**BaseMiniExt脚本说明**

**replaceUrl**

replaceUrl用于替换url或action中的{}内容，被默认应用于openWindow和execute方法中，其原型为：function replaceUrl(url, windowSetting)

替换数据取得顺序为settings.currentRow、地址栏、settings.paramFrom的选中行、settings.gridId的选中行、settings.treeId的当前节点。

注意事项：grid控件的行按钮会使grid的当前行选中或取消选中，行按钮自己写的openWindow等方法做替换时请注意。

**openWindow**

openWindow封装了mini.open方法，支持mini.open全部参数。其原型为：function openWindow(url, windowSettings)，封装功能如下：

* URL替换
* URL附加地址栏参数
* 窗口关闭刷新Grid控件
* 用户操作验证
* 高级应用见示例

URL替换:编辑Grid控件选中行

openWindow("Edit?ID={ID}",{mustSelectOneRow:true});

openWindow("Edit?ID={ID}",{mustSelectOneRow:true,gridId:'dataGrid'});

openWindow("Edit?ID={ID}",{mustSelectOneRow:true,paramFrom:'dataGrid'});

URL附加地址栏参数:新窗口URL附加当前窗口地址栏参数,addQueryString默认值为true

openWindow("Edit?ID={ID}",{addQueryString:true});

窗口关闭刷新Grid控件:

openWindow("Edit?ID={ID}",{refresh:false,gridId:'dataGrid'}); //窗口关闭后刷新grid，refresh默认值为true,gridId默认为dataGrid

用户操作验证:必须选择一行

openWindow("Edit?ID={ID}",{mustSelectOneRow:true,gridId:'dataGrid'});

类似参数还有如下：

var operateParamSettings = {

mustConfirm: false, //执行前是否需要用户确认

mustSelectRow: false, //必须选择行

mustSelectRowMsg: "当前没有选择要操作的记录，请重新确认！", //必须选择的提示信息

mustSelectOneRow: false, //必须选择一行

mustSelectOneRowMsg: "需要选择一条操作记录，请重新确认！", //必须选择一行的提示信息

mustSelectNode: false, //必须选择节点

mustSelectNodeMsg: "当前没有选择要操作的节点，请重新确认！", //必须选择的提示信息

mustSelectLeafNode: false, //必须选择叶子节点

mustSelectLeafNodeMsg: "当前没有选择叶子节点，请重新确认！" //必须选择叶子节点的提示信息

};

高级应用一：页面加载完成后，将页面树的选中节点为003.

openWindow("xxx",{data:"{CurrentSelectNodeID:'003'}"}); // xxx页面上需要有方法setData接收参数data

高级应用二：页面加载完成后，ajax调用GetSelectedNode后台方法，并将其返回值通过页面的setData设置

openWindow("xxx",{getDataAction:"GetSelectedNode"}); // xxx页面上需要有方法setData接收参数data

高级应用三：页面关闭后，调用后台DealData方法处理返回值

openWindow("xxx",{action:'DealData'}); //DealData方法会接受到参数RelationData

**execute**

execute封装了$.ajax方法,其原型为：function execute(action, execSettings) ，封装功能如下：

* Action替换
* Action附加地址栏参数
* 表单输入验证
* 用户操作验证
* 执行成功提示
* 执行前用户确认
* 增加执行参数
* 执行成功刷新重新加载页面Grid
* 执行成功关闭窗口
* 执行成功调用onComplete

Action替换:调用Delete方法删除选中行

execute("Delete?IDs={ID}",{mustSelectRow:true });

execute("Delete?IDs={ID}",{mustSelectRow:true, gridId:'dataGrid' });

execute("Delete?IDs={ID}",{mustSelectRow:true, paramFrom:'dataGrid' });

Action附加地址栏参数:

每次execute自动附件

表单输入验证:暂存时不做验证

execute("Save",{validateForm:false });

