1.介绍

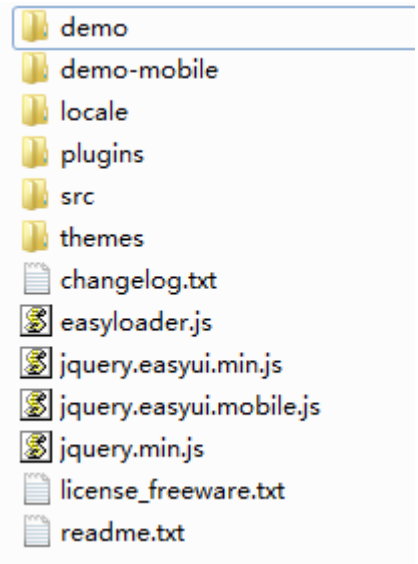
EasyUI 是一种基于 jQuery 的用户界面插件集合。使用 EasyUI 你不需要写很多代码，你只需要通过编写一些简单 HTML 标记，就可以定义用户界面。

1.1 下载

从官网下载 easyui 的 demo。下载地址：

http://www.jeasyui.net/download/

解压：



demo 中是各种控件的简单示例，适用于 PC 浏览器。
demo-mobie 是手机版的。
locale 是语言包，使 easyui 支持国际化 。
plugins 可以单独引用 easyui 的某个插件。
src 中是源码
themes 中是界面风格主题。

1.2 基本语法

使用 EasyUI 需要引入以下几个文件：

```
<link rel="stylesheet" type="text/css" href="themes/default/easyui.css">
<link rel="stylesheet" type="text/css" href="themes/icon.css">
```

```
<script type="text/javascript" src="jquery.min.js"></script>
<script type="text/javascript" src="jquery.easyui.min.js"></script>
```

jquery.easyui.min.js 中包含了 EasyUI 的全部插件，如果只使用单独少数的几个插件，为了减少网络流量，可以引用 plugins 下的插件 js。

用一个简单的示例演示 EasyUI 的效果，比如网站中常见的树形菜单：

```
<ul >
    <li><span>菜单 1</span></li>
    <li><span>菜单 2</span></li>
    <li>
        <span>菜单 3</span>
        <ul>
            <li><span>菜单 3-1</span></li>
            <li><span>菜单 3-2</span></li>
            <li>
                <span>菜单 3-3</span>
                <ul>
                    <li><span>菜单 3-3-1</span></li>
                    <li><span>菜单 3-3-2</span></li>
                </ul>
            </li>
        </ul>
    </li>
</ul>
```

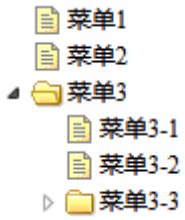
不使用 EasyUI 的时候，如果不自定义样式和鼠标事件，显示是这样的：

- 菜单1
- 菜单2
- 菜单3
  - 菜单3-1
  - 菜单3-2
  - 菜单3-3
    - 菜单3-3-1
    - 菜单3-3-2

使 EasyUI，只需要在需要处理的标签上加一个 class 即可，树形菜单的 class=”easyui-tree”

```
<ul class="easyui-tree">
    <li><span>菜单1</span></li>
    <li><span>菜单2</span></li>
```

显示结果如下图：



并且点击父菜单(图标是文件夹的)，可以展开和收起子菜单。

使用 EasyUI 的控件，有两种形式

第一种是需要给页面中的元素加上 class="easyui-xxx"，其中 xxx 就是控件的名字。如：

```
<ul class="easyui-tree">
    <li><span>菜单 1</span></li>
    <li><span>菜单 2</span></li>
</ul>
```

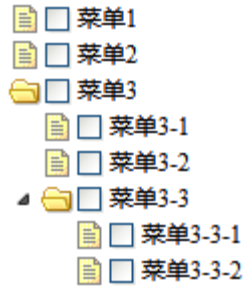
如果有其它的属性，可以直接写到标签中或 data-options 中：

```
<ul class="easyui-tree" checkbox="true">
```

或者：

```
<ul class="easyui-tree" data-options="checkbox:true">
```

运行结果是菜单节点前多了一个复选框：



第二种是使用 javascript 初始化控件,$(元素).xxx()，其中 xxx 就是控件名，如：

```
<ul id="test-menu" >
    <li><span>菜单 1</span></li>
    <li><span>菜单 2</span></li>
</ul>
<script type="text/javascript">
    $(function(){
        $("#test-menu").tree();
    })
</script>
```

如果有其它的属性，可以在方法中指定：

```
$(function () {
    $("#test-menu").tree({
        checkbox: true
    });
})
```

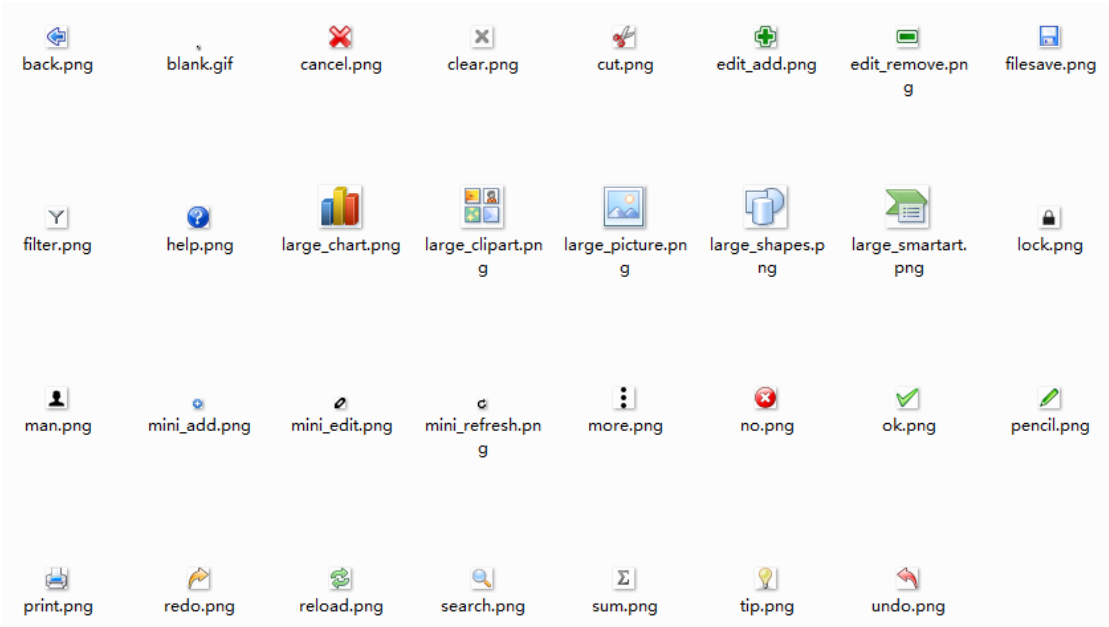
每个控件都有自己的方法和属性，会在后面的课程中一一介绍。

EasyUI 的基础用法和在线示例，参见官网：http://www.jeasyui.net/demo/380.html

EasyUI 控件使用有三个部分需要学习：属性、事件和方法。http://www.jeasyui.net/plugins/

1.3 系统图标

在 themes/icon.css 中定义了一些 class 样式，控件加上这个样式就会有一些小图标：



这些图标样式统一以 icon-xx 命名。具体参见 icon.cdd。图片在 themes/icons 文件夹中。

2.拖拽和调整大小

2.1 可拖动 draggable

设置一个元素可以用鼠标拖动，拖动是 easyui 的一个基础插件。

属性：

名称	类型	描述	默认值

proxy	string,function	拖动时要使用的代理元素，设置为 'clone' 时，克隆元素将被用作代理。如果指定一个函数，它必须返回一个 jQuery 对象。
		下面的实例演示了如何创建简单的代理对象。
		`$('.dragitem').draggable({`
		`  proxy: function(source){`
		`  var p = $('<div style="border:1px solid #ccc;width:80px"></div>');`
		`  p.html($(source).html()).appendTo('body');`
		`  return p;`
		`  }`
		`});`
			null

revert	boolean	如果设置为 true，拖动结束后元素将返回它的开始位置。	FALSE

cursor	string	拖动时的 css 光标（cursor）。	move
deltaX	number	拖动的元素相对于当前光标的 X 轴位置。	null
deltaY	number	拖动的元素相对于当前光标的 Y 轴位置。	null
handle	selector	启动可拖动（draggable）的处理（handle）。	null
disabled	boolean	如果设置为 true，则停止可拖动（draggable）。	FALSE
edge	number	能够在其中开始可拖动（draggable）的拖动宽度。	0
axis	string	定义拖动元素可在其上移动的轴，可用的值是 'v' 或 'h'，当设为 null，将会沿着 'v' 和 'h' 的方向移动。	null

事件：

名称	参数	描述
onBeforeDrag	e	拖动前触发，返回 false 就取消拖动。
onStartDrag	e	目标对象开始拖动时触发。
onDrag	e	拖动期间触发。返回 false 将不进行实际的拖动。
onStopDrag	e	拖动停止时触发。

方法：

名称	参数	描述
options	none	返回选项（options）属性（property）。
proxy	none	如果设置了代理（proxy）属性就返回拖动代理（proxy）。
enable	none	启用拖动动作。
disable	none	禁用拖动动作。

2.1.1 基础拖动

在元素上设置 class="easyui-draggable"，该元素即可被鼠标拖动：

```
<!--演示普通的拖动-->
<div id="drag01" class="easyui-draggable"
style="width:100px;height:100px;background-color: #cccccc">
</div>
```

当鼠标悬浮到元素上方时，鼠标会变成一个十字，按住鼠标左键，可以拖拽元素。

使用 js 初始化拖动可以实现同样的效果：

```
<!--演示使用 js 初始化 draggable-->
<div id="drag02" style="width:100px;height:100px;background-color: #cccccc">
    drag02
</div>
```

```
<script type="text/javascript">
    $(function(){
        $("#drag02").draggable();
    })
</script>
```

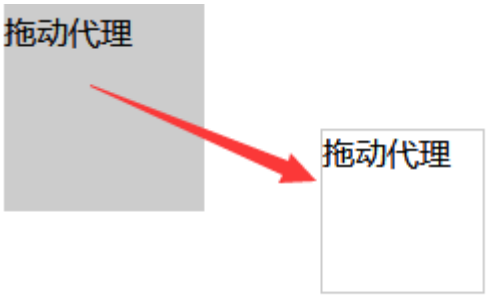
2.1.2 常用属性

handle 属性可以设置启动拖拽的元素，如：

```
<!--必须从拖动标题栏才可以-->
<div id="drag03" class="easyui-draggable" data-options="handle:'#title'"
     style="width:100px;height:100px;background-color: #cccccc">
    <div id="title" style="background:#f5daa7;">title</div>
</div>
```

鼠标必须放在 title 上才能拖动。

proxy 代理，拖动元素时，创建一个代理对象，在移动的过程中，拖动的是代理对象，停止拖动时，原对象会显示在终点位置：




```
<!--演示代理属性 proxy-->
<div id="drag04" style="width:100px;height:100px;background-color: #cccccc">
    拖动代理
</div>
<script type="text/javascript">
    $('#drag04').draggable({
        proxy: function (source) {
            var p = $('<div style="border:1px solid
#ccc;width:80px;height:80px"></div>');
            p.html($(source).html()).appendTo('body');
```

```
            return p;
        }
    });
</script>
```

