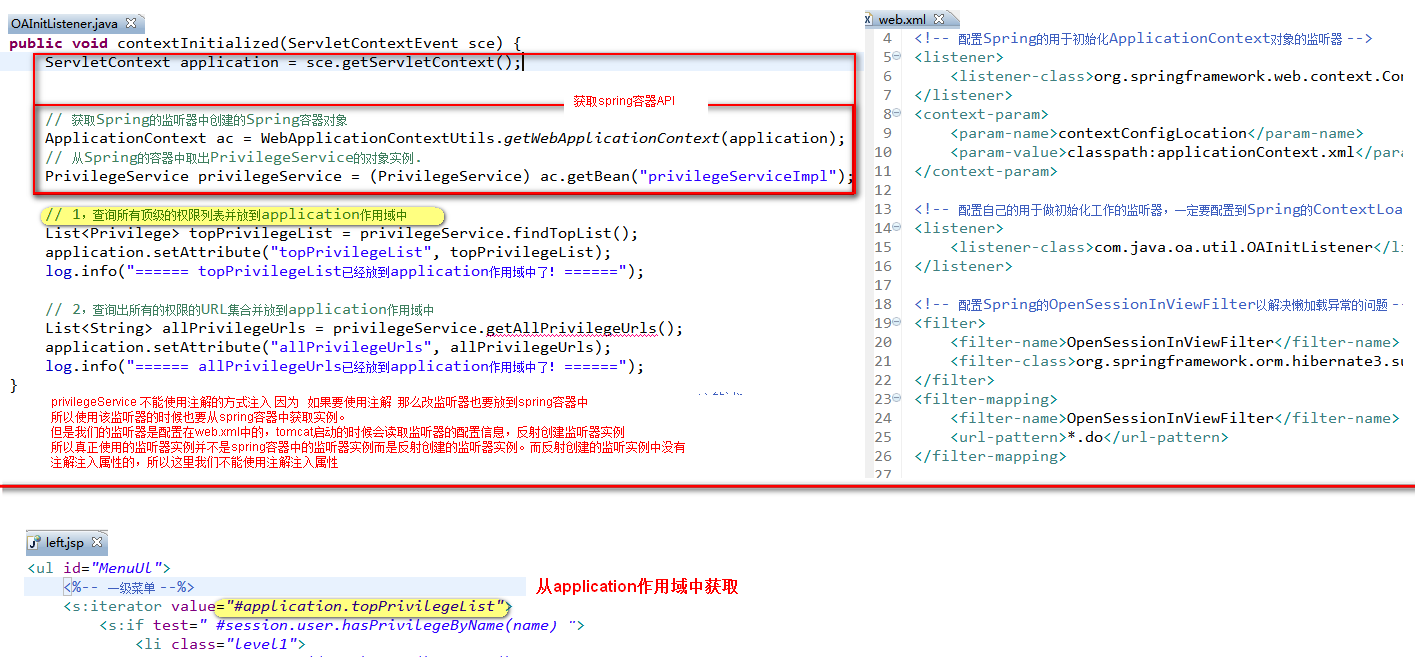
权限的基本数据是不会变化的，不需要每次都查询。所以我们可以将权限的基本数据存储起来放在Applicatioin 中这样所有人都可以访问到.什么时候往Applicatioin中放？答：当应用启动的时候就往Applicatioin中放.

使用监听器完成这个操作



在 web.xml文件中配置该监听器







注意：会有两处懒加载异常

1. 监听器查询出了所有的顶级权限 但是当查询 Privilege.children集合属性的时候发生了懒加载异常

原因：启动应用程序的时候查询了所有的顶级菜单，这算一次请求，当登录访问的时候左侧菜单被第 一次请求（这又是一次请求）这时才用到放在最大作用域中的顶级菜单，顶级菜单又懒加载子 级权限。但是这不是一次请求，而是两次请求所以会报懒加载异常。

解决办法：既然是一定要用到的属性那么就不要懒加载了。直接将hibernate的文件改成lazy=“false”