用户操作验证:必须选择要删除的行

execute("Delete?IDs={ID}",{mustSelectRow:true,gridId:'dataGrid'});

类似参数还有如下：

var operateParamSettings = {

mustConfirm: false, //执行前是否需要用户确认

mustSelectRow: false, //必须选择行

mustSelectRowMsg: "当前没有选择要操作的记录，请重新确认！", //必须选择的提示信息

mustSelectOneRow: false, //必须选择一行

mustSelectOneRowMsg: "需要选择一条操作记录，请重新确认！", //必须选择一行的提示信息

mustSelectNode: false, //必须选择节点

mustSelectNodeMsg: "当前没有选择要操作的节点，请重新确认！", //必须选择的提示信息

mustSelectLeafNode: false, //必须选择叶子节点

mustSelectLeafNodeMsg: "当前没有选择叶子节点，请重新确认！" //必须选择叶子节点的提示信息

};

执行成功提示:保存成功后提示：“保存成功”

execute("Save",{actionTitle:'保存'});

执行前用户确认:删除前提示：“确认要删除吗？”

execute("Delete",{actionTitle:'删除',mustConfirm:true});

增加执行参数:

addExecuteParam("ID","0");

执行成功刷新重新加载页面Grid:

execute("Delete",{refresh:true});

执行成功关闭窗口:

execute("Delete",{closeWindow:true});

执行成功调用onComplete:

execute("Save",{onComplete:function(text){ alert("保存成功！"); }});

**closeWindow**

closeWindow用于关闭openWindow打开的窗口,其原型为：function closeWindow(data) ，封装功能如下：

* 关闭窗口并返回值

关闭窗口并返回值:

closeWindow({returnValue:'ok'}); //{returnValue:'ok'}由openWindow的onDestroy方法接收

**showWindow**

showWindowm能将页面的mini-window控件展示为窗口,其原型为：function showWindow(windowId) ，封装功能如下：

* 展示mini-window控件为窗口

展示mini-window控件为窗口:

showWindow('queryWindow'); //页面上应该有个id为queryWindow的mini-window控件

**hideWindow**

hideWindow关闭showWindow打开的窗口,其原型为：function hideWindow(windowId) ，封装功能如下：

* 关闭showWindow打开的窗口

关闭showWindow打开的窗口:

hideWindow('queryWindow'); //页面上应该有个id为queryWindow的mini-window控件

**getQueryString**

getQueryString用于获取地址栏参数,其原型为：function getQueryString(key,url) ，封装功能如下：

* 获取地址参数

获取地址参数:

var code = getQueryString('Code'); //默认获取当前页面url的地址栏参数

**hasQueryString**

hasQueryString判断是否存在地址栏参数,其原型为：function hasQueryString(key) ，封装功能如下：

* 判断地址栏参数是否存在

判断地址栏参数是否存在:

var code = hasQueryString('Code');

**getValues**

getValues获取json数组属性值,其原型为：function getValues(rows, attr) ，封装功能如下：

* 获取json数组属性值

获取json数组属性值:

var IDs = getValues([{ID:'1',Name:'张三'},{ID:'2',Name:'李四'}],"ID"); //IDs的值是："1,2"

**addGridEnum**

addGridEnum用于给mini-grid添加一个枚举列,其原型为：function addGridEnum(gridId, fieldName, enumKey) ，封装功能如下：

* 给Grid添加一个枚举列

给Grid添加一个枚举列

addGridEnum("dataGrid", "sex", "System.Sex"); //给dataGrid的sex列添加枚举System.Sex

**addGridLink**

addGridLink用于给mini-grid添加一个超链接列,其原型为：function addGridLink(gridId, fieldName, url, gridLinkSettings) ，封装功能如下：