2.1.3 约束拖动

限制一个元素只能在指定范围内拖动，使用的是 onDrag 事件

```
<!--演示约束拖动-->
<div style=" width:500px;height:300px; border:1px solid #ccc;">
    <div class="easyui-draggable" data-options="onDrag:myOnDrag"
         style="width:100px;height:100px;background:#fafafa;border:1px solid #ccc;">
    </div>
</div>
<script type="text/javascript">
    function myOnDrag(e) {
        var d = e.data;
        $(d).html(d.left+"-"+d.top)
        if (d.left < 0) {//如果拖拽到最左边
            d.left = 0;//横坐标设置为 0
        }
        if (d.top < 0) {//如果拖拽到最上边
            d.top = 0
        }
        //如果当前横坐标+宽度(就是得到右边界)>父容器的宽度
        if (d.left + $(d.target).outerWidth() > $(d.parent).width()) {
            d.left = $(d.parent).width() - $(d.target).outerWidth();
        }
        if (d.top + $(d.target).outerHeight() > $(d.parent).height()) {
            d.top = $(d.parent).height() - $(d.target).outerHeight();
        }
    }
</script>
```

2.2 可放置 droppable

droppable 是可放置的意思，和 draggable 配合使用，droppable 可以接受 draggable 的元素。

属性：

名称	类型	描述	默认值
accept	selector	确定将被接受的可拖动元素。	null
disabled	boolean	如果设置为 true，则停止可放置（droppable）。	FALSE

事件：

名称	参数	描述
onDragEnter	e,source	当可拖动元素被拖进来时触发。source 参数指被拖动的 DOM 元素。
onDragOver	e,source	当可拖动元素被拖过时触发。source 参数指被拖动的 DOM 元素。
onDragLeave	e,source	当可拖动元素被拖离开时触发。source 参数指被拖动的 DOM 元素。
onDrop	e,source	当可拖动元素被放下时触发。source 参数指被拖动的 DOM 元素。

方法：

名称	参数	描述
options	none	返回选项（options）对象。
enable	none	启用可放置功能。
disable	none	禁用可放置功能。

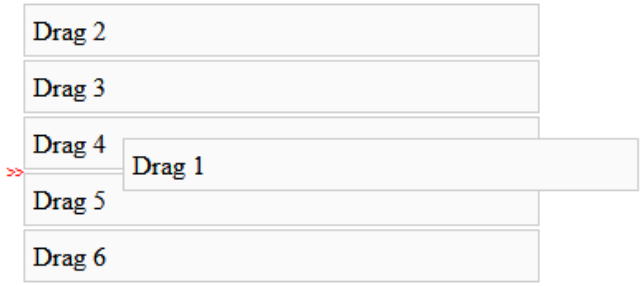
2.2.1 基本设置

在一个元素上设置 class=”easyui-droppable”后，它可以接受一个拖拽。或使用 js 设置：设置三个可拖拽的元素，drag1、drag2、drag3，右侧的 droppable 容器可接受 drag1 和 drag3。当可接受的对象拖入 droppable 容器时，边框变成红色。drag2 不被接受，所以释放后会回到原来的位置，不能放在右侧的 div 里。

```
<!--设置样式-->
<style type="text/css">
    .drags {
        width: 50px;
        height: 50px;
        margin: 5px;
        border: 1px solid #ccc
    }
    .container, #drop {
        width: 300px;
        height: 300px;
        border: 1px solid #ccc;
        float: left;
        margin: 5px;
    }
</style>
<!--可拖拽的元素-->
<div class="container">
    <div class="drags" id="drag1">drag1</div>
    <div class="drags" id="drag2">drag2</div>
    <div class="drags" id="drag3">drag3</div>
</div>
<!--放置的目标容器-->
<div id="drop">
</div>
<script type="text/javascript">
    $(function () {
        $(".drags").draggable({
            proxy: 'clone',//拖动代理对象
            revert: true   //释放后回到原来的位置
        });
        $('#drop').droppable({
            accept:'#drag1,#drag3',//可接受这两个对象放置
            onDragEnter:function(e,source){//当拖拽到当前元素上方时
                $(source).draggable('options').cursor='pointer';
                $(source).draggable('proxy').css('border','1px solid red');
            },
            onDragLeave:function(e,source){//当拖拽到离开当前元素时
                $(source).draggable('options').cursor='no-drop';
                $(source).draggable('proxy').css('border','1px solid #ccc');
            },
            onDrop:function(e,source){//当拖拽到放置到当前元素时
                $(this).append(source)
            }
```

```
        });
    })
</script>
```

2.2.2 改变项目顺序



拖拽列表中的某个元素，放置到列表中的新位置，设计思路：列表使用 ul-li，每一个 li 既是 draggable 可拖动的，也是 droppable。

```
<ul style="margin:0;padding:0;margin-left:10px;">
    <li class="drag-item">Drag 1</li>
    <li class="drag-item">Drag 2</li>
    <li class="drag-item">Drag 3</li>
    <li class="drag-item">Drag 4</li>
    <li class="drag-item">Drag 5</li>
    <li class="drag-item">Drag 6</li>
</ul>
<style type="text/css">
    .drag-item{
        list-style-type:none;
        display:block;
        padding:5px;
        border:1px solid #ccc;
        margin:2px;
        width:300px;
        background:#fafafa;
    }
    .indicator{
        position:absolute;
        font-size:9px;
        width:10px;
        height:10px;
        display:none;
        color:red;
    }
</style>
```

```
<script>
    $(function(){
        var indicator = $('<div class="indicator">&gt;&gt;</div>').appendTo('body');
        $('.drag-item').draggable({
            revert:true
        }).droppable({
            onDragOver:function(e,source){
                indicator.css({
                    display:'block',
                    left:$(this).offset().left-10,
                    top:$(this).offset().top+$(this).outerHeight()-5
                });
            },
            onDragLeave:function(e,source){
                indicator.hide();
            },
            onDrop:function(e,source){
                $(source).insertAfter(this);
                indicator.hide();
            }
        });
    });
```

2.3 尺寸调节 resizable

resizable 标记可以创建尺寸可调节的对象，class=”easyui-resizable”

属性：

名称	类型	描述	默认值
disabled	boolean	如果设置为 true，则禁止调整尺寸。	FALSE
handles	string	指可调整尺寸（resizable）的方向，’n’是北，’e’是东，等等。	n, e, s, w, ne, se, sw, nw, all
minWidth	number	调整尺寸时最小宽度。	10
minHeight	number	调整尺寸时最小高度。	10
maxWidth	number	调整尺寸时最大宽度。	10000
maxHeight	number	调整尺寸时最大高度。	10000
edge	number	被调整尺寸的边框的边缘。	5

事件：

名称	参数	描述
onStartResize	e	开始调整尺寸时触发。
onResize	e	调整尺寸期间触发。返回 false 时，DOM 元素将不进行实际的调整尺寸动作。

onStopResize	e	停止调整尺寸时触发。

方法：

名称	参数	描述
options	none	返回可调整尺寸（resizable）选项（options）。
enable	none	启用可调整尺寸（resizable）特性。
disable	none	禁用可调整尺寸（resizable）特性。

```
<div class="easyui-resizable" data-options="minWidth:100,minHeight:100"
style="width:200px;height:150px;border:1px solid #ccc;">
    <div style="padding:20px">Resize Me</div>
</div>
```

以上代码等价于：

```
<!--使用 js 创建尺寸可调节的元素-->
<div id="resize-01" data-options="" style="width:200px;height:150px;border:1px solid
#ccc;">
    <div style="padding:20px">Resize Me</div>
</div>
<script type="text/javascript">
    $(function(){
        $("#resize-01").resizable({minWidth:100,minHeight:100})
    })
</script>
```

3.基础控件

3.1 搜索框 searchbox

搜索框是一个带搜索按钮的输入框。

```
<input class="easyui-searchbox"  style="width:300px"/>
```

属性：

名称	类型	描述	默认值
width	number	组件的宽度。	auto
height	number	组件的高度。该属性自版本 1.3.2 起可用。	22
prompt	string	显示在输入框里的提示信息。	''
value	string	输入的值。	''
menu	selector	搜索类型的菜单。每个菜单项可以有下列的属性： name：搜索类型名称。	null

		selected：当前选择的搜索类型名称。 下面的实例演示了如何定义一个选中的搜索类型名称。 <input class="easyui-searchbox" style="width:300px" data-options="menu:'#mm'" /> Search Item1 Search Item2 Search Item3 	
searcher	function(value,name)	当用户按下搜索按钮或者按下 ENTER 键时，searcher 函数将被调用。	null

方法：

名称	参数	描述
options	none	返回选项（options）对象。

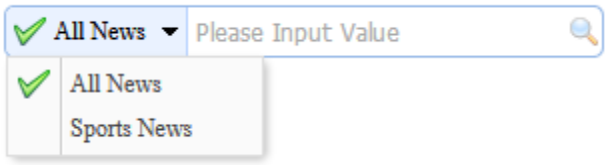
menu	none	返回搜索类型的菜单对象。
		下面的实例演示如何改变菜单项图标。
		var m = $('#ss').searchbox('menu'); // get the menu object
		var item = m.menu('findItem', 'Sports News'); // find the menu item
		// change the menu item icon
		m.menu('setIcon', {
		target: item.target,
		iconCls: 'icon-save'
		});
		// select the searching type name
		$('#ss').searchbox('selectName', 'sports');

textbox	none	返回文本框对象。
getValue	none	返回当前的搜索值。
setValue	value	设置新的搜索值。
getName	none	返回当前的搜索类型名称。
selectName	name	选择当前的搜索类型名称。 代码实例： $('#ss').searchbox('selectName', 'sports');
destroy	none	销毁该组件。
resize	width	重设组件的宽度。

点击搜索按钮的，获取输入框的值：

```
<input class="easyui-searchbox" data-options="prompt:'Please Input
Value',searcher:doSearch" style="width:300px"/>
<script type="text/javascript">
    function doSearch(value){
        alert('You input: ' + value);
    }
</script>
```

带分类的搜索框：



通过 menu 属性指定下拉框中的菜单 id

```
<div style="margin:20px 0;"></div>
<input class="easyui-searchbox" data-options="menu:'#mm',searcher:doSearch2"
style="width:300px"/>
<div id="mm">
    <div data-options="name:'all',iconCls:'icon-ok'">All News</div>
    <div data-options="name:'sports'">Sports News</div>
</div>
<script>
    function doSearch2(value,name){
        alert('You input: ' + value+'('+name+')');
    }
</script>
```

3.2 进度条 progressbar

进度条（progressbar）提供了一种显示长时间操作进度的反馈。进度可被更新以便让用户知道当前正在执行的操作。

属性：

名称	类型	描述	默认值
width	string	设置进度条（progressbar）的宽度。	auto
height	number	组件的高度。该属性自版本 1.3.2 起可用。	22
value	number	百分比值。	0
text	string	显示在组件上的文本模板。	{value}%

事件：

名称	参数	描述
onChange	newValue,oldValue	当值改变时触发。

		代码实例：
		`$('#p').progressbar({`
		`  onChange: function(value){`
		`  alert(value)`
		`  }`
		`});`