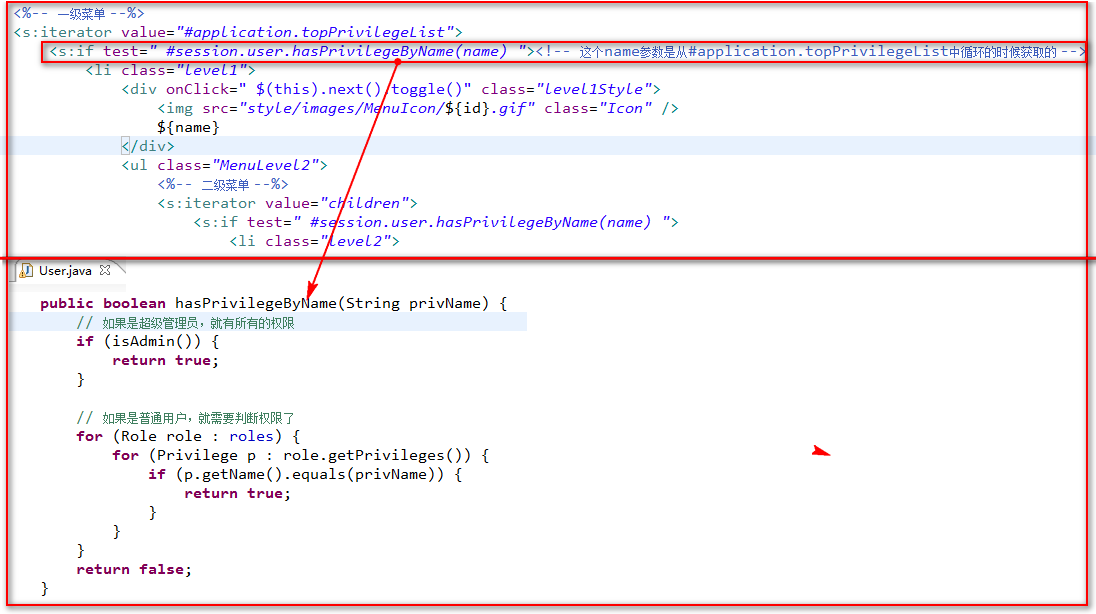


2.同上：登录的时候我们将user对象放到了session中 User 获取 role,role获取 privilege 又会有懒加载异常



显示当前用户的权限 前提：正常登录后

在left.jsp中进行判断 ognl表达式中可以调用方法，el表达式不可以调用方法



**注意如果当前用户是超级管理员admin 那么就有所有的权限 永远返回true**

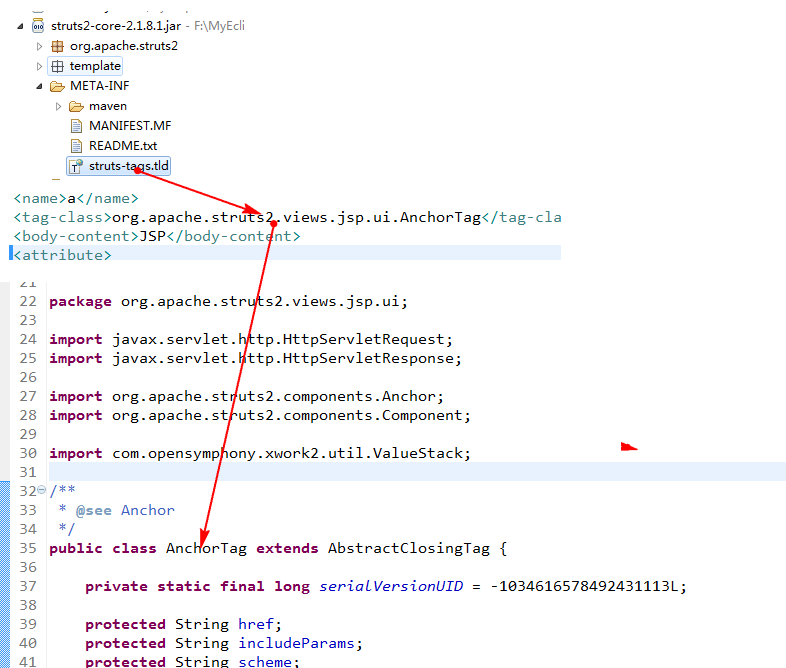
C right 页面

Right页面中也要有权限的操作 功能操作权限 比如 没有修改 和删除的权限 那么这两个连接就要隐藏以user列表页面为例



这样写比较麻烦有其他的简洁操作





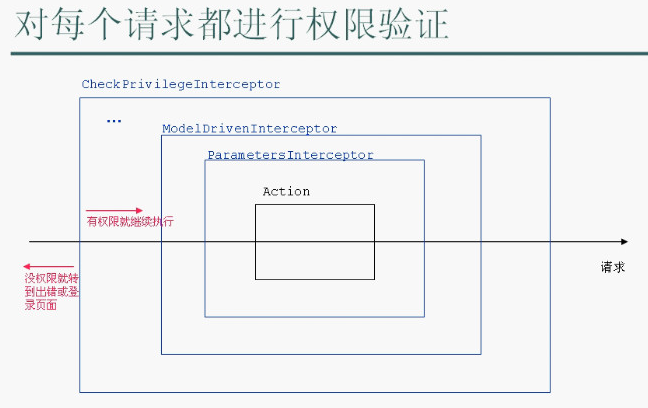


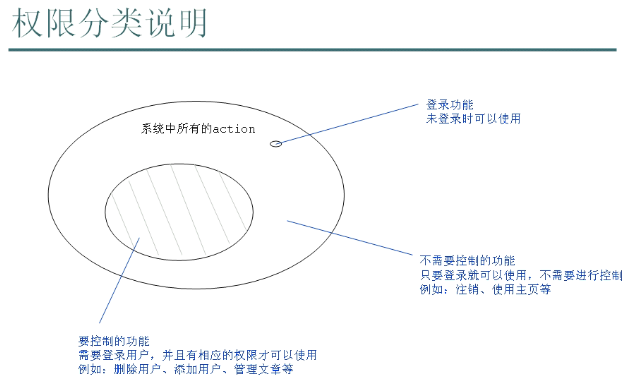




拦截验证每一个请求的权限

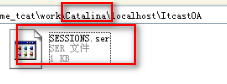
拦截 每一个action验证用户是否有权限访问 使用struts自定义拦截器**checkPrivilegeInterceptor.java**



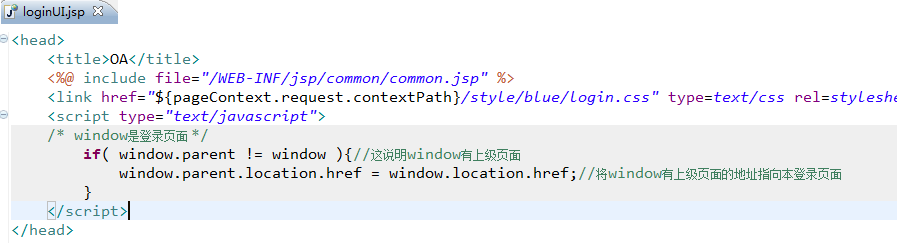




1. 给 user role privilege 添加序列化接口Serializable



2. 当session过期后（默认是30分钟） 再次点击现在不是登录状态了（例如退出系统后点击浏览器的返回按 钮），被拦截器阻拦下，发现没有登录，有加载登录页面这时就会出现请登录页面嵌套



用户是登录动作的主体所以可以将登录注销有关的操作放到UserAction中

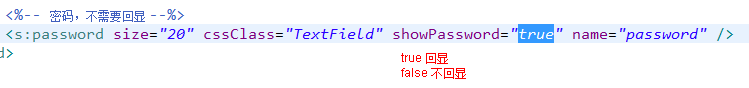
这里我们新建一个action专门处理**登录和注销**

1，LoginoutAction，有三个方法

2，记得加注解

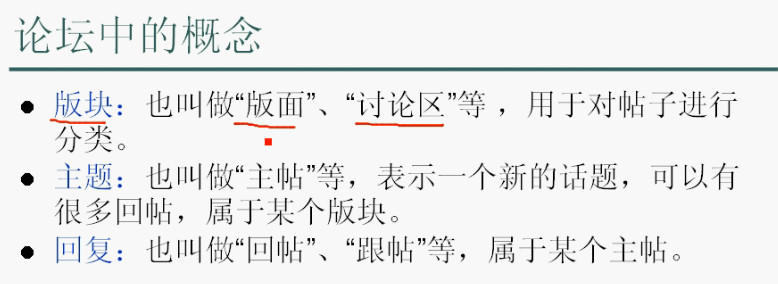
3，对应的jsp页面





4，配置struts.xml页面

# 论坛管理ForumManagerAction



帖子的分类：置顶、精华、普通。（热门、推荐、锁定、...）

普通帖：...

置顶帖：排到最前面。（公告帖）

精华帖：不影响排序，只是一个标志，可以列出所有的精华帖。

-------

热门帖：只是一个标志。

锁定帖：不能回复。

主题列表的排序：

最新状态的帖子排到最前面，置顶帖在最上面。

分页

数据列表的分页效果：

分页面 + 过滤条件 + 排序条件

回帖列表的排序：

最新的回复排到最下面（最后一页）

分页

前几个回复的称呼：

1楼、2楼、3楼、...

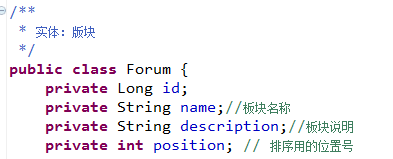
楼主、沙发、板凳、地板、...

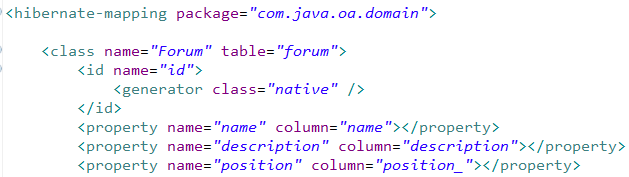
### 板块管理

所需功能：增删改查 上移下移



**暂时只写基础字段**





准备dao service action struts配置文件

然后完成基本的增删改查方法

版块管理中的上下移动

==============================================================

1，position的值不能重复。

2，列表显示时应按照positon的值排序（升序）。

3，上下移动就是与上面的或下面的Forum交换position的值。

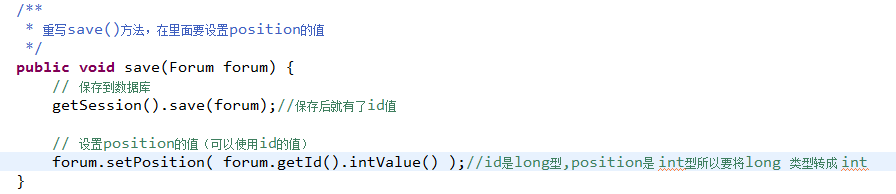
select \* from itcast\_forum where position\_ = (

select max(position\_) from itcast\_forum where position\_<9

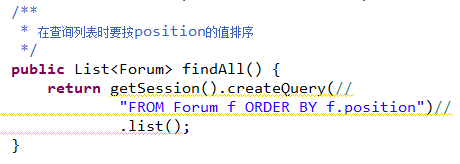
);

select \* from itcast\_forum where position\_<9 order by position\_ desc limit 0,1;

