

示例 SQL

MYSQL 版本

```
--  
-- Definition of table `tbl_cover_letter` 求职信表  
--  
DROP TABLE IF EXISTS `tbl_cover_letter`;  
CREATE TABLE `tbl_cover_letter` (  
  `cover_letter_uid` varchar(50) NOT NULL,  
  `cover_letter_company_name` varchar(100) default NULL,  
  `cover_letter_positon` varchar(50) default NULL,  
  `cover_letter_content` text,  
  `cover_letter_send_time` date default NULL,  
  `cover_letter_send_flag` bit(1) default NULL,  
  `cover_letter_person` varchar(50) default NULL,  
  PRIMARY KEY (`cover_letter_uid`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
--  
-- Definition of table `tbl_eduction_experience` 教育经历表  
--  
DROP TABLE IF EXISTS `tbl_eduction_experience`;  
CREATE TABLE `tbl_eduction_experience` (  
  `eduction_experience_uid` varchar(50) NOT NULL,  
  `eduction_experience_start_time` date default NULL,  
  `eduction_experience_end_time` date default NULL,  
  `eduction_experience_school_name` varchar(50) default NULL,  
  `eduction_experience_degree` varchar(50) default NULL,  
  `eduction_experience_person` varchar(50) default NULL,  
  PRIMARY KEY (`eduction_experience_uid`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
--  
-- Definition of table `tbl_peroject_experience` 项目经验表  
--  
DROP TABLE IF EXISTS `tbl_peroject_experience`;  
CREATE TABLE `tbl_peroject_experience` (  
  `peroject_experience_uid` varchar(50) NOT NULL,
```

```
`peroproject_experience_start_time` date default NULL,
`peroproject_experience_end_time` date default NULL,
`peroproject_experience_name` varchar(100) default NULL,
`peroproject_experience_position` varchar(50) default NULL,
`peroproject_experience_development_tool` varchar(50) default NULL,
`peroproject_experience_description` text,
`peroproject_experience_duty_description` text,
`peroproject_experience_person` varchar(50) default NULL,
PRIMARY KEY (`peroproject_experience_uid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

--

-- Definition of table `tbl_person` 人员信息表

--

```
DROP TABLE IF EXISTS `tbl_person`;
CREATE TABLE `tbl_person` (
  `person_uid` varchar(50) NOT NULL,
  `person_name` varchar(20) default NULL,
  `person_sex` varchar(5) default NULL,
  `person_birthday` date default NULL,
  `person_account` varchar(50) default NULL,
  `person_adress` varchar(50) default NULL,
  `person_email` varchar(50) default NULL,
  `person_work_start_time` date default NULL,
  `person_mobile` varchar(20) default NULL,
  `person_self_assessment` text,
  `person_job_category` varchar(10) default NULL,
  `person_salary_expected` varchar(20) default NULL,
  `person_fastest_entry_time` varchar(20) default NULL,
  PRIMARY KEY (`person_uid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

