云景开发手册

# 预备知识

## 更新说明

Bootstrap采用3.X

Knockout 采用3.2+

## 上网学习

http://www.w3school.com.cn/

Html

Jquery

CSS

http://www.cnblogs.com/TomXu/tag/Knockout/

Knockout 3.2

http://www.bootcss.com/

Bootstrap

orm

## 目录结构

**其他目录不建议项目组修改！**

/bin

**/config 配置文件（可修改）**

**/config/UIHtml 界面模板**

**/forms 表单（可修改）**

**/models模块（可修改）**

/resources 样式及图片

/script 脚本

## 学习进阶

1. 了解如何实现数据的增删改查；
2. 开发工作流表单
3. 系统框架设置
4. Gis

# 服务端

## 创建服务

服务端为dataservice，引用using IWorkFlow.Host; 继承 : BaseDataHandler

实现： public override string Key

{

get

{

return "org";

}

}

有两种开发模式：

1、直接从HttpContext传给函数；

[DataAction("login")]

public string Login(HttpContext context)

{

string uname = context.Request["username"];

string pwd2 = context.Request["password"];

...

调用方式：

var params = { username: $("#username").val(), password: $("#password").val()};

$.post("org.data?action=login", params, function (s) {

2、按要求传递给函数，如下： 默认userid都会传给每个函数

[DataAction("loginout", "userid")]

public string LoginOut(string userid)

{

security.LoginOutbyUserid(userid);

return "";

}

调用方式：

org.data?action=loginout&gid=

## 数据访问

### 强类型

Test t = new BIS001.Test();

t.ID = caseid;

t.Name = a;

t.Content = DateTime.Now;

**t.Condition.Add("ID = " + caseid);**

if (Utility.Database.Update<Test>(t) < 1)

{

Utility.Database.Insert<Test>(t);

}

### 后期生成实体

public void Insert(HttpContext context)

{

string content = context.Request["content"];

**DataRowMap** am = Utility.ToUpdateDataRowMap(content);

//取得最大值

am[am.PKey] = Utility.MaxValue(am, "F", 6);

Utility.Database.Insert(am);

context.Response.Write(Utility.ToJson<DataRowMap>(am));

}

### 数据操作条件

String caseid = “B00001”;

Test t = new BIS001.Test();

//单个条件

**t.Condition.Add("ID = " + caseid);**

**//多个条件时默认为and,例如非空**

**t.Condition.Add("ID = " + caseid);**

**t.Condition.Add(“PID is not null ”);或者PID is null**

**//OR条件，两个条件之间添加**

**t.Condition.Add("ID = " + caseid);**

**t.Condition.Add(“OR”);**

**t.Condition.Add(“PID is not null ");**

**//like条件**

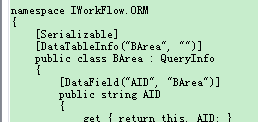
**t.Condition.Add("ID like %" + caseid + “%”);**

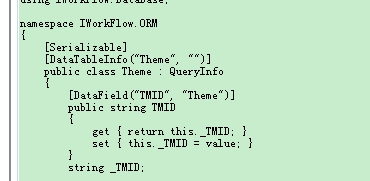
**//in或者not in条件**

**t.Condition.Add("ID in 1,2,3,4”);**

**注：所有的条件不需要添加单引号，ORM操作对字符型及数字类型不敏感。更新详细信息请参考：**IWorkFlow.DataBase. IQueryInfo接口

### ORM对象注意事项





**1.自动生成代码时DataTableInfo无法自动填充主键，需要手动输入  
 例如Theme表[DataTableInfo(“Theme”, “TMID”)]  
  
2.不希望更新的字段可设计DataField属性，例如自增主键字段不希望插入或更新的。** [DataField("autoid", "TempList",false)]

## 公共服务

(1)框架内数据库

**Mssql：**

<add name="**mydata**" connectionString="Data Source=192.168.0.10; Initial Catalog= STEPB;Persist Security Info=True;User ID= EPB;Password= EPB"/>

**Oracle:**

<add name="**myoracle**" connectionString="Data Source=ORA11;UID= EPB;PWD= EPB "/>

.net操作对象为 IWorkFlow.Host空间下Utility.Database

实体模型列子

[DataTableInfo("FX\_Group", "GID")]

public class Group : QueryInfo

{

[DataField("GID", "FX\_Group")]

public string GID

{

get { return this.\_GID; }

set { this.\_GID = value; }

}

string \_GID;

[DataField("GroupName", "FX\_Group")]

public string GroupName

{

get { return this.\_GroupName; }

set { this.\_GroupName = value; }

}

string \_GroupName;

[DataField("Descript", "FX\_Group")]

public string Descript

{

get { return this.\_Descript; }

set { this.\_Descript = value; }

}

string \_Descript;

}

类级别标记DataTableInfo;参数1:表/视图名，参数2:主键名

属性级别标记[DataField;参数1:字段名，参数2:表名（可不填）,参数3：是否被用来插入或更新bool型

实体过滤用法：(目前只支持与用法)

T t=new T();

t.Condition.add("caseid=15");

t.Condition.add("caseid<15");

t.Condition.add("caseid>15");

t.Condition.add("caseid<=15");

t.Condition.add("caseid>=15");

t.Condition.add("caseid in ii,dd,ff");

t.Condition.add("caseid like %ddd%");

t.Condition.add("caseid like ddd%");

t.Condition.add("caseid like %ddd");

实体分页用法：

t.Page= 2;

t.PageSize= 10;

t.OrderInfo="id desc";

可用方法

（1）IDbTransaction tran = Utility.Database.BeginDbTransaction();//开启一个事务

（2）Utility.Database.Commit(IDbTransaction tran);//开启一个事务

（3）Utility.Database.Rollback(IDbTransaction tran);// 回滚一个事务

（4）string sql= Utility.Database.GetBasePagerSql();//获取数据库基础分页sql

（5）DataSet ds=Utility.Database.ExcuteDataSet(string sql, IDbTransaction tran, DbParameter[] dbParms)//通过sql带事务带DbParameter返回DataSet

（6）IDataReader reader=Utility.Database.GetReader(string sql, params DbParameter[] dbParms))//通过sql返回IDataReader

（7）List<T> lst=Utility.Database.QueryList<T>(T t);

（8）List<T> lst=Utility.Database.QueryList<T>(T t,IDbTransaction tran);