在保存的时候我们需要设置位置号



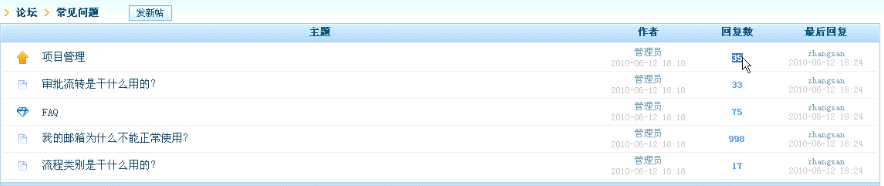
显示列表的时候按照位置号查询





### 主题管理

所需功能：增删改查 设置类别 移动主题到其他板块

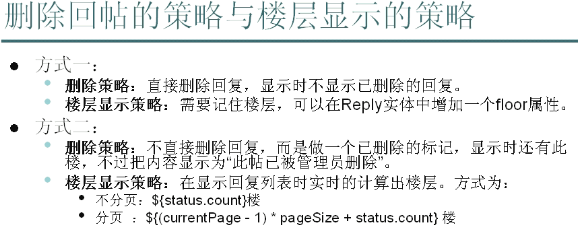


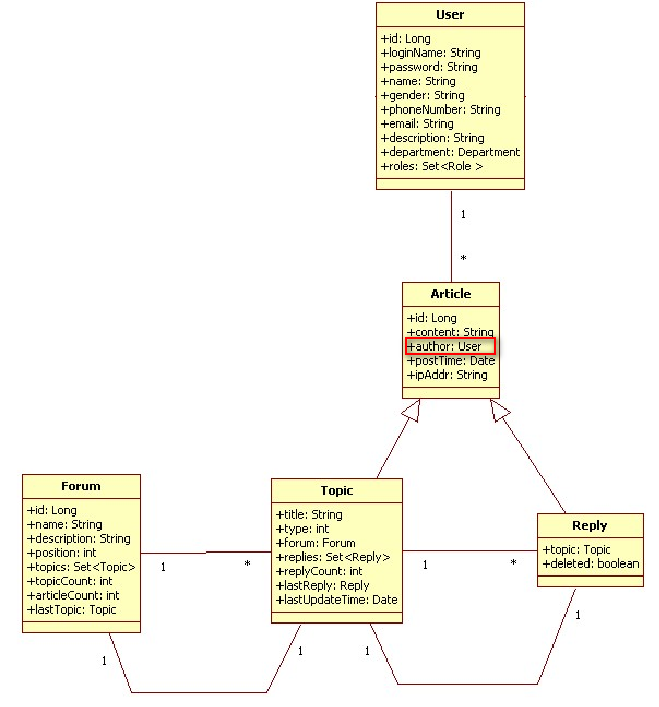
主题和回复又很多相同的字段这里我们新建一个Artical实体用于让主题实体和回复实体继承

### 回复管理

回帖的楼层是不能变的;

删除回帖 使用一个标志位





### 验证

客户端验证（JavaScript） -------- 服务器端验证（Java）

（格式验证） （格式验证 + 业务验证）

*jquery.validate.js*



# 论坛

版块列表 1个请求 ForumAction.list()

显示单个版块（主题列表） 1个请求 ForumAction.show()

发新帖 2个请求TopicAction.addUI() TopicAction.add()

显示单个主题（主帖+回帖列表） 1个请求TopicAction.show()

回帖 2个请求ReplyAction.addUI() ReplyAction.add()

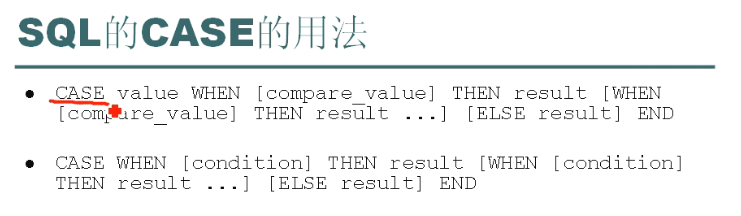




### 板块

1、左侧连接进来直接显示所有的板块列表

2、显示单个板块中的主题列表 排序：最新状态的排到前面，置顶帖在最上面。



1. 特殊属性的说明

|  |  |  |
| --- | --- | --- |
| Forum | topicCount | 主题数量 |
| articleCount | 文章数量（主题+回复） |
| lastTopic | 最后发表的主题 |
| Topic | replyCount | 回复数量 |
| lastReply | 最后发表的回复 |
| lastUpdateTime | 最后更新时间（主题发表的时间或最后回复的时间） |

1. 特殊属性的维护

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **发表新主题** | **发表新回复** |
| Forum | topicCount | 加1 |  |
| articleCount | 加1 | 加1 |
| lastTopic | 更新为当前的新主题 |  |
| Topic | replyCount | 0 | 加1 |
| lastReply | null | 更新为当前的新回复 |
| lastUpdateTime | 本主题的发表时间 | 更新为当前新回复的发表时间 |

