Spring集成jsf开发：

另：Mybatis，shiro；

Maven:

Jar；

配置文件：

Web.xml；

faces-config.xml:只需要和spring集成的配置；| 自定义组件是另外的；

Spring配置，applicationContext.xml；

Shiro配置，applicationContext-shiro.xml

Shiro需要的缓存，ehcache-shiro.xml

faces-config.xml:

//jsf,spring集成；DelegatingVariableResolver

<application>

<variable-resolver>

org.springframework.web.jsf.DelegatingVariableResolver

</variable-resolver>

<action-listener>com.cxdai.filter.ActionFilter</action-listener>

<locale-config>

<default-locale>en</default-locale>

<supported-locale>de</supported-locale>

<supported-locale>zh</supported-locale>

</locale-config>

</application>

~

ActionFilter类：

Action过滤，权限验证；

操作流程：

页面---action---service---mapper---xml；

~

页面控件，

Xml中的sql语句；

数据绑定机制：

JSF提供了数值绑定机制，使得在Faces标记中访问bean property 很容易，其原理是用 JSF-EL （表达式语言）把 JSF 组件绑定到受管理的 bean 属性。

·action中声明变量；

·页面，绑定action的变量；

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

登录：

登录页面，

A4j按钮；输入验证；返回登录错误提示；

~

loginAction，

声明变量：用户名，密码，错误提示信息；

//shiro认证，

**public** String login(){

String username = **this**.getUsername();

String password = **this**.getPassword();

UsernamePasswordToken token = **new** UsernamePasswordToken(username, password);

Subject currentUser = SecurityUtils.*getSubject*();

**try** {

currentUser.login(token);

//成功，跳转页面；

} **catch** (UnknownAccountException e) {

resultMsg = "用户名不存在";

}

//失败，返回错误提示；

return null;

}

//登出 | shiro方式；

**public** String logout(){

Subject currentUser = SecurityUtils.*getSubject*();

**try** {

currentUser.logout();

**return** "/index.xhtml?faces-redirect=true";

} **catch** (Exception e) {

log.error(e.getMessage());

}**finally**{

**return** "/index.xhtml?faces-redirect=true";

}

}

action权限检测：

页面url,action---配置文件(web.xml, faces-config.xml), ActionFilter类；

类中授权；

后台框架页面，islogin.xhtml

都是通过<ui:include src="dailyofficeworknav.xhtml"></ui:include>加入其它页面；

顶部，top.xhtml；

左侧菜单，dailyofficeworknav.xhtml

右侧，iframe；

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

分页：

列表页面，

<div class=*"buttondiv"*>

<ul>

<li class=*"lileave"*>

<font>第#{borrowCollectionAction.pages.curPage+1}页 / 共#{borrowCollectionAction.pages.totalPage &lt;

1?1:borrowCollectionAction.pages.totalPage}页共#{borrowCollectionAction.pages.totalCount==undefined?0:borrowCollectionAction.pages.totalCount}条

</font>

<a4j:commandButton value=*"首页"* styleClass=*"button"*

action=*"#{borrowCollectionAction.searchAll}"* execute=*"@form"* render=*"collectionSearchForm"*>

<a4j:param value=*"0"* assignTo=*"#{borrowCollectionAction.pages.curPage}"* />

<a4j:param value=*"1"* assignTo=*"#{borrowCollectionAction.searchBorrowCollection.status}"* />

</a4j:commandButton>

<a4j:commandButton value=*"上一页"* styleClass=*"button"*

action=*"#{borrowCollectionAction.searchAll}"* execute=*"@form"* render=*"collectionSearchForm"*>

<a4j:param value=*"#{borrowCollectionAction.pages.curPage-1}"* assignTo=*"#{borrowCollectionAction.pages.curPage}"* />

<a4j:param value=*"1"* assignTo=*"#{borrowCollectionAction.searchBorrowCollection.status}"* />

</a4j:commandButton>

<a4j:commandButton value=*"下一页"* styleClass=*"button"*

action=*"#{borrowCollectionAction.searchAll}"* execute=*"@form"* render=*"collectionSearchForm"*>

<a4j:param value=*"#{borrowCollectionAction.pages.curPage+1}"* assignTo=*"#{borrowCollectionAction.pages.curPage}"* />

<a4j:param value=*"1"* assignTo=*"#{borrowCollectionAction.searchBorrowCollection.status}"* />

</a4j:commandButton>

<a4j:commandButton value=*"末页"* styleClass=*"button"*

action=*"#{borrowCollectionAction.searchAll}"* execute=*"@form"* render=*"collectionSearchForm"*>

<a4j:param value=*"#{borrowCollectionAction.pages.totalPage-1}"* assignTo=*"#{borrowCollectionAction.pages.curPage}"* />

<a4j:param value=*"1"* assignTo=*"#{borrowCollectionAction.searchBorrowCollection.status}"* />

</a4j:commandButton>

</li>

</ul>

</div>

/////////////////////////////////////////////////////////////////////////

普通表单的数据添加、修改，都是通过实体类来传递页面数据；

~一些特殊的，通过action中声明参数，页面隐藏控件来传值；

主页面，可以直接取到include页面的值，使用跟include一样的参数；

1，参数传递：

Action中，声明变量，在页面和action间传递值；

**private** String borrowId;//借款记录id

**private** String resultMsg;//返回值

get/set下；

~

Id，直接使用；

页面，

<a4j:commandButton id=*"save"* value=*"保存"*

action=*"#{borrowMoneyAction.save}"*

execute=*"@form"*

data=*"#{borrowMoneyAction.resultMsg}"*

onclick="return validApprove();"

oncomplete=*"parent.waitHide();afterSaveDo2(event.data,#{rich:component('approveBorrowMoney')});"*>