（9）List<T> lst=Utility.Database.QueryList<T>(string sql);

（10）List<T> lst=Utility.Database.QueryList<T>(string sql,IDbTransaction tran);

（11）T t= Utility.Database.QueryObject<T>(T t);

（12）T t= Utility.Database.QueryObject<T>(T t,IDbTransaction tran);

（13）T t= Utility.Database.QueryObject<T>(string sql, params DbParameter[] dbParms);

（14）T t= Utility.Database.QueryObject<T>(string sql, IDbTransaction tran, params DbParameter[] dbParms);

（15）int t= Utility.Database.QueryCount<T>(T obj);

（16）int t= Utility.Database.QueryCount<T>(T obj,IDbTransaction tran);

（17）Object t= Utility.Database.QueryMaxValue(string tbName, string fieldName);//找到（tbName）表,(fieldName)字段的最大值

（18）int t=Utility.Database.ExecuteNonQuery(string sql,params DbParameter[] dbParms)//执行sql返回受影响行数

（19）int t=Update<T>(T obj)//执行更新返回受影响行数

（20）int t=Update<T>(T obj,IDbTransaction tran)//执行更新返回受影响行数(带事务)

（21）int t=Insert<T>(T obj)//执行插入返回受影响行数

（22）int t=Insert<T>(T obj,IDbTransaction tran)//执行插入返回受影响行数(带事务)

### 通用数据访问

### 日志

（23）Utity.WriteLog(string msg)//写系统日志

### 配置文件

### 当前用户权限

（1） IWorkFlow.BaseService.IWorkPrivilegeManage.AdminID//获取系统顶级管理员ID

（2） List<Privilege> lst=IWorkFlow.BaseService.IWorkPrivilegeManage.QueryPrivilegebyUserID（string userid）//获取该用户所有权限集对象

# 数据的增删改查

## 开始编码

1. 创建html文件，根据功能设计，放置相应的界面布局，该html是界面模板，不需要<html><head><body>等，也不能加入<script>，但是可以加入<style>设置css样式。

**注意：尽量采用bootstrap所设定的class类以及在config\UIHtml的html模板中所设定的class！！**

1. 创建js文件，首先构建功能模块，参见3.2！ 在this.show的函数中加入对html模板的加载，注意是异步加载！
2. 创建grid和form两类model，基本上多数功能模块都是这两类模块的组合，如下图：



FormModel

GridModel

**Js文件主要编写不用model之间的交互，以及model内部需要的一些特殊处理。**

1. 处理与服务端的交互，包括获取数据、保存数据、保存草稿等！

## 构建功能模块

$.Biz.dataManage = new function () {

//必须有show函数，params为参数，root为框架创建的模块容器div

this.show = function (params, root){}

};

调用方法如下：

/index.html#M0.Biz-dataManage:{id:001}

M0为所在的Tab页，Biz-dataManage为模块，其中”-“会自动替换为”.”， {id:001}为传入的参数，就是show函数所传递的params！

## 服务端交互

$.fxPost(serUrl + '?action=GetData', {param:value}, function (data) {

});

fxPost为框架函数，会将获取到的字符串序列化为json格式，自行处理服务端的错误信息，会自动处理防止取得缓存数据，同时在离线时取得本地缓存数据。若当前用户已过期，会自动提示重新登陆，无需刷新整个页面。

**对于2000行以内，字段数量不超过30个的数据，可以一次性拿到当前功能所需的所有的数据，以json格式返回。之外的需要自己写服务段查询及分页的代码。具体需要根据实际响应速度进行测试。**

通过调用表单的getData()函数，拿到需保存的数据，提交到服务端保存，调用getCacheData() 获取需到草稿的数据。

## 创建grid

获取数据后，将数据加载到数据网格（grid），做法如下：

var gridmode = $.Com.GridModel(options);

options包括：

{

beforeBind:function (vm, root){}; // 绑定前,一般用来创建一些vm的新的属性

afterBind:function (vm, root){}; //绑定后

edit:function (item, callback){}; //编辑,点击某行进行编辑,行,编辑后的回调

remove :function (row) {}; // 删除前回调,如返回false则阻止删除

keyColumns：string //主键字段名

columns: [] //字段设置

filters:[] //过滤条件

cssClass: string //表格样式

elementsCount: int //分页,设置每页显示行数

setRowClick:ture|false //是否整行选择

}

接口包括：

Show(root,data) //呈现数据,根div,需绑定的数据

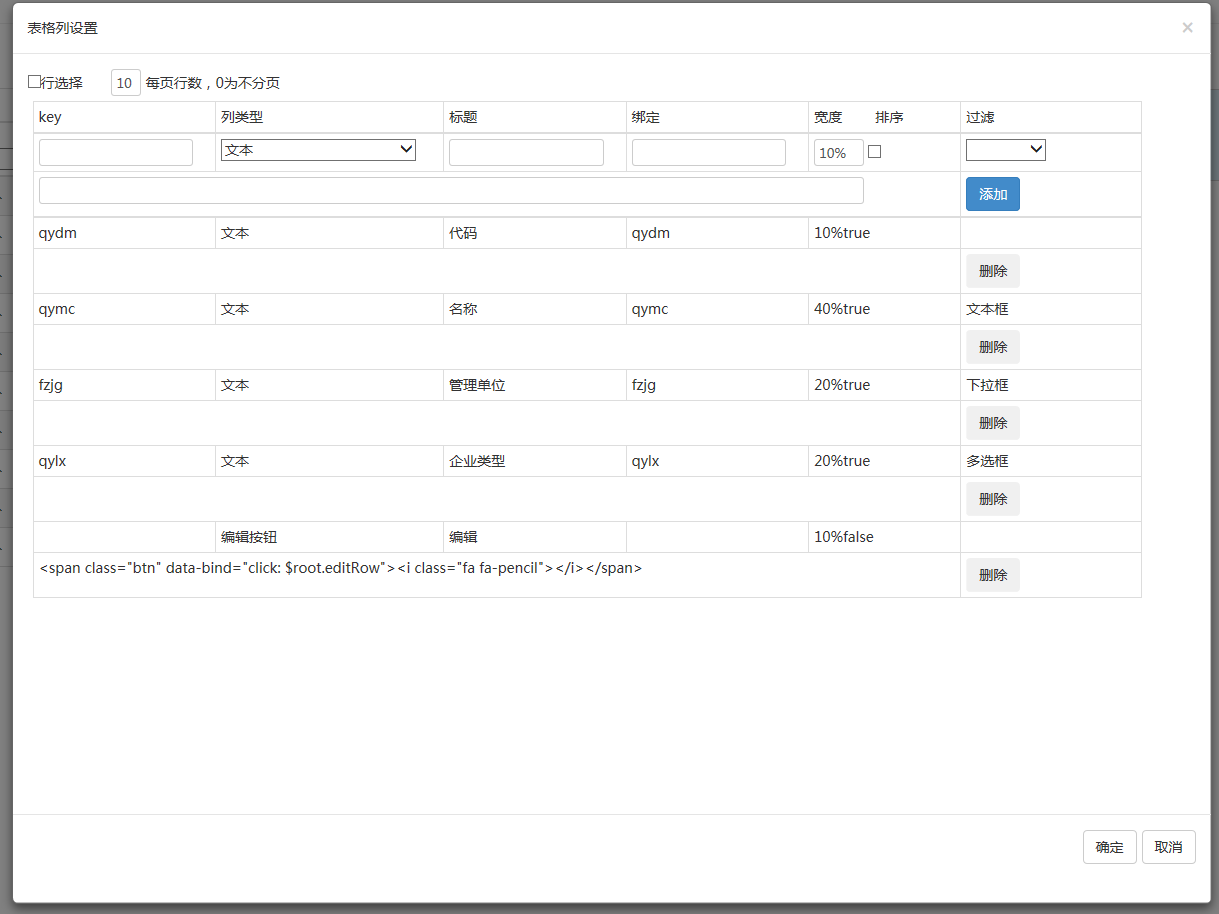
getData() //获取数据,可以提交给服务端保存，主要包括新增及编辑过的数据，其中新增的数据pk字段为空，另外还有删除的数据列表。

