# Kendo UI

## Grid

<http://blog.csdn.net/chelen_jak/article/list/20>

里面还有一些grid的操作，可学习

### 设置界面表格高度

在界面中装载表格的地方，使用 style 指定高度。

<**div style="height**: 300**px**;**"**>  
 <**div id="grid"**></**div**>  
</**div**>

### 获取表格中新增和修改数据

修改或新增的数据的dirty属性为true，然后通过数据的isNew方法，可以判断数据是不是新增的。

**var** submitdata=[];  
**var** grid=*$*(**"#grid"**).data(**"kendoGrid"**);  
**var** ds = grid.***dataSource***;  
*$*.each(ds.data(), **function** (idx, data) {  
 **if** (data.dirty === **true**) {  
 **var** d = data.toJSON();  
 d[**'\_\_status'**] = data.isNew() ? **'add'** : **'update'**;  
 d[**'\_\_id'**] = data.**uid**;  
 submitdata.push(d);  
 }  
});

### saveChanges

grid的saveChanges方法会自动获取表格新增和修改的数据，然后根据type自动调用新增或修改方法。

注意：如果想要在保存之后刷新表格，最好不要使用saveChanges方法。因为保存操作没有完成时，就已经发出了查询请求，此时，新增的数据将查询不到。如果非要使用，可以在一定延迟后执行刷新操作。

**function** *toSave*() {  
 *$*(**'#grid'**).data(**'kendoGrid'**).saveChanges();  
 setTimeout(*refresh*, 500);  
}  
**function** *refresh*() {  
 **var** pageNo = *$*(**'#grid'**).data(**'kendoGrid'**).dataSource.**\_page**; *$*(**'#grid'**).data(**'kendoGrid'**).dataSource.read(pageNo);  
}

### 表格中刷新当前页

//刷新Grid表格  
function refushGrid() {  
//获取当前页面编号  
var pageNo = $('#grid').data('kendoGrid').dataSource.\_page;  
//刷新页面  
$('#grid').data('kendoGrid').dataSource.read(pageNo);  
}

### 设置表格的某列不可编辑

1. 在数据源读取数据时通过 editable 属性指定：

var crudServiceBaseUrl = '${base.contextPath}',  
dataSource = new kendo.data.DataSource({  
transport : {  
 read : {  
 url: crudServiceBaseUrl + "/shop/query",  
 type: "POST",  
 //contentType: "application/json",  
 dataType: "json"  
  
},  
create : {  
 url: crudServiceBaseUrl + "/shop/submit",  
 contentType: "application/json",  
 type : "POST"  
},  
update: {  
 url: crudServiceBaseUrl + "/shop/submit",  
 contentType: "application/json",  
 type: "POST"  
},  
destroy: {  
 url: crudServiceBaseUrl + "/shop/remove",  
 contentType: "application/json",  
 type: "POST"  
},  
parameterMap: function (options, type) {  
 if (type !== "read" && options.models) {  
 var datas = Hap.prepareSubmitParameter(options, type);  
 console.info("datas",datas);  
return kendo.stringify(datas);  
} else if (type === "read") {  
return Hap.prepareQueryParameter(viewModel.model.toJSON(), options);  
}  
}  
},  
batch : true,  
serverPaging: true,  
serverSorting: true,  
pageSize : 10,  
schema : {  
data : 'rows',  
total : 'total',  
model : {  
id : "shopId",  
fields: {  
shopId : {type: "long" },  
source : {type: "string",validation:{required:true},editable:false},  
shopName : {type: "string",validation:{required:true}},  
shopCode : {type: "string",validation:{required:true}},  
defaultDeliveryMode : {type: "string",validation:{required:true}},  
availableAutomaticSync : {type: "string"}  
}  
}  
}  
});

1. 在表格列中通过 editor 配置设置：

{  
field: "warehouseCode",  
title: '仓库代码',  
attributes: {style: "text-align:left"},  
headerAttributes: {  
"class": "table-header-cell",  
style : "text-align: center"  
},  
width: 50,  
editor: function(container, options){  
container.html(options.model.warehouseCode);  
container.removeClass('k-edit-cell');  
}  
  