### 主题

1. 添加主题



**action**



**service**



1. 显示单个主题



### 回复

1. 发表回复页面
2. 发表回复

**Action**



**Service**



### 在线编辑器

在线编辑器：

FCKeditor （旧）（Html + Css + JavaScript + Servlet处理上传的文件）

CKeditor （新）

就是用JS实现的一个网页版的html代码生成器，最终的结果是一段html代码。

原理是动态生成html代码。

1，把目录 fckeditor/ 拷贝到工程中就可以使用了。

2，如果要使用文件上传的功能，还需要配置服务器端：

1，把相关jar包添加到项目中

2，把配置文件fckeditor.properties放在classpath根目录中。

3，在web.xml中配置一个接收上传文件的 ConnectorServlet。

说明：配置服务器端的样本与jar包在目录 "FCKeditor服务器端所需的配置等" 中。

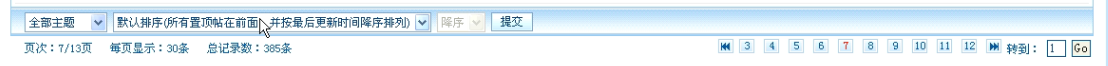
### 导入数据演示

使用工具SQLyog

### 分页

**论坛中的主题列表和回复列表都是需要分页的**

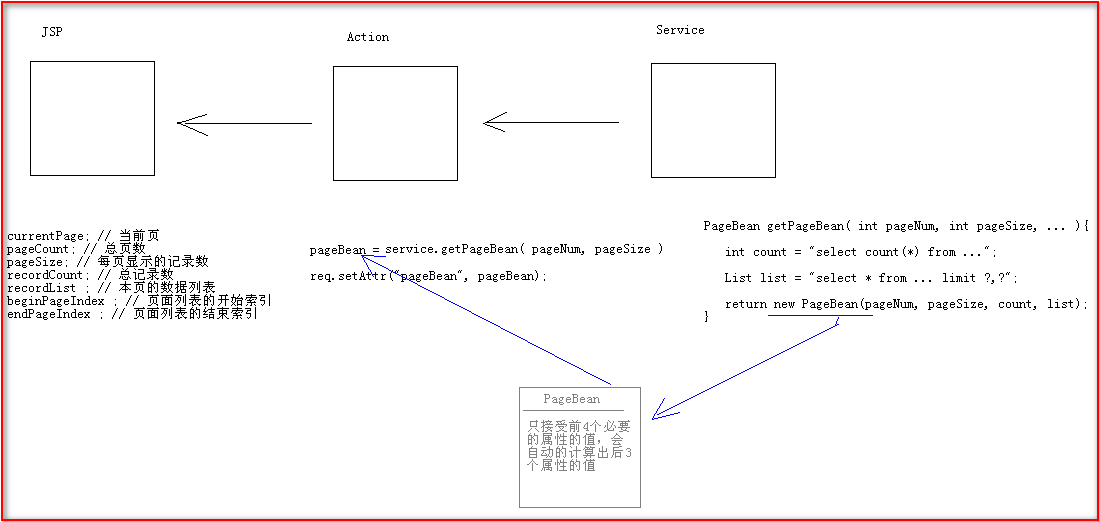
**主题页面的分页还有筛选的功能**



**回复页面的分页**



##### 分析



=================================================

功能---页次： ${currentPage}/${pageCount}页

功能---每页显示： ${pageSize}条

功能---总记录数： ${recordCount}条

功能---本页的数据列表: ${recordList}

<s:iterator value="%{recordList}">

id

name

...

</s:iterator>

功能---3 4 5 6 7 8 9 10 11 12 13

<s:iterator begin="%{beginPageIndex}" end="%{endPageIndex}" var="num">

<a href="javascript:gotoPage(${num})">${num}</a>

</s:iterator>

功能---转到：

<select id="pn" onchange=" gotoPage( this.value ) ">

<s:iterator begin="1" end="%{pageCount}" var="num">

<option value="${num}">${num}</option>

</s:iterator>

</select>

<script>

function gotoPage( pageNum ){

...

}

</script>

**分析后Jsp页面中需要的数据：**

currentPage; // 当前页

pageCount; // 总页数

pageSize; // 每页显示的记录数

recordCount; // 总记录数

recordList ; // 本页的数据列表

beginPageIndex ; // 页面列表的开始索引

endPageIndex ; // 页面列表的结束索引

============================================================

##### 2.创建pageBean

|  |
| --- |
| **public** **class** PageBean {  // 传递的参数或配置的值  **private** **int** currentPage; // 当前页  **private** **int** pageSize; // 每页显示的记录数  // 查询数据库  **private** **int** recordCount; // 总记录数  **private** List recordList; // 本页的数据列表  // 计算出来的  **private** **int** pageCount; // 总页数  **private** **int** beginPageIndex; // 页面列表的开始索引  **private** **int** endPageIndex; // 页面列表的结束索引  /\*\*  \* 只接受前4个必要的属性的值，会自动的计算出后3个属性的值  \*  \* **@param** currentPage  \* **@param** pageSize  \* **@param** recordCount  \* **@param** recordList  \*/  **public** PageBean(**int** currentPage, **int** pageSize, **int** recordCount, List recordList) {  **this**.currentPage = currentPage;  **this**.pageSize = pageSize;  **this**.recordCount = recordCount;  **this**.recordList = recordList;  // 计算pageCount  pageCount = (recordCount + pageSize - 1) / pageSize;  // 计算begPageIndex和endPageIndex  // a, 总页数不超过10页，就全部显示  **if** (pageCount <= 10) {  beginPageIndex = 1;  endPageIndex = pageCount;  }  // b, 总页数超过了10页，就显示当前页附近的共10个页码（前4个 + 当前页 + 后5个）  **else** {  // 显示当前页附近的共10个页码（前4个 + 当前页 + 后5个）  beginPageIndex = currentPage - 4; // 7 - 4 = 3  endPageIndex = currentPage + 5; // 7 + 5 = 12  // 当前面不足4个页码时，就显示前10页  **if** (beginPageIndex < 1) {  beginPageIndex = 1;  endPageIndex = 10;  }  // 当后面不足5个页码时，就显示后10页  **else** **if** (endPageIndex > pageCount) {  endPageIndex = pageCount;  beginPageIndex = pageCount - 10 + 1; // 注意在显示的时候是包含两个边界的  }  }  }  **关于 private** **int** pageSize; // 每页显示的记录数 **我们只让管理员管理，通过配置文件修改** |