* 添加一个超链接列
* 超链接列显示为特定字符串
* 支持全部openWindow参数
* onFilter过滤

给Grid添加一个超链接列:

addGridLink("dataGrid", "userName", "Edit?ID={ID}"); //使用当前行ID替换{ID}

超链接列显示为特定字符串:

addGridLink("dataGrid", "userName", "Edit?ID={ID}",{linkText:'修改'}); //userName列显示为“修改”

支持全部openWindow参数:

addGridLink("dataGrid", "Dept", urlConstant.singleEnum + "System.Dept", { action: 'SetDept', linkText: '设置当前部门', refresh: true, height: 620 });

onFilter过滤:

addGridLink("dataGrid", "Dept", urlConstant.singleEnum + "System.Dept", { onFilter: function (e) { if (e.row["ErrorCount"] >= 3) return true; else return false; } });

});

**addGridButton**

addGridButton用于给mini-grid添加一个按钮列,其原型为：function addGridButton(gridId, fieldName, gridButtonSettings) ，封装功能如下：

* 添加一个按钮列
* onButtonClick
* onFilter

添加一个按钮列:

addGridButton("dataGrid", "RetireUser", { action: "RetireUser?ID={ID}", actionTitle: "离退" });

onButtonClick:

addGridButton("dataGrid", "RetireUser", { onButtonClick:function(row){alert(row["ID"]); }, actionTitle: "离退" });

onFilter过滤:

addGridButton("dataGrid", "Unlock", { action: "Unlock", actionTitle: "解锁"

, onFilter: function (e) { if (e.row["ErrorCount"] >= 3) return true; else return false; }

});

**addSelector**

addSelector用于给控件添加一个选择器,其原型为：function addSelector(name, url, selectorSettings) ，封装功能如下：

* 添加一个选择器
* 多个返回值
* 选择结果加入Grid

添加一个选择器:

addSelector("dept", urlConstant.singleOrg, { width: 320, height: 580, allowResize: false }); //dept是个控件,可以为按钮或者buttonEdit

多个返回值

addSelector("dept", urlConstant.singleOrg, { returnParams:"value:ID,text:Name,deptCode:Code" }); //将返回值的ID赋值给dept控件的value，Name赋值给dept控件的text，Code赋值给deptCode控件

选择结果加入Grid:

addSelector("dept", urlConstant.singleOrg, { targetType:"grid",existValidateFields:"ID",gridId:"dataGrid" }); //选择结果加入dataGrid,通过ID字段判断重复行

**setFormDisabled**

setFormDisabled设置表单不可用,其原型为：function setFormDisabled(normalSettings) ，封装功能如下：

* 设置表单不可用

设置表单不可用：

setFormDisabled();

**setFormItemDisabled**

setFormItemDisabled设置表单控件不可用,其原型为：function setFormItemDisabled(name) ，封装功能如下：

* 设置表单控件不可用

设置表单控件不可用：

setFormItemDisabled("name");

**msgUI**

msgUI弹出提示框,其原型为：function msgUI(content, msgType, callbackFunc) ，封装功能如下：

* 弹出消息框
* 弹出警告框
* 弹出询问框
* 弹出三选项询问框
* 弹出错误信息框
* 弹出输入框

弹出消息框：

msgUI("保存成功！");

msgUI("保存成功！", 0); //msgType为0表示消息框，默认值

msgUI("保存成功！", 0, function(action){ if(action="ok") alert("你点的OK按钮"); });

弹出警告框:

msgUI("请小心操作！", 1);

弹出询问框:

msgUI("确定要删除吗？", 2);

弹出三选项询问框:

msgUI("文件尚未保存，是否保存文件？", 3);

弹出错误信息框:

msgUI("程序错误", 4);

弹出输入框:

msgUI("请输入要购买的数量！", 5);

**MiniExt脚本说明**

**mini.parse()**

禁止在页面中调用mini.parse方法，$(document).ready中进行mini控件的解析。页面上可以使用JQuery

**allowUrlOpenForm**

是否允许从地址栏直接打开Form页面开关

**onFormSetData**

onFormSetData：如果需要在表单设置数据时做一些操作，请在表单上定义此方法。原型为 function onFormSetData(data)