--

-- Definition of table `tbl_track_record` 工作经历表

--

```
DROP TABLE IF EXISTS `tbl_track_record`;
CREATE TABLE `tbl_track_record` (
  `track_record_uid` varchar(50) NOT NULL,
  `track_record_start_time` date default NULL,
  `track_record_end_time` date default NULL,
```

```
`track_record_company` varchar(50) default NULL,  
`track_record_dept` varchar(50) default NULL,  
`track_record_position` varchar(50) default NULL,  
`track_record_position_description` varchar(500) default NULL,  
`track_record_work_achievements` text,  
`track_record_belong_person` varchar(50) default NULL,  
PRIMARY KEY (`track_record_uid`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

示例本地化代码

cover_letter_company_name=求职公司名称
cover_letter_positon=求职公司岗位
cover_letter_content=求职信内容
cover_letter_send_time=发送日期
cover_letter_send_flag=是否发送
cover_letter_person=所属人

education_experience_start_time=开始日期
education_experience_end_time=结束日期
education_experience_school_name=学校名称
education_experience_degree=学历
education_experience_person=所属人

peroproject_experience_start_time=项目开始时间
peroproject_experience_end_time=项目结束时间
peroproject_experience_name=项目名称
peroproject_experience_position=所任职务
peroproject_experience_development_tool=开发工具
peroproject_experience_description=项目描述
peroproject_experience_duty_description=责任描述
peroproject_experience_person=所属人

person_name=姓名
person_sex=性别
person_birthday=生日
person_account=户口地
person_adress=居住地
person_email=电子邮件

person_work_start_time=开始工作时间

person_mobile=手机号码

person_self_assessment=自我评价

person_job_category=工作性质

person_salary_expected=薪水要求

person_fastest_entry_time=最快入职日期

track_record_start_time=开始工作日期

track_record_end_time=结束工作日期

track_record_company=公司名称

track_record_dept=所在部门

track_record_position=职位

track_record_position_description=职位描述

track_record_work_achievements=工作业绩

track_record_belong_person=所属人

工程创建

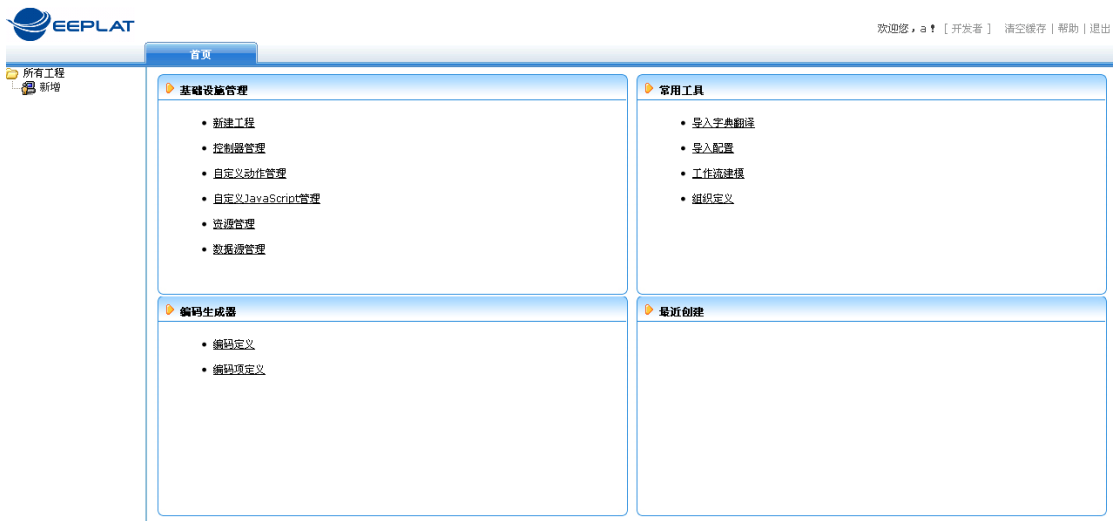
工程创建步骤:

登录平台=》新增工程=》初始化工程=》修改数据源=》初始化组织权限=》初始化数据表=》导入字典翻译=》清空缓存。

步骤详情如下:

1. 登录平台

按照上面源代码 svn 安装中测试给出的系统登录，登录系统，看到如下页面：



2. 新增工程

点击“新增”或者“新建工程”按钮，在弹出的页面中的填写数据，如下：

首页		新建工程	
新增数据			
名称	eeplatdemo *		
中文名	eeplatdemo *		
WEB应用名称	eeplatdemo *		
主标题			
主CSS			
SESSION过期时间			
上传路径			
模式	▼		
视感风格			
描述	<div></div>		
保存			

填写好数据，点击“保存”按钮，显示如下页面，表明新建成功。

所有工程

eeplatdemo

首页

新建工程

>> 浏览信息

名称	eeplatdemo
中文名	eeplatdemo
WEB应用名称	eeplatdemo
主标题	
主CSS	
SESSION过期时间	
上传路径	
模式	
业务对象	
面板	
视感风格	
描述	

初始化工程

打开工程登录

直接打开工程

分享到AppShare

3. 初始化工程

在如上页面中，点击“初始化工程”按钮，弹出如下页面：

首页

新建工程

>> 浏览信息

名称	eeplatdemo
中文名	eeplatdemo
WEB应用名称	eeplatdemo
主标题	
主CSS	
SESSION过期时间	
上传路径	
模式	
业务对象	eeplatdemo_全局
面板	
视感风格	
描述	

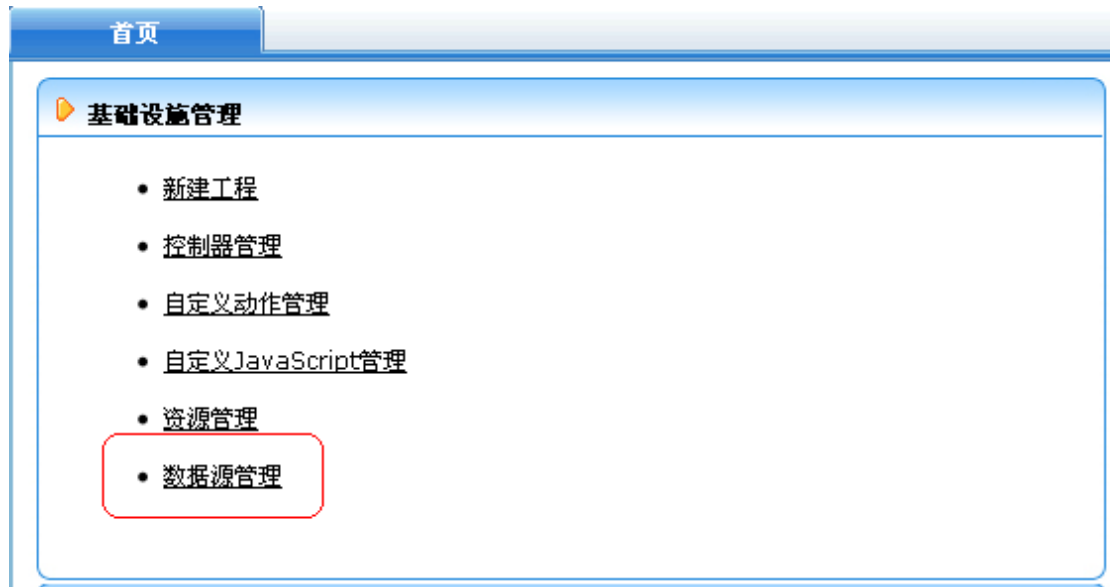
打开工程登录

直接打开工程

分享到AppShare

4. 修改数据源

打开首页，如下：



点击首页中“数据源管理”按钮，弹出如下页面：

查询结果							
JNDI名称	JNDI类型	方言	中文名	驱动类	驱动URL	用户名	密码
db2		DB2数据库示例		com.ibm.db2.jcc.DB2Driver	jdbc:db2://127.0.0.1:50000/test	test	test
gae		GAE_DataStore示例		org.jiqi.jdbc.Driver	jdbc:jiqi://local		
h2		H2_DataStore示例		org.h2.Driver	sample/sample	sa	
mysql		MYSQL数据库示例		com.mysql.jdbc.Driver	jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=utf-8	root	
oracle		ORACLE数据库示例		oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@127.0.0.1:1521:xe	edemo	edemo
sqlserver		SQLServer数据库示例		net.sourceforge.jtds.jdbc.Driver	jdbc:jtds:sqlserver://127.0.0.1:1433/test	test	test

选择与所使用的数据库对应的数据源，点击修改，将驱动 URL、用户名、密码设置为所做系统对应的数据库驱动 URL、用户名、密码并保存。设置之后重启 web 服务器。页面如下：

修改信息

修改信息

JNDI名称		JNDI类型	
方言	mysql		
中文名	MYSQL数据库示例		
驱动类	com.mysql.jdbc.Driver		
驱动URL	jdbc:mysql://127.0.0.1:3306/eeplat?useUnicode=		
用户名	root	密码	123456
池的大小		其它参数	
服务器初始化	是		

保存

5. 初始化组织权限（特别提醒：一个工程只能执行一次初始化组织权限操作）

重启 web 服务器后，登录平台在数据源管理中选择所用数据源，点击“初始化组织权限”按钮。出现如下页面，则初始化成功。

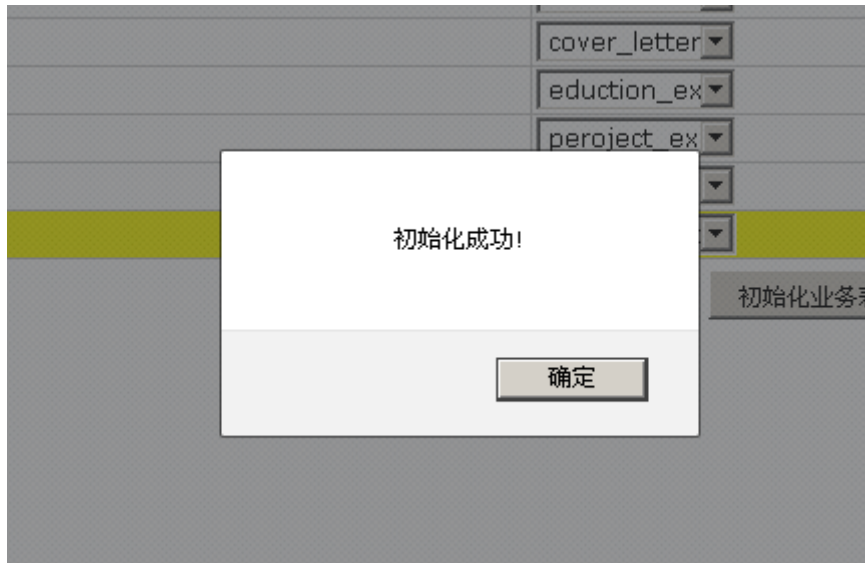


6. 初始化数据表（特别提醒：数据表的主键列请使用 uuid；本例中的数据库表 sql 语句见附录）

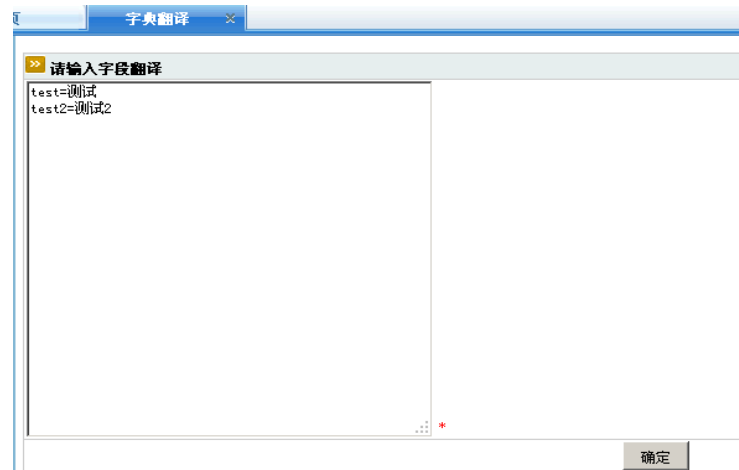
选择所用数据源，点击“初始化数据表”按钮，弹出如下页面：



然后设置主键列、特征值列（特别提醒：最好选择数据表的名称属性列或类似列，因为平台中下拉列表控制器显示的值为设置的特征值列）并选择业务包，然后点击初始化业务表。弹出如下页面，则初始化成功。



7. 导入字典翻译（**特别提醒：本例中的字段翻译见附录**）
在首页=》常用工具中，点击“导入字典翻译”按钮，进入如下页面：



字典翻译格式如上页面中所示，即“**[数据表/业务对象字段名]=[要显示的中文名]**”，按照此格式，输入翻译，点击确定即可。

8. 清空缓存

欢迎您，u！ [系统管理员] 清空缓存 | 帮助 | 退出



点击如上页面中的“清空缓存”按钮，成功后，点击左侧菜单所建工程，如下：



eeplatdemo	
查看信息 修改信息	
浏览信息	
名称	eeplatdemo
中文名	eeplatdemo
WEB应用名称	eeplatdemo
主标题	
主CSS	
SESSION过期时间	
上传路径	
模式	
业务对象	eeplatdemo_全局
面板	
视感风格	
描述	
打开工程登录 直接打开工程 分享到AppShare	

点击“打开工程登录”按钮，弹出登录页面，输入用户名: u, 密码: 1111。进入创建工程发布之后的页面。对应页面如下：

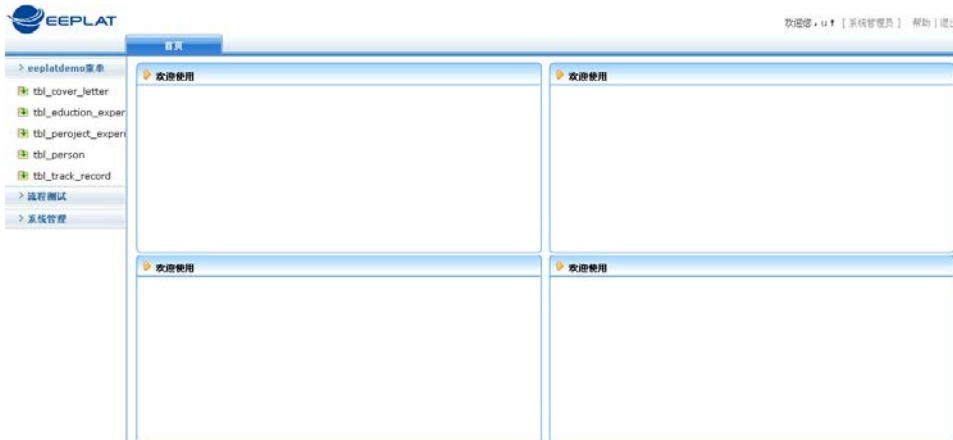


用户名

密码

验证码 7477

[登录](#)



EEPLAT 欢迎使用 u t [系统管理员] 帮助 | 退出

- > eeplatdemo菜单
 - tbi_cover_letter
 - tbi_education_experience
 - tbi_project_experience
 - tbi_person
 - tbi_track_record
- > 流程测试
- > 系统管理

欢迎使用

欢迎使用

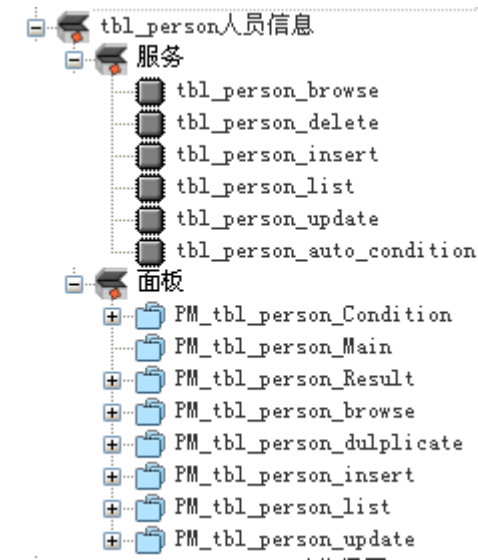
欢迎使用

欢迎使用

至此，工程的基本创建结束。下面会介绍一些常用的系统配置开发功能。

项目页面优化（菜单名称、表格元素显示等）

初始时，每个业务对象下都有默认的服务、面板，看其名称可知其用途。结构如下：



通过修改/删除菜单、面板（wiki 中有介绍）等，实现如下效果：



具体细节，参阅 wiki 中对应概念表述即可完成设置，故不再详细介绍。

关联数据表（处理表间关联）

关联数据表，是将有“一对多”关系的数据库表间建立关联关系。人员与求职信、教育经历、工作经历、项目经验均为一对多关系，需要将它们关联。

操作步骤：选择“多”的一方业务对象=》点击业务对象维护中“关联业务对象”按钮=》按页面显示选择相应的列等信息，点击“生成关联”按钮，即可。

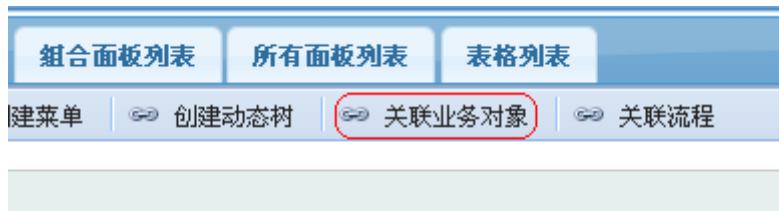
步骤明细如下：

1. 选择“多”的一方业务对象（以求职信数据表为例）

打开求职信表对应业务对象所在业务包，点击求职信业务对象,弹出页面，如下：

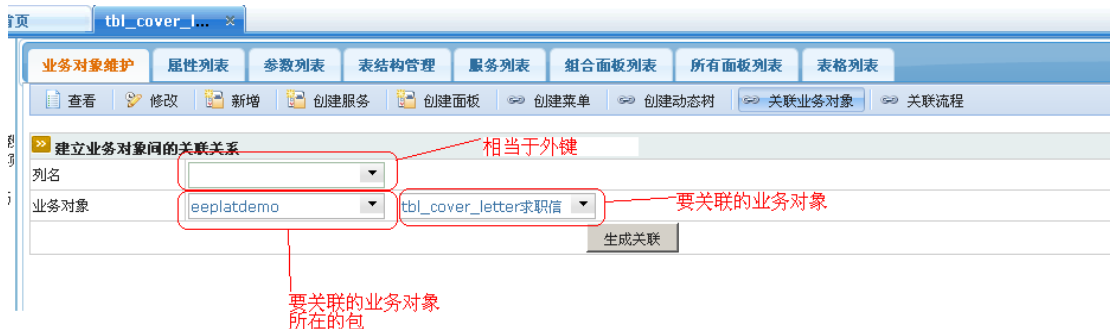


2. 点击业务对象维护中“关联业务对象”按钮，页面如下：



3. 按页面显示选择相应的列等信息，点击“生成关联”按钮。

在如下页面，是 2 步点击之后弹出页面，在页面中选择好相关项（列名：cover_letter_person；业务对象：tbl_person 人员信息），点击“生成关联”按钮即完成表/业务对象关联设置。这里要注意相关项要选对，如果设置错误，可重新更改设置。



以上即完成关联的配置。配置成功之后，在新增求职信时，上面配置的列名自动改为动态下拉菜单，供选择求职人员。效果如下：

新增数据			
求职公司名称	<input type="text"/>	求职公司岗位	<input type="text"/>
求职信内容	<div></div>		
发送日期	<input type="text"/>	是否发送	<input type="text"/>
所属人	张三		
	<div>张三 李四</div>	保存	

同样的操作步骤，将教育经历、项目经验、工作经历与人员信息关联。

附：刚打开新建面板，所属人处就默认选项，如不想要该项，可把所属人对应的表格元素的控制器改为：`form.DOResultListPopupNoSplitPageNoDefault`。

创建组合面板


人员与求职信、教育经历、工作经历、项目经验均为一对多关系，上一节已经将它们关联，那么接下来，在添加人员后，要在同一个面板中管理其求职信、教育经历、工作经历、项目经验。这就要求求职信将教育经历、工作经历、项目经验 4 个业务对象的 result 面板（建议复制 result 面板）组合，来实现。

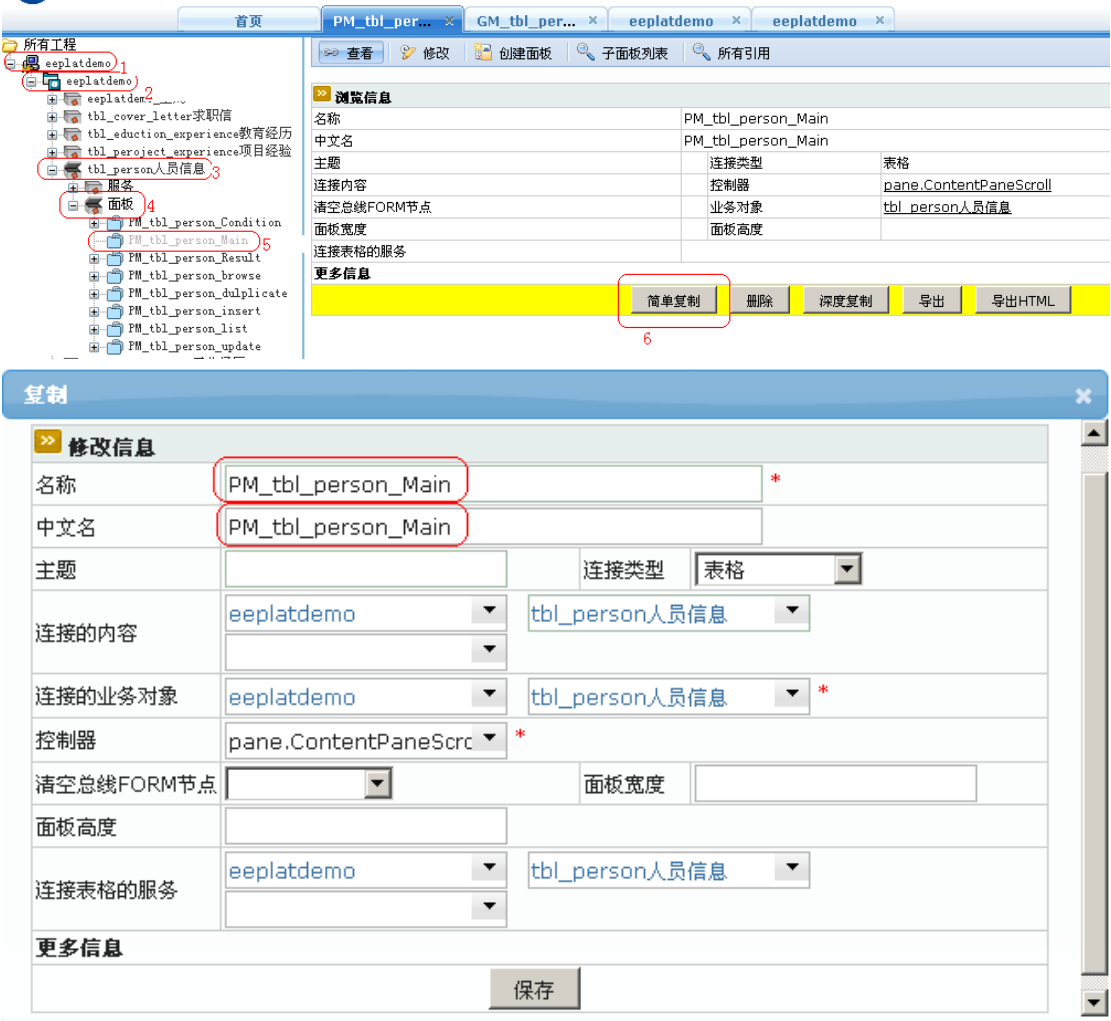
步骤：创建一个简单面板=》深度复制要组合在一起的业务对象的 result 面板并修改面板连接的服务及其新增服务=》增加组合面板=》在人员 result 表格中，添加按钮，连接面板设置为新建的面板。

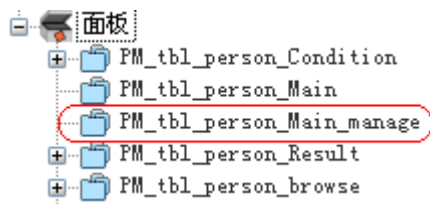
步骤明细：

1. 创建一个简单面板


依次打开：工程(eeplatdemo)->eeplatdemo->tbl_person 人员信息->面板，点击任一面板，点击弹出的页面中“简单复制”按钮，在第二个弹出面板中修改名称（如：PM_tbl_person_Main_manage）、中文名（如：管理）面板宽度（如：700）、高度（如：500），



保存之后双击，即可看到新建的面板。页面如下：





2. 深度复制要组合在一起的业务对象的 result 面板并修改面板连接的服务及其新增服务求职信、教育经历、工作经历、项目经验 4 个业务对象组合在一起，故对这四个业务对象进行操作。

以求职信为例：如 1 步中，找到求职信对象的 result 面板并点击=》点击弹出页面中的深度复制按钮，点击确定，成功之后双击  面板 按钮，看到复制的面板：

 PM_tbl_cover_letter_Result_copy ，将其表格元素中的“删除”的连接面板、目标面板改为  PM_tbl_cover_letter_Result_copy ，同时“修改”、“新增”对应面板中表格的

“保存”按钮的连接面板、目标面板改为  PM_tbl_cover_letter_Result_copy ；然后修



改其及其表格的名称、中文名、主题=》在  服务 中找到  tbl_cover_letter_list 服

务，复制该服务，成功复制后双击  服务 按钮，选择复制生成的服务



 tbl_cover_letter_list_copy ，对其进行修改之后为： tbl_cover_letter_list_person

修改明细如下：

修改服务信息			
名称	tbl_cover_letter_list_person *		
中文名	tbl_cover_letter_list_person 修改名称及中文名，名称处请用英文小写		
业务对象	tbl_cover_letters求职信		
主SQL	select * from tbl_cover_letter where cover_letter_person = ? cover_letter_person是与人员业务对象关联的列，这样可以查询选定人员的求职信		
事务控制		参数是否过滤	
类型	查询 *	自定义动作	
刷新总线类型			
更多信息			
保存			

服务参数列表						