},

1. 设置多列不可编辑

dataSource = new kendo.data.DataSource({  
 transport: {  
 read: {  
 url: BaseUrl + "/mst/base/items/query",  
 type: "POST",  
 dataType: "json"  
 },  
 ......  
 parameterMap: function (options, type) {  
 if (type !== "read" && options.models) {  
 var datas = Hap.prepareSubmitParameter(options, type)  
 return kendo.stringify(datas);  
 } else if (type === "read") {  
 return Hap.prepareQueryParameter(viewModel.model.toJSON(), options)  
 }  
 }  
 },  
 batch: true,  
 serverPaging: true,  
 pageSize: 20,  
 schema: {  
 data: 'rows',  
 total: 'total',  
 model: {  
 id: "baseItemId",  
 fields: {  
 price : {  
 validation:{  
 checkPrice :function(input){  
 if(input.is("[name='price']") ){  
 if(input.val() < minPrice){  
 input.attr("data-checkPrice-msg", "<@spring.message 'homs.pricerule'/>");  
 return false;  
 }  
 return true;  
 }  
 }  
 }  
 }  
 },  
 editable : function(col){  
 if(col =='skuId'||col == 'itemName'||col =='itemBarcode' ||col =='primaryUom'  
 || col == 'itemType'){  
 return false;  
 }  
 return true;  
 }  
 }  
 }  
});

### 设置表格列必须

1. 通过数据源读取数据时设置：
2. schema : {  
   data : 'rows',  
   total : 'total',  
   model : {  
   id : "shopId",  
   fields: {  
   shopId : {type: "long" },  
   source : {type: "string",validation:{required:true},editable:false},  
   shopName : {type: "string",validation:{required:true}},  
   shopCode : {type: "string",validation:{required:true}},  
   defaultDeliveryMode : {type: "string",validation:{required:true}},  
   availableAutomaticSync : {type: "string"}  
   }  
   }  
   }

### 设置第一行为行序号

1. 设置行序号由1到数据源总体数量。第一页从1开始，第二页从pagesize+1开始。

columns: [{  
 **field** : **""**,  
 **title** : **'序号'**,  
 **attributes** : {  
 **style** : **"text-align:center"** },  
 **width** : 50,  
 **headerAttributes** : {  
 **"class"** : **"table-header-cell"**,  
 **style** : **"text-align: center"** },  
 **template** : **"<span class='row-number'></span>"** },

……  
 dataBound : **function**() {  
 **var** rows = **this**.items();  
 **var** page = **this**.pager.page() - 1;  
 **var** pagesize = **this**.pager.pageSize();  
 *$*(rows).each(  
 **function**() {  
 **var** index = *$*(**this**).index() + 1 + page \* pagesize;  
 **var** rowLabel = *$*(**this**).find(**".row-number"**);  
 *$*(rowLabel).html(index);  
 });  
 }  
}).data(**"kendoGrid"**);

1. 设置表格第一列为行序号

*//将rownumber中的#改成序号  
$*(**function**(){  
 **var** rownumList = *$*(**'[data-index="0"]'**);  
 **for** (**var** i=0;i<rownumList.**length**;i++)  
 {  
 **var** rownumText = *$*(rownumList[i]).html();  
 **if**(**"#"**==rownumText){  
 *$*(rownumList[i]).html(**"序号"**);  
*$*(rownumList[i]).eq(0).parent().parent().prev().children().eq(0).css(**"width"**,**"50px"**);  
*$*(rownumList[i]).eq(0).parent().parent().parent().parent().parent().next().children(**"table"**).children(**"colgroup"**).children().eq(0).css(**"width"**,**"50px"**);  
 }  
 }  
});

### 获取表格中鼠标焦点所在行的数据

$("#grid").on("click", "tr.k-state-current", function (e) {  
var data = $("#grid").data("kendoGrid").dataItem($(e.target).closest("tr")); //获取选中行对象  
});

### 获取表格中选中的数据

第一种方法

previewFunction:function(e){  
var grid = $('#grid').data('kendoGrid');  
//获取选中项  
var grid\_data = grid.selectedDataItems();  
  
data\_selected = new kendo.data.DataSource({  
data: grid\_data  
}) ;  
grid.setDataSource(data\_selected);  
},