**FuncType=View**

地址栏参数跟FuncType=View的作用是：调用setFormDisabled(); 和隐藏.mini-toolbar

**clearDataForm**

清空表单数据

clearDataForm();

clearDataForm({formId:'dataForm'});

**clearQueryForm**

清空查询表单数据

clearQueryForm();

clearQueryForm({formId:'queryForm'});

**search**

查询

search();

search({formId:'queryForm',gridId:'dataGrid'});

**quickSearch**

快速查询

quickSearch();

quickSearch({queryBoxId:'key',gridId:'dataGrid'});

**add**

add封装了openWindow方法，支持openWindow全部参数。其原型为：function add(windowSettings)，封装功能如下：

* 默认参数
* createId

默认参数：

add();

openWindow('Edit',{title:'增加'});

createId：url直接跟新ID参数，用于解决页面刷新丢数据问题

add({createId:true});

**edit**

edit封装了openWindow方法，支持openWindow全部参数。其原型为：function edit(windowSettings)，封装功能如下：

默认参数：

edit();

openWindow('Edit',{title:'编辑'});

**view**

view封装了openWindow方法，支持openWindow全部参数。其原型为：function view(windowSettings)，封装功能如下：

默认参数：

view();

openWindow('Edit',{title:'查看',funcType:'view'});

**del**

del封装了execute方法，支持execute全部参数。其原型为：function del(execSettings)，封装功能如下：

默认参数：

del();

execute('Delete',{actionTitle:'删除',mustSelectRow:true,mustConfirm:true,validateForm:false});

**save**

save封装了execute方法，支持execute全部参数。其原型为：function save(execSettings)，封装功能如下：

默认参数：

save();

execute('Save',{actionTitle:'保存',closeWindow:true});

**saveList**

saveList封装了execute方法，支持execute全部参数。其原型为：function saveList(execSettings)，封装功能如下：

默认参数：

saveList();

execute('SaveList',{actionTitle:'保存'});

**addRow**

addRow为grid控件增加一行。其原型为：function addRow(newRowJson, settings)，封装功能如下：

* 默认参数
* createId

默认参数：

addRow();

createId：添加新行直接产生ID

addRow({createId:true});

**delRow**

delRow用于删除grid的一行。其原型为：function delRow(execSettings)，封装功能如下：

默认参数：

delRow();

delRow({gridId:'dataGrid'});

**searchTree**

searchTree用于查询tree。其原型为：function searchTree(settings)，封装功能如下：

默认参数：

searchTree();

searchTree({treeId:'dataTree',queryTreeBoxId:'treeKey'});

**saveNode**

saveNode封装了execute方法，支持execute全部参数。其原型为：function saveNode(execSettings)，封装功能如下：

默认参数：

saveNode();

execute('SaveNode',{actionTitle:'保存',onComplete:returnForm}); //执行完毕后调用returnForm方法

**nodeViewing**

nodeViewing封装了openWindow方法，支持openWindow全部参数。其原型为：function nodeView(winSettings)，封装功能如下：

默认参数：

nodeViewing();

openWindow('NodeEdit',{title:'查看',paramFrom:'dataTree',mustSelectNode:true,funcType:view});

**nodeEditing**

nodeEditing封装了openWindow方法，支持openWindow全部参数。其原型为：function nodeEditing(winSettings)，封装功能如下：

默认参数：

nodeEditing();

openWindow('NodeEdit',{title:'编辑',paramFrom:'dataTree',mustSelectNode:true,onDestroy:nodeEdited}); //窗口关闭调用nodeEdited

nodeEdited：将返回值设置到树节点

**nodeAdding**

nodeAdding封装了openWindow方法，支持openWindow全部参数。其原型为：function nodeAdding(winSettings)，封装功能如下：