getCacheData() //获取需到草稿的数据，包括grid全部的数据，不管是否编辑过，另外还有删除的数据列表。

**注意：请特别注意getData与getCacheData的区别，可以通过断点调试察看数据的不同。主要的区别在于是否包含未编辑过的数据。**

其中： columns: []、filters:[]、cssClass: string 不推荐使用，建议采用grid的模板的机制来设置grid的样式。具体做法可以使用调试模式下grid的设置界面。





其中列类型有：

1. 文本：
2. 转义列，就是将数据集合的代码翻译成文字的功能，需要配合客户端缓存，比如有一个企业类型的字段，数据库保存的是0、1、2…，客户端缓存了{0:’国企’,1:’外资’,2:’民营’}，则可以设置为转义列，系统会自动显示对应的中文意思。
3. 序号；
4. 文本框，grid会提供文本框编辑的功能，并自动更新到对应的数据集上；
5. 下拉框，也是提供数据编辑的功能；
6. 删除按钮：<span class="btn" data-bind="click: $root.removeRow"><i class="fa fa-times"></i></span>
7. 编辑按钮：<span class="btn" data-bind="click: $root.editRow"><i class="fa fa-pencil"></i></span>

过滤条件包括：

1. 文本框：模糊查询
2. 下拉框：自动获取当前加载的数据集全部可能的值，也可以另外设置下拉数据源；
3. 多选框：自动获取所有可能的值，可多选；
4. 数值区间，提供大于等于和小于等于的两个输入框
5. 日期区间：提供大于等于和小于等于的两个输入框，自动设置为日期选择



## 子表单

创建：

var basemodel = $.Com.FormModel(options);