业务对象、参数及参数类型要正确选择为关联业务对象的current类型的参数						
新增 修改 删除 参数检查定义 复制到						
全选	顺序	类型	参数	模式	业务对象	
<input type="checkbox"/>	 	Current	tbl_person current		tbl_person人员信息	

这 之 后 ， 修 改

( PM_tbl_cover_letter_Result_copy 修改之后的名称) 里表格连接的服务为  tbl_cover_letter_list_person ，如下：

浏览信息			
名称	GM_tbl_cover_letter_Result_person		
中文名	GM_tbl_cover_letter_Result_person		
列数目		数据数目	10
控制器	grid.GridList	服务	tbl cover letter list person
可选择列		文字换行(nowrap)	
主题	求职信		

同样的按照上面对服务的操作，对求职信的新增服务进行修改，修改前后内容如下：

Insert 添加服务：

改前：

类型	参数	模式	业务对象
Key	cover letter uid		tbl cover letter求职信
FORM	cover letter company name		tbl cover letter求职信
FORM	cover letter positon		tbl cover letter求职信
FORM	cover letter content		tbl cover letter求职信
FORM	cover letter send time		tbl cover letter求职信
FORM	cover letter send flag		tbl cover letter求职信
FORM	cover letter person		tbl cover letter求职信

改后：

类型	参数	模式	业务对象
Key	cover letter uid		tbl cover letter求职信
FORM	cover letter company name		tbl cover letter求职信
FORM	cover letter positon		tbl cover letter求职信
FORM	cover letter content		tbl cover letter求职信
FORM	cover letter send time		tbl cover letter求职信
FORM	cover letter send flag		tbl cover letter求职信
Current	tbl person currunt		tbl person人员信息

对教育经历、工作经历、项目经验业务对象重复 2 步操作。

3. 增加组合面板

点击新建的面板=》点击弹出页面中的“子面板按钮”=》点击新面板中的“增加组合面板”按钮=》在弹出面板中选择求职信表对应的业务对象、2 步中深度复制的 result 面板，设置排序标识，设置好后确定即可(要添加求职信、教育经历、工作经历、项目经验，故共需添加 4 次)。相关页面如下：

.tdemo
 platdemo
 eeplatdemo_全局
 tbl_cover_letter求职信
 tbl_education_experience教育经历
 tbl_perproject_experience项目经验
 tbl_person人员信息
 服务
 面板
 PM_tbl_person_Condition
 PM_tbl_person_Main
 PM_tbl_person_Main_manage
 PM tbl person Result

查看
 修改
 创建面板
 子面板列表
 所有引用

增加组合面板
 修改组合关系
 移除组合关系

名称	中文名	连接类型	连接内容
----	-----	------	------

增加组合面板

>> 增加组合面板

选择业务包	eeplatdemo
选择业务对象	tbl_cover_letter求职信 1
选择面板	PM_tbl_cover_letter_R * 2
条件类型	
条件	
排序标示	1 3

确定

查看	修改	创建面板	子面板列表	所有引用
----	----	------	-------	------

增加组合面板	修改组合关系	移除组合关系
--------	--------	--------

名称	中文名	连接类型	连接内容	控制器	排序标示
PM_tbl_cover_letter_Result	PM_tbl_cover_letter_Result	表格	GM_tbl_cover_letter_Result	pane.ContentPane 1	
PM_tbl_education_experience_Result	PM_tbl_education_experience_Result	表格	GM_tbl_education_experience_Result	pane.ContentPane 2	
PM_tbl_perproject_experience_Result	PM_tbl_perproject_experience_Result	表格	GM_tbl_perproject_experience_Result	pane.ContentPane 3	
PM_tbl_track_record_Result	PM_tbl_track_record_Result	表格	GM_tbl_track_record_Result	pane.ContentPane 4	

4. 在人员 result 表格中，添加按钮，连接面板设置为新建的面板
如上步中，找到 result 面板并打开，点击 result 表格，页面如下：

所有工程

- eeplatdemo
 - eeplatdemo_全局
 - tbl_cover_letter求职信
 - tbl_education_experience教育经历
 - tbl_perproject_experience项目经验
 - tbl_person人员信息
 - 服务
 - 面板
 - PM_tbl_person_Condition
 - PM_tbl_person_Main
 - PM_tbl_person_Main_manage
 - PM_tbl_person_Result
 - GM_tbl_person_Result**
 - PM_tbl_person_browse





查看 修

>> 浏览信息

名称	
中文名	
列数目	
控制器	
可选择列	
主题	
更多信息	

>> 浏览信息

在弹出的页面中，选择表格元素中的“查看”（或“复制”）⇒ 点击“复制到”按钮⇒ 确定
弹出的新面板中复制到的路径是人员的 result 表格，点击确定⇒ 选择表格元素中复制的“查看”（或“复制”）⇒ 点击“修改”按钮⇒ 修改元素的中文名为：管理、修改连接面板为 1 步中新建的面板，点击“保存”按钮。

>> 浏览信息			
<div>  新增  复制到 2  修改  关联管理 </div>			
全选 <input type="checkbox"/>	排序	中文名	业务对象属性
<input type="checkbox"/>	↑ ↓	姓名	person_name
<input type="checkbox"/>	↑ ↓	性别	person_sex
<input type="checkbox"/>	↑ ↓	生日	person_birthday
<input type="checkbox"/>	↑ ↓	户口地	person_account
<input type="checkbox"/>	↑ ↓	电子邮件	person_email
<input type="checkbox"/>	↑ ↓	开始工作时间	person_work_start_time
<input type="checkbox"/>	↑ ↓	手机号码	person_mobile
<input type="checkbox"/>	↑ ↓	最快入职日期	person_fastest_entry_time
<input checked="" type="checkbox"/>	↑ ↓	查看	1
<input type="checkbox"/>	↑ ↓	修改	
<input type="checkbox"/>	↑ ↓	删除	
<input type="checkbox"/>	↑ ↓	新增	
<input type="checkbox"/>	↑ ↓	复制	

修改信息

选择业务包	eeplatdemo
选择业务对象	tbl_person人员信息
选择表格	GM_tbl_person_Result
<div>确定</div>	

>> 浏览信息

新增
 复制到
 修改
 关联管理
 删除

全选	排序	中文名	业务对象属性
<input type="checkbox"/>	↑ ↓	姓名	person_name
<input type="checkbox"/>	↑ ↓	性别	person_sex
<input type="checkbox"/>	↑ ↓	生日	person_birthday
<input type="checkbox"/>	↑ ↓	户口地	person_account
<input type="checkbox"/>	↑ ↓	电子邮件	person_email
<input type="checkbox"/>	↑ ↓	开始工作时间	person_work_start_time
<input type="checkbox"/>	↑ ↓	手机号码	person_mobile
<input type="checkbox"/>	↑ ↓	最快入职日期	person_fastest_entry_time
<input type="checkbox"/>	↑ ↓	查看	
<input checked="" type="checkbox"/>	↑ ↓	查看	

>> 修改信息

中文名	管理		
重命名			
连接的服务	eeplatdemo	tbl_person人员信息	
连接的面板	eeplatdemo	tbl_person人员信息	管理
目标面板	eeplatdemo	tbl_person人员信息	_opener
控制器		form.TPaneSelected	*
是否只读	<input type="radio"/> 是 <input type="radio"/> 否	允许为空	<input type="radio"/> 是 <input type="radio"/> 否
是否新行	<input type="radio"/> 是 <input type="radio"/> 否	相关配置	
相关约束		表格外	上部
校验类型		是否摘要	
点击执行的JS		确认JS	

更多信息
 保存

如上，则完成组合面板的创建，效果如下：

管理

» 查询结果

查看

修改

删除

新增

复制

求职公司名称	求职公司岗位	求职信内容	发送日期	是否发送	所属人
第1页 每页10条 共0页 共0条记录					

» 查询结果

查看

修改

删除

新增

复制

开始日期	结束日期	学校名称	学历	所属人
第1页 每页10条 共0页 共0条记录				

» 查询结果

查看

修改

删除

新增

复制

项目开始时间	项目结束时间	项目名称	所任职务	开发工具	项目描述	责任描述	所属人
第1页 每页10条 共0页 共0条记录							

» 查询结果

查看

修改

删除

新增

复制

开始工作日期	结束工作日期	公司名称	所在部门	职位	职位描述	工作业绩	所属人
第1页 每页10条 共0页 共0条记录							

为了美观一些，把新建的管理面板的控制器改为：pane.Tab，同时将组合的四个面板的表格主题修改，效果如下：

管理

求职信

教育经历

项目经验

工作经历

» 求职信

查看

修改

删除

新增

复制

求职公司名称	求职公司岗位	求职信内容	发送日期	是否发送	所属人
第1页 每页10条 共0页 共0条记录					

规则应用（以级联删除为例）

规则的应用，这里以级联删除为例。

前面已经实现各业务对象间的关联，但是在执行删除人员操作后，求职信、教育经历、工作经历、项目经验数据表中的相应数据却没有一起删除。怎样实现级联删除呢？用**规则**或者**自定义动作**，都可以实现。这里用规则来实现。

以求职信业务对象为例，步骤：

1. 复制求职信删除服务并修改

修改明细如下：

名称	tbl_cover_letter_delete_copy				
中文名	tbl_cover_letter_delete_copy				
主SQL	delete from tbl_cover_letter where cover_letter_uid = ?				
事务控制	参数是否过滤				
类型	删除	自定义动作			
刷新总线类型	Current	tbl_person_currunt	tbl_person人员信息		
更多信息					
服务参数列表					
新增 修改 删除 参数检查定义 复制到					
全选	顺序	类型	参数	模式	业务对象
<input type="checkbox"/>	<input type="checkbox"/>	Current	tbl_cover_letter_currunt		tbl_cover_letter求职信

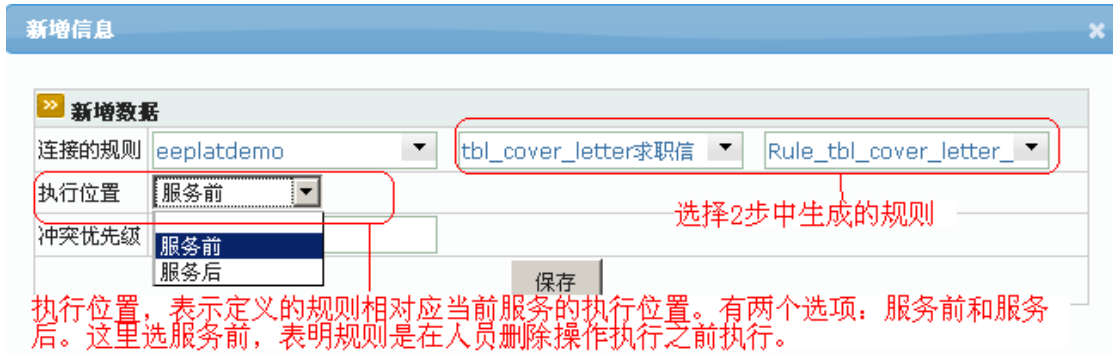
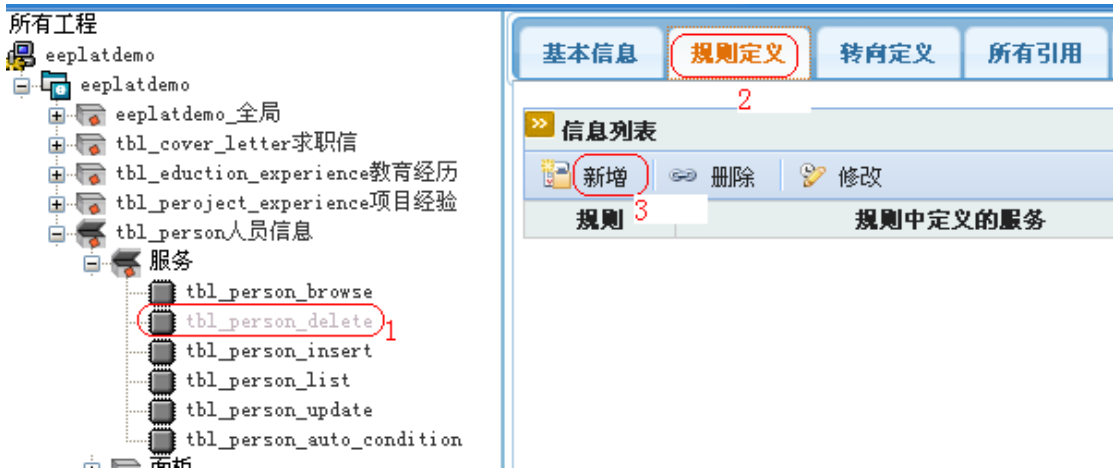
改后明细：

名称	tbl_cover_letter_delete_byperson				
中文名	tbl_cover_letter_delete_byperson				
主SQL	delete from tbl_cover_letter where cover_letter_person = ?				
事务控制	主SQL				
类型	删除	自定义动作			
刷新总线类型					
更多信息					
服务参数列表					
新增 修改 删除 参数检查定义 复制到					
全选	顺序	类型	参数	模式	业务对象
<input type="checkbox"/>	<input type="checkbox"/>	Current	tbl_person_currunt		tbl_person人员信息

2. 点击修改后的服务的“生成规则按钮”并点击确定，如下：

修改	复制	新增	服务多态	生成规则	删除
服务信息管理					
名称	tbl_cover_letter_delete_byperson				
中文名	tbl_cover_letter_delete_byperson				
主SQL	delete from tbl_cover_letter where cover_letter_person = ?				

3. 找到 person（人员）业务对象的删除服务并点击，在弹出页面中点击“规则定义”=> 点击新增，在新弹出面板中选择 2 步中生成规则并设置执行位置等项目信息并保存，相关页面如下：






至此，级联删除设置完毕。这样在执行删除人员操作后，求职信数据表中的相应数据也会一起删除。

控制器扩展（以警示灯功能实现为例）

控制器的概念、分类、实现方式等请参照平台 wiki。

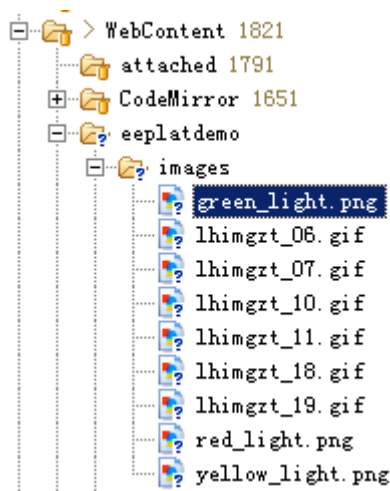
下面以警示灯功能实现为例，介绍实现 `com.exedosoft.plat.ui.jquery.form.DOBaseForm` 类的表格元素类型的控制器。

为了演示控制器扩展，这里实现对人员薪金要求的警示。如：在 1-3000 显示, 3001-6000 显示, 6000 以上显示。（图片素材在警示灯图片文件夹中）

步骤：建控制器类=》进入平台控制器管理=》将新建的控制器添加到平台中=》在面板中配置新建的控制器并看显示效果。

步骤明细如下：

1. 将警示灯图片放到 WebContent->eeplatdemo->images 文件夹下，如下所示：



参照 `com.exedosoft.plat.ui.jquery.form` 下的 Java 实现的控制器（如：`DOValueInputFileHD`）新建一个 Java 类，命名为：`WarningLight`。建好类后重启 tomcat。代码明细如下：