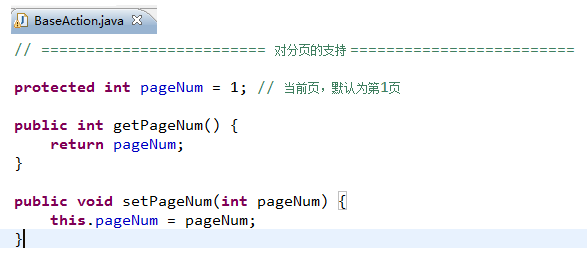
**记住一点：pageBean中已经包含了页面中需要展示的数据列表信息**

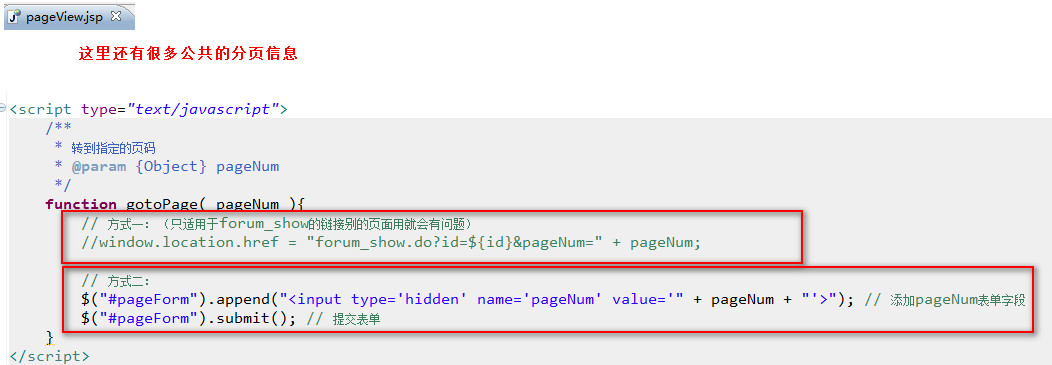
##### 3.实现分页 版本1

**步骤：**

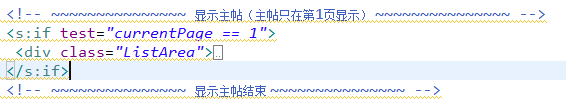


**一点改进：提取公共代码**





**在显示回复分页的时候主贴只在第一页显示**



**在显示回复分页的时候楼层的显示**



##### 4.实现分页 版本2

现在有两个分页

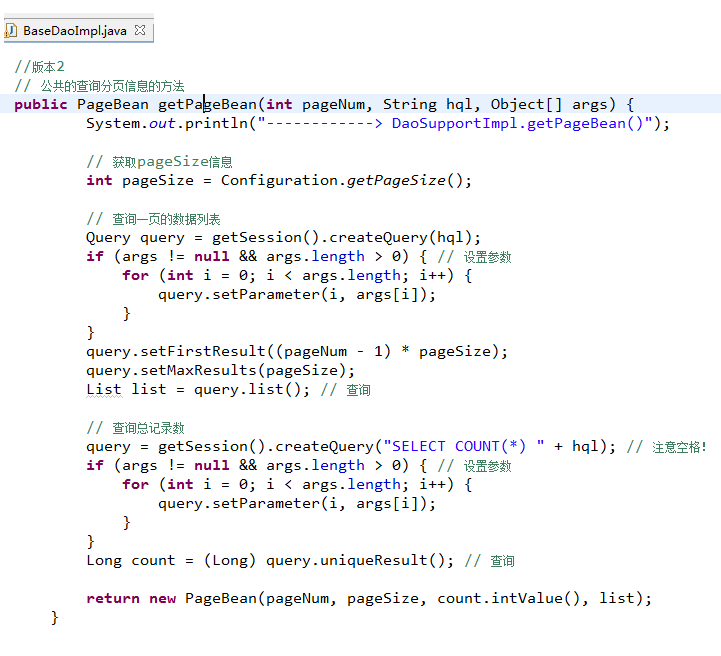
板块中主题的分页 核心代码在TopicDaoImpl中 为版本1

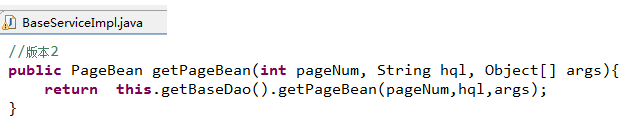
主题中回复的分页 核心代码在ReplyDaoImpl中 为版本1

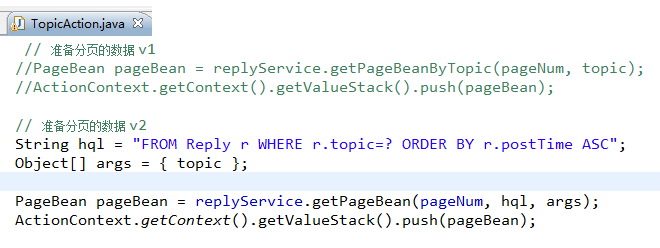
提取公共代码到baseDaoImpl中并在baseServiceImpl中调用

然后 修改板块中主题的分页 为版本2

然后 修改主题中回复的分页 为版本2





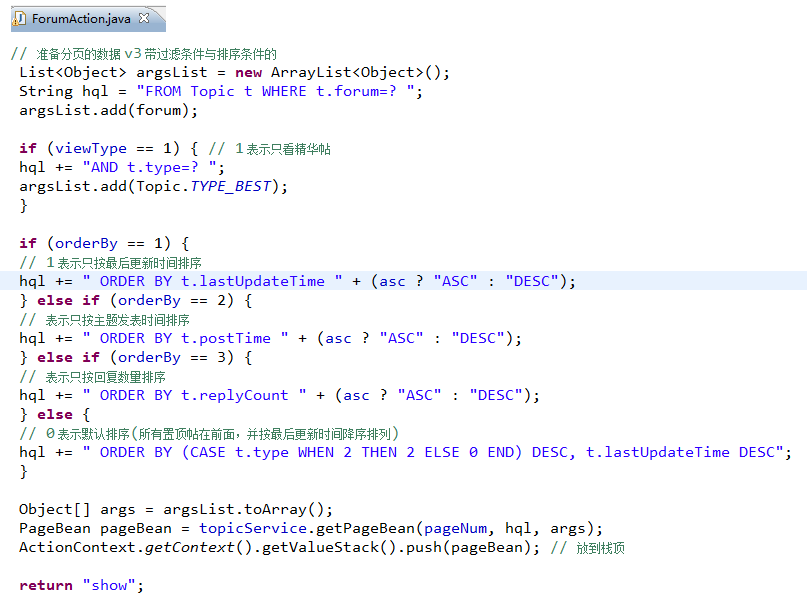


##### 带条件的分页 版本3

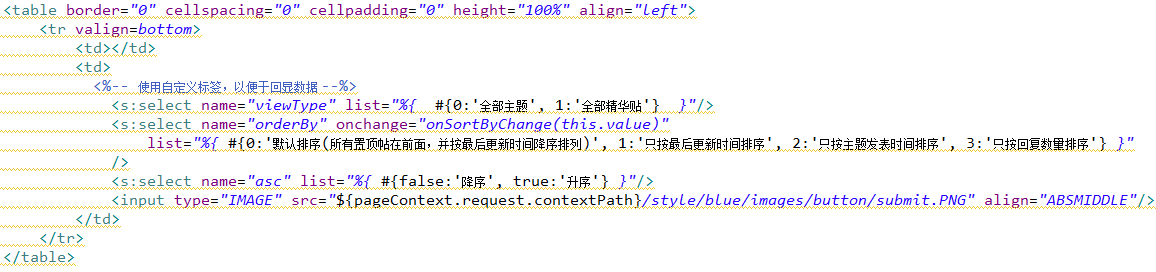
**以板块中主题的分页为例**

条件就是参数 我们将条件参数传到action中然后拼接成sql语句根据条件查询





当通过连接跳转到下一页的时候要将现在的查询参数也带过去，回显查询条件



截止到版本三的一些问题

1. Action拼接hql语句

HQL

参数

HQL:

FROM

实体

WHERE

条件1 AND 条件2 AND ... 条件n

ORDER BY

属性1, 属性2, ... 属性n

id=?

id between ? and ?

