URL表单解决方案

# 概要说明

系统目前支持两种URL表单解决方案。

## 方案一

在任务审批页面中，通过JQUERY的load方法，将表单加载到任务审批页面中，数据处理可采用前置或后置俩种处理器方式进行保存。

优点：

1. 表单处理的事务和平台的事务整合到一起。
2. 不用额外调用启动流程的API 。
3. 和系统可以完整的整合。
4. 利于新开发的系统。

局限性：

1. URL页面需要在同一个应用当中，不同域的URL加载有问题。
2. 需用户写页面处理方法并了解平台调用方法，例：在具体实现方式中往dao类增加数据保存方法。
3. 页面逻辑和数据处理逻辑分开写，对平台的依存性很强，对于现有已实现的系统功能需做较大改动才能实现整合。

## 方案2：

某些原有的系统需加入流程进行改造，如按方案一做修改，工作量大、

风险也大。

方案2的做法是：

1. 在原有的系统中调用启动流程的API进行启动流程，当然需要传入业务主键ID 。
2. 在任务审批页面中：
3. 使用iframe加载明细表单，显示业务表单的数据。
4. 在任务审批工具条中增加一个编辑数据连接，弹出页面对数据进行编辑，保存。
5. 数据处理完毕后，在完成流程任务。

优点：

1. 对原有系统改造较小。
2. URL地址可以位于不同的系统，前提是配置了单点登录。
3. 系统的独立性比较好。

缺点：

1. 数据的保存操作和流程操作需要分开执行。
2. 需要和流程API打交道。

倘若是重新开发系统的功能，建议使用方案1，改造系统选择方案2 。

# 具体实现方式：

## 方案1在系统中的实现

建立自定义表

create table person\_data(

id NUMBER(18) NOT NULL,

username VARCHAR2(127) NOT NULL,

idCard VARCHAR2(40) NOT NULL,

CONSTRAINT PK\_person\_data PRIMARY KEY (id)

)

可用代码生成器配置生成以下文件，我设置的路径是 extform

PersonDataController.java,

PersonDataDao.java,

PersonDataService.java,

PersonData.java,

PersonData.map.xml,

personDataEdit.jsp

说明：personDataEdit.jsp：用来第一次提交表单数据，personDataGet.jsp用来查看之前保存的表单数据，同时可以更新表单数据。

操作步骤：

本系统提供代码生成器生成后台和前台的代码，因此在url表单使用时，只需修改以下两个文件即可：

1. 修改PersonDataDao.java文件，往该类中添加add和updateById方法，具体详细写法如下所示：

1.需要注意的是方法的参数必须为ProcessCmd对象。

2. 在处理器中获取表单提交的数据方法如下：

cmd 为ProcessCmd对象。

Map map= cmd.getFormDataMap();

在程序中，程序将所有的表单的数据封装到map对象中，用户使用表单名称获取表单的数据。

3.注意在启动流程的时候需要将增加的数据主键，写回到cmd对象中。

代码如下：

cmd.setBusinessKey(id.toString());

add方法：

/\*\*

\* 新增数据

\* **@param** cmd

\* **@throws** Exception

\*/

**public** **void** add(ProcessCmd cmd) **throws** Exception

{

PersonData personData = **new** PersonData();

Map map= cmd.getFormDataMap();

Long id=UniqueIdUtil.*genId*();

personData.setId(id);

personData.setUsername(map.get("username").toString());

personData.setIdcard(map.get("idcard").toString());

**this**.add(personData);

**//写回主键，这个数据将保存到流程中。**

cmd.setBusinessKey(id.toString());

}

updateById方法：

/\*\*

\* 更新数据

\* **@param** cmd

\* **@throws** Exception

\*/

**public** **void** updateById(ProcessCmd cmd) **throws** Exception

{

Map map= cmd.getFormDataMap();

String id=(String) map.get("id");

PersonData personData = **this**.getById(**new** Long(id));

personData.setUsername(map.get("username").toString());

personData.setIdcard(map.get("idcard").toString());

**this**.update(personData);

}

1. 修改personDataEdit.jsp文件，将代码具体修改如下所示：



1.在表单中加载自己的js。

有时需要使用在页面中使用自己的js。如果直接在视图页面中如下写:

<script src=”js地址”></script>

这样js是不会执行的，这时我们需要使用动态加载js的方法，页面中有一个afterOnload()方法。

这个方法就是为动态加载js用的，这个方法在整个页面加载完成之后进行调用。

如果需要加载自己的js，那么在afterLoadJs数组中增加自定义的js路径。

2.在页面中使用规则验证。

使用方法如下：

在需要验证的控件中，增加一个validate 的属性，这个属性为一个json的数据。

具体验证规则说明:

系统提供了内置规则，内置规则定义在CustomValid.js文件中。

|  |  |
| --- | --- |
| 规则 | 说明 |
| required:true | 必填 |
| number:true | 任何数字 |
| variable:true | 只是数字，字母或下划线 |
| minlength:100 | 长度不少于100 |
| maxlength:100 | 长度不超过100 |
| rangeLength:[0,100] | 字符长度必须在0之100间 |
| email:true | 邮箱 |
| url:true | 所填为一个地址 |
| date:true | 日期 |
| digits:true | 整数 |
| equalTo:”相等比较的ID” | 输入值的比较，规则为控件的id。 |
| range:[0,100] | 在0到100范围之内数字 |
| maxIntLen：3 | 整数位最大长度为3 |
| maxDecimalLen:3 | 小数位最大长度为3 |