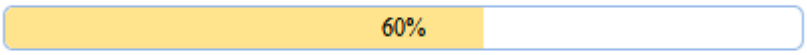
方法：

名称	参数	描述
options	none	返回选项（options）对象。

resize	width	调整组件尺寸。
		代码实例：
		`$('#p').progressbar('resize');  // 调整进度条为初始宽度`
		`$('#p').progressbar('resize', 350);  // 调整进度条为一个新的宽度`

getValue	none	返回当前的进度值。
setValue	value	设置一个新的进度值。

基本的进度条：




```
<div class="easyui-progressbar" data-options="value:60" style="width:400px;"></div>
```

进度条常用操作：

```
<div id="bar" class="easyui-progressbar" data-options="value:60"
style="width:400px;"></div>
<input type="button" value="开始" onclick="start()"/>
<script type="text/javascript">
    function start(){
        //获得进度条当前的值
        var value = $('#bar').progressbar('getValue');
        if (value < 100){
            value += Math.floor(Math.random() * 10);
            //给进度条设置新的值
            $('#bar').progressbar('setValue', value);
        }
    }
</script>
```

3.3 分页插件 pagination

3.3.1 基础分页

分页插件 pagination 用于创建分页插件，包括每页几条、总页数 、页码等


```
<div class="easyui-pagination" data-options="total:100"></div>
```

引用对应的语言包，可以实现国际化。如引入汉化包：

```
<script type="text/javascript" src="/easyui/locale/easyui-lang-zh_CN.js"></script>
```

3.3.2 常用属性

属性：

名称	类型	描述	默认值
total	number	记录总数，应该在创建分页（pagination）时设置。	1
pageSize	number	页面尺寸。（注：每页显示的最大记录数）	10
pageNumber	number	创建分页（pagination）时显示的页码。	1

pageList	array	用户能改变页面尺寸。pageList 属性定义了能改成多大的尺寸。
		代码实例：
		`$('#pp').pagination({`
		`pageList: [10,20,50,100]`
		`});`
			[10,20,30,50]

loading	boolean	定义数据是否正在加载。	FALSE

buttons	array,selector	定义自定义按钮，可能的值：
		1、数组，每个按钮包含两个属性：
		iconCls：CSS class，它将显示一个背景图片
		handler：当按钮被点击时的处理函数
		2、选择器，指示按钮。
		按钮可通过标记声明：
		`<div class="easyui-pagination"`
		`style="border:1px solid #ccc" data-options="`
		`total: 114,`
		`buttons: [{`
		`iconCls:'icon-add',`
		`handler:function(){alert('add')}`
			null

		<pre>},'-',{
		iconCls:'icon-save',
		handler:function(){alert('save')}
		}]">
		 </pre>
		按钮也可以使用 javascript 创建：
		<pre>$('#pp').pagination({
		total: 114,
		buttons: [{
		iconCls:'icon-add',
		handler:function(){alert('add')}
		},'-',{
		iconCls:'icon-save',
		handler:function(){alert('save')}
		}]
		});</pre>	

layout	array	分页布局定义。该属性自版本 1.3.5 起可用。
		布局项目包括一个或多个下列值：
		1、list：页面尺寸列表。
		2、sep：页面按钮分割。
		3、first：第一个按钮。
		4、prev：前一个按钮。
		5、next：后一个按钮。
		6、last：最后一个按钮。
		7、efresh：刷新按钮。
		8、manual：允许输入域页码的手动页码输入框。
		9、links：页码链接。
		代码实例：
		<pre>$('#pp').pagination({
		layout:['first','links','last']
		});</pre>	

links	number	链接数量，只有当 'links' 项包含在 'layout' 中时才是有效的。该属性自版本 1.3.5 起可用。	10
showPageList	boolean	定义是否显示页面列表。	TRUE
showRefresh	boolean	定义是否显示刷新按钮。	TRUE
beforePageText	string	在 input 组件之前显示 label。	Page
afterPageText	string	在 input 组件之后显示 label。	of {pages}
displayMsg	string	Display a page information.	显示 {from} to {to} of {total}

			页面信息。

事件：

名称	参数	描述

onSelectPage	pageNumber, pageSize	当用户选择新的页面时触发。回调函数包含两个参数：
		pageNumber：新的页码
		pageSize：新的页面尺寸
		代码实例：
		`$('#pp').pagination({`
		`onSelectPage:function(pageNumber, pageSize){`
		`$(this).pagination('loading');`
		`alert('pageNumber:'+pageNumber+',pageSize:'+pageSize);`
		`$(this).pagination('loaded');`
		`}`
		`});`

onBeforeRefresh	pageNumber, pageSize	刷新按钮点击之前触发，返回 false 就取消刷新动作。
onRefresh	pageNumber, pageSize	刷新之后触发。
onChangePageSize	pageSize	当用户改变页面尺寸时触发。

方法：

名称	参数	描述
options	none	返回选项（options）对象。
loading	none	把分页（pagination）变成正在加载（loading）状态。
loaded	none	把分页（pagination）变成加载完成（loaded）状态。

refresh	options	刷新并显示分页信息。该方法自版本 1.3 起可用。
		代码实例：
		`$('#pp').pagination('refresh'); // 刷新分页栏信息`
		`$('#pp').pagination('refresh',{ // 改变选项，并刷新分页栏信息`
		`total: 114,`
		`pageNumber: 6`
		`});`

select	page	选择一个新页面。页面索引从 1 开始。该方法从版本 1.3 起可用。
		代码实例：
		`$('#pp').pagination('select'); // 刷新当前页面`
		`$('#pp').pagination('select', 2); // 选择第二页`

3.3.3 自定义分页

添加自定义按钮：

```
$("#page").pagination({
    total: 114,
    buttons: [{
        iconCls: 'icon-add',
        handler: function () {
            alert('add');
        }
    }, {
        iconCls: 'icon-cut',
        handler: function () {
            alert('cut');
        }
    }, {
        iconCls: 'icon-save',
        handler: function () {
            alert('save');
        }
    }]
})
```

添加其它组件：

```
<div class="easyui-pagination" data-options="total:114,buttons:$('#buttons')"></div>
<div id="buttons">
    <table style="border-spacing:0">
        <tr>
            <td>
                <input class="easyui-searchbox" style="width:150px">
            </td>
            <td>
                <a href="javascript:void(0)" class="easyui-linkbutton"
data-options="iconCls:'icon-save',plain:true"></a>
            </td>
        </tr>
    </table>
</div>
```

4.布局控件

4.1 面板 panel

panel 面板是一个容器，是创建其它组件的基础。它是一个 div，可以添加标题栏、窗口工具栏、底部工具栏、远程加载数据等。

属性：

名称	类型	描述	默认值
id	string	面板（panel）的 id 属性。	null
title	string	显示在面板（panel）头部的标题文字。	null
footer	string	面板的底部工具栏	null
iconCls	string	在面板（panel）里显示一个 16x16 图标的 CSS class。	null
width	number	设置面板（panel）的宽度。	auto
height	number	设置面板（panel）的高度。	auto
left	number	设置面板（panel）的左边位置。	null
top	number	设置面板（panel）的顶部位置。	null
cls	string	给面板（panel）添加一个 CSS class。	null
headerCls	string	给面板（panel）头部添加一个 CSS class。	null
bodyCls	string	给面板（panel）主体添加一个 CSS class。	null
style	object	给面板（panel）添加自定义格式的样式。 改变面板（panel）边框宽度的代码实例： `<div class="easyui-panel" style="width:200px;height:100px" data-options="style:{borderWidth:2}"></div>`	{}
fit	boolean	当设置为 true 时，面板（panel）的尺寸就适应它的父容器。下面的实例演示了自动调整尺寸到它的父容器的最大内部尺寸的面板（panel）。 `<div style="width:200px;height:100px;padding:5px"><div class="easyui-panel" style="width:200px;height:100px" data-options="fit:true,border:false">Embedded Panel</div></div>`	FALSE
border	boolean	定义了是否显示面板（panel）的边框。	TRUE
doSize	boolean	如果设置为 true，创建时面板（panel）就调整尺寸并做成布局。	TRUE
noheader	boolean	如果设置为 true，面板（panel）的头部将不会被创建。	FALSE

<table>
<tr><td>content</td><td>string</td><td>面板（panel）主体内容。</td><td>null</td></tr>
<tr><td>collapsible</td><td>boolean</td><td>定义是否显示折叠按钮。</td><td>FALSE</td></tr>
<tr><td>minimizable</td><td>boolean</td><td>定义是否显示最小化按钮。</td><td>FALSE</td></tr>
<tr><td>maximizable</td><td>boolean</td><td>定义是否显示最大化按钮。</td><td>FALSE</td></tr>
<tr><td>closable</td><td>boolean</td><td>定义是否显示关闭按钮。</td><td>FALSE</td></tr>
<tr><td>halign</td><td>string</td><td>指定标题栏的位置，left 或者 right</td><td>top</td></tr>
<tr><td>titleDirection</td><td>string</td><td>当设置了 halign 的时候，设置标题文字的方向，up 或者 down</td><td>down</td></tr>
<tr><td rowspan="8">tools</td><td rowspan="8">array,selector</td><td>自定义工具组，可能的值：</td><td rowspan="8">[]</td></tr>
<tr><td>1、数组，每个元素包含 iconCls 和 handler 两个属性。</td></tr>
<tr><td>2、选择器，指示工具组。</td></tr>
<tr><td>面板（panel）工具组可通过已存在 标签声明：</td></tr>
<tr><td><pre>
<div class="easyui-panel"
style="width:300px;height:200px"
    title="My Panel"
data-options="iconCls:'icon-ok',tools:'#tt'">

   <a href="#" class="icon-add"
onclick="javascript:alert('add')">
   <a href="#" class="icon-edit"
onclick="javascript:alert('edit')">

</pre></td></tr>
<tr><td>面板（panel）工具组可通过数组定义：</td></tr>
<tr><td><pre>
 <div class="easyui-panel"
style="width:300px;height:200px"
    title="My Panel"
data-options="iconCls:'icon-ok',tools:[
                {
                    iconCls:'icon-add',

handler:function(){alert('add')}
                },{
                    iconCls:'icon-edit',

handler:function(){alert('edit')}
                }]">

</pre></td></tr>
<tr><td>collapsed</td><td>boolean</td><td>定义初始化面板（panel）是不是折叠的。</td><td>FALSE</td></tr>
</table>

minimized	boolean	定义初始化面板（panel）是不是最小化的。	FALSE
maximized	boolean	定义初始化面板（panel）是不是最大化的。	FALSE
closed	boolean	定义初始化面板（panel）是不是关闭的。	FALSE
href	string	一个 URL，用它加载远程数据并且显示在面板（panel）里。请注意，除非面板（panel）打开，否则内容不会被加载。这对创建一个惰性加载的面板（panel）很有用： Open	null
cache	boolean	设置为 true 就缓存从 href 加载的面板（panel）内容。	TRUE
loadingMessage	string	当加载远程数据时在面板（panel）里显示一条信息。	Loading…

extractor	function	定义如何从 ajax 响应中提取内容，返回提取的数据。
		extractor: function(data){
		var pattern = /<body[^>]*>((.|[\n\r])*)</body>/im;
		var matches = pattern.exec(data);
		if (matches){
		return matches[1]; // only extract body content
		} else {
		return data;
		}
		}	