options包括：

{

beforeBind:function (vm, root){}; // 绑定前,一般用来创建vm的新的属性

afterBind:function (vm, root){}; //绑定后

beforeSave:function (vm, root){}; //保存前,一般用来数据效验

}

接口包括：

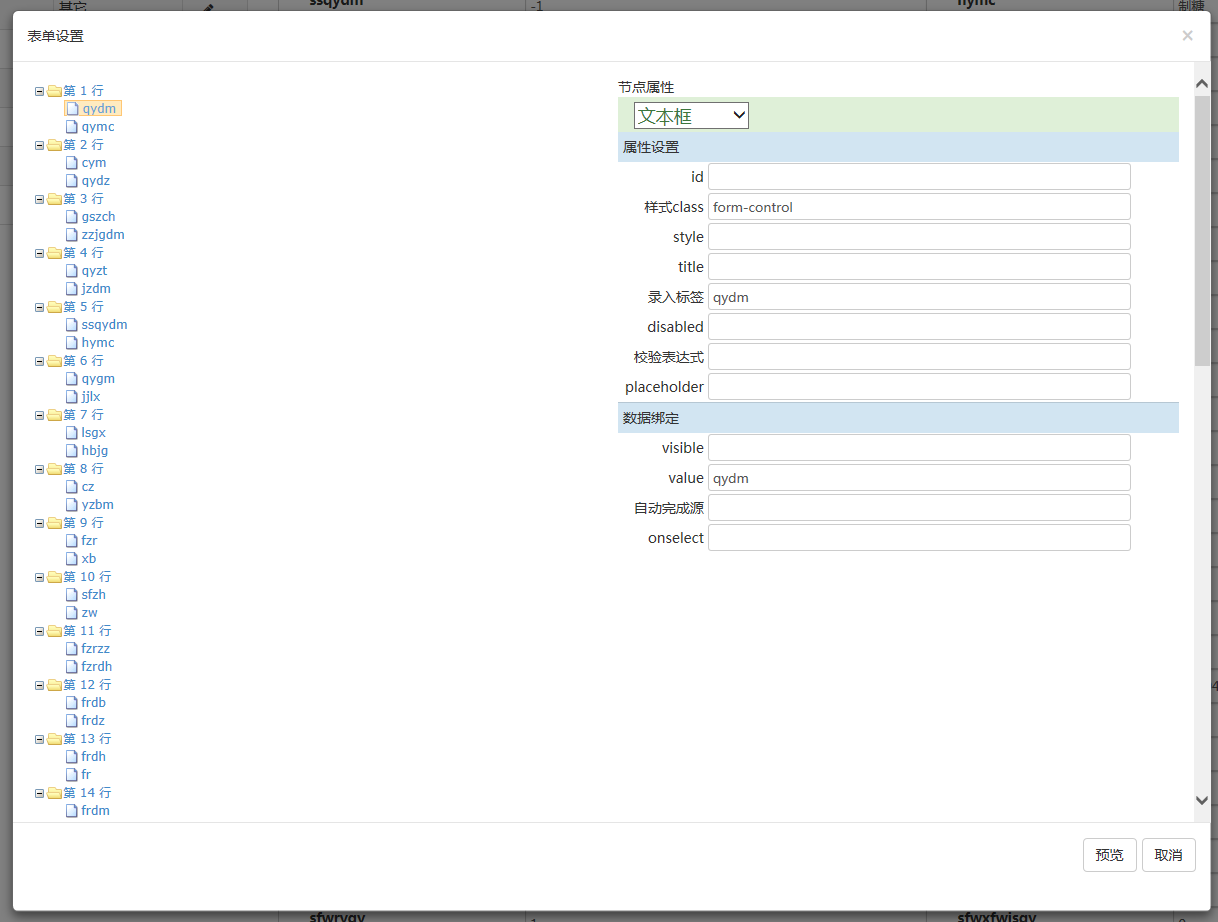
Show(root,data) //呈现数据,根div,需绑定的数据

getData() //获取数据,可以提交给服务端保存

getCacheData() //获取需到草稿的数据，调用该函数不会触发数据的校验！

**注意：请特别注意getData与getCacheData的区别，可以通过断点调试察看数据的不同。主要的区别在于是否进行数据校验！**

点击修改配置，显示如下界面



## 录入控件

具体参考表单设计器。以下为一些需要说明的控件：

### 下拉框

可选项建议放在统一的客户端缓存，具体配置参见：5.8 客户端数据缓存。

## 只读

系统提供只读设置的函数：

$.Com.setReadOnly(root, {'办理情况':'keep'},true);

setReadOnly: function (target, options, isreadonly) {

// target：目标div

// optins={'data-part':'readonly','data-part':'keep','data-part':'hide'}

// data-readonly-type:控件只读方式（可选值：readonly-只读、hide-隐藏）

系统提供了表单只读的设置功能，采用嵌套设置模式，有两个标签，如下：

<div **data-part="主表"**>

<div>

<label>单位名称：</label>

<div>

<input data-id="unitname" type="text" data-bind="value:UnitName"/>

<button type="button" data-bind="click: \_unitSel" data-readonly-type ="hide">选择</button>

</div>

</div>

<div>

<label>单位地址：</label>

<div> <input data-bind="value:UnitAdd" type="text" /> </div>

</div>

<div **data-part="办理情况"**>

<label>办理情况：</label>

<div> <textarea data-bind="value:DealCondition" ></textarea> </div>

</div>

</div>

系统会自动设置表单为只读，但是办理情况部分会保留可写状态，选择按钮会被隐藏。

## 数据校验

**jquery-html5Validate.js 基于HTML5表单验证的jQuery插件**

http://www.zhangxinxu.com/wordpress/?p=2857

# 公共组件及函数

详细参考jquery.iworkflow.js

## iwfWindow 弹窗

## listView2 树形列表

$.fn.listView2 = function (options) {};

options说明：

{ expandbyClick:false, //是否点击父节点展开子节点

flowlayout:100; //叶子节点宽度

expandable:fasle, //是否可展开

data: [

{ title: ‘新建业务’, text: ‘新建’, iconCls: ‘icon-arrow-left’,

css: ‘btn-primary’,id:‘001’,handler: function () {alert("");}

},

{ type: 'split' },

{ type: 'group',title: '分组的待办箱', text: '待办箱', iconCls: 'icon-save',

id:'002', expend:true,expandbyClick:true,unselectable:true, children:

[ { id: 'cd-001', title: '信访', text: '信访',url:'' },

{ id: 'cd-003', title: '收文', text: ' 收文 ',url:'' } ]

},

{ title: '系统设置', text: '系统设置',url:'' }

]

}

## iwfToolbar 工具栏

data: [

{ title: '保存', text: '保存', iconCls: 'icon-save', css: 'btn-primary' ,

handler: function (data,sender) {alert("");}},

{ type: 'split' },

{ content:'<select> ...</select>' },

{ type: 'group', children: [

{type: 'radio', name: 'codewin', id: 'cd-001', title: '设计', text: '设计', value: true },

{type: 'radio', name: 'codewin', id: 'cd-003', title: 'JS', text: ' JS ', value: false }

]

},

{

type: 'menu', iconCls: 'fa fa-table', text: '表格', handler: tableAction,

flowlayout:100, children: [

{ id: 'td-001', name: '插入表格' },

{ id: 'td-005', name: '合并下一行' }

]

如果没有children,通过show(root),可以扩展下拉菜单的面板

}, { type: 'split' }

,{ title: '设置为业务模块', text: '业务模块', handler: addToPrivilege, css: 'btn-info' }

, {type: 'group',float:'right', children: [

{

type: 'checkbox', name: 'ddd', checked: true, title: '在用', text: '在用',

value:'useing', handler: function (data,element) {}

},

{

type: 'checkbox', name: 'ddd', checked: true, title: '备用', text: '备用',

value:'backup',handler: function (data,element) {**}**

} ]

]

css: {

root: 'iwf-toolbar',

button: 'btn btn-default',

link: 'iwf-toolbar-link',

text: 'iwf-toolbar-text2',

select: 'active',

split: 'btn-split',

circle: 'btn-circle',

menu: 'btn-menu'

},

## iwfTab tab页

## DropdownMenu 下拉菜单

## ComDatetimepicker 日期选择

## setCookies()

## getCookies()

## delCookies()

## uniq(json,pro) 取唯一值

这里面的json是一个二维表，就是以下格式json1=[{pro1:’a1’,pro2:’b1’},{pro1:’a2’,pro2:’b2’}]，然后获取其属性的唯一值，比如uniq(json1,pro1)返回[‘a1’,’a2’]

## formatDate(date) 日期格式化

自动将输入的日期格式化为界面友好的字符串，比如：今天 10:20 、昨天9：29、本月5日。

## addStyle(href) 加载css

## addScript(href) 加载js

## getDict 获取缓存字典

参见5.8 客户端数据缓存

## Word正文控件

打开正文函数：

function ShowWindow(type, template, url, action, userName, bookmarks) {

var x;

var y;

if (window.frameElement) {

x = $(window.frameElement).position().left;

y = $(window.frameElement).position().top;

} else {

x = 0;

y = 0;