第二种方法

previewFunction:function(e){  
var grid = $('#grid').data('kendoGrid');  
//获取选中项  
  
var selectedRows = grid.select();  
var selectedDataItems = [];  
for (var i = 0; i < selectedRows.length; i++) {  
var dataItem = grid.dataItem(selectedRows[i]);  
selectedDataItems.push(dataItem);  
}  
  
data\_selected = new kendo.data.DataSource({  
data: selectedDataItems  
}) ;  
grid.setDataSource(data\_selected);  
},

### 设置表格第一列为行序号

表格中有一个rownumber属性和一个selectable属性。其中，rownumber属性用于控制显示表格前的序号（此时表格序号对应的 field 显示的是“#”号，上面的函数就是将对应的 field 变成“序号”的），如果不想显示选择框，可以将表格的 selectable属性去掉。



*<!--将rownumber中的#改成序号 -->*$(function () {  
var rownumList = $('[data-index="0"]');  
for (var i = 0; i < rownumList.length; i++) {  
var rownumText = $(rownumList[i]).html();  
if ("#" == rownumText) {  
$(rownumList[i]).html("序号");  
$(rownumList[i]).eq(0).parent().parent().prev().children().eq(0).css("width", "50px");  
$(rownumList[i]).eq(0).parent().parent().parent().parent().parent().next().children("table").children("colgroup").children().eq(0).css("width", "50px");  
}  
}  
$("#pricerowGrid").kendoGrid({  
dataSource: priceRowDataSource,  
height: '300px',  
resizable: true,  
scrollable: true,  
navigatable: false,  
rownumber: true,  
selectable: 'multiple, rowbox',  
pageable: {  
pageSizes: [5, 10, 20, 50],  
refresh: true,  
buttonCount: 5  
},  
...  
});

### 双击表格中的某一行，跳转到对应界面

方法：$(selector).on(events,[selector],[data],fn)

//双击弹框，pricerowGrid 是一个选择  
pricerowGrid.on('dblclick', '.k-grid-content tr', function (event) {  
var data = $('#pricerowGrid').data("kendoGrid").dataItem($(event.target).closest("tr"));  
window.top.openTab(data.pricerowId, "价格行" + data.pricerowId, "${base.contextPath}/mst/productDetail/PriceRowDetail.html?pricerowId=" + data.pricerowId + "&productId=" + data.productId + "&code=" + data.code);  
});

也可以调用一个弹窗函数，传递参数过去

grid.on('dblclick', '.k-grid-content tr', function (event) {  
// console.log("点击",djiango);  
var data = $('#Grid').data("kendoGrid").dataItem($(event.target).closest("tr"));  
var id = data.id;  
var code = data.code;  
entryFunction(id, code); //跳转函数  
});  
  
//跳到详情界面  
function entryFunction(id, code){  
window.top.openTab(code,"短信模板","${base.contextPath}/om/om\_mailtemplate\_detail.html?id="+id);  
}

### 表格中行上移下移

实体类里设置一个字段，用这个字段排序。

{  
  
command: [  
{  
name: 'ding',  
template: '<**a class="k-button k-button-icontext k-grid-ding" title="上移"**><**span class="fa fa-flash"**></**span**></**a**>',  
click: function (e) {  
e.preventDefault();  
var grid=$('#grid').data("kendoGrid");  
var dataItem = grid.dataItem($(event.target).closest("tr"));  
  
var dataItems=grid.dataSource.data();  
for(var i in dataItems)  
{  
if(i!=0) {  
if (dataItems[i].id == dataItem.id) {  
var dataChange=dataItems[i-1];  
var priorityChange=dataItem.priority;  
dataItem.priority=dataChange.priority;  
dataChange.priority=priorityChange;  
var map=[];  
map.push(dataItem);  
map.push(dataChange);  
$.ajax({  
url: BaseUrl + "submit",  
type : "POST" ,  
contentType: "application/json",  
data:JSON.stringify(map),  
success:function (data) {  
if(data.success)  
{  
grid.dataSource.read();  
}else {  
kendo.ui.showErrorDialog({  
title:"提示",  
message:data.msg  
})  
}  
}  
})  
}  
}  
}  
  