事件：

名称	参数	描述
onLoad	none	当远程数据被加载时触发。
onBeforeOpen	none	面板（panel）打开前触发，返回 false 就停止打开。
onOpen	none	面板（panel）打开后触发。
onBeforeClose	none	面板（panel）关闭前触发，返回 false 就取消关闭。下面声明的面板（panel）不会关闭。 <div class="easyui-panel" style="width:300px;height:200px;"

		title="My Panel" data-options="onBeforeClose:function(){return false}"> The panel cannot be closed.
onClose	none	面板（panel）关闭后触发。
onBeforeDestroy	none	面板（panel）销毁前触发，返回 false 就取消销毁。
onDestroy	none	面板（panel）销毁后触发。
onBeforeCollapse	none	面板（panel）折叠前触发，返回 false 就停止折叠。
onCollapse	none	面板（panel）折叠后触发。
onBeforeExpand	none	面板（panel）展开前触发，返回 false 就停止展开。
onExpand	none	面板（panel）展开后触发。
onResize	width,height	面板（panel）调整尺寸后触发。 width：新的外部宽度 height：新的外部高度
onMove	left,top	面板（panel）移动后触发。 left：新的左边位置 top：新的顶部位置
onMaximize	none	窗口最大化后触发。
onRestore	none	窗口还原为它的原始尺寸后触发。
onMinimize	none	窗口最小化后触发。

方法：

名称	参数	描述
options	none	返回选项（options）属性（property）。
panel	none	返回外部面板（panel）对象。
header	none	返回面板（panel）头部对象。
body	none	返回面板（panel）主体对象。
setTitle	title	设置头部的标题文本。
open	forceOpen	当 forceOpen 参数设置为 true 时，就绕过 onBeforeOpen 回调函数打开面板（panel）。
close	forceClose	当 forceClose 参数设置为 true 时，就绕过 onBeforeClose 回调函数关闭面板（panel）。
destroy	forceDestroy	当 forceDestroy 参数设置为 true 时，就绕过 onBeforeDestroy 回调函数销毁面板（panel）。

refresh	href	刷新面板（panel）加载远程数据。如果分配了 'href' 参数，将重写旧的 'href' 属性。
		代码实例：
		// open a panel and then refresh its contents.
		$('#pp').panel('open').panel('refresh');
		// refresh contents with a new URL.

		$('#pp').panel('open').panel('refresh','new_content.php') ;
resize	options	设置面板（panel）尺寸并做布局。Options 对象包含下列属性： width：新的面板（panel）宽度 height：新的面板（panel）宽度 left：新的面板（panel）左边位置 top：新的面板（panel）顶部位置 代码实例： $('#pp').panel('resize',{ width: 600, height: 400 });
move	options	移动面板（panel）到新位置。Options 对象包含下列属性： left：新的面板（panel）左边位置 top：新的面板（panel）顶部位置
maximize	none	面板（panel）适应它的容器的尺寸。
minimize	none	最小化面板（panel）。
restore	none	把最大化的面板（panel）还原为它原来的尺寸和位置。
collapse	animate	折叠面板（panel）主体。
expand	animate	展开面板（panel）主体。

4.1.1 基础面板

在一个普通的 div 上添加 class=”easyui-panel”

```
<!--最基础的面板-->
<div class="easyui-panel">
    <p>panel content.</p>
    <p>panel content.</p>
</div>
```

运行效果：

页面上会生成一个带边框的 div。

给面板设置基本属性，如 title:

```
<!--演示 title 和 style-->
<div class="easyui-panel" title="基础面板" style="width:200px;padding:10px">
    <p>panel content.</p>
```

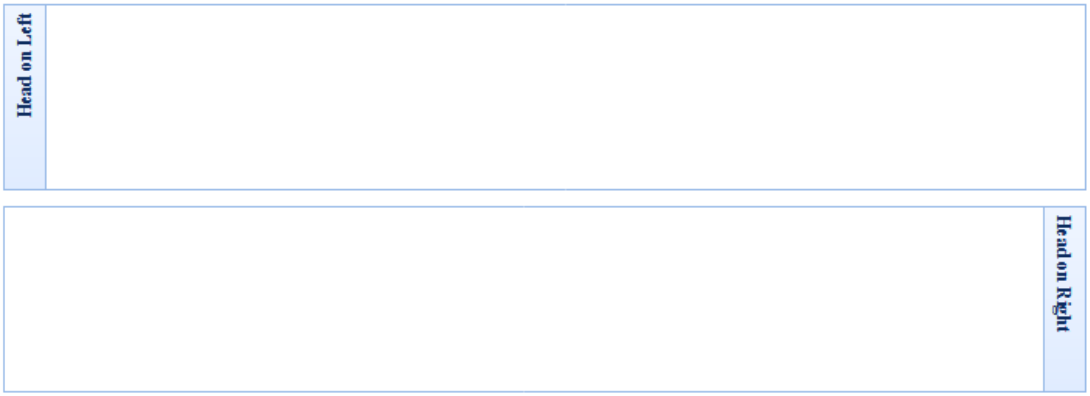
```
    <p>panel content.</p>
</div>
```

运行效果：

title 被处理成一个标题栏。通过 style 指定了面板的 css 样式。

title 默认是在顶部的，可以通过 halign 指定它在左边或右边

```
<!--左边的标题栏-->
<div class="easyui-panel" title="Head on Left" halign="left" titleDirection="up"
style="width:700px;height:120px;">
</div>

<!--右边的标题栏-->
<div class="easyui-panel" title="Head on Right" halign="right"
style="width:700px;height:120px;">
</div>
```



titleDirection 指定文字的方向。up 是由上往下，down 是由下往上。

4.1.2 流式面板

在 style 中将面板的宽度和高度指定成百分比的形式，面板的宽高会自动适应父容器的大小：

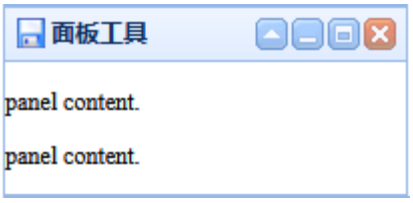
```
<!--流式面板，自适应父容器宽度-->
<div class="easyui-panel" title="流式面板" style="width:20%;">
    <p>panel content.</p>
    <p>panel content.</p>
</div>
```

调整浏览器窗口大小，可以看到流式面板的宽度跟随浏览器窗口变化。

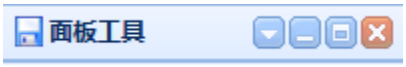
4.1.3 面板常用属性

面板控件有很多属性，如：

```
<!--控件的其它属性，属性可以直接写在标签上，也可以写到 data-options 中-->
<div class="easyui-panel" style="width:20%;" iconCls="icon-save"
    data-options="title:'面板工具
',collapsible:true,minimizable:true,maximizable:true,closable:true">
    <p>panel content.</p>
    <p>panel content.</p>
</div>
```



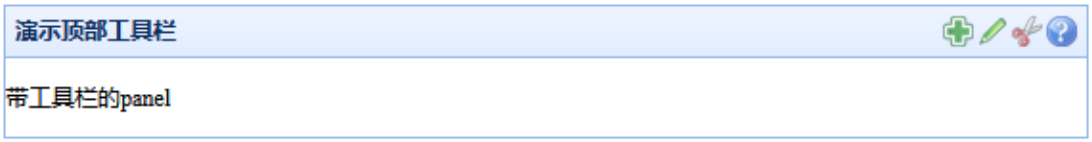
iconCls 指定面板的图标。icon-save 对应的是一个软盘的小图标。
collapsible 指定面板是否可收起，收起面板后，面板只剩下工具栏，内容部分被隐藏。



minimizable 指定面板是否可以最小化。
maximizable 指定面板是否可以最大化。
closable 指定面板是否可以关闭。

4.1.4 顶部和底部工具栏

顶部工具栏：

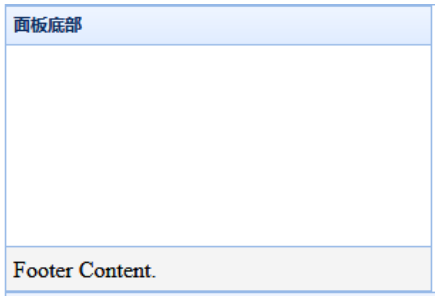


实现代码，使用 tools 属性指定工具栏，tools 的值是工具栏所在容器的 id。

```
<div class="easyui-panel" title="演示顶部工具栏" data-options="tools:'#tt'"
style="width: 30%">
    <p>带工具栏的 panel</p>
</div>
<div id="tt">
    <a href="javascript:void(0)" class="icon-add"
onclick="javascript:alert('add')"></a>
    <a href="javascript:void(0)" class="icon-edit"
onclick="javascript:alert('edit')"></a>
    <a href="javascript:void(0)" class="icon-cut"
onclick="javascript:alert('cut')"></a>
```

```
    <a href="javascript:void(0)" class="icon-help"
onclick="javascript:alert('help')"></a>
</div>
```

底部工具栏：



通过 footer 属性指定底部工具栏，footer 的值是工具栏所在容器的 id。

```
<div class="easyui-panel" title="面板底部" style="width:300px;height:200px;"
data-options="footer:'#ft'">
</div>
<div id="ft" style="padding:5px;">
    Footer Content.
</div>
```

4.1.5 javascript 操作面板

```
<!--使用 js 控制面板元素-->
<div style="margin:20px 0 10px 0;">
    <input type="button" value="打开" onclick="$('#js-panel').panel('open')"/>
    <input type="button" value="关闭" onclick="$('#js-panel').panel('close')"/>
    <input type="button" value="展开" onclick="$('#js-panel').panel('expand',true)"/
    <input type="button" value="收起"
onclick="$('#js-panel').panel('collapse',true)"/>
</div>
<div id="js-panel">
    js 控制的面板
</div>
<script type="text/javascript">
    $(function () {
        $("#js-panel").panel({
            title: "js 控制的 pannel",
            collapsible: true,
            closable: true,
            width:500,
            height:150,
            tools:[{
                iconCls:'icon-add',
                handler:function(){alert('new')}
```

```
            },{
                iconCls:'icon-save',
                handler:function(){alert('save')}
            }]
        });
    })
</script>
```

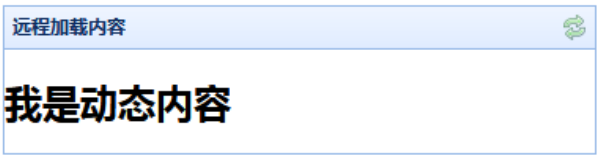
4.1.6 动态加载面板内容

通过 panel 的 href 属性可以指定远程 url。

```
<!--动态获取内容的面板-->
<div id="dynamic-panel" class="easyui-panel" title="远程加载内容" style="width: 20%"
href="dynamic.html" data-options="
            tools:[{
                iconCls:'icon-reload',
                handler:function(){
                    $('#dynamic-panel').panel('refresh', 'dynamic.html');
                }
            }]
        ">
</div>
```

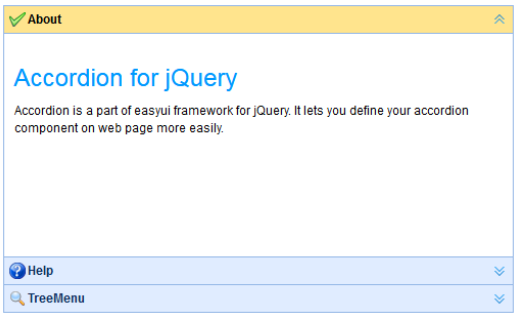
其中 dynamic.html 是远程服务器的请求地址或者另一个页面的地址，如创建一个 dynamic.html 的页面，输入如下内容：

```
<h1>我是动态内容</h1>
```