id is null

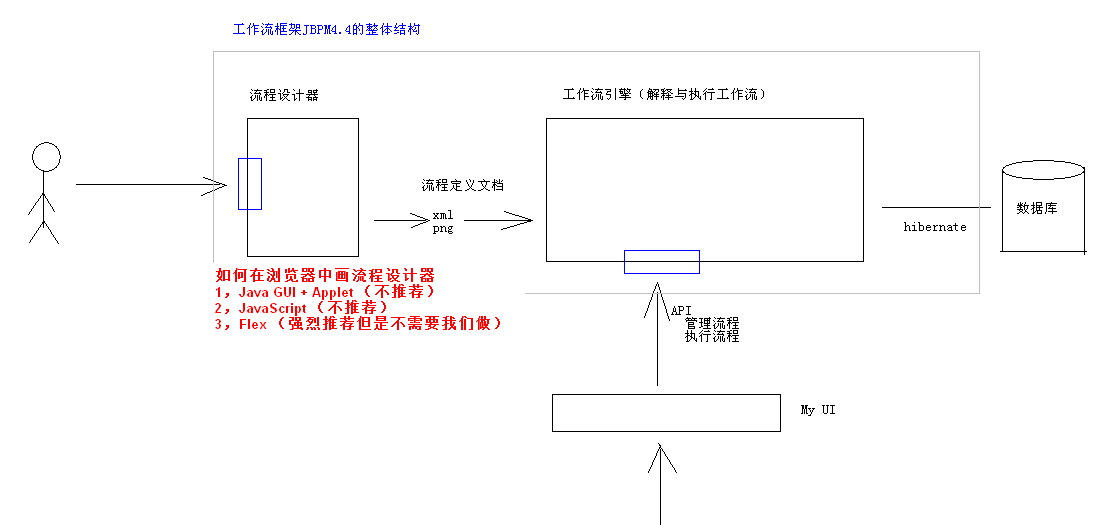
##### 带条件的分页 最终版

**QueryHelper帮助类 作用是拼接hql语句**

##### 给用户列表添加分页

**注意分页后页面中的数据就在pageBean中的recordList 了**

# JBPM



### Helloworld

1、添加jar包 15个

2、添加配置文件 2个 并修改数据库配置

**public** **class** HelloWorld\_API {

//0，创建流程引擎

//1，部署流程定义文档

//2，启动流程实例

//3，查询我的任务列表

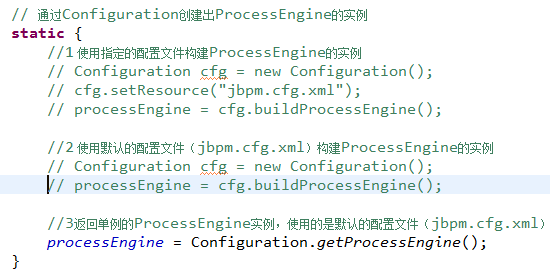
//4，办理任务

}

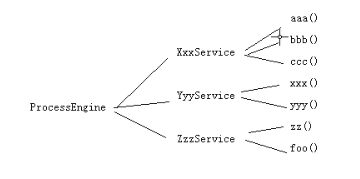
### 管理流程执行流程

##### 1.产生ProcessEngine

Configuration 表示配置信息。三种方式



##### 2.ProcessEngine 含有所有与流程有关的操作全局只有一个就可以了所以一般写成静态



xxService中的每个方法都是一个与流程有关的业务操作。

在方法中已经管理了事务。

链式的方法调用。

**各个Service的作用：**

|  |  |
| --- | --- |
| **RepositoryService** | **管理流程定义** |
| **ExecutionService** | **管理流程执行的，包括启动、推进、删除Execution等操作** |
| **TaskService** | **管理任务的** |
| HistoryService | 历史管理（执行完的数据管理，主要是查询） |
| IdentityService | jBPM的用户、组管理 |
| ManagementService |  |

**其中有三个xxService很重要**

getRepositoryService() 管理流程定义有关的API。

getExecutionService() 执行流程实例的有关API。

getTaskService() 跟Task有关的操作的API。

##### 3.我们对流程的操作一般分为两类

**一、管理流程定义** ProcessDefinitionTest

部署（添加）

删除

查询（查询所有、自定义条件查询）

获取部署添加的文件信息（查看流程图）

！！没有更新操作！策略为：重新部署 + 版本递增。

**二、执行流程实例** ProcessInstanceTest

启动流程实例

查询我的任务列表

办理任务

直接向后执行一步

##### 4.管理流程定义 ProcessDefinitionTest 涉及的对象的概念

**部署对象 Deployment**

表示一次部署的多个文件的信息。

对应的表：

jbpm4\_deployment

jbpm4\_lob

**流程定义 ProcessDefinition**

表示一个工作流程，包含每个环节的详细信息。

这是解析xx.jpdl.xml文件得到的数据。

此类没有对应的表，他对应的是.jpdl.xml文件。

##### 5.执行流程实例 ProcessInstanceTest 涉及的对象的概念

**执行对象 Execution**

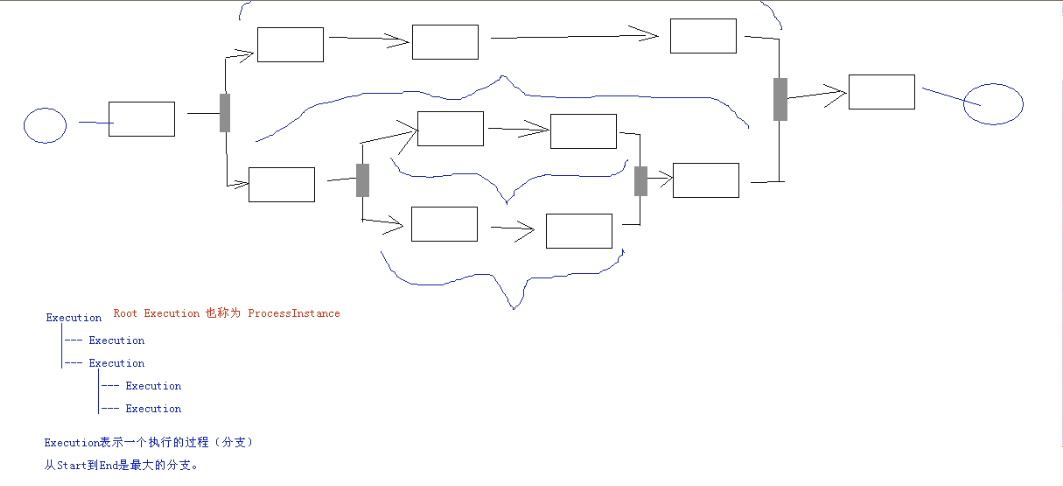
表示一个执行的路径（分支）。最大的分支就是从Start到End的那个。

最大的分支称为Root Execution 也称为 **流程实例 ProcessInstance**。

对应的表：

jbpm4\_execution 正在执行的信息表

jbpm4\_hist\_procinst 已经执行完的历史信息表



**流程实例 ProcessInstance**

表示按某流程定义执行的所有执行**时**的信息。

**ProcessInstance就是Execution对象（的子类型）**

对应的表：

jbpm4\_execution 正在执行的信息表

jbpm4\_hist\_procinst 已经执行完的历史信息表

**任务 Task**

在执行流程的过程中，在Task环节生成的任务信息。

对应的表：

jbpm4\_task 正在执行的信息表

jbpm4\_hist\_task 已经执行完的历史信息表

**流程变量**

A.设置流程变量

如果流程变量是一个实体类 必须实现**Serializable序列化接口**



B.获取流程变量

### 流程设计器

**流程的组成：**

1，流转 Transition （连线...）

2，活动 Activity （节点、环节、步骤...）

3，事件 Event

**A活动 Activity**

预先定义好的活动：

Start 开始活动

End 结束活动

State 状态活动不是任务 那如何执行 直接向后执行一步

Decision 判断活动

Fork/Join 分支/聚合活动

Task 任务活动

Custom活动 + ExternalActivityBehaviour接口

==============Task活动（个人任务的）==================

Task活动

名称

办理人

指定（个人任务的）办理人的方式：

1，直接指定为某人：

在.jpdl.xml写 <task assignee="员工" ...>

2，使用流程变量：

在.jpdl.xml写 <task assignee="#{employee}" ...>

3，使用AssignmentHandler

在.jpdl.xml中的<task>中写子元素<assignment-handler

class="com.AssignmentHandlerImpl"/>

要求自己的AssignmentHandlerImpl要实现AssignmentHandler接口

代码：

assignable.setAssignee(userId); // 指定此任务的办理人

重新指定任务的办理人：

processEngine.getTaskService().assignTask(taskId, userId);

=================Task活动===============

==============Task活动（个人任务的）==================

指定（组任务的）办理人的方式：

1，直接指定为某人：

在.jpdl.xml写 <task candidate-users="经理A,经理B,经理C" ...>

2，使用流程变量：

在.jpdl.xml写 <task candidate-users="#{managerIds}" ...>

要求managerIds变量的值是一个字符串，例如："经理A,经理B,经理C"

3，使用AssignmentHandler

在.jpdl.xml中的<task>中写子元素<assignment-handler

class="cn.itcast.AssignmentHandlerImpl"/>

要求自己的AssignmentHandlerImpl要实现AssignmentHandler接口

代码：

assignable.addCandidateUser("经理A"); // 添加一个候选人，这就是一个组任务了

assignable.addCandidateUser("经理B"); // 添加一个候选人，这就是一个组任务了

assignable.addCandidateUser("经理C"); // 添加一个候选人，这就是一个组任务了

==============Task活动（个人任务的）==================

**B事件 Event**

Event 事件

事件类型

流程开始 start

流程结束 end

活动开始 start

活动结束 end

指定代码

<on ...>

### JBPM4.4与Spring整合

目的：

1，由Spring管理单例的ProcessEngine.