}  
}  
],  
// locked: true,  
title: "上移",  
width: 120,  
headerAttributes: {  
style: "text-align:center"  
},  
attributes: {  
style: "text-align:center"  
}  
}

### 表格中声明组件

{  
field: "payMode",  
title: '退款方式',  
attributes: {style: "text-align:center"},  
headerAttributes: {  
"class": "table-header-cell",  
style: "text-align: center"  
},  
width: 100,  
template: function (dataItem) {  
var v = dataItem.payMode;  
$.each(PayTypeData, function (i, n) {  
if(n.value == v){  
v = n.meaning;  
return v;  
}  
})  
return v;  
},  
editor: function (container, options) {  
$('<**input name="' + options.field + '"**/>')  
.appendTo(container)  
.kendoDropDownList({  
dataTextField: "meaning",  
dataValueField: "value",  
dataSource: PayTypeData  
});  
}  
},

### 表格中添加编辑/删除列

{  
title: '<@spring.message "hap.delete"/>',  
width: 50,  
//如果该行id不为空，则添加一个编辑超链接  
template: function (rowdata) {  
if (!!rowdata.asRefundEntryId) {  
//点击链接后不会跳转，会调用 onClick 方法。  
return '<**a href="javascript:void(0);" onclick="***editFunctionResources*(\**''**+rowdata.**asRefundEntryId** +**'\')"**><@spring.message "hap.delete"/></**a**>'  
}  
return '';  
},  
attributes: {style: "text-align:center"},  
headerAttributes: { style: "text-align:center"}  
},

### 表格中设置选择一列的值，设置其他列的值随之改变

{  
field: "code",  
title: '<@spring.message "hmall.product.code"/>',  
width: 120,  
template:function(dataItem){  
if(dataItem.code !=null){  
return dataItem.code;  
}else{  
return '';  
}  
},  
editor : function(container, options){  
$('<**input required name="' + options.field + '"**/>').appendTo(container).kendoLov($.extend(${lovProvider.getLov(base.contextPath, base.locale, "LOV\_PRODUCT")},  
{  
select : function(e){  
options.model.set("productId",e.item.productId);  
options.model.set("code",e.item.code);  
options.model.set("productName",e.item.name);  
options.model.set("productType",e.item.productType);  
},  
query:function(e){  
  
}  
}))  
},

### 表格中声明时间组件

注意：要指定日期格式，否则表格中不显示

{  
field: "estimateConTime",  
title: '<@spring.message "hmall.OrderEntry.estimateConTime"/>',  
width: 180,  
format: "{0:yyyy-MM-dd HH:mm:ss}",  
editor: function (container, options) {  
$('<**input id="'+ options.field +'" name="' + options.field + '" data-value-primitive="true"**/>')  
.appendTo(container)  
.kendoDateTimePicker({  
  
}).data("kendoDateTimePicker");  
}  
}

在前端修改界面显示格式：

1. 在数据源dataSource中设置字段类型：

**schema**: {  
 **data**: **'rows'**,  
 **total**: **'total'**,  
 **model**: {  
 **id**: **"inventoryId"**,  
 **fields**: {  
 **startDate**:{**type**:**'date'**,**defaultValue**:**""**,**validation**:{**required**: **true**}},  
 **endDate**:{**type**:**'date'**,**defaultValue**:**""**},  
 }  
 }  
}

1. 编写一个JS函数，用于日期格式转换：

Date.**prototype**.Format = **function** (fmt) { *//author: meizz* **var** o = {  
 **"M+"**: **this**.getMonth() + 1, *//月份* **"d+"**: **this**.getDate(), *//日* **"h+"**: **this**.getHours(), *//小时* **"m+"**: **this**.getMinutes(), *//分* **"s+"**: **this**.getSeconds(), *//秒* **"q+"**: ***Math***.floor((**this**.getMonth() + 3) / 3), *//季度* **"S"**: **this**.getMilliseconds() *//毫秒* };  
 **if** (/(y+)/.test(fmt)) fmt = fmt.replace(RegExp.**$1**, (**this**.getFullYear() + **""**).substr(4 - RegExp.**$1**.**length**));  
 **for** (**var** k in o)  
 **if** (**new** RegExp(**"("** + k + **")"**).test(fmt)) fmt = fmt.replace(RegExp.**$1**, (RegExp.**$1**.**length** == 1) ? (o[k]) : ((**"00"** + o[k]).substr((**""** + o[k]).**length**)));  
 **return** fmt;  
}