使用 refresh 方法还能重新载入面板内容。

4.2 折叠面板(手风琴)accordion



折叠面板 accordion 依赖 panel，允许您提供多个面板（panel），同时显示一个或多个面板（panel）。每个面板（panel）都有展开和折叠的内建支持。点击面板（panel）头部可展开或折叠面板（panel）主体。面板（panel）内容可通过 ajax 指定 'href' 属性来加载。用户可定义被选中的面板（panel）。如果未指定，则默认选中第一个面板（panel）。

容器选项：

名称	类型	描述	默认值
width	number	折叠面板（Accordion）容器的宽度。	auto
height	number	折叠面板（Accordion）容器的高度。	auto
fit	boolean	设置为 true，就使折叠面板（Accordion）容器的尺寸适应它的父容器。	FALSE
border	boolean	定义是否显示边框。	TRUE
animate	boolean	定义当展开或折叠面板（panel）时是否显示动画效果。	TRUE
multiple	boolean	设置为 true，则可同时展开多个面板（panel）。该属性自版本 1.3.5 起可用。	FALSE
halign	string	left 或 right,水平面板	
titleDirection	string	up 或 down,水平面板中标题文字的方向	
selected	number	初始化选中的面板（panel）索引。该属性自版本 1.3.5 起可用。	0

面板选项：

折叠面板（Accordion）的面板（panel）选项继承自面板（panel），下面是附加的属性：

名称	类型	描述	默认值
selected	boolean	设置为 true 就展开面板（panel）。	FALSE
collapsible	boolean	定义是否显示可折叠按钮。如果设置为 false,将不能通过点击来展开/折叠面板（panel）。	TRUE

事件：

名称	参数	描述
onSelect	title,index	当面板（panel）被选中时触发。
onUnselect	title,index	当面板（panel）未被选中时触发。该事件自版本 1.3.5 起可用。
onAdd	title,index	当添加一个新面板（panel）时触发。
onBeforeRemove	title,index	当移除一个面板（panel）之前触发，返回 false 就取消移除动作。
onRemove	title,index	当移除一个面板（panel）时触发。

方法：

名称	参数	描述
options	none	返回折叠面板（accordion）的选项。
panels	none	获取全部的面板（panel）。

resize	none	调整折叠面板（accordion）的尺寸。
getSelected	none	获取第一个选中的面板（panel）。
getSelections	none	过去所有选中的面板（panel）。该方法自版本 1.3.5 起可用。
getPanel	which	获取指定的面板（panel）。'which' 参数可以是面板（panel）的标题（title）或索引（index）。

getPanelIndex	panel	获取指定的面板（panel）索引。该方法自版本 1.3 起可用。
		下面的实例显示如何获取选中的面板（panel）索引。
		`var p = $('#aa').accordion('getSelected');`
		`if (p){`
		`  var index = $('#aa').accordion('getPanelIndex', p);`
		`  alert(index);`
		`}`

select	which	选择指定的面板（panel）。'which' 参数可以是面板（panel）的标题（title）或索引（index）。
unselect	which	未选择指定的面板（panel）。'which' 参数可以是面板（panel）的标题（title）或索引（index）。该方法自版本 1.3.5 起可用。

add	options	添加一个新的面板（panel）。默认情况下，新添加的面板（panel）会被选中。如需添加一个未被选中的新面板（panel），请传递 'selected' 属性，并将其设置为 false。
		代码实例：
		`$('#aa').accordion('add', {`
		`  title: 'New Title',`
		`  content: 'New Content',`
		`  selected: false`
		`});`

remove	which	移除指定的面板（panel）。'which' 参数可以是面板（panel）的标题（title）或索引（index）。

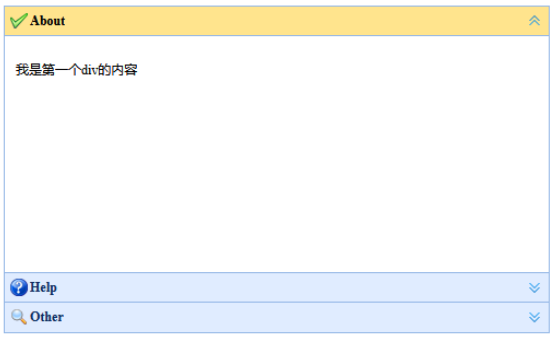
4.2.1 基础手风琴

折叠面板，父容器的 class="easyui-accordion"，父容器中的子节点 div 会被处理成 panel，自带折叠效果：

```
<!--基础折叠面板-->
<div class="easyui-accordion" style="width:500px;height:300px;">
    <div title="About" data-options="iconCls:'icon-ok'" style="padding:10px;">
        <p>我是第一个 div 的内容</p>
    </div>
    <div title="Help" data-options="iconCls:'icon-help'" style="padding:10px;">
```

```
        <p>第二个 div</p>
    </div>
    <div title="Other" data-options="iconCls:'icon-search'" style="padding:10px;">
        <p>新的 div</p>
    </div>
</div>
```

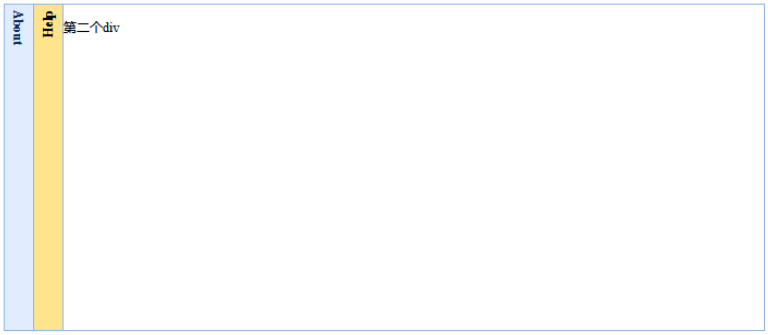
运行效果：



默认展开的是第一个面板，通过 selected=面板索引，可以指定要展开哪个面板。或者在某个面板上设置 selected=true。
流式手风琴和流式面板一样，通过 style 将容器的宽高设定为百分比，略。

4.2.2 水平手风琴

水平手风琴，标题栏是在侧面的，通过 halign 指定标题栏是在左面还是右面，titleDirection 指定标题栏的文字方向。

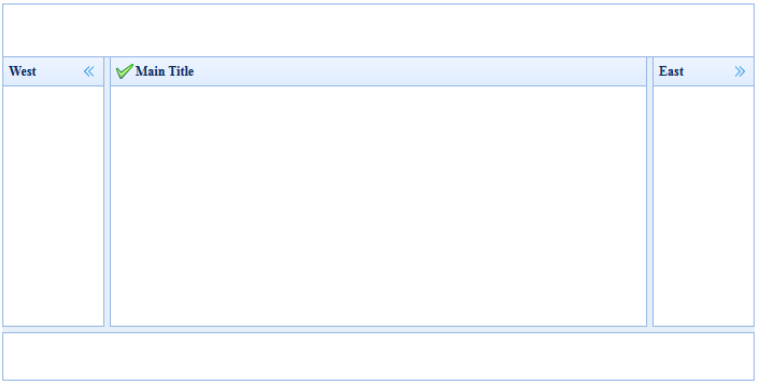



```
<div class="easyui-accordion" halign="left" style="width:700px;height:300px;">
    <div title="About">
        <p>我是第一个 div 的内容</p>
    </div>
    <div title="Help" titleDirection="up">
        <p>第二个 div</p>
    </div>
</div>
```

4.2.3 展开多个面板

默认的手风琴每次可打开一个面板。使用 multiple:true 属性(要写在 data-options 中)，可以设置打开多个面板。

4.3 布局 layout



layout 是布局控件，class=”easyui-layout”。layout 依赖于 panel 和 resizable。layout 的规则是上北(north)下南(south)左西(west)右东(east)和中间(center)。

在容器上设置 class=”easyui-layout”，并在子容器上通过 region 指定方位。

布局选项：

名称	类型	描述	默认值
fit	boolean	当设置为 true 时，就设置布局（layout）的尺寸适应它的父容器。当在 ’body’ 标签上创建布局（layout）时，它将自动最大化到整个页面的全部尺寸。	false

区域面板选项：

区域面板选项（Region Panel Options）是定义在面板（panel）组件中，下面是一些共同的和新增的属性：

名称	类型	描述	默认值
title	string	布局面板（layout panel）的标题文本。	null
region	string	定义布局面板（layout panel）的位置，其值是下列之一：north、south、east、west、center。	
border	boolean	当设置为 true 时，就显示布局面板（layout panel）的边框。	TRUE
split	boolean	当设置为 true 时，就显示拆分栏，用户可以用它改变面板（panel）的尺寸。	FALSE
iconCls	string	在面板（panel）头部显示一个图标的 CSS class。	null
href	string	从远程站点加载数据的 URL 。	null
collapsible	boolean	定义是否显示可折叠按钮。	TRUE
minWidth	number	面板（panel）最小宽度。	10

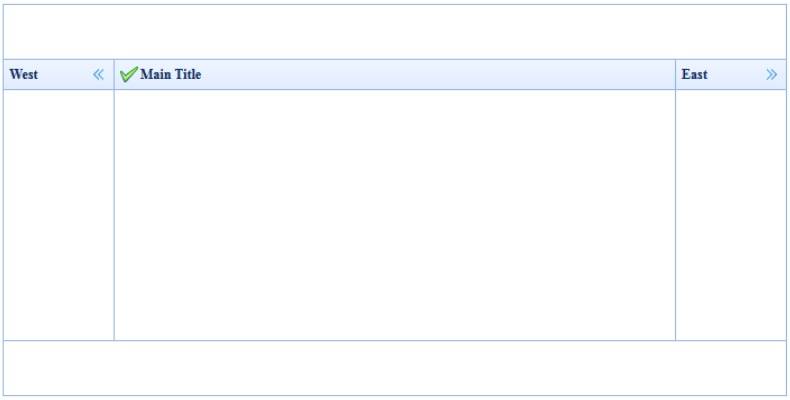
minHeight	number	面板（panel）最小高度。	10
maxWidth	number	面板（panel）最大宽度。	10000
maxHeight	number	面板（panel）最大高度。	10000

方法：

名称	参数	描述
resize	none	设置布局（layout）的尺寸。
panel	region	返回指定的面板（panel），'region' 参数可能的值是：'north'、'south'、'east'、'west'、'center'。
collapse	region	折叠指定的面板（panel），'region' 参数可能的值是：'north'、'south'、'east'、'west'。
expand	region	展开指定的面板（panel），'region' 参数可能的值是：'north'、'south'、'east'、'west'。
add	options	添加一个指定的面板（panel），options 参数一个配置对象，更多细节请参阅标签页面板（tab panel）属性。
remove	region	移除指定的面板（panel），'region' 参数可能的值：'north'、'south'、'east'、'west'。