```
public class WarningLight extends DOBaseForm {

    @Override
    public String getHtmlCode(DOIModel iModel) {
        // TODO Auto-generated method stub
        DOFormModel property = (DOFormModel) iModel;
        return formSelectStr(property);
    }

    String formSelectStr(DOFormModel property) {

        // 获取表格元素设置的业务对象属性的值
        String value = property.getValue();
        int salary = Integer.parseInt(value);
```

```

// 获取三个警示灯图片
String imgRed = "eeplatdemo/images/red_light.png";
String imgGreen = "eeplatdemo/images/green_light.png";
String imgYellow = "eeplatdemo/images/yellow_light.png";

// 定义light变量，显示图片用
String light = "";

// 定义StringBuffer变量，用来在表格控制器中显示警示灯图片
StringBuffer sb = new StringBuffer();

if (value != null && !value.trim().equals("")) {
    // 判断薪水以确定警示灯颜色
    if ((salary >= 1) && (salary <= 3000)) {
        light = imgGreen;
    } else if ((salary > 3000) && (salary <= 6000)) {
        light = imgYellow;
    } else if (salary > 6000) {
        light = imgRed;
    }
    // 在表格中显示的警示灯
    sb.append("<span
style='").append(property.getStyle()).append(">")
        .append("<img
src=\"").append(light).append(\">").append(
            "</img>").append("</span>");
}

return sb.toString();
}
}

```

2. 在平台开发首页的基础设施管理中选择控制器管理并点击。相关页面如下：



点击后弹出页面：

首页

控制器管理

请输入查询条件

名称

中文名

类型

Java Class

查询

查询结果

查看

修改

删除

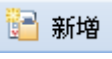
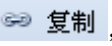
新增

复制

导出

名称	中文名	类型	Java Class	控制器类别
com.exedosoft.plat.ui.jquery.form.DOInputTextBetween	form.DOInputTextBetween	FORM	com.exedosoft.plat.ui.jquery.form.DOInputTextBetween	录入相关
com.exedosoft.plat.ui.jquery.pane.LayOutHeader	pane.LayOutHeader	PANE	com.exedosoft.plat.ui.jquery.pane.LayOutHeader	c_pane
com.exedosoft.plat.ui.jquery.pane.LayOutHeader	pane.LayOutHeader	PANE	com.exedosoft.plat.ui.jquery.pane.LayOutHeader	c_pane

3. 将新建的 Java 控制器添加到平台中

在控制器管理页面中，点击  按钮，在弹出的新增页面中输入相关项并保存（也可选择一个 Form 类型的控制器然后点击  复制，在弹出的复制页面中修改）。配置明细如下：

名称

WarningLight

设置控制器名，须用英文字母

中文名

form.WarningLight

设置一个自己可以识别的中文名

类型

FORM

有多种类型，这里选择Form，因为是表格元素控制器

Java Class

com.exedosoft.plat.ui.jquery.form.WarningLight

为自己添加的控制器类的路径

输出JSP

FORM类别

标准属性

CSS风格

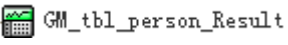
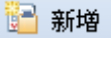
视觉风格

DOJO

风格对应(DOJO)

风格对应(MOBILE)

风格对应(JQUERY)

4. 在人员 result 面板的  GM_tbl_person_Result 表格中，点击  新增，在新增面板中配置 3 步中新加的控制器并保存。

配置明细如下：

新增数据

中文名

警示灯

设置页面显示的名称

重命名

选择人员的薪水属性

连接的表格

tbl_person人员信息

GM_tbl_person_Result

连接的业务对象属性

tbl_person人员信息

person_salary_expect

连接的服务

tbl_person人员信息

连接的面板

tbl_person人员信息

目标面板

tbl_person人员信息

控制器

显示相关

*

form.WarningLight

*

是否只读

☐是 ☐否

允许为空

☐是 ☐否

是否新行

☐是 ☐否

控制器

选择新建的警示灯配置

相关约束

表格外

这样警示灯控制器就扩展完毕，看一下效果吧：

>> 查询结果					
 查看	 修改	 删除	 新增	 复制	 管理
警示灯	姓名	性别	生日	户口地	电子邮件
	张三	男	2007-02-15	北京	test@test.com
	李四	男	2007-01-30	天津	lisi@eeplat.com
	王五	男	2012-01-29	上海	wangwu@eeplat.com
    第1页 每页10条 共1页 共3条记录					

自定义动作扩展（以删除多条记录为例）

自定义动作是平台功能的扩展，即为了完成平台不能配置的功能。

自定义动作有两种实现方式：

(1) 通过脚本在线编写 (2) 实现 `com.exedosoft.plat.action.Action` 接口或继承 `com.exedosoft.plat.action.DOAbstractAction` 抽象类

自定义动作的应用场景：

(1) 服务使用自定义动作：这时服务的执行委托给自定义动作，不再执行服务定义的 SQL 语句

(2) 参数使用自定义动作：自定义动作的返回值为参数的值

(3) 编码（CodeMain）使用自定义动作：自定义动作的返回值编码的值

(4) 编码项(CodeItem)目使用自定义动作：自定义动作的返回值为编码项目的值

(5) 通过平台提供的 `javascript callAction` 的函数，直接调用自定义动作，完成前台和后台的直接交互

这里以两种实现方式分别写一个批量删除人员的自定义动作扩展，作为例子。


（一）脚本在线编写

步骤：新建自定义动作=》新建删除服务配置其自定义动作为新添加的动作=》配置人员 result 表格并在其中添加批量删除按钮使之连接的服务为新建的删除服务

步骤明细：

1. 新建自定义动作

点击开发平台首页中基础设施管理的自定义动作管理，进入自定义动作管理页面，然后点击

 按钮，在弹出的新增页面中配置批量删除人员的自定义动作并保存，配置明细页面如下：

名称	<code>deleteAllPersonByScript</code> 自定义动作名称，用英文
中文名	<code>批量删除人员_byScript</code> 定义易于辨别的中文名
JAVA CLASS	
脚本实现	<pre>1 var ret ; //返回值声明 2 3 var DEFAULT_FORWARD = "success"; 4 var NO_FORWARD = "noforward"; 5 6 var sSubmit = doservice.getService("tbl_person_delete");//调用删除人员的服务 7 var checks=doform.getValueArray("checkinstance");//获取Form节点中name为checkinstance的属性的值 8 9 if(checks !=null && checks.length >0){ 10 // 循环调用删除服务 11 for(var i = 0 ; i < checks.length ; i++){ 12 var rid = checks[i] ; 13 sSubmit.invokeUpdate(rid) ; //rhino js是老版本的,这个rid参数要用[]包起来 14 } 15 SessionContext.getInstance().getThreadContext().setEchoValue("批量删除完成！"); 16 ret = DEFAULT_FORWARD ;</pre> 脚本实现里，用来编写批量删除的实现

脚本实现代码明细：

```
var ret ; //返回值声明
```

```

var DEFAULT_FORWARD = "success";
var NO_FORWARD = "noforward";

var sSubmit = doservice.getService("tbl_person_delete");//调用删除人员的服务
var checks=doform.getValueArray("checkinstance");//获取 Form 节点中 name 为
checkinstance 的属性的值

if(checks !=null && checks.length >0){
    // 循环调用删除服务
    for(var i = 0 ; i < checks.length ; i ++){
        var rid = checks[i] ;
        sSubmit.invokeUpdate(rid) ; //rhino js 是老版本的, 这个 rid 参数要用[]包起来
    }
    SessionContext.getInstance().getThreadContext().setEchoValue("批量删除完成!")
) ;
    ret = DEFAULT_FORWARD ;
}else{
    SessionContext.getInstance().getThreadContext().setEchoValue("没有数据!") ;
    ret = NO_FORWARD ;
}
ret ; //返回值

```

2. 新建删除服务配置其自定义动作为新添加的动作

打开人员业务对象的服务包，新建一个删除服务，配置明细如下：

名称	tbl_person_delete_all_byscript *		
中文名	tbl_person_delete_all_byscript		
业务对象		tbl_person人员信息	
主SQL			
事务控制		参数是否过滤	
类型	删除 *	自定义动作	批量删除人员_byScript
刷新总线类型			

选择新增的自定义动作，这里显示的是自定义动作的中文名

3. 配置人员 result 表格并在其中添加批量删除按钮使之连接的服务为新建的删除服务

配置人员 result 表格如下：

修改信息			
名称	GM_tbl_person_Result *		
中文名	GM_tbl_person_Result		
连接的服务	eeplatdemo	tbl_person人员信息	tbl_person_auto_cond
列数目		数据数目	10
数字列		可选择列	CheckBox
选择控制器	grid.GridList *		
连接的业务对象	eeplatdemo	tbl_person人员信息 *	可选择列选择checkbox
主题	查询结果		
更多信息			

在人员 result 表格中新增“批量删除”表格元素，配置如下：

新增信息			
中文名	批量删除	定义意思明确的中文名	
重命名			
连接的表格		tbl_person人员信息	GM_tbl_person_Result
连接的业务对象属性		tbl_person人员信息	
连接的服务		tbl_person人员信息	tbl_person_delete_all
连接的面板		tbl_person人员信息	
目标面板		tbl_person人员信息	选择新建的删除服务
控制器	常用按钮 *	form.TServiceSelectedF *	
是否只读	<input type="radio"/> 是 <input type="radio"/> 否	允许为空	<input type="radio"/> 是 <input type="radio"/> 否
是否新行	<input type="radio"/> 是 <input type="radio"/> 否	控制器选择: form.TServiceSelectedUf	
相关约束		表格外	上部
校验类型		是否摘要	
点击执行的JS		确认JS	
更多信息			


（二）继承 com.exedosoft.plat.action.DOAbstractAction 抽象类

步骤：新建自定义动作类=》将其添加到平台自定义动作中=》新建删除服务配置其自定义动作作为新添加的动作=》配置人员 result 表格并在其中添加批量删除按钮使之连接的服务为新建的删除服务

步骤明细：

1. 新建 package，路径如下：com.eeplat.action.myself，在其中新建人员批量删除自定义

动作类:DeleteAllPerson, 然后重启 tomcat。(可参照  com.exedosoft.plat.action

里的  CoreSaveAllWfAction.java)

代码明细如下：