2，由Spring管理事务.

**直接在OA项目中**

1、Jar包冲突



2、配置文件

A. 将jbpm的映射文件加入到hibernate的映射文件中

B. jbpm.cfg.xml

<import resource=*"jbpm.businesscalendar.cfg.xml"* />

<!-- 删除此项配置

<import resource="jbpm.tx.hibernate.cfg.xml" />

-->

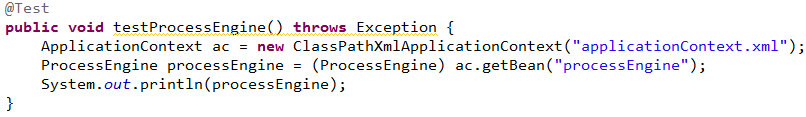
<!-- 导入jbpm.tx.spring.cfg.xml文件 -->

<import resource=*"jbpm.tx.spring.cfg.xml"* />

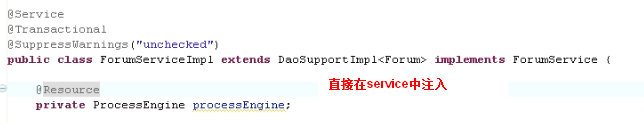
1. 在spring配置文件中配置processEngine

|  |
| --- |
| **<!-- 配置JBPM的ProcessEngine，注意配置文件的路径要写正确！ -->**  **<bean id=*"springHelper"* class=*"org.jbpm.pvm.internal.processengine.SpringHelper"*>**  **<property name=*"jbpmCfg"* value=*"jbpm.cfg.xml"*></property>**  **</bean>**  **<beanid=*"processEngine"*factory-bean=*"springHelper"*factory-method=*"createProcessEngine"* />** |

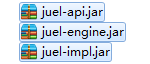
1. 测试



1. 使用

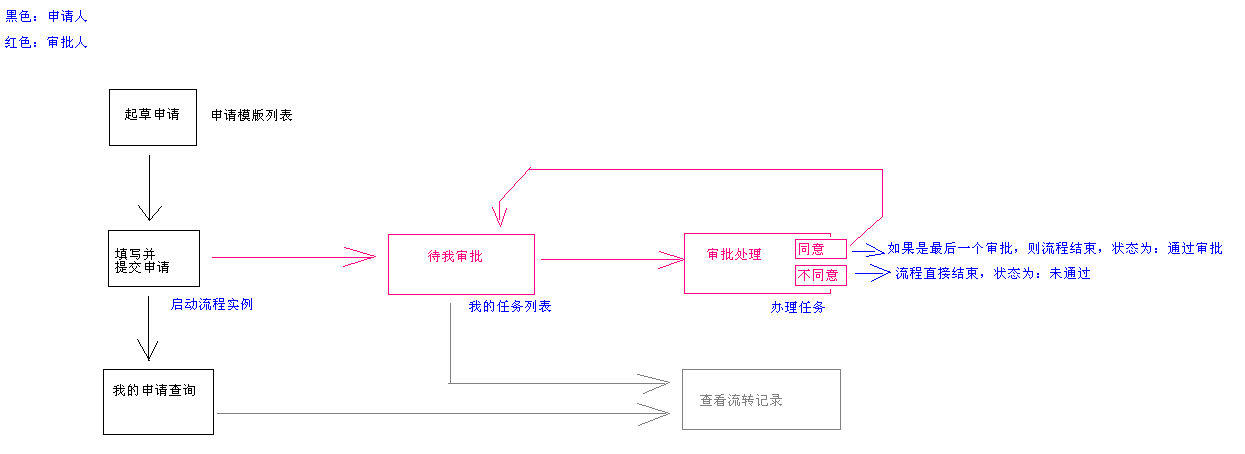


1. 发布到tomcat发现有jar包冲突



将这三个包在工程中删除放到tomcat/lib中并删除tomcat/lib中的el-api.jar

# 流转审批



##### 审批流程管理

就是流程定义的增删查

只需要action 和 service 记住这个service要单独加上@Transactional 事务

============================================================

【Strtus2中的文件上传】

一、JSP中

<form method="post" enctype="multipart/form-data" ...>

<input type="file" name="myupload"> **<!-- 此类型的字段不能回显 -->**

...

二、Action中

public class MyAction{

private File myupload; // 上传的文件

private String myuploadFileName; // 上传文件的原始的名称

private String myuploadContentType; // 上传文件的类型

// setter与getter略..

}

如果upload==null表示没有上传文件（没有选择文件）。

============================================================

============================================================

【Strtus2中的文件下载】

一、Action中

public class MyAction{

private InputStream inputStream; // 下载文件用的输出流

// getter与setter方法..

public String mydownload(){

inputStream = new FileInputStream("d:/a.png");

return “mydownload”;

}

}

二、struts.xml中

<result name=“mydownload” type="stream">

<param name="inputName">inputStream</param>

</result>

============================================================

如果我们上传的流程定义压缩包是中文的，里面的xml和png都是中文的此时就会有乱码问题

因为此时流程定义的key值是中文的，那么在删除和查看流程图的时候会出现这种情况：





============================================================

【**在JSP中保证使用utf-8编码传递的中文参数（GET方式）**】

情况一：在<s:a>标签中应使用<s:param ..> 这种形式会自动进行url编码

<s:a action="...">

<s:param name="key" value="%{key}"></s:param>

**不要用**

<s:a action="processDefinition\_delete?key=%{key}">删除</s:a>

情况二：在javascript代码中

pdId = encodeURI(pdId); // 使用js的函数进行URL编码

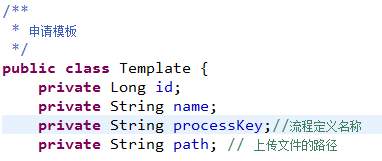
var url = "processDefinition\_showProcessImage.do?id=" + pdId + "&t=" + new Date();

================================================================

##### 申请模板管理

1.设计实体

该实体和流程有关系要在该实体中保存流程名称



2. 将映射文件添加到hibernate.cfg.xml中

1. 增删改查
2. 删除注意删除数据库的同时也要删除模板文件
3. Struts上传与下载
4. 修改时要注意 如果upload==null表示没有上传文件（没有选择文件）



================================================================

【上传的文件的存储方案】

方案一：存到数据库中（blob）

方案二：存到文件夹中（服务器端）【推荐】

1，文件重名，就般使用UUID做为文件名。如果需要原文件名，就存到数据库中。

2，文件应分文件夹存放，不要在一个文件夹中存放过多的文件。可以使用日期划分文件夹。

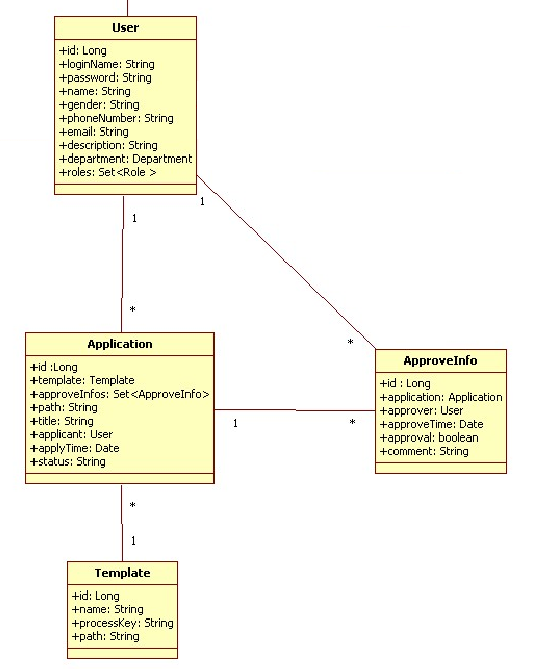
================================================================