<a4j:param value=*"getCheckboxIds('borrowSearchForm:borrowId')"* assignTo=*"#{borrowMoneyAction.borrowId}"* noEscape=*"true"* />

</a4j:commandButton>

消息提示，---返回普通提示；返回异常信息；返回自定义异常提示信息；

Map<String,Object> resultMap=new HashMap<String,Object>();

**try** {

//数据操作的结果，返回提示； approveNewBorrowService.saveNewBorrow(Long.*parseLong*(borrowId),okflag,memo,resultMap);

resultMsg=String.*valueOf*(resultMap.get("msg"));

} **catch** (Exception e) {

resultMsg=e.getMessage(); //可以接收到下面service中自定义的异常类信息；

//可以自己写异常；

//e.printStackTrace();

//resultMsg = "保存失败，请核对填写的数据";

log.error(e.getMessage());

}

Service中，自己写异常类：

**public** **void** saveBorrowMoney(**long** borrowId, **int** okflag, String memo,

Map<String, Object> resultMap) **throws** FuiouModException, Exception {

//成功，返回提示；

resultMap.put("msg", "放款成功");

} **else** {

**throw** **new** Exception("数据已过期,请刷新后重试");

}

自定义异常类，

**public** **class** FuiouModException **extends** Exception{

/\*\*

\* **@fields** serialVersionUID

\*/

**private** **static** **final** **long** ***serialVersionUID*** = -6597013124481645689L;

**private** String msg;

**public** FuiouModException(String msg) {

**this**.msg=msg;

}

/\* (non-Javadoc)

\* @see java.lang.Throwable#getMessage()

\*/

@Override

**public** String getMessage() {

**return** **this**.msg;

}

}

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

页面项目实际：

//一个面板，里面一个表格，表格循环数据；| <a4j:repeat

<rich:panel>

<font>当前借款人代收记录</font>

<table width=*"100%"* class=*"fulltable"*>

<tr class=*"tr\_head"*>

<th width=*"5%"*>序号</th>

<th width=*"10%"*>计划代收日期</th>

<th width=*"10%"*>代收本金</th>

<th width=*"10%"*>代收利息</th>

<th width=*"10%"*>代收总额</th>

<th width=*"10%"*>代收状态</th>

<th width=*"10%"*>复核结果</th>

<th width=*"31%"*>复核人</th>

</tr>

<a4j:repeat rowKeyVar=*"i"* value=*"#{borrowAction.collectionList}"* var=*"item"*>

<tr class=*'tr\_#{i%2}'*>

<td>#{i+1}</td>

<td><h:outputText value=*"#{item.planDate}"*><f:convertDateTime pattern=*"yyyy-MM-dd HH:mm:ss"* /></h:outputText></td>

<td>#{item.planPrinciple}</td>

<td>#{item.planInterest}</td>

<td>#{item.planTotal}</td>

<td>#{item.status}</td>

<td>#{item.approveResult}</td>

<td>#{item.approveName}</td>

</tr>

</a4j:repeat>

</table>

<div style="height:*20px*;"></div>

</rich:panel>

文本控件+隐藏控件：

文本控件显示不可修改值，隐藏控件传递值到action；

<h:outputText id=*"shownewCalculationDate"*

value=*"#{contractApplyAction.financialExamination.calculationDate}"*>

<f:convertDateTime pattern=*"yyyy-MM-dd"*></f:convertDateTime>

</h:outputText>

<h:inputText id=*"newCalculationDate"*

style="display:*none*"

value=*"#{contractApplyAction.financialExamination.calculationDate}"*>

<f:convertDateTime pattern=*"yyyy-MM-dd"*></f:convertDateTime>

</h:inputText>

~

另，使用<h:inputHidden

<h:outputText id=*"shownewenddate"*

value=*"#{contractApplyAction.financialExamination.redemptionDate}"*>

<f:convertDateTime pattern=*"yyyy-MM-dd"*></f:convertDateTime>

</h:outputText>

<h:inputHidden id=*"newenddate"*

value=*"#{contractApplyAction.financialExamination.redemptionDate}"* >

<f:convertDateTime pattern=*"yyyy-MM-dd"*></f:convertDateTime>

</h:inputHidden>

页面加载时，控件赋值：---控件的value为另外实体类的属性；

~所以不能直接使用include页面中的实体类来赋值；

~还是从后台绑定时，给出绑定参数(声明变量，实体类，可以有多个实体类，然后赋值)

el表达式使用：---判断值，显示不同内容；

<c:choose>

<c:when test=*"${contractApplyAction.currentExamination.customerName eq contractApplyAction.lastHistory.customerName}"*>

<h:outputText value=*"#{contractApplyAction.lastHistory.customerName}"* />

</c:when>

<c:otherwise>

<h:outputText value=*"#{contractApplyAction.currentExamination.customerName}"* class=*"hcolor"* />

<h:outputText value=*"#{contractApplyAction.lastHistory.customerName}"* />

</c:otherwise>

</c:choose>