```
public class DeleteAllPerson extends DOAbstractAction {

    @Override
    public String excute() throws ExedoException {
        // TODO Auto-generated method stub
        // 获取Form节点
    }
}
```

```

BOInstance form = DOGlobals.getInstance().getSessoInContext()
    .getFormInstance();
// 调用删除人员的服务
DOService deleteService =
DOService.getService("tbl_person_delete");
// 获取Form节点中name为checkinstance的属性的值
// 因为表格用的GridList控制器，在其配置的checkbox的name
为checkinstance，（明细请
// 参照/exedo/webv3/template/grid/GridList.ftl）
// 因此获取Form节点中name为checkinstance的属性的值即可获得在页面中选择
的要删除的人员列表
String[] checks = form.getValueArray("checkinstance");
if (checks == null && checks.length <= 0) {
    this.setEchoValue("没有数据！"); // 在页面弹出提示框
    return NO_FORWARD;
} else {
    // 循环调用删除服务
    for (int i = 0; i < checks.length; i++) {
        deleteService.invokeUpdate(checks[i]);
    }
}
return DEFAULT_FORWARD;
}
}
}

```

2. 点击开发平台首页中基础设施管理的自定义动作管理，进入自定义动作管理页面，将 1 步中新建的自定义动作类加入到平台中。配置页面明细如下：

名称	DeleteAllPerson	*
中文名	批量删除人员	
JAVA CLASS	com.eeplat.action.myself.DeleteAllPerson	所建的自定义动作类的路径

3. 新建删除服务配置其自定义动作为新添加的动作
打开人员业务对象的服务包，新建一个删除服务，配置明细如下：

新增数据			
名称	tbl_person_delete_all *		
中文名	tbl_person_delete_all		
主SQL			
事务控制		参数是否过滤	
类型	删除 *	自定义动作	批量删除人员
刷新总线类型			
更多信息			
保存			

4. 配置人员 result 表格并在其中添加批量删除按钮使之连接的服务为新建的删除服务
配置人员 result 表格如下：

修改信息			
名称	GM_tbl_person_Result *		
中文名	GM_tbl_person_Result		
连接的服务	eeplatdemo	tbl_person人员信息	tbl_person_auto_cond
列数目		数据数目	10
数字列		可选择列	CheckBox
选择控制器	grid.GridList *		
连接的业务对象	eeplatdemo	tbl_person人员信息 *	可选择列选择checkbox
主题	查询结果		
更多信息			

在人员 result 表格中新增“批量删除”表格元素，配置如下：

新增信息			
中文名	批量删除	定义意思明确的中文名	
重命名			
连接的表格		tbl_person人员信息	GM_tbl_person_Result
连接的业务对象属性		tbl_person人员信息	
连接的服务		tbl_person人员信息	tbl_person_delete_all
连接的面板		tbl_person人员信息	
目标面板		tbl_person人员信息	选择3步中新建的删除服务
控制器	常用按钮 *	form.TServiceSelectedF	*
是否只读	<input type="radio"/> 是 <input type="radio"/> 否	允许为空	<input type="radio"/> 是 <input type="radio"/> 否
是否新行	<input type="radio"/> 是 <input type="radio"/> 否	控制器选择: form.TServiceSelectedUf	
相关约束		表格外	上部
校验类型		是否摘要	
点击执行的JS		确认JS	
更多信息			

至此，两种方式的批量删除的自定义动作扩展结束。完成后的效果图如下：

查询结果						
<div> 查看 修改 删除 新增 复制 管理 批量删除 </div>						
全选 <input type="checkbox"/>	警示灯	姓名	性别	生日	户口地	电子邮件
<input type="checkbox"/>		张三	男	2007-02-15	北京	
<input type="checkbox"/>		李四	男	2007-01-30	天津	lisi@eeplat.com
<input type="checkbox"/>		王五	男	2012-01-29	上海	wangwu@eeplat.com

配置的表格的checkbox

添加的批量删除按钮

缺省 workflow 示例

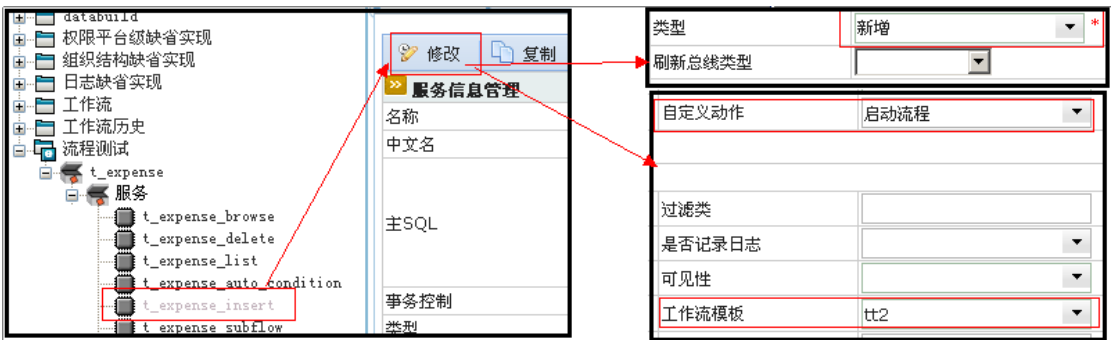
以平台提供的缺省报销流程的实例（tt2）为例，报销单业务对象：t_expense。Svn 版本为：1974。

工作流的配置过程，参见 eeplat wiki 中工作流概念的流程示例运行简介（网址：<http://code.google.com/p/eeplat/wiki/ConfigWorkflow>）。

【另补充】：

业务对象手动关联工作流的说明：

打开 t_expense 业务对象的服务包，点击 insert 服务=》点击修改按钮，将“自定义动作”属性设置为“启动流程”，“工作流模板”属性设置为“tt2”（平台提供的缺省报销流程的实例），之后保存即可。



◆ 工作流提交时选择下一节点执行人的实现










1. 配置提交报销单节点权限验证类型为“运行时指定用户”；

节点属性编辑	
节点ID：	tt2_n16
节点名称：	提交报销单
流程状态：	
回退时流程状态：	
权限验证类型：	运行时指定用户
权限控制类：	

2. 新建人员选择面板。打开平台缺省提供的 user 业务对象，简单复制 PM_do_org_user_Main 面板、深度复制 PM_do_org_user_Condition 和 PM_do_org_user_Result 面板，分别命名为 PM_do_org_wfiuser_Main、PM_do_org_wfiuser_Condition、PM_do_org_wfiuser_Result 在复制的 PM_do_org_wfiuser_Result 面板中配置 form.DOSelectData 控制器（位置设置在表格下方）的按钮（用于确认选择的人员）。同时，将 PM_do_org_wfiuser_Condition、PM_do_org_wfiuser_Result 配置为 PM_do_org_wfiuser_Main 的组合面板，配置如下：

» 浏览信息			
名称	PM_do_org_wfiuser_Main		
中文名	PM_do_org_wfiuser_Main		
主题	人员选择	连接类型	表格
连接内容		控制器	pane.ContentPaneScroll
清空总线FORM节点		业务对象	do_org_user
面板宽度	550	面板高度	450
连接表格的服务			
更多信息			
简单复制		删除	深度复制
		导出	导出HTML

浏览信息							
名称	GM_do_org_wfiuser_Condition						
中文名	GM_do_org_wfiuser_Condition						
列数目		数据数目					
控制器	grid.GridConditionAutoTr		服务				
可选择列		文字换行(nowrap)					
主题	请输入查询条件						
更多信息							
表格元素列表							
<div><div>新增</div><div>复制到</div><div>修改</div><div>删除</div><div>关联管理</div><div>提交范围</div><div>权限设置</div><div>设为摘要</div><div>设为普通</div></div>							
全选	排序	中文名	业务对象属性	连接的服务	连接的面板	目标面板	控制器
<input type="checkbox"/>	<div><div>↑</div><div>↓</div></div>	姓名	<u>name</u>				<u>form.DOInputText</u>
<input type="checkbox"/>	<div><div>↑</div><div>↓</div></div>	部门	<u>deptuid</u>	<u>do_org_dept_top</u>			<u>form.DOSelectTree</u>
<input type="checkbox"/>	<div><div>↑</div><div>↓</div></div>	查询			PM_do_org_wfiuser_Result		<u>form.TPane</u>

名称	GM_do_org_wfiuser_Result						
中文名	GM_do_org_wfiuser_Result						
列数目		数据数目	10				
控制器	grid.GridList	服务	do_org_user_auto_condition				
可选择列	Radio	文字换行(nowrap)					
主题	用户查询						
更多信息							
» 表格元素列表							
<div><div> 新增</div><div> 复制到</div><div> 修改</div><div> 删除</div><div> 关联管理</div><div> 提交范围</div><div> 权限设置</div><div> 设为摘要</div><div> 设为普通</div></div>							
全选 <input type="checkbox"/>	排序	中文名	业务对象属性	连接的服务	连接的面板	目标面板	控制器
<input type="checkbox"/>	<div><div>↑</div><div>↓</div></div>	姓名	name		PM_do_org_user_browse	_opener	form.DOValuePaneByRole
<input type="checkbox"/>	<div><div>↑</div><div>↓</div></div>	工号	user_code				form.DOValueSimple
<input type="checkbox"/>	<div><div>↑</div><div>↓</div></div>	选择					form.DOSelectData

<div>查看</div> <div>修改</div> <div>创建面板</div> <div>子面板列表</div> <div>所有引用</div>			
<div>增加组合面板</div> <div>修改组合关系</div> <div>移除组合关系</div>			
名称	中文名	连接类型	连接内容
PM_do_org_wfiuser_Condition	PM_do_org_wfiuser_Condition	表格	GM_do_org_wfiuser_Condition
PM_do_org_wfiuser_Result	PM_do_org_wfiuser_Result	表格	GM_do_org_wfiuser_Result

- 在工作流提交报销单面板中添加“下一步执行人”表格元素，控制器选择常用按钮中的 DOSelectInvoke，并配置其“重命名”属性为英文名称（若不配置，“中文名”属性为中文，会导致后面的 selectdata 选择之后，selectinvoke 不显示选择的数据）和其弹出的面板（即人员选择面板）。

中文名	下一步执行人		
重命名	scheduleIds		
连接的服务	监察工作	tbl_project	
连接的面板	监察工作	tbl_project	PM_do_org_wfiuser_M
目标面板	监察工作	tbl_project	_opener
控制器	常用按钮	form.DOSelectInvoke	*

至此，即完成 workflow 提交时选择下一步执行人的功能。效果图如下：

下一步执行人

查找 *

人员选择

请输入查询条件

姓名 部门

查询

用户查询

序号	选择	姓名	工号
1	<input type="radio"/>	test1	007
2	<input checked="" type="radio"/>	xfag	00100
3	<input type="radio"/>	test2	008

第1页 每页10条 共1页 共3条记录

选择

◆ 工作流监控的图形显示实现

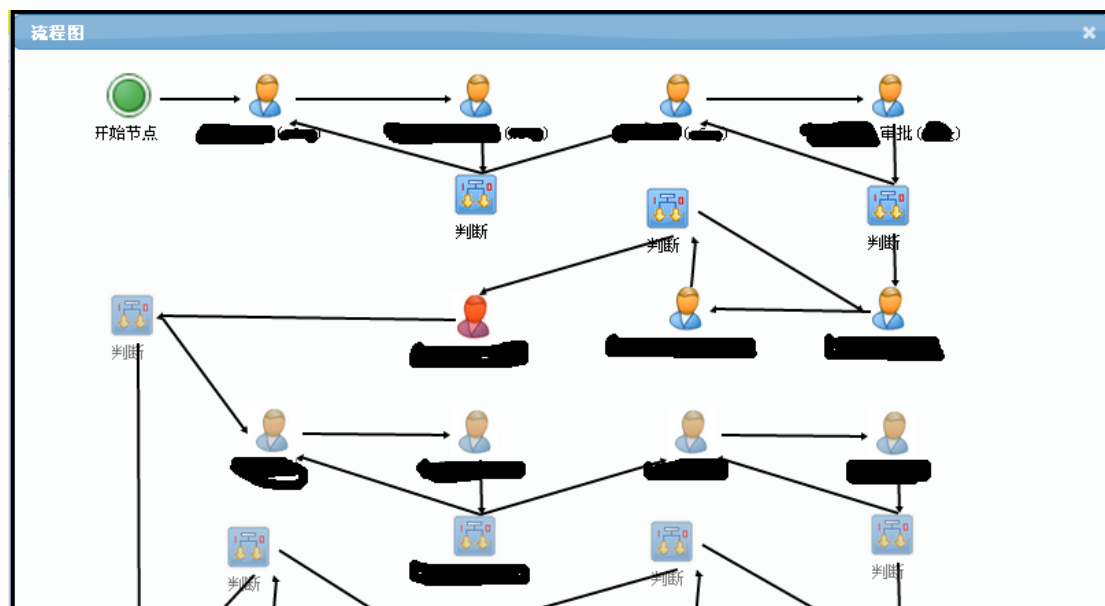
- 1、打开 eeplat 首页资源管理面板，点击新增按钮，将 WebContent/exedo/flow / monitor.jsp 这个页面创建为资源。
- 2、打开工作流业务包中的工作流个人任务业务对象，在其面板包中新建面板，面板的控制器是 pane.DOIframe 或者 DOIframe 内容是 monitor.jsp，配置如下：

名称	PM_wf_monitor *		
中文名	PM_wf_monitor		
主题	流程图	连接类型	资源
连接的内容	工作流	工作流个人任务	wfmonitor.jsp
控制器	pane.DOIframe *		
连接的业务对象	工作流	工作流个人任务	*
青空总线FORM节点		面板宽度	800

- 3、在平台缺省实现的工作流我已处理面板或者待办列表面板，添加监控按钮，连接的面板为上步中新建的面板。配置如下：

中文名	监控		
重命名			
连接的服务	工作流	工作流个人任务	
连接的面板	工作流	工作流个人任务	PM_wf_monitor
目标面板	工作流	工作流个人任务	_opener
控制器	常用按钮	form.TAPane	*

至此，即完成 workflow 监控的图形显示的实现。效果图如下：



◆ 其他
待续。。。。。

编码定义与编码项应用简例

流水号为年份+序号，在平台中自动生成的实现：

1、将业务对象流水号对应参数类型设置为 code，如下页面：

名称	file_flow_num	*
中文名	file_flow_num	
类型	Code	*

2、在编码项管理中，添加年份、序号编码项，配置页面如下：

年份配置：

编码项名称	公文流水号年份			*	
编码项描述					
编码项类型	当期时间	*	编码项长度	4.0	*
截取或补充方向			格式化	%1\$tY	
扩展Class					
占位符					
参数		tab_file	file_flow_num		

序号配置：

>> 修改信息

编码项名称	公文流水号			*	
编码项描述					
编码项类型	纯流水序号	*	编码项长度	6.0	*
截取或补充方向			格式化		
扩展Class					
占位符	0				
参数		tab_file	file_flow_num		

[注]参数为业务对象中对应流水号属性的参数。

3、在编码管理中添加编码（名称自定），并定义上步中建好的编码项（注意顺序），页面如下：

>> 编码项列表					
🔑 修改	✖ 删除	📄 新增			
编码项名称	编码项描述	编码项类型	编码项长度	占位符	排序标示
公文流水号			6.0	0	2.0
公文流水号年份			4.0		1.0

如上步骤完成，在执行添加服务时即可实现流水号自动生成。

使用表格元素的条件类型属性控制表格元素的显示与否

配置实现步骤：

- 1、在表格面板，选择一个表格元素，点击修改按钮；
- 2、在弹出的修改信息面板中，可以看到“条件类型”属性，选择 Script，并在条件输入框中写如 js 语句，配置明细及 js 语句格式如下图：

条件类型	Script
	<pre>1 var ret=false; 2 var doin=doinstance.getValue("statistics_letter_aves"); 3 if(doin.equals("TEST")){ 4 ret=true; 5 } 6 ret;</pre>

提醒：这里属于后台的 js 调用，可以参照平台 wiki 里的 JS API（服务器端）介绍，如下：

后台的 Javascript 可以调用所有的 Java API,但是需要注意 JavaScript 的用法。

后台的 Javascript 的常用内置变量：

- douser:当前的登陆用户
- doform: 当前的提交的表单内容
- doinstance:当前记录

常用内置变量用法：

内置变量.getValue("属性名称") 就可以获取属性对应的值，如获取当前登陆者的部门：