}

var window\_height = pageHeight() - 78;

var window\_width = document.documentElement.clientWidth;

var dialog\_Word = $.dialog({

content: 'url:' + url,

title: '正文信息',

id: type,

left: x,

top: y,

width: window\_width,

height: window\_height,

drag: false,

lock: true,

data: { Type: type, Template: template, WorkFlowCaseId: wftool.wfcase.caseid, Action: action, UserName: userName, Bookmarks: bookmarks },

min: false,

max: false,

close: function () {

if (action != "Look" && confirm("是否保存正文？")) {//查看时不提示保存

this.content.DocumentSave();

}

this.content.WebOfficeClose();//关闭文档，释放资源

InitWordControl();

},

init: function () {

//this.iwin.document.getElementById('caseid').value = caseId;

//this.max();

}

}).max();

}

function pageHeight() {

var c = document.documentElement.clientHeight;

var b = document.body.scrollHeight;

return (c > b) ? c : b;

}

ShowWindow函数参数说明：

type-word类型，生成的文件名、模板路径文件夹和保存的文件路径跟该类型有关系，需要  
在web.config中配置；

template-模板文件的名称，url-显示正文的网页路径；

action-操作类型；

userName-当前操作用户，用于留痕；

bookmarks-书签，用于替换模板中对应的书签。

# 框架

## 用户及权限管理

查看系统的用户及权限管理模块

## 页面容器 尚不可用！

页面容器提供了一种便捷的功能扩展机制，效果如下：

Index.html#M1. iframemodel:{url:’www.baidu.com’, title:’baidu’,keyword=’环保APP’}

url为页面路径，title为标题，其他会自动拼成?keyword=’’&…传递给页面。

## 界面主题

themes: {

PCLayout: [

{

id: "default",

TempFile: 'config/UIHtml/indexpc.html',

Name: "默认",

img: "resources/theme/default.png",

CssFile: "v3/css/bootstrap.css",

descript: "默认主题，适用于各种分辨率",

maxWidth: 2560,

minWidth: 1024,

uiMode: "mouse"

},

{

id: "pc01",

TempFile: 'config/UIHtml/indexpc3.html',

Name: "纯色标题栏",

img: "resources/theme/default.png",

CssFile: "v3/css/bootstrap3.css",

descript: "简单桌面布局，适用于各种分辨率",

maxWidth: 2560,

minWidth: 1024,

uiMode: "mouse"

},

{

id: "pcmenu",

TempFile: 'config/UIHtml/indexpcmenu.html',

Name: "开始菜单",

img: "resources/theme/default.png",

CssFile: "v3/css/bootstrap.css",

descript: "具有开始菜单，适用于各种分辨率",

maxWidth: 2560,

minWidth: 1024,

uiConfig: "UIConfigMenu.json",

uiMode: "mouse"

}

]

}界面主题包括：TempFile：模板文件，CssFile：bootstrap样式文件，maxWidth: 最大分辨率, minWidth: 最小分辨率，uiMode：操作模式（"mouse"|”touch”）， uiConfig：界面布局配置文件，portalConfig：门户界面配置文件。

**TempFile包含了对界面样式的设定，配合对应的bootstrap样式，设置专有的css class。**

## 界面布局

界面布局采用如下的配置字符串格式：

[

{**module**:'casemodule', **model**:'doingcase',**text**:'我的业务', **params**:'', **closable**:false,**children**:[

{**url**:'casemodule.doingcase:{title:"待办箱"}',**text**:'待办箱'},

{url:'', text: '-' },

{url:'casemodule.havebeendone:{title:"在办箱"}',text:'在办箱'},

{url:'casemodule.donecase:{title:"办结箱"}',text:'办结箱'},

{url:'casemodule.newsmodel:{title:"新闻公告"}',text:'新闻公告'}]

},

{module:'sysmodule', model:'MaxPrivilege',text:'系统功能',param:'', closable:false, children:[

{url:'sysmodule.usermanage:{title:"用户管理"}',text:'用户管理'},

{url:'sysmodule.workflowdesign:{title:"工作流管理"}',text:'工作流管理'},

{url:'sysmodule.groupmanage:{title:"页面排版"}',text:'页面排版'}]

}

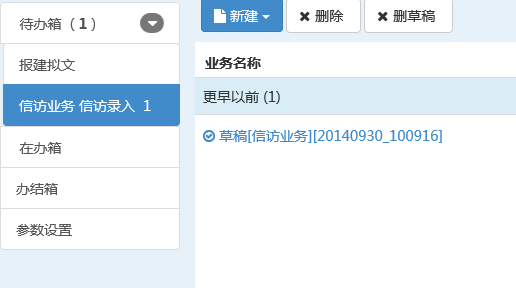
]

框架可以让每个用户自由配置界面，并保存在cookie中。

## 待办箱分组

在config.js文件中的 doingCaseConfig部分，

设置之后效果如下：



默认会给每一个流程的每一个步骤设置一个待办箱子项，比如上图的”信访业务”流程的”信访录入”步骤,可以给每个步骤指定处理页面，如下所示：

**{ text: '信访审批', FlowID: 'W000026', ActID: 'A006', url:'xinfangbanli:{}' }**

可以将一个流程的全部步骤设置为一个子项，如下所示：

**{ text: '信访审批', FlowID: 'W000026', url: '' }**

可以将多个流程的多个步骤合并在一个待办箱子项，如下所示：

**{ text: '集中办理', Acts: [{ "FlowID": "W000071", "ActID": "A001" }, { "FlowID": "W000070", "ActID": "A002" }], url: '' },**

同时，系统提供了always关键字，标示一直出现在待办箱，不论有没有对应的业务

**{ text: '报建拟文', FlowID: 'W000018', ActID: 'A020', url: 'usermanage:{title:"用户管理"}', always: true },**

## 新建业务菜单设置

开发人员可以订制新建业务菜单的界面布局，包括业务的排序、图标、样式、分组等。格式如下，加入到config.js文件中。

newCaseMenu: [{

"id": 0,

"text": "环境监察",

"type": "group",

"css": "list-group-item treeview-bar",

"unselectable": true,

"children": [

{

"id": "W000040",

"text": "信访业务",

"css": "treeview-item btn btn-default panel-body",

"iconCls": "fa fa-share-square-o fa-lg"

}

]

}]

