# 登陆功能

1. 查询数据库，来验证登陆
2. 登陆成功的用户存入session
3. Login拦截器，判断session中有用户，才能放行操作，没有则返回登陆界面
4. 注销用户按钮，从session中移除用户，返回登陆界面

# 请假管理

#### 采取做法：后台直接丢数据，让前台自己拼凑

1. 列出当前用户申请请假的列表
   1. 根据列表的申请状态来定义不同的操作（0：初始录入、1：审核中、2：审核通过）
   2. 后台：workflowAction-listLeavebill
   3. 前台：grid
2. 添加申请请假填写表单功能（可以和 修改界面共用）
   1. 跳转到请假表单页面
   2. 前台：toolbar
3. 启动请假申请（正式启动工作流）
   1. 前台：grid中的操作按钮
   2. 设置业务状态为1
   3. 启动参数有3个：流程定义id、业务标识、流程实例参数（即map数组，主要存放申请人id）
      1. 流程定义id：前台传参
      2. 绑定流程与业务，使其相关联，以bkey来保存业务类+ id 按钮
      3. 流程实例参数，用session中的user属性
4. **查看审核记录按钮**
   1. 使用请假单id，查询请假单实例，压入栈顶，用于表单回显
   2. 使用请假单Id，查询历史评论表
5. 操作按钮分别为（启动申请、**修改、删除**）
6. 请假表单，审核中和审核通过状态时，不能启动，修改和删除（将来考虑可撤回）

# 绘制流程图

1. 流程申请人（发起人），位于第一个task节点，通过在assignee设置${performer}来定义流程发起人的变量，在完成task1的时候给其设置该流程变量
2. 可通过task节点的formKey来动态指定该流程对应的表单连接（因为每个流程都有属于自己的表单），再完成该task前，把formkey取出，丢入值栈，**再指定struts的result将从值栈中取出**。

# 部署管理

1. 前台输入文件名称、文件，随后提交（可用form或ajax）
2. 后台取得文件流，进行部署操作（可选择modeldriven）
3. 后台提供流程定义查询（用于启动流程，可查看流程图片）
   1. 通过部署id，图片名称，来获取数据库中的数据流
   2. 用servletActionContext。GetResponse()。GetoutPutStream()来输出图片流
4. 后台提供部署对象查询（可删除部署）

# 任务管理

1. 从session中取出当前用户
2. 用用户来查询正在执行的任务列表，获取任务集合
3. 通过实现taskListener接口来设置任务候选人
   1. 设置方法为获取当前用户id
   2. 查询出用户的经理人
      1. 先用webapplicationContextUtils来取得ac
      2. 在用ac取得用户service的bean
      3. 用service获取当前用户对象，取出其上级
   3. 设置进assignee中，重写notify方法
      1. 用notify的参数delegate来设置
4. 点击“任务办理”按钮，进入表单界面
   1. 利用任务id查询出流程id再查询出businessKey，通过bk来获取业务表单
   2. 将业务表单压入栈顶
   3. 前端只读表单用disable来操作
   4. 根据连线来添加操作按钮，如“驳回”、“同意”
      1. 根据任务id获取流程定义对象（processDefinitionEntity）
      2. 根据流程定义对象获取当前活动（ActivitiImpl）
      3. 根据当前活动获取完成之后的连线（getOutgoingTransaction）
      4. 遍历PvmTransition调用getProperty(‘name’)的方法，来获取连线的名称
      5. 将每个连线名称存入list
      6. 将list存入栈顶
   5. 办理任务界面，如果流程的发起人，是当前登录用户的话，那完成按钮改为“提交申请”（思路，assignee）
      1. 解决方案：http://www.kafeitu.me/activiti/2012/05/20/set-process-start-user.html
   6. 根据任务id查询评论表，将评论list压入栈顶，供前台使用
      1. 根据当前任务id，查询流程实例
      2. 根据流程实例，查询历史任务列表
      3. 遍历历史任务列表，用taskid来获取评论列表
      4. 利用taskService的getTaskcomment来获取评论
      5. 整合评论列表压入栈顶，供前台使用
   7. 点击操作按钮来完成任务
      1. 用activiti自定义表添加批注，添加批注之前需要设置当前审核人的id，（用Authentication的set方法来设置），不设置的话评论表会没有评论的用户id
      2. 前台传递连线名称，后台获取
      3. 如果连线名称为“默认提交”，则不进行连线判断变量的设置，反之则反
      4. 完成任务的同时，设置流程变量（用于判断连线分支）
      5. 完成任务后，要指定下一个任务的办理人（transactionListener）
      6. 完成任务后，判断如果流程结束，业务表单的状态从1变成2
         1. 根据任务id，获取历史流程实例
         2. 用历史流程实例，获取EndTime
         3. 如果EndTime不为空，则表示流程实例结束，然后转换业务状态
5. 显示流程图按钮
   1. 根据任务id查询流程图，并显示到页面
      1. 根据任务id，获取流程部署对象id，压入栈顶
      2. 根据任务id，获取当前活动节点变量的map，压入栈顶
         1. 根据任务id，获取流程定义对象（ProcessDefinitionEntity）
         2. 根据流程对象获取当前活动对象id（可能是个集合）
         3. 通过活动对象id，用流程定义entity获取活动对象
         4. 把当前活动对象的坐标，长宽封装进map中，返回map
   2. 流程图页面结构
      1. 流程图img，src=action名
      2. 当前活动节点框，div：设置颜色，边框，坐标，长宽（均动态获取）
      3. 流程图和活动节点都要用绝对位置

# 存在问题

modelDriven不能取得url的参数

# Xml to 菜单

1. 登陆：需要参数：用户名、密码、xmlFilePath
2. 从本地读取xml（outservlet）
   1. 利用io，路径一般为 xmlFilePath=d:\\xmlFilePath\\
   2. 分别读取到user基本信息，及user功能信息
   3. User基本信息直接存入session
   4. User功能信息封装成menu，将menu置入session，供前台使用（具体操作跳至第2点）
3. 转换user功能部分xml到json，置入menu
   1. 直接转json ，自己写
   2. 先转成bean再转成json 参考：zkpt的bean和utils
4. 前台获取json，调用就是方法拼凑菜单
   1. 通过session获取json
   2. 调用trendsMenus.js来拼凑
5. 自己写测试类，用于生成对应格式的xml
   1. 需要得到不同权限的用户