```
var deptuid = douser.getValue("deptuid");
```

不常用的有：

- docontext:全局 Session, 对应 JAVA 类 SessionContext
- doservice:当前的服务,对应 JAVA 类 DOService
- domodel:当前的模型,对应 JAVA 类 DOIModel

图表使用示例（控制器扩展延伸）

以平台缺省的 `t_expense` 中的测试图表实现为例，说明图表的使用。（示例中用的 `svn` 版本为：1821）

报销单查询结果	
导出Excel	导出Excel(模板)
测试图表	
报销事由	报销人
123	123
456	456
789	789
011	000
200	200
test	test

说明：本图表示例可以是控制器扩展的延伸。因为控制器的实现有如下 4 种方式：

1. 通过脚本在线编写，采用在线的 JavaScript 脚本和 freeMarker 脚本开发控制器，此类控制器在线开发即时执行，无需编译，存储在系统模型数据库中，因而在在线开发方面具有很好的优越性；此类控制器从名称上最后一部分名字以 TDB 开始，如：pane.TDBPaneTemplate、grid.TDBGridTemplate 等。
2. 实现 `com.exedosoft.plat.ui.DOIView` 接口，可以实现所有类型的控制器，即控制器的实现为一个 Java 类，此类控制器的开发、扩展修改需要编译后才能被使用；
3. 实现 `com.exedosoft.plat.ui.DOIViewTemplate` 接口或继承 `com.exedosoft.plat.ui.DOViewTemplate` 抽象类，结合 FreeMarker 模板文件可以实现所有类型的控制器，由于采用了模板，所以此类控制器的开发相对更加简洁
4.
 - o 实现 `com.exedosoft.plat.ui.jquery.form.DOBBaseForm`，实现表格元素类型的控制器
 - o 实现 `com.exedosoft.plat.ui.jquery.menu.DOBBaseMenu`，实现菜单类型的控制器
 - o 实现 `com.exedosoft.plat.ui.jquery.pane.TPaneTemplate`，结合 FreeMarker 模板，实现面板类型的控制器

在《eeplat_6 控制器扩展》文档中，介绍了第四种方式实现控制器扩展的例子，而第二和第三中方式与第四中方式类似，固未再做示例介绍。下面实现图表使用的示例中，将采用第一种方式实现控制器即在线脚本编写控制器。

配置实现步骤：

新建控制器=>新建或修改面板，将其控制器选择为新建的控制器

- 1、打开 eeplat 首页控制器管理面板，添加 pane 类型的控制器 TDBChart（推荐复制现有的 pane 类型控制器之后进行修改），页面配置如下：

名称	TDBChart	*
中文名	pane.TDBChart	
类型	PANE	*
Java Class		

逻辑脚本

```

1 var s = "";
2 var data = new HashMap();
3 data.put("model", domodel);
4 data.put("contextPath", DOGlobals.PRE_FULL_FOLDER);
5 data.put("webmodule", DOGlobals.URL);
6 var service = DOService.getService("t_expense_list");
7 var str = "<graph>";
8 var list = service.invokeSelect();
9 for(var it = list.iterator(); it.hasNext());{
10 var anIns = it.next();
11 str = str + "<set name=" + anIns.getValue("title") + " value=" + anIns.getValue("expense_money") + " col
12 }
13 str = str + "</graph>";
14 System.out.println("gXml" + str)
15 data.put("gXml",str);
16 s = Packages.com.exedosoft.plat.template.HtmlTemplateGenerator.getContentFromDBTemplate(domodel.getCo

```

模板脚本

```

1 <div id="chart1div">
2     FusionCharts
3 </div>
4 <script language="JavaScript">
5     var chart1 = new FusionCharts("/${webmodule}/FusionChartsFree/FCF_Column3D.swf", "cha
6     chart1.setDataXML("${gXml}");
7     chart1.render("chart1div");
8 </script>

```

附 js 脚本:

```

var s = "";
var data = new HashMap();
data.put("model", domodel);
data.put("contextPath", DOGlobals.PRE_FULL_FOLDER);
data.put("webmodule", DOGlobals.URL);
var service = DOService.getService("t_expense_list");
var str = "<graph>";
var list = service.invokeSelect();//老版本的 rhino js 需加中括号, 即为:service.invokeSelect([])
for(var it = list.iterator(); it.hasNext());{
var anIns = it.next();
str = str + "<set name=" + anIns.getValue("title") + " value=" + anIns.getValue("expense_money") + " color='D64646' />";
}
str = str + "</graph>";
System.out.println("gXml" + str)
data.put("gXml",str);
s =
Packages.com.exedosoft.plat.template.HtmlTemplateGenerator.getContentFromDBTemplate(domodel.getController().getObjUid(), data);
s;

```

附 HTML 代码:

```
<div id="chart1div">
    FusionCharts
</div>
<script language="JavaScript">
    var chart1 = new
FusionCharts("/${webmodule}/FusionChartsFree/FCF_Column3D.swf", "chart1Id", "400", "300",
"0", "1");

    chart1.setDataXML("${gXml}");
    chart1.render("chart1div");
</script>
```

- 2、打开在流程测试业务包，点击图表测试面板。在打开的面板中，点击修改按钮，将该面板控制器修改为 1 步中新建的控制器。



图表使用示例（控制器扩展延伸）

以平台缺省的 `t_expense` 中的测试图表实现为例，说明图表的使用。（示例中用的 `svn` 版本为：1974）

报销单查询结果	
导出Excel	导出Excel(模板)
测试图表	
报销事由	报销人
123	123
456	456
789	789
011	000
200	200
test	test

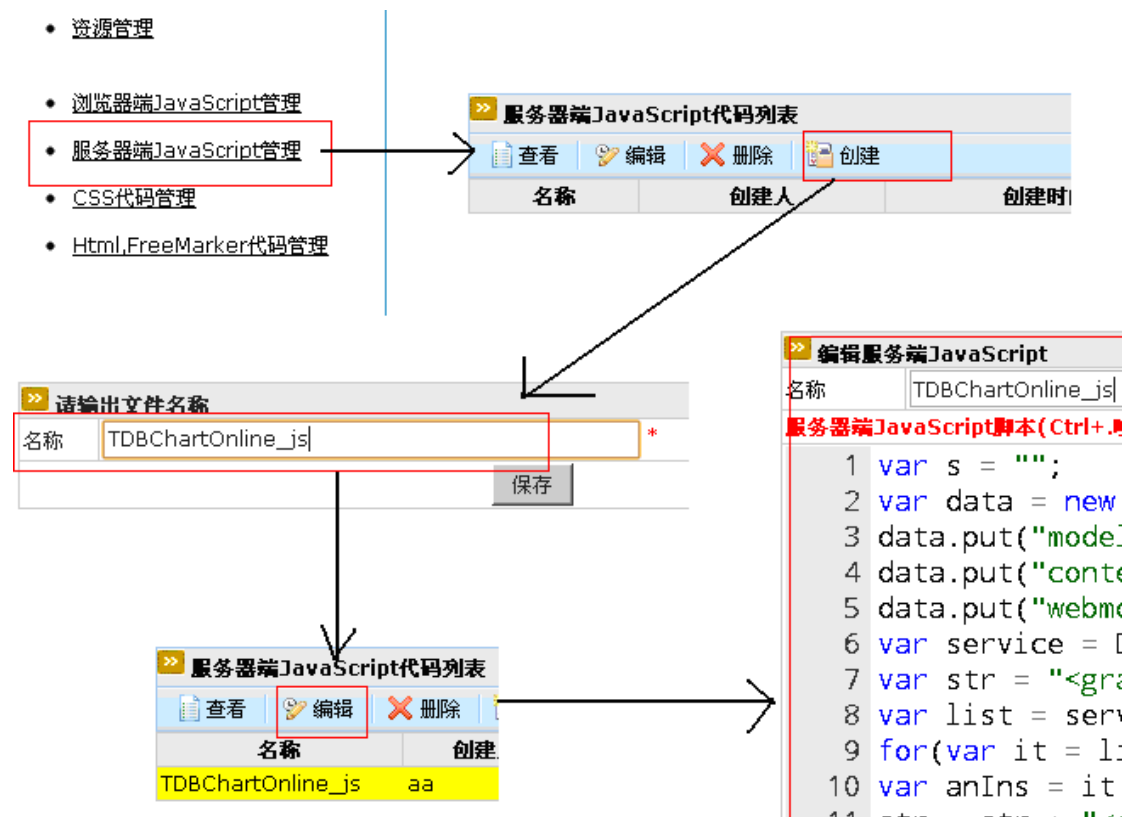
说明：本图表示例可以是控制器扩展的延伸。因为控制器的实现有如下 4 种方式：

1. 通过脚本在线编写，采用在线的 JavaScript 脚本和 freeMarker 脚本开发控制器，此类控制器在线开发即时执行，无需编译，存储在系统模型数据库中，因而在在线开发方面具有很好的优越性；此类控制器从名称上最后一部分名字以 TDB 开始，如：pane.TDBPaneTemplate、grid.TDBGridTemplate 等。
2. 实现 `com.exedosoft.plat.ui.DOIView` 接口，可以实现所有类型的控制器，即控制器的实现为一个 Java 类，此类控制器的开发、扩展修改需要编译后才能被使用；
3. 实现 `com.exedosoft.plat.ui.DOIViewTemplate` 接口或继承 `com.exedosoft.plat.ui.DOViewTemplate` 抽象类，结合 FreeMarker 模板文件可以实现所有类型的控制器，由于采用了模板，所以此类控制器的开发相对更加简洁
4.
 - o 实现 `com.exedosoft.plat.ui.jquery.form.DOBBaseForm`，实现表格元素类型的控制器
 - o 实现 `com.exedosoft.plat.ui.jquery.menu.DOBBaseMenu`，实现菜单类型的控制器
 - o 实现 `com.exedosoft.plat.ui.jquery.pane.TPaneTemplate`，结合 FreeMarker 模板，实现面板类型的控制器

在《eeplat_6 控制器扩展》文档中，介绍了第四种方式实现控制器扩展的例子，而第二和第三中方式与第四中方式类似，固未再做示例介绍。下面实现图表使用的示例中，将采用第一种方式实现控制器即在线脚本编写控制器。

配置实现步骤：

- 1、打开 eeplat 首页服务器端 JavaScript 管理面板，点击创建按钮，创建名为 TDBChartOnline_js 的 JavaScript 代码。之后在 js 代码列表中选中刚创建的 js 代码，点击编辑按钮，在编辑页面输入 js 脚本。步骤如下图：



附 js 脚本:

```

var s = "";
var data = new HashMap();
data.put("model", domodel);
data.put("contextPath", DOGlobals.PRE_FULL_FOLDER);
data.put("webmodule", DOGlobals.URL);
var service = DOService.getService("t_expense_list");
var str = "<graph>";
var list = service.invokeSelect();//老版本的 rhino js 需加中括号, 即为:service.invokeSelect([])
for(var it = list.iterator(); it.hasNext();){
var anIns = it.next();
str = str + "<set name=" + anIns.getValue("title") + " value=" +
anIns.getValue("expense_money") + " color='D64646' />";
}
str = str + "</graph>";
System.out.println("gXml" + str)
data.put("gXml",str);
s
=
Packages.com.exedosoft.plat.template.HtmlTemplateGenerator.getContentFromDBTemplate(domodel.getController().getTemplate(), data);
s;

```

2、打开 eeplat 首页 Html,FreeMarker 代码管理面板, 点击创建按钮, 创建名为 TDBChartOnline_html 的 HTML 代码。之后在 HTML 代码列表中选中刚创建的 HTML 代码, 点击编辑按钮, 在编辑页面输入 HTML 代码。步骤如下图:

- 浏览器端JavaScript管理
- 服务器端JavaScript管理
- CSS代码管理
- Html,FreeMarker代码管理

>> HTML代码

查看 编辑 删除 创建

名称	创建人
----	-----

>> 请输出文件名称

名称 TDBChartOnline_html

保存

>> 编辑HTML

名称 TDBChartOnline_html

HTML代码

```

1 <div id="chart
2     Fu
3     </div>
4 <script langua
5     va
6     ch.
7     ch.

```

>> HTML代码

查看 编辑 删除 创建

名称	创建人
TDBChartOnline_html	aa

附 HTML 代码:

```
<div id="chart1div">
```

```
    FusionCharts
```

```
</div>
```

```
<script language="JavaScript">
```

```
    var
```

```
    chart1
```

```
=
```

```
new
```

```
FusionCharts("/${webmodule}/FusionChartsFree/FCF_Column3D.swf", "chart1Id", "400", "300",
"0", "1");
```

```
    chart1.setDataXML("${gXml}");
```

```
    chart1.render("chart1div");
```

```
</script>
```

3、打开 eeplat 首页控制器管理面板，添加 pane 类型的控制器 TDBChartOnline，页面配置如下：

名称	TDBChartOnline	*
中文名	pane TDBChartOnline	名称及中文名
类型	PANE	* 类型为pane
Java Class		
逻辑脚本	TDBChartOnline_js	对应1步中创建的js代码
模板脚本	TDBChartOnline_html	对应2步中创建的html代码
输出JSP		

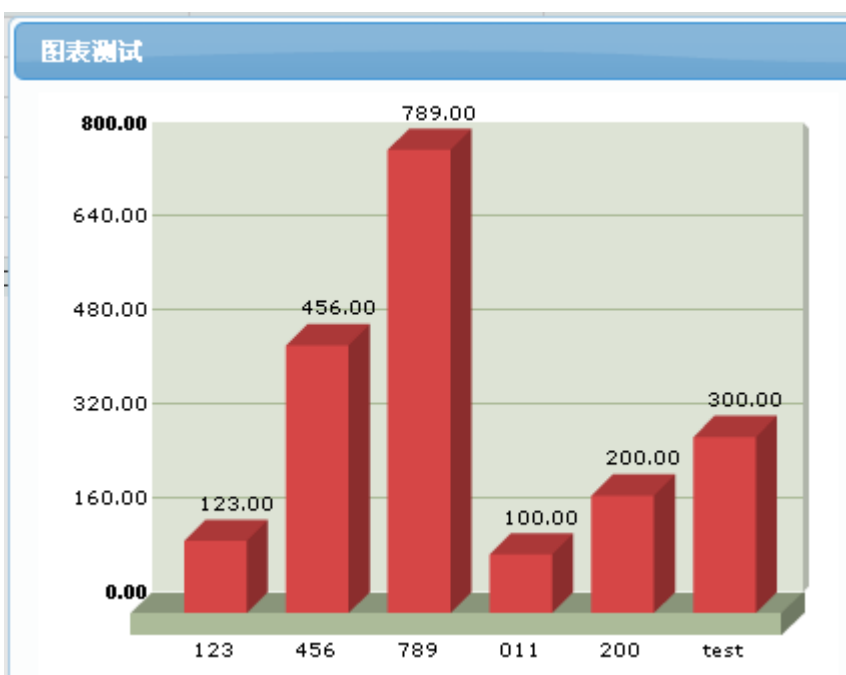
4、打开在流程测试业务包，点击图表测试面板。在打开的面板中，点击修改按钮，将该面板控制器修改为 3 步中新建的控制器。



查看	修改	创建面板	子面板列表	所有引用
>> 修改信息 点击修改按钮				
名称	PM_t_expense_testChart *			
中文名	图表测试			
主题	图表测试	连接类型		
连接的内容	流程测试	t_expense		
控制器	pane.TDBChartOnline *	选择3步中创建的控制器		
连接的业务对象	流程测试	t_expense	*	
清空总线FORM节点		面板宽度		
面板高度				
连接表格的服务	流程测试	t_expense		

至此，即完成图表的配置，进入发布之后的系统，在流程测试菜单中，打开报销单查询面板，点击图表测试按钮，效果如下：

» 报销单查询结果		
导出Excel	导出Excel(模板)	测试图表
报销事由	报销人	
123	123	2
456	456	2
789	789	2
011	000	2
200	200	2
test	test	2
第1页 每页10条 共1页 共6条记录		



角色授权（菜单权限、操作权限、数据权限）统一勾选的实现

写在前面：如果增加了角色授权统一勾选功能之后， workflow 相关待办任务的待办列表打开速度过慢（一分钟或以上），则给 workflow 相关表加索引。

执行索引的 sql 语句（执行之前，把以下 sql 语句中相关表的索引删掉）如下：

```
CREATE INDEX nodeinstance_pi_index ON do_wfi_nodeinstance (PI_UID);
CREATE INDEX nodeinstance_exestatus_index ON do_wfi_nodeinstance (ExeStatus);
CREATE INDEX nodeinstance_node_uid_index ON do_wfi_nodeinstance (node_uid);
CREATE INDEX nodeinstance_union ON do_wfi_nodeinstance (ExeStatus,PI_UID,node_uid);

CREATE INDEX do_wfi_ni_dependency_Pre_NID_UID ON do_wfi_ni_dependency (Pre_NID_UID);
CREATE INDEX do_wfi_ni_dependency_Post_NID_UID ON do_wfi_ni_dependency (Post_NID_UID);

CREATE INDEX process_exestatus_index ON do_wfi_processinstance (ExeStatus);

CREATE INDEX do_authorization_ouuid ON do_authorization (ouuid);
CREATE INDEX do_authorization_whatuid ON do_authorization (whatuid);
CREATE INDEX do_authorization_parteruid ON do_authorization (parteruid);
CREATE INDEX do_authorization_union ON do_authorization (parterUid,ouUid,whatUid);

CREATE INDEX do_org_user_role_userid_index ON do_org_user_role (USER_UID);
CREATE INDEX do_org_user_role_roleid_index ON do_org_user_role (ROLE_UID);

CREATE INDEX do_wfi_his_ni_dependency_Pre_NID_UID ON do_wfi_his_ni_dependency (Pre_NID_UID);
CREATE INDEX do_wfi_his_ni_dependency_Post_NID_UID ON do_wfi_his_ni_dependency (Post_NID_UID);

CREATE INDEX nodeinstance_his_pi_index ON do_wfi_his_nodeinstance (PI_UID);
CREATE INDEX nodeinstance_his_exestatus_index ON do_wfi_his_nodeinstance
```

```
nce (ExeStatus);
CREATE INDEX nodeinstance_his_union ON do_wfi_his_nodeinstance (ExeStatus,PI_UID);

CREATE INDEX process_history_exestatus_index ON do_wfi_his_processinstance (ExeStatus);
```

配置实现步骤:

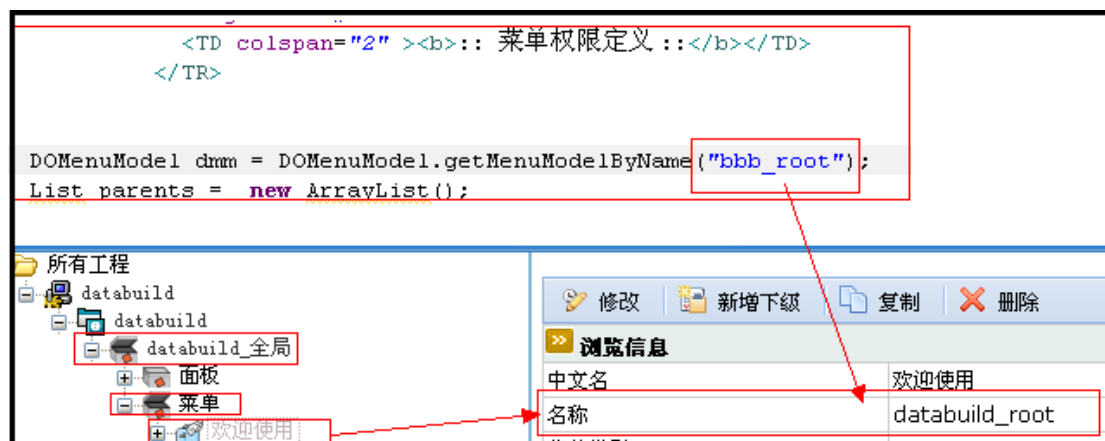
- 1、将平台 WebContent/exedo/webv3 文件夹下的 DO_Auth_Role.jsp 中第 232 行左右处的 “DOMenuModel dmm = DOMenuModel.getMenuModelByName("bbb_root");” 中的 “bbb_root” 改为自己项目的根菜单的名字。

补充:

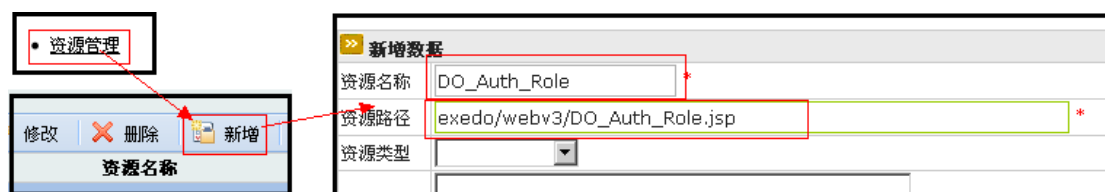
A、如果 DO_Auth_Role.jsp 文件中国际化的 js 文件没有加入，那么加上如下语句：