4.3.1 基础布局

布局外部的容器加 class="easyui-layout"，内部子元素设置 region 属性定位。




```
<div class="easyui-layout" style="width:700px;height:350px;">
    <div data-options="region:'north'" style="height:50px"></div>
    <div data-options="region:'south'" style="height:50px;"></div>
    <div data-options="region:'east'" title="East" style="width:100px;"></div>
    <div data-options="region:'west'" title="West" style="width:100px;"></div>
    <div data-options="region:'center',title:'Main Title',iconCls:'icon-ok'">
    </div>
</div>
```

4.3.2 全屏布局

在 body 上添加 class=”easyui-layout”，布局将填满整个浏览器。

```
<body class="easyui-layout">
<div data-options="region:'north'" style="height:50px"></div>
<div data-options="region:'south'" style="height:50px;"></div>
<div data-options="region:'east'" title="East" style="width:100px;"></div>
<div data-options="region:'west'" title="West" style="width:100px;"></div>
<div data-options="region:'center',title:'Main Title',iconCls:'icon-ok'">
</div>
```

4.3.3 流式布局

将 region 元素的宽度设置成百分比，即可实现宽度随父容器调节。

4.3.4 自适应高度布局

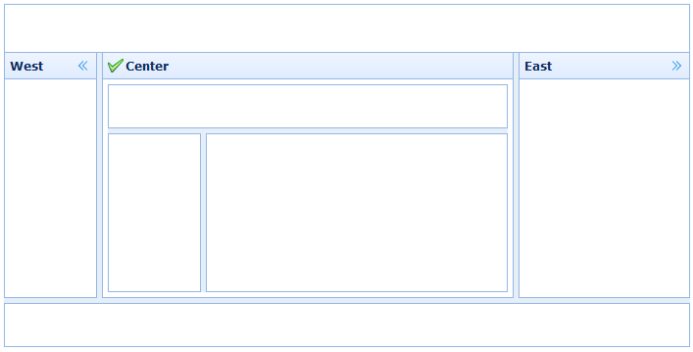
通过 resize 方法重置面板高度。

```
<div style="margin:20px 0;">
    <input type="button" onclick="addItem()" value="添加">
    <input type="button" onclick="removeItem()" value="移除">
</div>
<div id="auto-layout" class="easyui-layout" style="width:700px;height:350px;">
    <div data-options="region:'north'" style="height:50px"></div>
    <div data-options="region:'south'" style="height:50px;"></div>
    <div data-options="region:'west'" title="West" style="width:100px;"></div>
    <div data-options="region:'center',title:'Main Title',iconCls:'icon-ok'">
        <p>Panel Content.</p>
        <p>Panel Content.</p>
        <p>Panel Content.</p>
        <p>Panel Content.</p>
    </div>
</div>
<script type="text/javascript">
    $(function () {
        $('#auto-layout').layout();
        setHeight();
    });

    function addItem() {
        $('#auto-layout').layout('panel', 'center').append('<p>More Panel
```

```
Content.</p>');
        setHeight();
    }

    function removeItem() {
        $('#auto-layout').layout('panel', 'center').find('p:last').remove();
        setHeight();
    }

    function setHeight() {
        var c = $('#auto-layout');
        var p = c.layout('panel', 'center');   //获得中间的面板
        var oldHeight = p.panel('panel').outerHeight();//面板原来的高度
        p.panel('resize', {height: 'auto'});//重置面板高度
        var newHeight = p.panel('panel').outerHeight();//面板现在的高度
        // 重置面板高度=原来的高度+增量
        c.layout('resize', {
            height: (c.height() + newHeight - oldHeight)
        });
    }
</script>
```

4.3.5 嵌套布局

布局支持嵌套：




```
<div class="easyui-layout" style="width:700px;height:350px;">
    <div data-options="region:'north'" style="height:50px"></div>
    <div data-options="region:'south',split:true" style="height:50px;"></div>
    <div data-options="region:'east',split:true" title="East"
style="width:180px;"></div>
    <div data-options="region:'west',split:true" title="West"
style="width:100px;"></div>
    <div data-options="region:'center',iconCls:'icon-ok'" title="Center"
style="padding:5px">
```

```
        <div class="easyui-layout" data-options="fit:true">
            <div data-options="region:'north',split:true" style="height:50px"></div>
            <div data-options="region:'west',split:true" style="width:100px"></div>
            <div data-options="region:'center'"></div>
        </div>
    </div>
</div>
```

4.3.6 添加和移除布局

```
<input onclick="addEast()" value="添加" type="button"/>
<input onclick="removeEast()" value="移除" type="button"/>
<input onclick="collapseNorth()" value="折叠" type="button"/>
<div id="cc" class="easyui-layout" style="width:700px;height:350px;">
    <div data-options="region:'north'" style="height:50px"></div>
    <div data-options="region:'south',split:true" style="height:50px;"></div>
    <div data-options="region:'west',split:true" title="West"
style="width:100px;"></div>
    <div data-options="region:'center',title:'Center'"></div>
</div>
<script type="text/javascript">
    function addEast() {
        var options = {
            region: "east",
            width: 100,
            split: true,
            title: 'new east'
        };
        $('#cc').layout('add', options);
    }
    function removeEast() {
        $('#cc').layout('remove', 'east');
    }
    function collapseNorth(){
        $('#cc').layout('collapse', 'north');
    }
</script>
```

4.3.7 加载动态数据

和 panel 的操作一样，使用 href 加载远程数据：

```
<div data-options="region:'center',title:'Main
Title',iconCls:'icon-ok',href:'dynamic.html'">
```

动态分页数据：

```
<div data-options="region:'center',title:'内容',href:'dynamic.html',split:true,
tools:[{iconCls:'icon-add',handler:function(){
alert('add')}}],footer:'#page'"></div>
```

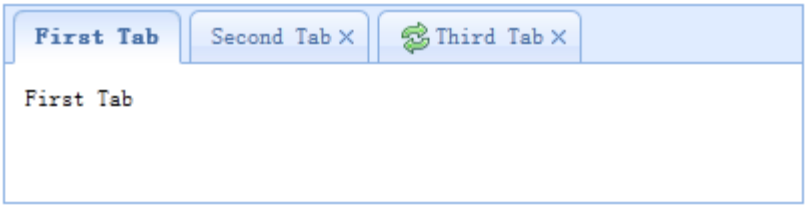
分页连接

```
<div id="page" class="easyui-pagination"
data-options="total:100,pageList:[10,25,50,100],
buttons:[{iconCls:'icon-add',handler:function(){alert('add')}}]"></div>
```

函数：

```
$(function(){
    $('#page').pagination({
        onSelectPage:function(pageNumber, pageSize){
            $(this).pagination('loading');
            alert('pageNumber:'+pageNumber+',pageSize:'+pageSize);
            $(this).pagination('loaded');
            $("#js-layout").layout("panel",
"center").panel("refresh","dynamic2.html")
        }
    });

})
```

4.4 选项卡 tabs



选项卡依赖于 panel 和 linkbutton。

属性：

名称	类型	描述	默认值
width	number	标签页（Tabs）容器的宽度。	auto
height	number	标签页（Tabs）容器的高度。	auto
plain	boolean	当设置为 true 时，就不用背景容器图片来呈现 tab 条。	FALSE
fit	boolean	当设置为 true 时，就设置标签页（Tabs）	FALSE

<table>
<tr><td></td><td></td><td>容器的尺寸以适应它的父容器。</td><td></td></tr>
<tr><td>border</td><td>boolean</td><td>当设置为 true 时，就显示标签页（Tabs）容器边框。</td><td>TRUE</td></tr>
<tr><td>scrollIncrement</td><td>number</td><td>每按一次 tab 滚动按钮，滚动的像素数。</td><td>100</td></tr>
<tr><td>scrollDuration</td><td>number</td><td>每一个滚动动画应该持续的毫秒数。</td><td>400</td></tr>
<tr><td rowspan="11">tools</td><td rowspan="11">array,selector</td><td>放置在头部的左侧或右侧的工具栏，可能的值：</td><td rowspan="11">null</td></tr>
<tr><td>1、数组，指示工具组，每个工具选项都和链接按钮（Linkbutton）一样。</td></tr>
<tr><td>2、选择器，指示包含工具的 <div>。</td></tr>
<tr><td>代码实例：</td></tr>
<tr><td>通过数组定义工具。</td></tr>
<tr><td><pre>
$('#tt').tabs({
    tools:[{
        iconCls:'icon-add',
        handler:function(){
            alert('add')
        }
    },{
        iconCls:'icon-save',
        handler:function(){
            alert('save')
        }
    }]
});
</pre></td></tr>
<tr><td>通过已有的 DOM 容器定义工具。</td></tr>
<tr><td><pre>
$('#tt').tabs({
    tools:'#tab-tools'
});
<div id="tab-tools">
    <a href="#" class="easyui-linkbutton"
plain="true" iconCls="icon-add">
    <a href="#" class="easyui-linkbutton"
plain="true" iconCls="icon-save">
    </div>
</pre></td></tr>
<tr><td>toolPosition</td><td>string</td><td>工具栏位置。可能的值：'left'、'right'。该属性自版本 1.3.2 起可用。</td><td>right</td></tr>
<tr><td>tabPosition</td><td>string</td><td>标签页（tab）位置。可能的值：'top'、'bottom'、'left'、'right'。该属性自版本 1.3.2 起可用。</td><td>top</td></tr>
<tr><td>headerWidth</td><td>number</td><td>标签页(tab)头部宽度,只有当 tabPosition 设置为 'left' 或 'right' 时才有效。该属性自版本 1.3.2 起可用。</td><td>150</td></tr>
<tr><td>tabWidth</td><td>number</td><td>tab 条的宽度。该属性自版本 1.3.4 起可</td><td>auto</td></tr>
</table>

		用。	
tabHeight	number	tab 条的高度。该属性自版本 1.3.4 起可用。	27
selected	number	初始化选定的标签页索引。该属性自版本 1.3.5 起可用。	0
showHeader	boolean	当设置为 true 时，显示标签页（tab）头部。该属性自版本 1.3.5 起可用。	TRUE

事件：

名称	参数	描述
onLoad	panel	当一个 ajax 标签页面板（tab panel）完成加载远程数据时触发。
onSelect	title,index	当用户选择一个标签页面板（tab panel）时触发。
onUnselect	title,index	当用户未选择一个标签页面板（tab panel）时触发。该事件自版本 1.3.5 起可用。
onBeforeClose	title,index	当一个标签页面板（tab panel）被关闭前触发，返回 false 就取消关闭动作。下面的实例演示如何在关闭标签页面板（tab panel）前显示确认对话框。
onClose	title,index	当用户关闭一个标签页面板（tab panel）时触发。
onAdd	title,index	当一个新的标签页面板（tab panel）被添加时触发。

```
// using the async confirm dialog
$('#tt').tabs({
    onBeforeClose: function(title,index){
        var target = this;
        $.messager.confirm('Confirm','Are you sure you want to close '+title,function(r){
            if (r){
                var opts = $(target).tabs('options');
                var bc = opts.onBeforeClose;
                opts.onBeforeClose = function(){};  // allowed to close now
                $(target).tabs('close',index);
                opts.onBeforeClose = bc;  // restore the event function
            }
        });
        return false;  // prevent from closing
    }
});
```

onUpdate	title,index	当一个标签页面板（tab panel）被更新时触发。
onContextMenu	e, title,index	当一个标签页面板（tab panel）被右键点击时触发。