默认参数：

nodeAdding();

openWindow('NodeEdit',{title:'增加',paramFrom:'dataTree',mustSelectNode:true,onDestroy:nodeAdded}); //窗口关闭调用nodeAdded

nodeAdded：将返回值设置到树节点

**nodeDeleting**

nodeDeleting封装了execute方法，支持execute全部参数。其原型为：function nodeDeleting(execSettings)，封装功能如下：

默认参数：

nodeDeleting();

execute('DeleteNode',{title:'删除',paramFrom:'dataTree',mustSelectNode:true,onComplete:nodeDeleted}); //执行完毕调用nodeAdded

nodeDeleted：将返回值设置到树节点

**onNodeDroping**

onNodeDroping设置到mini-tree控件的ongivefeedback事件，用于控制不能拖放到非同级节点的前后，源码如下：

源码：

function onNodeDroping(e) {

//不能拖放到非同级节点的前后

if ((e.effect == "before" || e.effect == "after") && e.targetNode.ParentID != e.node.ParentID)

e.effect = "no";

}

**onNodeDrop**

onNodeDrop设置到mini-tree控件的onbeforedrop事件，用于节点的拖拽，源码如下：

源码：

function onNodeDrop(e) {

addExecuteParam("dragNode", mini.encode(e.dragNode));

addExecuteParam("dropNode", mini.encode(e.dropNode));

addExecuteParam("dragAction", e.dragAction);

execute("DropNode");

}

**returnForm**

returnForm的作用是关闭窗口并返回form数据

源码：

function returnForm(windowSettings) {

windowSettings = $.extend({}, windowParamSettings, windowSettings);

var form = new mini.Form(windowSettings.formId);

closeWindow(form.getData());

}

**returnList**

returnList的作用是关闭窗口并返回grid数据

源码：

function returnList(windowSettings) {

windowSettings = $.extend({}, windowParamSettings, windowSettings);

var grid = mini.get(windowSettings.gridId);

closeWindow(grid.getSelecteds());

}

**addSingleUserSelector**

addSingleUserSelector封装了addSelector方法，支持addSelector全部参数。其原型为：function addSingleUserSelector(name,selectorSettings)，封装功能如下：

默认参数：

addSingleUserSelector('userName');

addSingleUserSelector('userName',{OrgIDs:'00001'});

类似的方法还有：

addMultiUserSelector

addSingleOrgSelector

addMultiOrgSelector

addSingleRoleSelector

addMultiRoleSelector

**flowLoadMenubar**

flowLoadMenubar根据表单数据加载流程按钮条。其原型为：function flowLoadMenubar(jsonFormData)

**onFormSubmitting**

onFormSubmitting，如果表单上存在，则在表单提交前调用。其原型为：function onFormSubmitting(flowRouting)

**onFormSubmit**

onFormSubmit，如果表单上存在，则在表单提交后调用。其原型为：function onFormSubmit(flowRouting)

**onFormSaving**

onFormSaving，如果表单上存在，则在表单提保存前。其原型为：function onFormSaving()

**onFormSaved**

onFormSaved，如果表单上存在，则在表单提保存后。其原型为：function onFormSaved()

**flowTrace**

flowTrace，打开流程跟踪页面，其源码为：

function flowTrace(winSettings) {

var settings = $.extend({ title: '流程跟踪', width: 1000 }, winSettings);

openWindow('/MvcConfig/Workflow/Trace/Diagram?ID={ID}&FuncType=FlowTrace', settings);

}

**flowAdd**

flowAdd，打开表单启动流程，其原型为：function flowAdd(flowCode, windowSettings)

示例：

flowAdd("test");

flowAdd("test",{url:'Edit'});

**flowEdit**

flowEdit，打开表单启动流程，其原型为：function flowEdit(flowCode, windowSettings)

示例：

flowEdit("test");

flowEdit("test",{url:'Edit'});

**grid展示图表**

见示例

示例：

@Html.GridChartSettings()