## 客户端数据缓存

在**config\DataDictConfig.json**文件中包含了对客户端数据缓存的设置，在系统初始化的时候自动加载到客户端，如下：

{

"test": "select TMID,Name from FX\_Theme as test;",

"test0": "select Name from FX\_Theme as test",

"test1": [ ],

"test2": {"1": "a","2": "b" }

}

从数据库中读取，如只有1个字段，则构造为数组，两个字段，则为json。

通过$.Com.getDict (dictType)可以获取到缓存数据，比如$.Com.getDict(test1)返回[]。

## 消息推送

在config.js文件有一个属性：

messageInterval: 15000, //消息轮询间隔

用于设定消息轮训的间隔，当然，以后引入实时设计后会取消该设置！

主要是在服务段调用消息推送：

MessageManager.addMsg(cMsg msg)

public class cMsg

{

public string guid { get; set; }

//标题

public string text { get; set; }

//说明

public string descript { get; set; }

//是否已经显示过

public bool haveShow { get; set; }

//接收人，如果有多个接收人就有多条业务

public string recipient { get; set; }

//图标

public string icon { get; set; }

//地址，没有index.html#

public string url { get; set; }

//创建时间，默认用创建时间排序

DateTime createTime { get; set; }

//类型，预留，用来将消息分类处理，比如说系统消息，即时消息、任务推送等，或短信提醒类消息等等

//once/keeponeday/always/sys

//不显示列表、保留一天、一直保留、直到已阅、系统消息

public string type { get; set; }

}

## 离线设置

正在优化中…

## 门户

本开发框架提供了门户功能，门户的也是通过配置字符串将model组合，构建起门户，字符串格式如下：

{**columns**:[{**width**:'40%'},{width:'35%'},{width:'25%'}],

**porlet**:[

{ **id**: 'portlet1', **title**: '我的OA', **column**: 0, **closable**: true, **fold**: true ,**model**:'MyCaseOAModel',**params**:'title:""' },

{ id: 'portlet0', title: '我的业务', column: 0, closable: true, fold: true ,model:'MyCaseModel',params:'title:""' },

…

{ id: 'portlet20', title: '排污费入库统计', column: 2, closable: true, fold: true, model:'ChargeThumbNailModel',params:'title:""'}

]

}

每个用户可以自由配置，并保存在cookie！

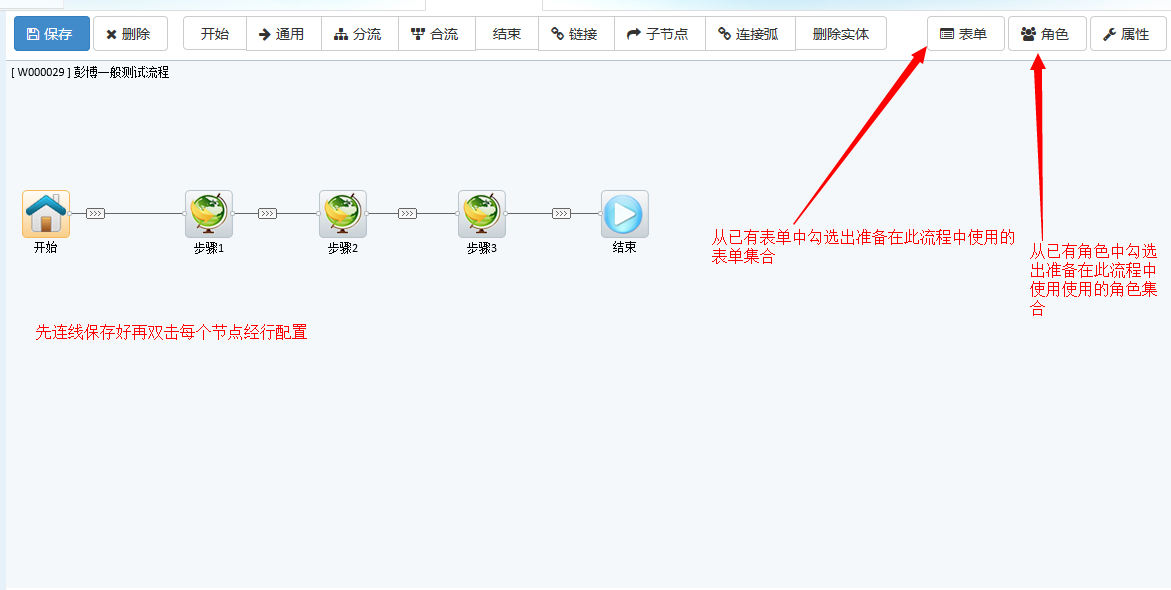
可以为不同的界面布局设置不同的门户配置，比如大屏幕可以有三列或四列，而手机只有1列，小屏幕可以为两列。

# 工作流

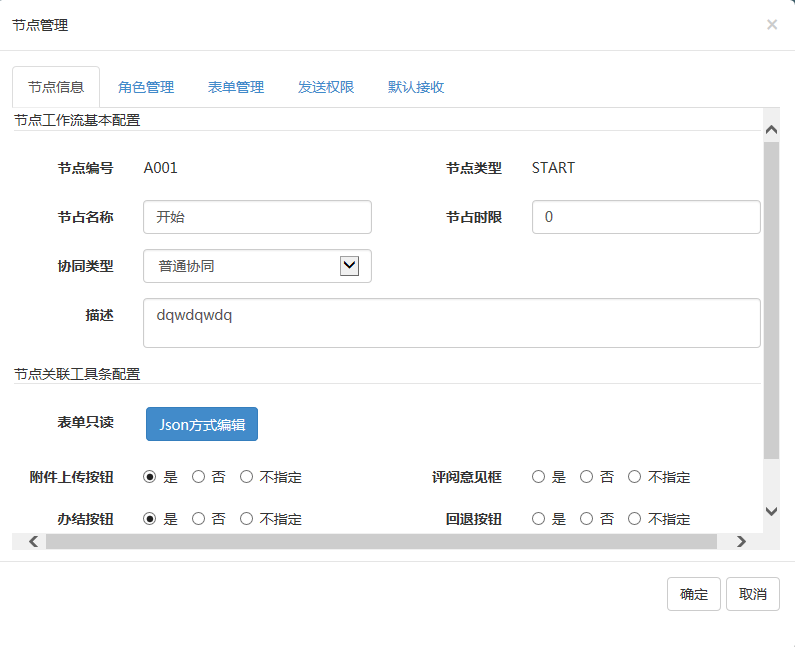
## 工作流设计

查看工作流设计模块，系统会将工作流保存在config\WFModel目录下。

设计截图描述



双击单个节点，配置



## 工具栏开发

工具条实例化时 options对象具有以下参数

{

Url: "/PBTest.data",

saveData: function (obj) {

return "fsdfs";