1. 在表格中声明日期组件，并调用格式转换函数：

{  
 **field**: **"startDate"**,  
 **title**: **'<@spring.message "生效日期"/>'**,  
 **width**: 110,  
 template:**function** (dataItem) {  
 **if**(dataItem[**'startDate'**] != **null** && dataItem[**'startDate'**] !=**""**){  
 **return** dataItem[**'startDate'**].Format(**"yyyy-MM-dd"**);  
 }**else**{  
 **return ""**;  
 }  
 },  
 editor: **function** (container, options) {  
 **var** endDate = options.model.endDate;  
 **var** nowMaxDate = **new** *Date*(**"9999-12-29"**);  
 **if**(endDate != **null** && endDate != **""** && endDate!= **undefined**){  
 nowMaxDate = endDate;  
 }  
 *$*(**'<input required name="'** + options.field + **'"/>'**)  
 .**appendTo**(container)  
 .kendoDatePicker({  
 **max**:nowMaxDate,  
 **format**: **"{0:yyyy-MM-dd}"**,  
 change:**function** () {  
 **if**(**this**.value() == **""** || **this**.value() == **null**){  
 options.model[**"startDate"**]=**""**;  
 }  
 }  
 });  
 }  
},

### 设置表格列背景色



{  
 **field** : **"taxFlag"**,  
 **attributes** : {**style** : **"text-align:center;background-color:NavajoWhite;"**},  
 **title** : **'是否纳税'**,  
 **width** : **'90px'**,  
 **headerAttributes** : {**"class"** : **"table-header-cell"**,**style** : **"text-align: center"**},  
 template: **function** (dataItem) {  
 **var** v = dataItem.**taxFlag**;  
 *$*.each(sysYesOrNo, **function** (i, n) {  
 **if** ((n.value || **''**).toLowerCase() == (v || **''**).toLowerCase()) {  
 v = n.meaning;  
 **return** v||**""**;  
 }  
 })  
 **return** v||**""**;  
 },

### 根据字段条件设置列背景色

对账标识为Y时为红色，非Y时为黄色



{  
 **field**: **"escOrderCode"**,  
 **title**: **'顾客订单编号'**,  
 **width**: 200,  
 **attributes**: {  
 **style**: **"#:data.isBalance=='Y'?'background-color:Tomato;color:white':(data.isBalance=='Y'?'background-color:limegreen;color:white':(data.isBalance=='N'?'background-color:yellow':''))#"** }  
},

### 根据字段条件设置行背景色

1. 通过设置列为行中的每一列设置属性

**columns**: [  
 {  
 **field**: **"isBalance"**,  
 **title**: **'<@spring.message "对账标识"/>'**,  
 **headerAttributes**: {  
 **"class"**: **"table-header-cell"**,  
 **style**: **"text-align: center"** },  
 **attributes**: {**style**: **"text-align:center"**},  
 **width**: 80  
 },  
 {  
 **field**: **"escOrderCode"**,  
 **title**: **'顾客订单编号'**,  
 **width**: 200,  
 **attributes**: {  
 **style**: **"#:data.isBalance=='Y'?'background-color:Tomato;color:white':(data.isBalance=='Y'?'background-color:limegreen;color:white':(data.isBalance=='N'?'background-color:yellow':''))#"** }  
 },  
 {  
 **field**: **"infoCode"**,  
 **title**: **'订单编号'**,  
 **width**: 120,  
 **attributes**: {  
 **style**: **"#:data.isBalance=='Y'?'background-color:Tomato;color:white':(data.isBalance=='Y'?'background-color:limegreen;color:white':(data.isBalance=='N'?'background-color:yellow':''))#"** }  
 },