方法：

名称	参数	描述
options	none	返回标签页（tabs）选项（options）。
tabs	none	返回全部的标签页面板（tab panel）。
resize	none	调整标签页（tabs）容器的尺寸并做布局。

add	options	添加一个新的标签页面板（tab panel），options 参数是一个配置对象，更多详细信息请参见标签页面板（tab panel）属性。
		当添加一个新的标签页面板（tab panel）时，它将被选中。
		如需添加一个未选中的标签页面板（tab panel），请记得设置 'selected' 属性为 false。
		`// add a unselected tab panel`
		`$('#tt').tabs('add',{`
		`  title:'new tab',`
		`  selected:false`
		`  //...`
		`});`

close	which	关闭一个标签页面板（tab panel），'which' 参数可以是要被关闭的标签页面板（tab panel）的标题（title）或索引（index）。
getTab	which	获取指定的标签页面板（tab panel），'which' 参数可以是标签页面板（tab panel）的标题（title）或索引（index）。
getTabIndex	tab	获取指定的标签页面板（tab panel）索引。

getSelected	none	获取选中的标签页面板（tab panel）。下面的实例演示如何获取选中的标签页面板（tab panel）的索引。
		`var tab = $('#tt').tabs('getSelected');`
		`var index = $('#tt').tabs('getTabIndex',tab);`
		`alert(index);`

select	which	选择一个标签页面板（tab panel），'which' 参数可以是标签页面板（tab panel）的标题（title）或索引（index）。
unselect	which	选择一个标签页面板（tab panel），'which' 参数可以是标签页面板（tab panel）的标题（title）或索引（index）。该方法自版本 1.3.5 起可用。
showHeader	none	显示标签页（tabs）头部。该方法自版本 1.3.5 起可用。
hideHeader	none	隐藏标签页（tabs）头部。该方法自版本 1.3.5 起可用。
exists	which	指示指定的面板是否已存在，'which' 参数可以是标签页面板（tab panel）的标题（title）或索引（index）。

<table>
<tr><td rowspan="8">update</td><td rowspan="8">param</td><td>更新指定的标签页面板（tab panel），param 参数包含两个属性：</td></tr>
<tr><td>tab：被更新的标签页面板（tab panel）。</td></tr>
<tr><td>options：面板（panel）的选项（options）。</td></tr>
<tr><td>代码实例：</td></tr>
<tr><td><pre>
// update the selected panel with new title and content
var tab = $('#tt').tabs('getSelected'); // get selected panel
$('#tt').tabs('update', {
    tab: tab,
    options: {
        title: 'New Title',
        href: 'get_content.php' // the new content URL
    }
});

// call 'refresh' method for tab panel to update its content
var tab = $('#tt').tabs('getSelected'); // get selected panel
tab.panel('refresh', 'get_content.php');
</pre></td></tr>
<tr><td rowspan="2">enableTab</td><td rowspan="2">which</td><td>启用指定的标签页面板（tab panel），'which' 参数可以是标签页面板（tab panel）的标题（title）或索引（index）。该方法自版本 1.3 起可用。</td></tr>
<tr><td>代码实例：</td></tr>
<tr><td><pre>
$('#tt').tabs('enableTab', 1); // enable the second tab panel
$('#tt').tabs('enableTab', 'Tab2');   enable the tab panel that has 'Tab2' title
</pre></td></tr>
<tr><td rowspan="2">disableTab</td><td rowspan="2">which</td><td>禁用指定的标签页面板（tab panel），'which' 参数可以是标签页面板（tab panel）的标题（title）或索引（index）。该方法自版本 1.3 起可用。</td></tr>
<tr><td>代码实例：</td></tr>
<tr><td><pre>
$('#tt').tabs('disableTab', 1); // disable the second tab panel.
</pre></td></tr>
<tr><td rowspan="3">scrollBy</td><td rowspan="3">deltaX</td><td>通过指定的像素数滚动标签页（tab）头部，负值表示滚动到右边，正值表示滚动到左边。该方法自版本 1.3.2 起可用。</td></tr>
<tr><td>代码实例：</td></tr>
<tr><td><pre>
// scroll the tab header to left
$('#tt').tabs('scroll', 10);
</pre></td></tr>
</table>

标签面板属性：

名称	类型	描述	默认值
id	string	标签页面板（tab panel）的 id 属性。	null
title	string	标签页面板（tab panel）的标题文字。	
content	string	标签页面板（tab panel）的内容。	
href	string	加载远程内容来填充标签页面板（tab panel）的 URL。	null
cache	boolean	当设置为 true 时，在设定了有效的 href 特性时缓存这个标签页面板（tab panel）。	TRUE

iconCls	string	显示在标签页面板(tab panel)标题上的图标的 CSS class。	null
width	number	标签页面板（tab panel）的宽度。	auto
height	number	标签页面板（tab panel）的高度。	auto
collapsible	boolean	当设置为 true 时，允许标签页面板（tab panel）可折叠。	FALSE

一些附加的属性。

名称	类型	描述	默认值
closable	boolean	当设置为 true 时，标签页面板（tab panel）将显示一个关闭按钮，点击它就能关闭这个标签页面板（tab panel）。	FALSE
selected	boolean	当设置为 true 时，标签页面板（tab panel）将被选中。	FALSE

4.4.1 基础选项卡

父容器上设置 class="easyui-tabs"，子 div 节点会被处理成选项卡：

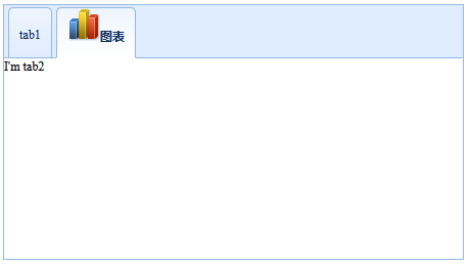

```
<div class="easyui-tabs" style="width:700px;height:250px">
    <div title="tab1">
        <h1>TAB1</h1>
    </div>
    <div title="tab2">
        I'm tab2
    </div>
    <div title="Help" data-options="iconCls:'icon-help',closable:true">
        This is the help content.
    </div>
</div>
```

4.4.2 流式选项卡

将选项卡容器的 width 设置成百分比。选项卡可以固定宽度，使用 tabWidth 设置像素。

4.4.3 图片选项卡

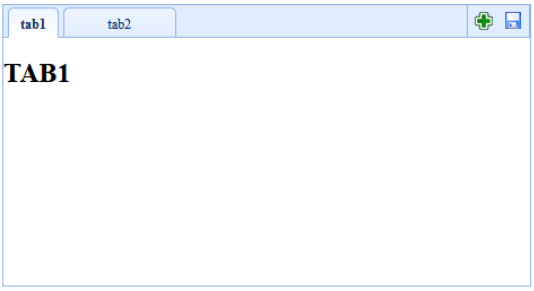
通过 title 属性，可以设置选项卡显示的图标或者样式。title 的值可以是一段 html




```
<!--图片选项卡-->
<div class="easyui-tabs" style="width:60%;height:250px" data-options="tabHeight:50">
    <div title="tab1">
        <h1>TAB1</h1>
    </div>
    <div title="<img src='/easyui/themes/icons/large_chart.png'/>图表">
        I'm tab2
    </div>
</div>
```

4.4.4 选项卡工具组

使用 tools 属性指定工具组，工具组是添加到选项卡面板上的：



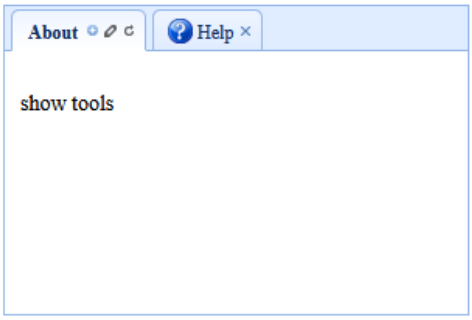

```
<div class="easyui-tabs" style="width:60%;height:250px" data-options="tools:[{
      iconCls:'icon-add',
      handler:function(){
         alert('add')
      }
   },{
      iconCls:'icon-save',
      handler:function(){
         alert('save')
      }
   }]">
```

tools 也可以是一个选择器：


```
<div class="easyui-tabs" style="width:60%;height:250px"
data-options="tools:'#tab-tools'">
    <div title="tab1"><h1>TAB1</h1></div>
    <div title="tab2" data-options="tabWidth:100">I'm tab2</div>
</div>
<!--选项卡工具组-->
<div id="tab-tools">
    <input class="easyui-searchbox"  style="width:300px"/>
</div>
```

4.4.5 选项卡工具条

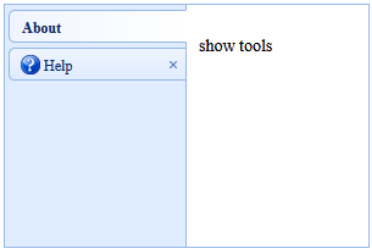
工具条是在单独一个选项卡上设置的，在选项卡上设置 tools，语法与工具组一致：




```
<div class="easyui-tabs" style="width:300px;height:200px">
    <div title="About" data-options="tools:'#p-tools'">
        <p style="font-size:14px">show tools</p>
    </div>
    <div title="Help" data-options="iconCls:'icon-help',closable:true">
        This is the help content.
    </div>
</div>
<div id="p-tools">
    <a href="javascript:void(0)" class="icon-mini-add" onclick="alert('add')"></a>
    <a href="javascript:void(0)" class="icon-mini-edit" onclick="alert('edit')"></a>
</div>
```

4.4.6 选项卡位置

通过 tabPosition 属性可以设置选项卡位置，有上(top)下(bottom)左(left)右(right)四个位置:




```
<div class="easyui-tabs" style="width:300px;height:200px"
data-options="tabPosition:'left'">
    <div title="About" style="padding:10px">
        <p style="font-size:14px">show tools</p>
    </div>
    <div title="Help" data-options="iconCls:'icon-help',closable:true"
style="padding:10px">
        This is the help content.
    </div>
</div>
```

4.4.7 动态加载内容

给某个选项卡加上 href 属性即可实现动态加载，注意选项卡必须是选中状态时，才会加载动态内容。

4.4.8 javascript 操作选项卡

```
<input type="text" name="tabName" id="tabName">
<input type="button" value="添加" onclick="addTab()"/>
<input type="button" value="移除" onclick="removeTab()"/>
<div id="js-tab" style="width:60%;height:250px">
    <div title="tab1"></div>
</div>

<script type="text/javascript">
    $(function () {
        $("#js-tab").tabs({})//初始化选项卡
    });
    function addTab() {
```

```
        var name = $("#tabName").val();
        //如果选项卡已存在就选中它
        if ($("#js-tab").tabs("exists", name)) {
            $("#js-tab").tabs("select", name);
        } else {//如果选项卡不存在就新增一个
            $("#js-tab").tabs("add", {
                title: name,
                selected: true,
                closable:true
            });
        }
    }
    function removeTab() {
        //移除一个选项卡
        var name = $("#tabName").val();
        $("#js-tab").tabs("close", name);
    }
</script>
```

5.菜单和按钮

5.1 菜单 menu

menu 是一个基础组件，通常用于上下文菜单、元素的右键菜单，能用于导航和执行命令。
创建菜单 class=”easyui-menu”