},

getRemark: function () {

return "PB意见";//得到意见

},

getCaseName: function () {

return "PB会签高级测试";

},

afterSave: function () {

alert("触发保存完毕事件");

},

afterSend: function () {

alert("触发发送完毕事件");

},

afterInit: function () {

alert("触发工具条加载完毕事件");

},

beforeBack: function () {

alert("退件之前事件触发");//this=当前button

},

afterUploadOK: function (event, data) {

alert("上传成功了事件触发");

},

showRemarkList: true,//是否显示评阅意见<div id="WF\_RemarkList"></div>

showRemarkBox: true,//是否出现意见框<div id="WF\_RemarkBox"></div>

showAttachment: true,//是否出现附件<div id="WF\_Attachment"></div>

showSaveBtn: true,//是否出现保存按钮

showBackBtn: true,//是否出现回退按钮

showSendBtn: true,//是否出现发送按钮

showEndBtn: true,//是否出现办结按钮

showFlowChart: true,//是否出现流程图

showActionAndReceiverSelect: true//是否出现选步骤和人按钮

}

工具条实例化后对象wftool具有方法如下

wftool.SetReceivers(["U0000124", "U0000128"]);//设置接收人为【林雁生,陈树强】

wftool.SetActAndReceivers("A001", ["U0000117", "U0000119"], function () { });//设置步骤同时设置接收人

wftool.GetSelectActInfo();//获取选中步骤信息

wftool.GetCurActInfo ();//获取当前步骤信息

wftool.GetDom("send");//获取工具条元素，元素标识以iwftype=“什什么”为主

## 工具栏服务端

与工作流对应服务写法已PBTest.data为例子

至少实现以下3个服务

[DataAction("GetData", "userid", "caseid")]

public string GetData (string userid, string caseid)

{

return 你的返回值//这里返回值样子必须是以下类型的Json，你要的数据就放在data里

class JSON

{

public bool success;

public string content;

public string msg;

public string tips;

public Object data;

}

}

[DataAction("save", "BizParams", "userid", "content")]

public string Save(string BizParams, string userid, string content)

{

SkyLandDeveloper developer = SkyLandDeveloper.FromJson(BizParams);

IDbTransaction tran = Utility.Database.BeginDbTransaction();

try

{

string caseid = developer.Create(userid, tran);//获取创建后的

//这里写你的代码

developer.Commit(tran);

return developer.JsonResult(true, "你的成功提示信息");//

}

catch (Exception e)

{

developer.RollBack(tran);

return developer.JsonResult(false, "你的失败提示信息");

}

}

[DataAction("send", "BizParams", "userid", "content")]

public string send (string BizParams, string userid, string content)

{

SkyLandDeveloper developer = SkyLandDeveloper.FromJson(BizParams);

IDbTransaction tran = Utility.Database.BeginDbTransaction();

try

{

string caseid = developer.Create(userid, tran);//获取创建后的

//这里写你的代码

//developer**.**ExpireDate **=** DateTime**.**Now**.**AddDays**(10);//这是设置下一步时限的例子**

//developer**.**caseName=”你的额名字”；//这是设置业务名的例子

//developer**.**limit=10；//这是设置整个流程时长的例子

//developer.attachkeys可以获得所有页面的上附件的Key的list集合

【附件系统表：FX\_AttachMent】

developer**.**Send**(**userid**,** tran**); //多一句话**

developer.Commit(tran);

return developer.JsonResult(true, "你的成功提示信息");//

}

catch (Exception e)

{

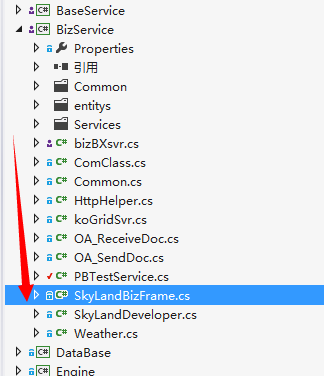
developer.RollBack(tran);

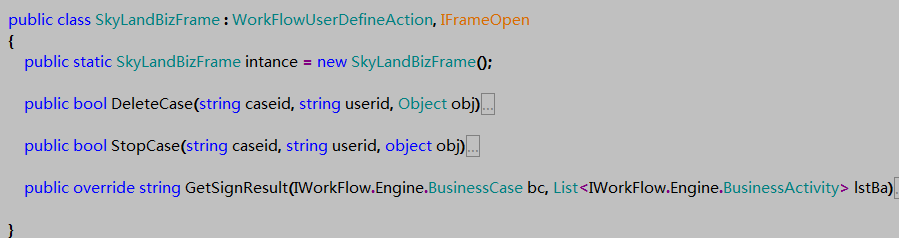
return developer.JsonResult(false, "你的失败提示信息");

}

}

另外以下位置将方便的提供你删除业务的回调和停办业务的回调，已经会签完成的回调





## 服务端

可用方法

(1)找到所有未办结业务

List<IWorkFlow.Engine.BusinessCase> lst=IWorkFlow.Engine.EngineHost.Instance.QueryAll();

(2)找到所有未办结业务(且属于该userid的待办业务)

List<IWorkFlow.Engine.CaseItem> lst=IWorkFlow.Engine.EngineHost.Instance.QueryMyCase(string userid);

(3)找到所有未办结业务(且属于该userid的已办（或者叫在办）业务)

List<IWorkFlow.Engine.CaseItem> lst=IWorkFlow.Engine.EngineHost.Instance.QueryHaveDoneCase(string userid);

(4)找到所有未办结业务(且业务ID等于caseid的那唯一一条业务)

IWorkFlow.Engine.BusinessCase lst=IWorkFlow.Engine.EngineHost.Instance.QuerythisCase(string caseid）

(5)找到所有停办的业务

List<IWorkFlow.Engine.CaseItem> lst=IWorkFlow.Engine.EngineHost.Instance.QueryStopCase(）

(6)带事务重构服务端缓存（一般在异常出错时调用（谨慎用））

bool flag=IWorkFlow.Engine.EngineHost.Instance.ReLoadOneCase(string caseid, Object tran)