……

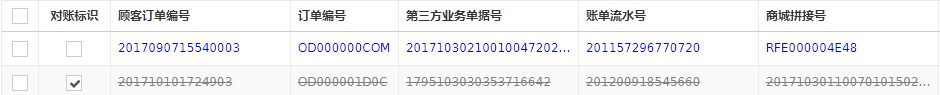
1. 通过动态根据条件设置表格行样式(分页时无法调用，注意分页)
2. 在dataBound中设置：

*$*(**"#Grid"**).kendoGrid({  
 **…**  
 **columns**: [

…

],  
 dataBound: **function** (e) {  
 **this**.dataItems().forEach(**function** (item, i) {  
 **var** isSuit = item.**isSuit**;  
 **var** parentLine = item.parentLine;  
 **if** (isSuit != **null** && isSuit == **"Y"** && parentLine == **null**)  
 e.sender.items().eq(i).css(**'background-color'**, **'#EEFFBB'**)  
 **else if**(parentLine != **null**)  
 e.sender.items().eq(i).css(**'background-color'**, **'#DCDCDC'**)  
 **else** e.sender.items().eq(i).css(**'background-color'**, **'#E8FFF5'**)  
 })  
 }  
}).data(**"kendoGrid"**);

### 动态根据条件设置表格行样式



下面这段代码可以根据某个字段动态修改表格行样式，但是这段代码必须在表格渲染之后执行。

可以在document加载完毕后执行，也可以通过一个按钮动态调用。

css可以为行动态添加样式，比如背景色，比如字体颜色等。

**var** grid = *$*(**"#grid"**).data(**"kendoGrid"**); *// 行样式***var** rows = grid.tbody.find(**"tr"**);  
rows.each(**function** (i, row) {  
 **var** data = grid.dataItem(row);  
 **var** bCancel = data.**isBalance** === **"Y"**;  
 **if** (bCancel) {  
 *$*(row).css(**"color"**, **"gray"**).css(**"text-decoration"**,**"line-through"**);  
 }  
 **else** {  
 *$*(row).css(**"color"**, **"blue"**);  
 }  
});

## Grid校验

### 相邻列之间数据校验

通过表格中设置的input元素可以在dataSource里面获取到input前面和后面一个元素。

通过input.context中的nextElementSibling和previousElementSibling元素可以获取到后一个和前一个相邻元素。

通过input.context中的parentElement元素可以获取到行元素，可以获取到该行所有的列元素。

Grid中列设置：

{  
 **field**: **"decideRefundAmount"**,  
 **title**: **'退款金额'**,  
 **attributes**: {**style**: **"text-align:center"**},  
 **width**: 100,  
 editor: **function** (container, options) {  
 *$*(**'<input name="'** + options.field + **'" />'**)  
 .**appendTo**(container)  
 .kendoNumericTextBox({  
 **min**: 0,  
 }).data(**"kendoNumericTextBox"**);  
 }  
}

dataSource.model.field中的校验逻辑：

**decideRefundAmount**: {  
 **validation**: {  
 numValidate: **function** (input) {  
 **if** (input.is(**"[name = 'decideRefundAmount']"**) && input.val() != **''**) {  
 **var** num1 = parseFloat(input.context.previousElementSibling.**innerText**);  
 **var** num2 = parseFloat(input.val());  
 **if** (num2 > num1) {  
 input.attr(**"data-numValidate-msg"**, **'退款金额应不超过可退金额'**);  
 **return false**;  
 }  
 }  
 **return true**;  
 }  
 },  
},

## 弹窗

### 新打开一个弹窗界面

openTab的第一个参数 "FND"+regionId 要具有唯一性，不然可能会发生弹窗混乱的现象。

**window**.**top**.*openTab*(**"FND"**+regionId, **"省份维护"**, **"${base.contextPath}/fnd/region/region\_province.html?regionId="** + regionId);

### 弹窗刚好遮住父界面（比较美观）

弹窗按钮：注意参数中引号的使用

**'<a href="#" style="margin-right:8px;margin-left: 8px;" onclick="***changeTable***('**+dataItem.meterId+**',\''**+dataItem.**meterName**+**'\')"><font color="#4169E1" size="2">换表</font></a>'**

即使只传一个meterName参数，也需要使用转义符，否则会报错。如果只传一个meterId参数，(数字)可以不用转义符。且引号三层嵌套时，最外层必须用单引号，因为转义符不能和双引号一起使用。