```
<% if ("en".equals(DOGlobals.getValue("lang.local"))){ %>
<script type="text/javascript"
src="<%=request.getContextPath()%>/exedo/webv3/js/main/lang_en.js" ></script>
<% }else{ %>
<script type="text/javascript"
src="<%=request.getContextPath()%>/exedo/webv3/js/main/lang_zh.js" ></script>
<% }%>
```

B、把 globals.xml 文件中的 `<property name="aop.formmodel.class">com.exedosoft.plat.FAOP</property>` 删掉或屏蔽掉。FAOP 是扩展权限功能，就是自己可以通过 AOP 扩展权限的定义。但是这个类只是返回 false 所以是永远没有权限。它有条件的返回 true，并且和权限表结合才有意义。



- 2、打开 eeplat 首页资源管理面板，点击新增按钮，将 DO_Auth_Role.jsp 创建为资源，配置如下：



- 3、在角色业务对象的面板中，选择一个面板，点击简单复制，创建一个新面板，用来连接

新增的资源，配置如下：

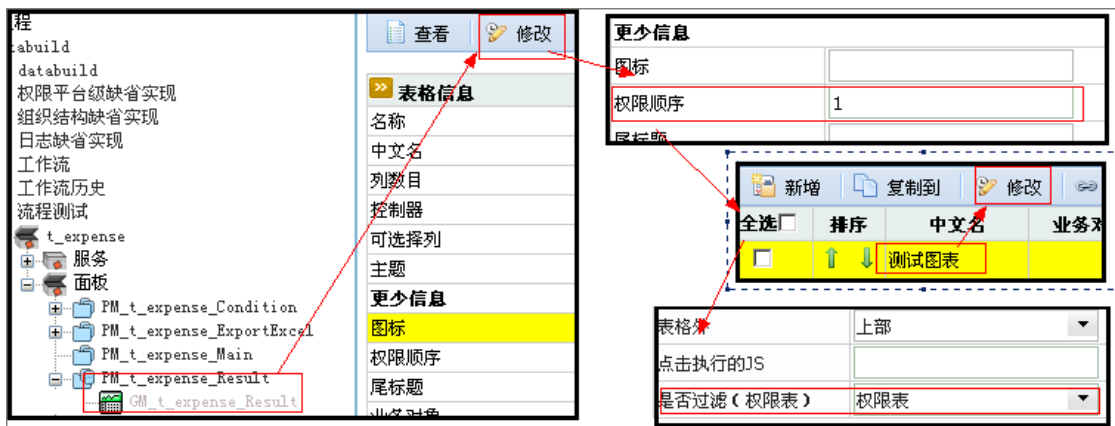


4、打开角色业务对象的 result 表格面板，新增名为设置权限的按钮表格元素，其连接面板为上步新建的面板，其目标面板为_opener_window，配置如下：



至此，完成角色授权统一勾选面板的配置，配置页面（说明：配置页面中的数据权限、操作权限及菜单权限里的数据，均需设置相应的过滤，具体设置，将在文档最后的附注中说明）效果如下：

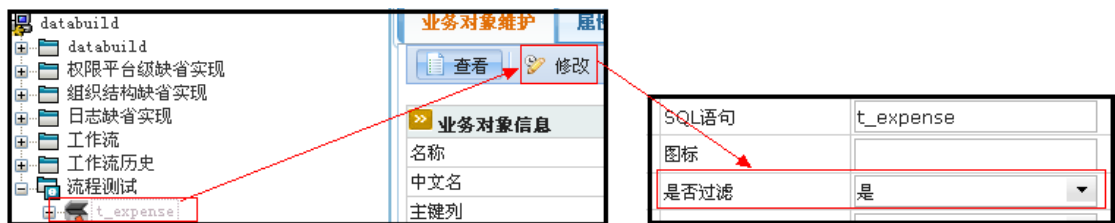
钮，设置“是否过滤（权限表）”属性为“权限表”（说明：“权限表和数据” 是通过自己的类扩展，也就是自己可以完全接管权限控制，不走系统缺省提供的权限机制）后保存即可。
配置如下所示：



3、数据权限过滤的设置：

数据权限，是业务对象实际的业务数据（如用户表的一条用户记录）的权限，是纪录级别的。
以 t_expense 业务对象为例。

打开业务对象维护面板，点击修改按钮=》设置“是否过滤”属性为“过滤”后保存即可。
配置如下所示：



JS 应用示例 (EEPLAT 版本: 1974)

以获取用户管理页面用户所在单位 ID 的值，然后再获得其所在单位的名称为例。

步骤:

新建表格元素并设置 js 属性（以“失去焦点时 JS”为例）=》编写 js 代码（=》编写自定义动作代码）。

步骤详情如下:

1. 新建表格元素

在组织结构缺省实现中的用户业务对象的用户管理修改面板，新建中文名为 ID 的表格元素，用来显示用户 ID，配置如下图:



设置新建元素的“失去焦点时 JS”属性，填写将要编写的 js 方法名，配置如下图:



2. 编写 js 代码

在 exedo/webv3/js 的 my.js 文件中（或者在平台中的“浏览器端 JavaScript 管理”中新建 js 记录），编写 js 代码，代码如下:

```
function getId(){
var uid = $("#gm_do_org_user_update_ofdept_deptuid").val();
alert("用户部门ID: "+uid);

var paras = "objuid="+uid;
//通过callService方法调用服务，也可通过callAction方法调用自定义动作
```


修改信息

>> 修改信息

ID	40288031278ed91501278	名称	admin	*
工号	000001			
部门	卫生部			
性别	男	日期		
EMAIL	admin@admin.com	电话		
手机				
家庭住址				

192.168.1.169:8080 says: 上的网页

用户所在单位名称：卫生部

☐ 禁止此页再显示对话框。

确定

用户修改密码示例 (EEPLAT 版本: 1974)

本例利用 eeplat 中重置密码自定义动作 (ResetPassword, 在 com.exedosoft.plat.login 中) 做密码修改的例子。

步骤:

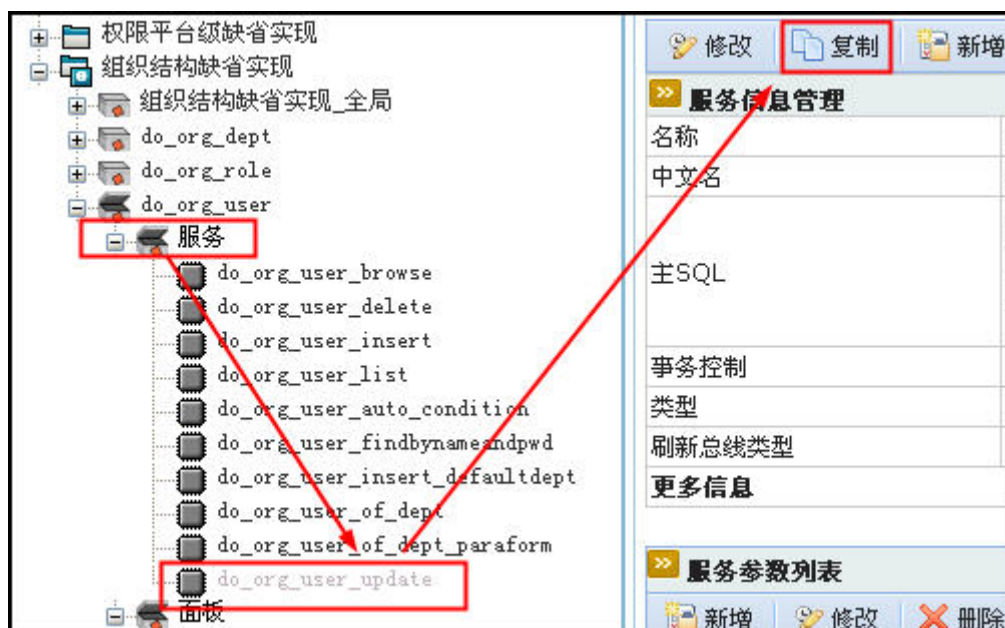
新建修改密码和查询旧密码服务=》修改 ResetPassword 类的代码=》新建修改密码面板=》配置菜单。

步骤详情如下:

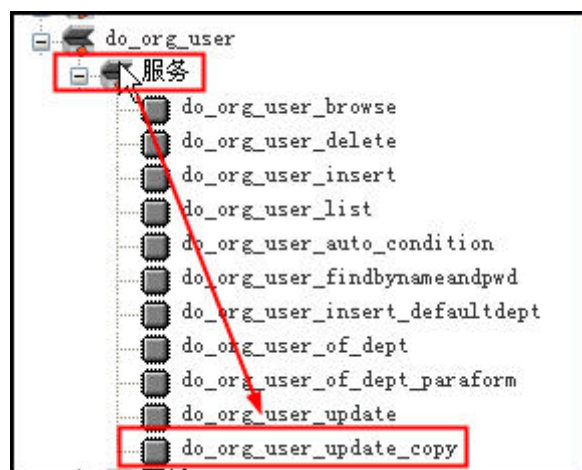
1. 新建修改密码和查询旧密码服务

A. 新建修改密码服务:

a. 点击打开用户业务对象, 并打开服务, 选择 update 服务并点击, 然后点击右侧页面中的复制按钮, 如下图所示:



b. 双击服务, 选择复制生成的 update 服务, 如下图所示:



c. 点击修改按钮，修改服务的名称及主 sql 语句，如下图所示：

>> 修改服务信息	
名称	do_org_user_update_resetpassword *
中文名	do_org_user_update_resetpassword
业务对象	组织结构缺省实现 do_org_user
主SQL	update do_org_user set password=? where objuid = ?
事务控制	
类型	修改 *
刷新总线类型	
更多信息	

d. 之后，修改服务参数列表。如下图所示：

>> 服务参数列表			
新增 修改 删除 参数检查定义 复制到			
全选	顺序	类型	参数
<input type="checkbox"/>	↑ ↓	FORM	new_password1
<input type="checkbox"/>	↑ ↓	FORM	objuid

B. 新建查询旧密码服务：

查询旧密码服务 eeplat 中有默认实现，配置如下图所示：

修改 复制 新增 服务多态 生成规则 删除 导出			
>> 服务信息管理			
名称	do_org_user_findbynameandpwd		
中文名	do_org_user_findbynameandpwd		
主SQL	select * from do_org_user where name=? and password = ?		
事务控制		参数是否过滤	
类型	查询	自定义动作	用户登陆
刷新总线类型			
更多信息			
>> 服务参数列表			
新增 修改 删除 参数检查定义 复制到			
全选	顺序	类型	参数
<input type="checkbox"/>	↑ ↓	FORM	name
<input type="checkbox"/>	↑ ↓	MD5	password_MD5

2. 修改 ResetPassword 类的代码

修改代码如下图所示：

服务信息管理

名称

do_org_user_resetpassword_zdy

中文名

do_org_user_resetpassword_zdy

主SQL

事务控制

参数是否过滤

类型

修改

自定义动作

用户重置密码

刷新总线类型

更多信息

服务参数列表

新增

修改

删除

参数检查定义

复制到

全选

顺序

类型

参数

模式

c. 深度复制 update 面板，名称定为：PM_do_org_user_update_resetpassword。

d. 打开表格面板，修改连接的服务为上面复制修改后的 browse 服务，并配置表格元素，如下图所示：

名称	业务对象属性	连接的服务	连接的面板	目标面板	控制器	是否新行	是否摘要
name					form.DOInputText		
旧密码					form.DOInputText	是	
新密码					form.DOInputText	是	
重置新密码					form.DOInputText	是	
保存		do_org_user_resetpassword_zdy	PM_do_org_user_resetpassword	PM_do_org_user_resetpassword	form.TServiceUf	是	

【注】

“名称”的配置：

修改信息

中文名

名称

重命名

连接的服务

official

official_全局

连接的面板

official

official_全局

目标面板

official

official_全局

控制器

form.DOInputText

是否只读

☒是 ☐否

允许为空

☒是 ☐否

是否新行

☐是 ☒否

相关配置

相关约束

表格外

“旧密码”的配置：

修改信息			
中文名	旧密码		
重命名	old_password	→ 用于自定义动作类中获取旧密码的值	
连接的服务	official	official_全局	
连接的面板	official	official_全局	
目标面板	official	official_全局	
控制器	录入相关	form.DOInputPassword	*
是否只读	<input type="radio"/> 是 <input type="radio"/> 否	允许为空	<input type="radio"/> 是 <input checked="" type="radio"/> 否
是否新行	<input checked="" type="radio"/> 是 <input type="radio"/> 否	相关配置	
相关约束		表格外	
校验类型	055	点击执行的JS	

“新密码”的配置：

修改信息			
中文名	新密码		
重命名	new_password1	→ 用于自定义动作类中获取新密码的值	
连接的服务	official	official_全局	
连接的面板	official	official_全局	
目标面板	official	official_全局	
控制器	录入相关	form.DOInputPassword	*
是否只读	<input type="radio"/> 是 <input type="radio"/> 否	允许为空	<input type="radio"/> 是 <input checked="" type="radio"/> 否
是否新行	<input checked="" type="radio"/> 是 <input type="radio"/> 否	相关配置	
相关约束		表格外	
校验类型	;255	点击执行的JS	
确认JS		是否过滤（权限表）	

“重复新密码”的配置：

修改信息			
中文名	重复新密码		
重命名	new_password2	→ 用于自定义动作类中获取重复新密码的值	
连接的服务	official	official_全局	
连接的面板	official	official_全局	
目标面板	official	official_全局	
控制器	录入相关	form.DOInputPassword	*
是否只读	<input type="radio"/> 是 <input checked="" type="radio"/> 否	允许为空	<input type="radio"/> 是 <input checked="" type="radio"/> 否
是否新行	<input checked="" type="radio"/> 是 <input type="radio"/> 否	相关配置	
相关约束		表格外	
校验类型	;255	点击执行的JS	
确认JS		是否过滤（权限表）	

4. 配置菜单

在菜单中，新增修改密码菜单，配置如下图所示：



至此，配置用户修改密码结束。效果图如下：