(7)工作流回退（业务号,流转号,提交人,意见,事务）

bool flag=IWorkFlow.Engine.EngineHost.Instance.Back(string caseID, string baID, string senderid, string remark, object args)//

(8)工作流退送（业务号,流转号,回退节点ID,提交人,意见,事务）

bool flag=IWorkFlow.Engine.EngineHost.Instance.BackSend(string caseID, string baID, string ReceiveAct, string senderid, string remark, object args)//

(9)工作流发送（业务号,流转号,提交人,接受节点,接受人,意见,事务）

bool flag=IWorkFlow.Engine.EngineHost.Instance.CommonSend(string caseid, string baid, string senderid, string receAct, List<string> receivers, string remark, object args)//

(10)工作流办结（业务号,流转号,提交人,意见,事务）

bool flag=IWorkFlow.Engine.EngineHost.Instance.End(string caseID, string baID, string userid, string remark, object args)

(11)工作流多重定义发送（业务号,流转号,提交人,接受节点,接受人,意见,事务）但是如果当前节点是结束节点则会自动办结

bool flag=IWorkFlow.Engine.EngineHost.Instance.Send(string caseID, string baID, string senderid, List<string> receivers, string receAct, string remark, object args)

(12)工作流创建业务(流程号,创建人,业务名,流程时限)

IWorkFlow.Engine.BusinessCase bc=IWorkFlow.Engine.EngineHost.Instance.Create(string flowID, string creater, string name, int limit)

(13)工作流设置流程时限

bool flag=IWorkFlow.Engine.EngineHost.Instance.SetBusinessCaseLimit(string caseID, int limitday, object args)

(14)工作流设置当前步的本人意见

bool flag=IWorkFlow.Engine.EngineHost.Instance.SetBusinessActRemark(string caseID, string senderid, string remark, object args)

(15)工作流删除某条业务(带事务)

bool flag=IWorkFlow.Engine.EngineHost.Instance.Delete(string caseID, string creater, object args)

(16)工作流获取当前节点ID的节点实体对象(流程ID,节点ID)

ActivityModel ent= GetCurModelByAct(string flowid, string actid)

(17)工作流停办一条业务(流程ID,停办人,停办意见,事务)

IWorkFlow.Engine.EngineHost.Instance.StopOneCase（string caseid, string senderid, string remark, Object args）

(18)工作流停办恢复一条业务(流程ID,启动人,启动意见,事务)

IWorkFlow.Engine.EngineHost.Instance.StopWakeOneCase（string caseid, string senderid, string remark, Object args）

## 工作流表单



### 页面结构

<div class="**col-lg-8**">

<!--意见填写区域，预备加入工作流控件-->

<div class="row"><div class="col-md-12" data-id="WF\_RemarkBox"></div></div>

<!-- 表单 -->

<div data-id="formContainer" **class="pageContainer">**

<div **class="bg-info row"** style="padding: 0 15px;">

<h3> 信访登记表</h3>

<div class="btn-group" **style="float: right; margin-top:-40px;">**

<button /> <button data-id="btnQD" value="启动监察" class="btn btn-default" />

</div>

</div>

<div **class="page-header sub-header">** 信访基本内容 </div>

<div **class="row form-horizontal form-bordered"** data-id="baseInfo">

<div **class="col-md-9">**

<div class="form-group">

<label class="**col-sm-2 control-label**">单位名称：</label>

<div class="**col-sm-4**">

<div class="input-group">

<input data-id="unitname" type="text" class="form-control" data-bind="value:UnitName" placeholder="请选择..." />

<span class="input-group-btn"><button class="btn btn-default" type="button" data-bind="click: \_unitSel">...</button> </span>

</div>

</div>

<label class="col-sm-2 control-label">单位地址：</label>

<div class="col-sm-4">

<input data-bind="value:UnitAdd" type="text" class="form-control" placeholder="" />

</div>

</div>

</div>

<div **class="col-md-3">**

<label class="control-label">信访类型：</label> <div class="form-group"> <input type="text" /> </div>

</div>

</div>

<div class="page-header sub-header">

信访来源

<span **style="float: right; margin-top:-10px;">**

<button data-id="addSource" class="btn btn-default"><i class="fa fa-plus"></i> 添加</button>

</span>

</div>

<div class="row">

<div class="col-md-12" data-id="sourceList2">

<table **class="table table-striped table-bordered table-condensed"**></table>

</div>

</div>

<!-- 审批意见 -->

<div class="**col-lg-4 visible-lg**" data-id="WF\_RemarkList">

</div>

</div>

</div>

### 代码结构

放在Forms目录下,html为模板文件,建议与js同名

定义

$.Biz.B\_Complaint = function (){

//下面是表单初始化所需的参数

this.options ={

HtmlPath: "Forms/B\_Complaint/B\_Complaint.html", //当前表单模板

Url: "B\_ComplaintSvc.data", //表单对应服务地址

getCaseName: function () { return "关于“" + root.find("[data-id='unitname']").val() + "”的信访"; },

afterSave: function (){}, //点击保存后的回调

}

//页面呈现：

//参数： 页面的根div,工作流工具栏,页面初始化数据

this.show = function (formdiv, formdata,toolbj) {

}

}

//加入说明版本

$.Biz.B\_Complaint.prototype.version = "1.0";

**在forms目录下,修改models.js文件,加入路径引用：**

"$.Biz.B\_Complaint": "Forms/B\_Complaint/B\_Complaint.js“

### 草稿及离线支持

通过实现this.getCacheData = function ()

系统可以实现定时草稿保存和离线保存，保存草稿系统会跳过数据校验，默认每分钟会自动保存一次草稿，如果网络中断，会保存在本机。

# GIS开发

## 工作空间

## 图层

## 工具栏

## 侧边栏窗口

## 图层控制

# 部署

## 部署环境

要求windows server 2008或以上版本，.net framework4.5或以上，iis

## 配置文件

配置文件除了web.config之外，全部放在config目录下：

config.js：主要是界面配置，直接加载在客户端页面；

DataDictConfig.json：客户端数据缓存

DbGroup.xml、DbTables.xml： 数据库中英文对照表

wfForms.config：工作流的表单设置

其他为界面和门户配置！

UIHtml：界面模板

WFModel：工作流设置

# 常见问题

## 登录异常

## URL格式说明