**function** *changeTable*(meterId, meterName) {  
 **newWindow** = *$*(**"#changeTableWin"**).kendoWindow({  
 **actions**: [**"Close"**],  
 **visible**: **false**,  
 **iframe**: **true**,  
 **modal**: **true**,  
 **title**: **"换表"**,  
 **content**: ***BaseUrl***+**"/meter/change\_meter\_table.html?meterId="**+meterId+**"&meterName="**+meterName,  
 close: **function** () {  
 *$*(**'#grid'**).data(**"kendoGrid"**).dataSource.page(1);  
 }  
 }).data(**"kendoWindow"**);  
 **var** dialog = *$*(**"#changeTableWin"**).data(**"kendoWindow"**);  
 dialog.maximize().open();  
}

### 弹窗完全覆盖原界面

*location*.**href** = **"region\_area.html?regionId="** + regionId + **"&parentId="** + pid + **"&ppid="** + ppid;

### 普通弹窗

**var** url = ***BaseUrl***+**"/om/om\_balance\_accounts.html"**;

**var *dialog*** = *$*(**"#editWin"**).kendoWindow({  
 **actions**: [ **"Maximize"**, **"Minimize"**, **"Close"**],  
 **width**: 500, *//也可以 width: '100%' 的形式* **height**: 165,  
 **title**: **'<@spring.message "hap.edit"/>'**,  
 **content**: ***url***,  
 **iframe**: **true**,  
 **visible**: **false**,  
 **modal**: **true**,  
 close: **function**() {  
*// location.reload();* }  
 }).data(**"kendoWindow"**);  
  
 ***dialog***.center().open();  
 }

## 下拉框

valuePrimitive

<http://www.scscms.com/html/article/20131112-dropdownlist.html>

## 复选框

### 修改复选框大小

Style中的zoom属性可以调整复选框大小，设置为100%时为默认大小，然后可以设置大于1或小于1。

<**input class="col-sm-1" id="red" style="margin-top**: 3**px**;**zoom**: 150%;**" name="color" type="checkbox" value="red"** />

### 复选框竖直排列



1、要将复选框竖直排列，需要将复选框放到一个div标签里，然后放到一个fieldset标签中。

<**fieldset id="favoritecolor"**>  
 <**h2**>Favorite color：</**h2**>  
 <**div class="row"**>  
 <**input class="col-sm-1" id="red" style="margin-top**: 3**px**;**zoom**: 150%;**" name="color" type="checkbox" value="red"** />  
 <**label class="col-sm-2" style="margin-top**: 5**px**;**"**>Red</**label**>  
 <**input class="k-textbox col-sm-4" name="red" type="text" value="红色"** />  
 </**div**>  
 <**div class="row" style="margin-top**: 10**px**;**"**>  
 <**input class="col-sm-1" id="green" style="margin-top**: 3**px**;**zoom**: 150%;**" name="color" type="checkbox" value="green"** />  
 <**label class="col-sm-2" style="margin-top**: 5**px**;**"**>Green</**label**>  
 <**input class="k-textbox col-sm-4" type="text" value="绿色"** />  
 </**div**>  
 <**div class="row" style="margin-top**: 10**px**;**"**>  
 <**input class="col-sm-1" id="yello" style="margin-top**: 3**px**;**zoom**: 150%;**" name="color" type="checkbox" value="yello"** />  
 <**label class="col-sm-2" style="margin-top**: 5**px**;**"**>Yello</**label**>  
 <**input class="k-textbox col-sm-4" type="text" value="黄色"** />  
 </**div**>  
</**fieldset**>

## 按钮

### 防止按钮重复提交

1. Hap提供的锁定按钮方法

Btn\_reset1是被点击按钮的ID

**var** options = **new** *Object*();  
options.**target** = **"#btn\_reset1"**;  
**Hap**.lockBtn(options)  
 ...  
 Hap.unLockBtn(options);