如果系统提供的验证规则不够，那么用户可以在系统的自定义验证规则中做添加。

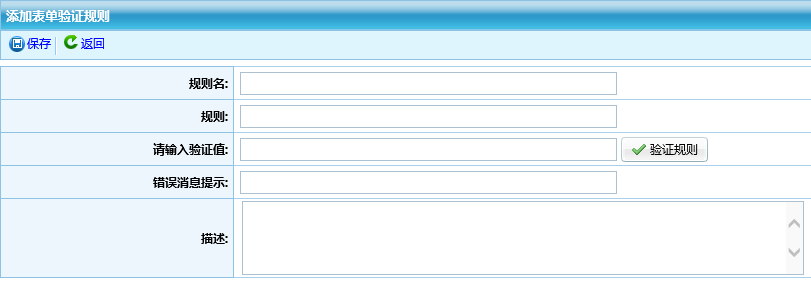


在表单中使用自定义规则方法如下：

在需要验证的控件编写如下规则：

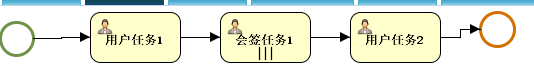
validate=”{汉字:true}” 。

添加表单验证规则。



规则使用正则表达式。

流程配置如下：

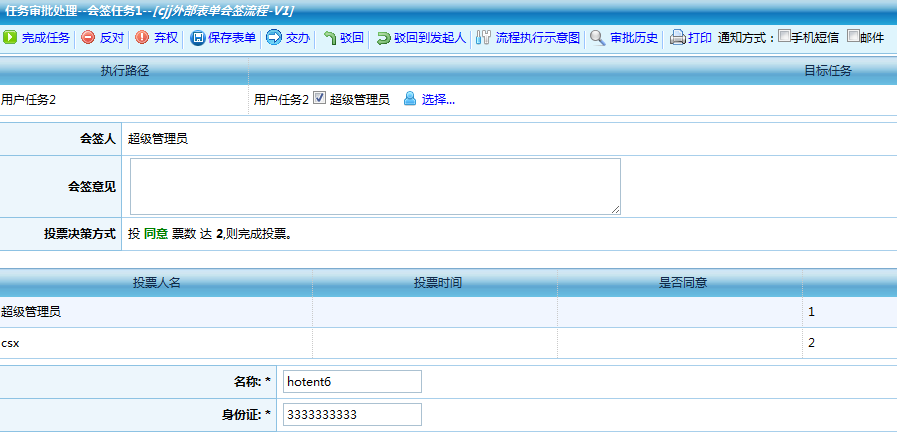


在流程定义，表单管理中作如下配置





实现效果如下

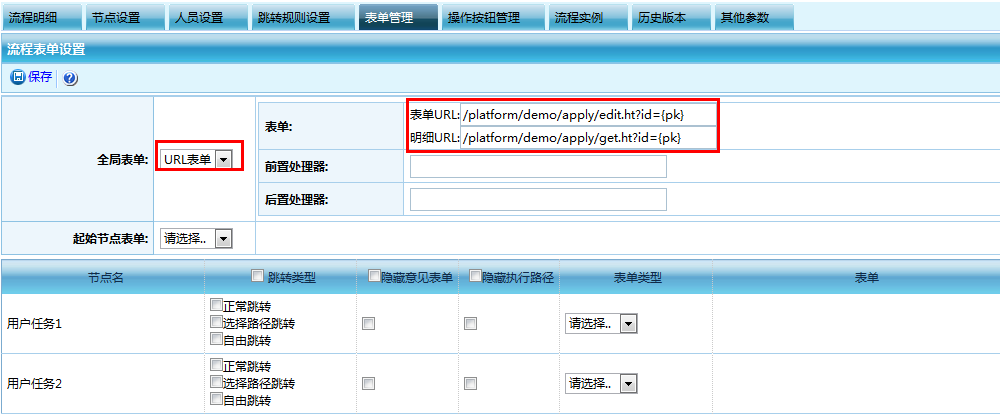


## 方案2在系统中的实现

1. 在流程定义中设置表单。

注意需要设置两个。

分别填写**表单UR**L（可对表单进行编辑的url）和**明细URL**



URL写法：

1.如果需要使用的表单不在同一应用下，则需要写以http开头的全路径如。

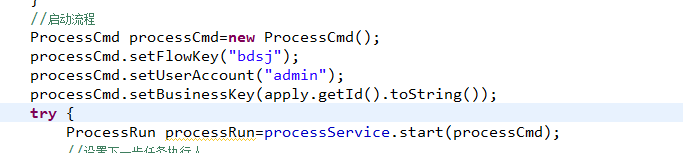
2. 如果表单页面和平台在同一个应用下,路径从根开始写，不需要上下文路径，

**注意**:两种url写法都需要传入一参数，表单主键，形式如：

[http://localhost:8089/bpm/platform/demo/apply/edit.ht?**id**={pk}](http://localhost:8089/bpm/platform/demo/apply/edit.ht?id=%7bpk%7d)。

主键必须使用**{pk}**。

1. 在需要改造的页面中调用如下API启动流程。

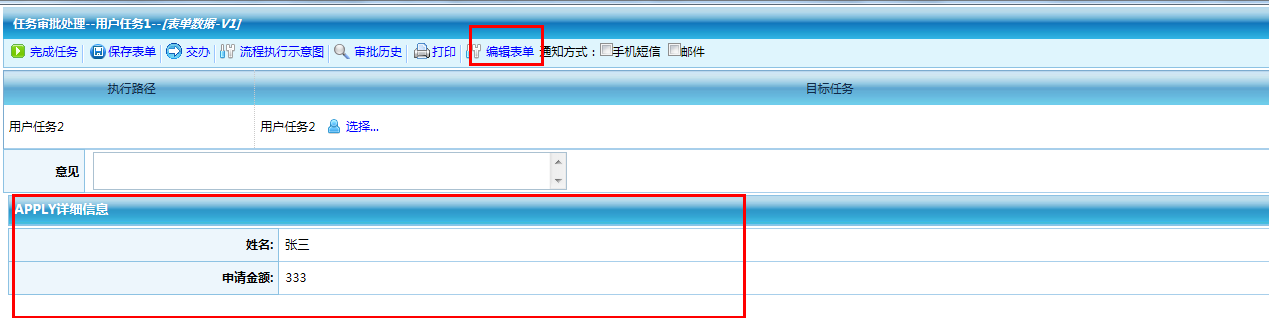


启动流程需要最少三个参数：

1. 流程KEY或流程定义ID。
2. 启动人员的帐号，这个帐号是BPMX3平台的用户帐号。
3. 业务主键。

启动流程后，那么就可以BPMX3平台处理流程执行了。

1. 实际效果如下



1. 在执行表单页面中，显示业务表单明细。
2. 点击编辑表单按钮对业务表单进行编辑。