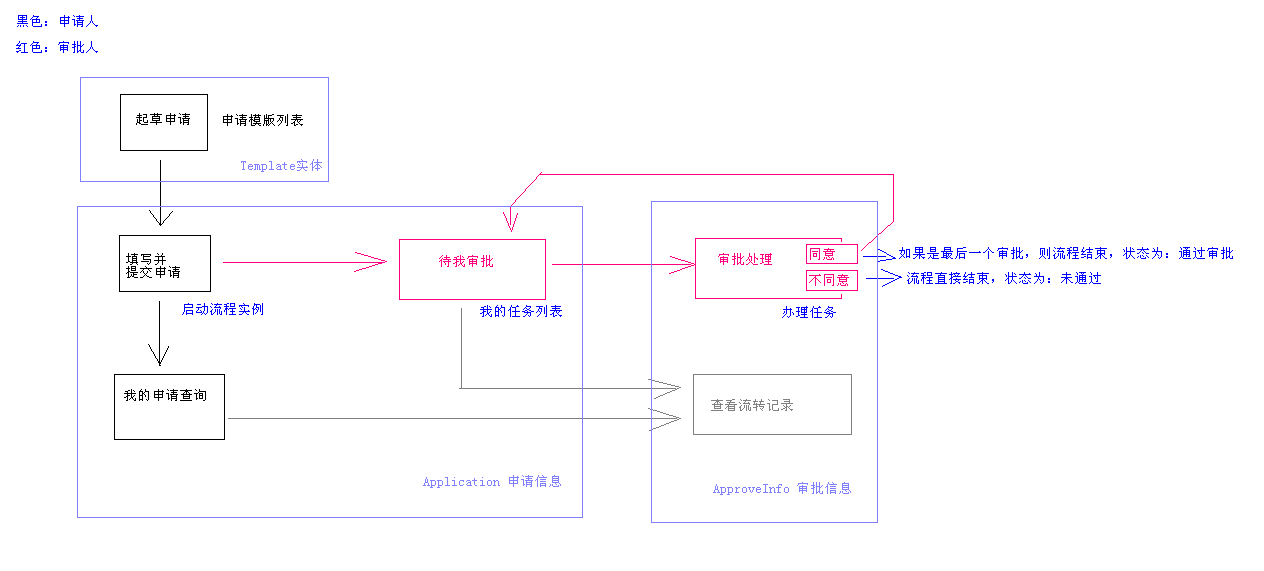
##### 3.起草申请（申请模板的选择）

**Application 申请信息**

**ApproveInfo 审批信息**







1. 设计实体
2. 写映射文件
3. 修改Template.java和Template.hbm.xml文件关于Application的关系
4. 将映射文件添加到hibernate.cfg.xml文件中

##### 4审批流转的约定

一个task有两个必须要有的信息 任务名称 和 办理人

审批流程：

第一个环节是“提交申请”，以后每个环节基本上都是“审批”环节。

1，第一个活动是“提交申请”，办理人是申请人，也就是提交申请时的当前登录用户。

2，在我们的程序中，使用User对象表示用户，在JBPM中使用字符串表示参与者。

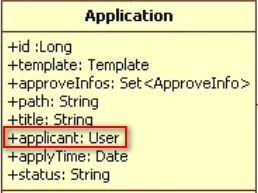
可以使用User中的一个不会重复的属性值作为JBPM的用户标识。

比如可以是id.toString()或是loginName.

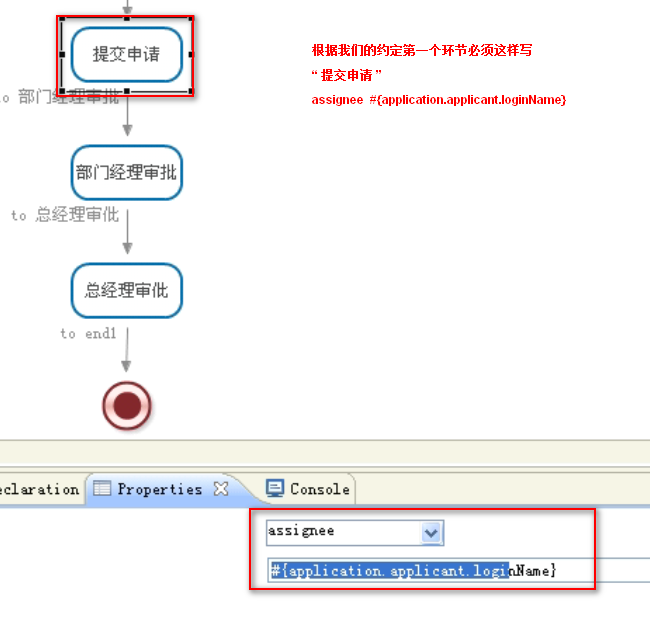
3，在第一个“提交申请”的环节中会产生一个“申请信息”，

要求在流程中后面的每个环节中都应能方便的获取这个“申请信息”。

可以在“提交申请”环节设置一下流程变量：setVariable("application", application);



4，（对应第一条）“提交申请”环节的办理人应写为：assignee="#{application.applicant.loginName}"



============================================================================================

##### 5流转

1、准备数据：

1，流程 定义 一条线 和 两条线 两种

2，模版

3，用户等数据

姓名 登录名 角色

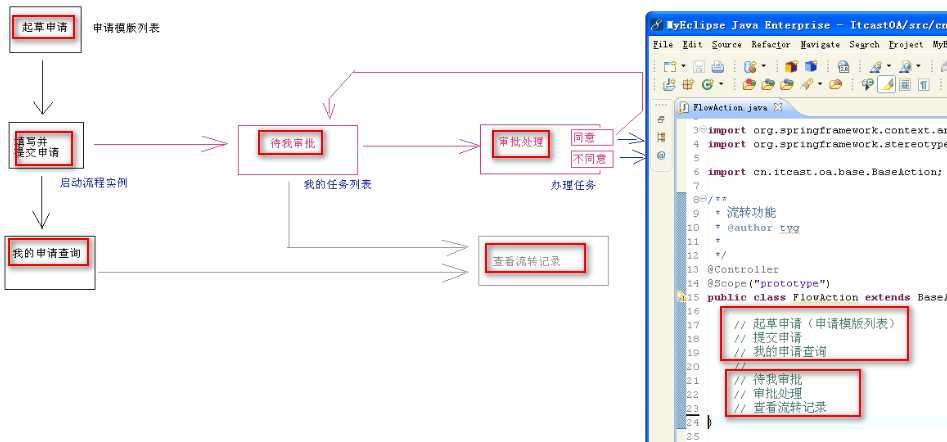
-----------------------------

张三 zs 员工

李四 ls 部门经理

王五 ww 总经理

2、功能分析



A 起草申请要跳转到模板选择页面

B 选择模板后跳转到提交申请页面

C 申请成功后跳转到申请查询页面

3.处理多条线路



4.