Hap方法代码js源码：

Hap.lockBtn = **function** (options) {  
 options = $.extend({  
 **"target"**:**""**,  
 **"message"**: **""** },options || {});  
 **var** target = options.**target** || **".hap-submit-load"**;  
 **var** dom = $(target);  
 **var** message = options.**message**;  
 **if**(message){  
 dom.val(message);  
 }  
 dom.addClass(**"disabled"**);  
 dom.css(**"cursor"**,**"wait"**)  
  
}

1. Hap提供的添加遮罩层方法

使用遮罩层防止按钮重复点击：

**function** *testMask*() {  
 **Hap**.blockUI();  
 setTimeout(*closeMask*, 5000);  
}  
  
**function** *closeMask*() {  
 **Hap**.unblockUI();  
}

### 遮罩层

Hap提供了遮罩层

**Hap**.blockUI(); //添加遮罩层  
**Hap**.unblockUI(); //隐藏遮罩层

遮罩层源码：

*/\*\*  
 \* 表单提交遮罩  
 \** ***@param*** *options:  
 \* options.target 目标dom的id等标识  
 \* options.message:提示信息内容  
 \* options.textOnly 仅文字提示  
 \* options.iconOnly 仅图标提示  
 \* options.boxed 带边框的提示内容  
 \* options.animate 提示动画  
 \*  
 \*/*Hap.blockUI = **function**(options) {  
 options = $.extend(**true**,{}, {boxed:**true**}, options);  
 **var** html = **''**;  
  
 **if** (options.animate) {  
 html = **'<div class="loading-message '** + (options.boxed ? **'loading-message-boxed'** : **''**) + **'">'** + **'<div class="block-spinner-bar"><div class="bounce1"></div><div class="bounce2"></div><div class="bounce3"></div></div>'** + **'</div>'**;  
 } **else if** (options.iconOnly) {  
 html = **'<div class="loading-message '** + (options.boxed ? **'loading-message-boxed'** : **''**) + **'"><img src="'** + \_basePath + **'/lib/assets/global/img/loading-spinner-grey.gif" align=""></div>'**;  
 } **else if** (options.textOnly) {  
 html = **'<div class="loading-message '** + (options.boxed ? **'loading-message-boxed'** : **''**) + **'"><span>&nbsp;&nbsp;'** + (options.message ? options.message : **'LOADING...'**) + **'</span></div>'**;  
 } **else** {  
 html = **'<div class="loading-message '** + (options.boxed ? **'loading-message-boxed'** : **''**) + **'"><img src="'** + \_basePath + **'/lib/assets/global/img/loading-spinner-grey.gif" align=""><span>&nbsp;&nbsp;'** + (options.message ? options.message : **'LOADING...'**) + **'</span></div>'**;  
 }  
  
 **if** (options.target) { *// element blocking* **var** el = $(options.target);  
 **if** (el.height() <= ($(window).height())) {  
 options.cenrerY = **true**;  
 }  
 el.block({  
 message: html,  
 baseZ: options.zIndex ? options.zIndex : 1000,  
 centerY: options.centerY !== undefined ? options.centerY : **false**,  
 css: {  
 top: **'10%'**,  
 border: **'0'**,  
 padding: **'0'**,  
 backgroundColor: **'none'** },  
 overlayCSS: {  
 backgroundColor: options.overlayColor ? options.overlayColor : **'#555'**,  
 opacity: options.boxed ? 0.05 : 0.1,  
 cursor: **'wait'** }  
 });  
 } **else** { *// page blocking* $.blockUI({  
 message: html,  
 baseZ: options.zIndex ? options.zIndex : 1000,  
 css: {  
 border: **'0'**,  
 padding: **'0'**,  
 backgroundColor: **'none'**,  
 },  
 overlayCSS: {  
 backgroundColor: options.overlayColor ? options.overlayColor : **'#555'**,  
 opacity: options.boxed ? 0.05 : 0.1,  
 cursor: **'wait'** }  
 });  
 }  
};  
  
*/\*\*  
 \* 取消遮罩  
 \** ***@param*** *target  
 \*/*Hap.unblockUI = **function**(target) {  
 **if** (target) {  
 $(target).unblock({  
 onUnblock: **function**() {  
 $(target).css(**'position'**, **''**);  
 $(target).css(**'zoom'**, **''**);  
 }  
 });  
 } **else** {  
 $.unblockUI();  
 }  
};