属性：

菜单项（menu item）代表一个显示在菜单中的单独的项目。它包含下列属性：

名称	类型	描述	默认值
id	string	菜单项（menu item）的 id 属性。	
text	string	项目文本。	
iconCls	string	在项目左边显示一个 16x16 图标的 CSS class。	
href	string	当点击菜单项（menu item）时设置页面位置。	
disabled	boolean	定义是否禁用菜单项（menu item）。	FALSE
onclick	function	当点击菜单项（menu item）时被调用的函数。	

菜单属性：

名称	类型	描述	默认值

zIndex	number	菜单（Menu）的 z-index 样式，从它开始增加。	110000
left	number	菜单（Menu）的左边位置。	0
top	number	菜单（Menu）的顶部位置。	0
minWidth	number	菜单（Menu）的最小宽度。该属性自版本 1.3.2 起可用。	120
hideOnUnhover	boolean	如果设置为 true，当鼠标离开它时自动隐藏菜单（menu）。该属性自版本 1.3.5 起可用。	TRUE

菜单事件：

名称	参数	描述
onShow	none	当菜单（menu）显示之后触发。
onHide	none	当菜单（menu）隐藏之后触发。

onClick	item	当点击菜单项（menu item）时触发。下面的实例演示如何处理所有菜单项点击：
		`<div class="easyui-menu" data-options="onClick:menuHandler" style="width:120px;">`
		`  <div data-options="name:'new'">New</div>`
		`  <div data-options="name:'save',iconCls:'icon-save'">Save</div>`
		`  <div data-options="name:'print',iconCls:'icon-print'">Print</div>`
		`  <div class="menu-sep"></div>`
		`  <div data-options="name:'exit'">Exit</div>`
		`</div>`
		`<script type="text/javascript">`
		`  function menuHandler(item){`
		`  alert(item.name)`
		`  }`
		`</script>`

名称	参数	描述
options	none	返回选项（options）对象。
show	pos	在指定的位置显示菜单（menu）。 pos 参数有两个属性： left：新的左边位置。 top：新的顶部位置。
hide	none	隐藏菜单（menu）。
destroy	none	销毁菜单（menu）。

getItem	itemEl	获取包含 'target' 属性（指示项目 DOM 元素）的菜单项（menu item）属性。下面的实例演示如何通过 id 获取指定的项目：
		`<div id="mm" class="easyui-menu" style="width:120px">`
		`  <div>New</div>`

		Open
		 Save
		
		var itemEl = $('#m-open')[0]; // the menu item element
		var item = $('#mm').menu('getItem', itemEl);
		console.log(item);

setText	param	给指定的菜单项（menu item）设置文本。'param' 参数包含两个属性：
		target：DOM 对象，被设定的菜单项（menu item）。
		text：string，新的文本值。
		代码实例：
		var item = $('#mm').menu('findItem', 'Save');
		$('#mm').menu('setText', {
		target: item.target,
		text: 'Saving'
		});

setIcon	param	给指定的菜单项（menu item）设置图标。'param' 参数包含两个属性： target：DOM 对象，即菜单项（menu item）。 iconCls：新图标的 CSS class。 代码实例： $('#mm').menu('setIcon', { target: $('#m-open')[0], iconCls: 'icon-closed' });

findItem	text	找到指定的菜单项（menu item），返回对象与 getItem 方法相同。
		代码实例：
		// find 'Open' item and disable it
		var item = $('#mm').menu('findItem', 'Open');
		$('#mm').menu('disableItem', item.target);

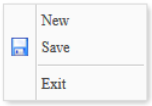
appendItem	options	追加一个新的菜单项（menu item），'param' 参数指示新的项目属性。默认情况下，新增的项目将作为顶级菜单项（menu item）。如需追加一个子菜单项，需设置 'parent' 属性，用来指示已经有子项目的父项目元素。
		代码实例：
		// append a top menu item
		$('#mm').menu('appendItem', {
		text: 'New Item',
		iconCls: 'icon-ok',
		onclick: function(){alert('New Item')}
		});
		// append a menu separator

		$('#mm').menu('appendItem', {
		separator: true
		});
		// append a sub menu item
		var item = $('#mm').menu('findItem', 'Open'); // find 'Open' item
		$('#mm').menu('appendItem', {
		parent: item.target, // the parent item element
		text: 'Open Excel',
		iconCls: 'icon-excel',
		onclick: function(){alert('Open Excel')}
		});

removeItem	itemEl	移除指定的菜单项（menu item）。
enableItem	itemEl	启用菜单项（menu item）。
disableItem	itemEl	禁用菜单项（menu item）。

5.1.1 基础菜单

创建一个简单的右键菜单




```
<div id="mm" class="easyui-menu" data-options="onClick:menuHandler"
style="width:120px;">
    <div data-options="name:'new'">New</div>
    <div data-options="name:'save',iconCls:'icon-save'">Save</div>
    <div class="menu-sep"></div>
    <div data-options="name:'exit'">Exit</div>
</div>
<script>
    function menuHandler(item){
        alert(item.name);
    }
    $(function(){
        $(document).bind('contextmenu',function(e){
            e.preventDefault();//阻止默认事件
            $('#mm').menu('show', {
                left: e.pageX,
                top: e.pageY
            });
        });
```

```
    });
</script>
```

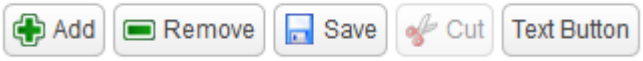
5.1.2 自定义菜单

```
<div id="mm" class="easyui-menu" style="width:120px;">
    <div data-options="iconCls:'icon-add'">New</div>
    <div>
        <span>Open</span>
        <div style="text-align:left;padding:10px">
            <div style="font-weight:bold;font-size:16px">Select your Language:</div>
            <ul style="margin:0;padding:0 0 0 40px">
                <li><a href="javascript:void(0)">Java</a></li>
                <li><a href="javascript:void(0)">Basic</a></li>
                <li>
                    <span>Other</span>
                    <input>
                </li>
            </ul>
        </div>
    </div>
    <div>
        <span>Sub</span>
        <div>
            <div><span>sub1</span></div>
            <div>
                <span>sub2</span>
                <div>
                    <div><span>sub3</span></div>
                    <div><span>sub4</span></div>
                </div>
            </div>
        </div>
    </div>
</div>
<script>
    $(function () {
        $(document).bind('contextmenu', function (e) {
            e.preventDefault();
            $('#mm').menu('show', {
                left: e.pageX,
                top: e.pageY
            });
```

```
        });
    });
</script>
```

5.2 链接按钮 linkbutton

通常情况下，使用<button> 元素来创建按钮，而链接按钮（Link Button）则是使用 <a> 元素来创建的。所以实际上一个链接按钮（Link Button）就是一个显示为按钮样式的<a> 元素。它可显示图标和文本，或者仅仅显示图标和文本中的一个。按钮宽度可动态收缩/扩展以适应其文本标签。



属性：

名称	类型	描述	默认值
id	string	该组件的 id 属性。	null
disabled	boolean	如果设置为 true，则禁用按钮。	FALSE
toggle	boolean	如果设置为 true，则允许用户切换按钮的选中状态。该属性自版本 1.3.3 起可用。	FALSE
selected	boolean	定义按钮状态是否已选择。该属性自版本 1.3.3 起可用。	FALSE
group	string	指示按钮所属的分组名称。该属性自版本 1.3.3 起可用。类似于 radio 的效果，实现按钮单选，与 toggle 结合使用	null
plain	boolean	如果设置为 true，则显示一个简单的效果(没有边框)。	FALSE
text	string	按钮文本。	''
iconCls	string	在左边显示一个 16x16 图标的 CSS class。	null
iconAlign	string	按钮图标的位置。可能的值：'left'、'right'。该属性自版本 1.3.2 起可用。	left

方法：

名称	参数	描述
options	none	返回选项（options）属性（property）。
disable	none	禁用按钮。 代码实例： `$('#btn').linkbutton('disable');`
enable	none	启用按钮。 代码实例： `$('#btn').linkbutton('enable');`
select	none	选中按钮。该方法自版本 1.3.3 起可用。
unselect	none	未选中按钮。该方法自版本 1.3.3 起可用。

5.2.1 基础链接按钮

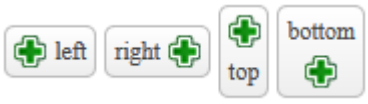

```
<!--纯文字链接按钮-->
<a href="#" class="easyui-linkbutton">Text Button</a>
<!--带图标的链接按钮-->
<a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-add'">icon</a>
<!--图标右对齐-->
<a href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-remove',iconAlign:'right'">iconAlign</a>
<!--禁用-->
<a href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-cut',disabled:true">disabled</a>
<!--选中-->
<a href="#" class="easyui-linkbutton" data-options="selected:true">selected</a>
<!--简单样式-->
<a href="#" class="easyui-linkbutton" data-options="plain:true">plain</a>
```

5.2.2 流式链接按钮

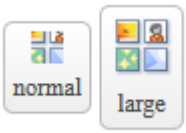
通过 style 将按钮的宽度 width 设置成百分比。

```
<a href="#" class="easyui-linkbutton" style="width:10%">fluid</a>
```

5.2.3 图标链接按钮




```
<!--默认图标在左边 left,通过 iconAlign 设置上下左右-->
<a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-add'">left</a>
<a href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-add',iconAlign:'right'">right</a>
<a href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-add',iconAlign:'top'">top</a>
<a href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-add',iconAlign:'bottom'">bottom</a>
```



大图标按钮，通过 size 属性指定大小

```
<a href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-large-clipart',iconAlign:'top'">normal</a>
<a href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-large-clipart',size:'large',iconAlign:'top'">
```

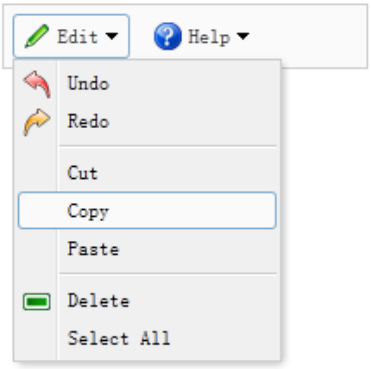
5.2.4 分组按钮

通过 group 将多个按钮分组，作用和 radio 单选类似，一个 group 中每次只能选中一个按钮。和 toggle 结合使用，切换按钮的选中状态。


```
<a href="#" class="easyui-linkbutton" data-options="toggle:true,group:'g1'">Button
1</a>
<a href="#" class="easyui-linkbutton"
data-options="toggle:true,group:'g1',selected:true">Button 2</a>
<a href="#" class="easyui-linkbutton" data-options="toggle:true,group:'g1'">Button
3</a>
```

5.3 菜单按钮 menubutton

菜单按钮（menubutton）是下拉菜单的一部分。它与链接按钮（linkbutton）及菜单（menu）有关。显示链接按钮（linkbutton），隐藏菜单（menu）。当用户点击或移动鼠标到链接按钮（linkbutton）上时，将显示菜单（menu）以允许用户点击菜单。



属性：

该属性扩展自链接按钮（linkbutton），下面是菜单按钮（menubutton）增加的属性。

名称	类型	描述	默认值

plain	boolean	如果设置为 true，则显示一个简单的效果。	TRUE
menu	string	用于创建对应菜单（menu）的选择器。	null
duration	number	当悬停在按钮上时，以毫秒为单位定义的，显示菜单（menu）的持续时间。	100

方法：

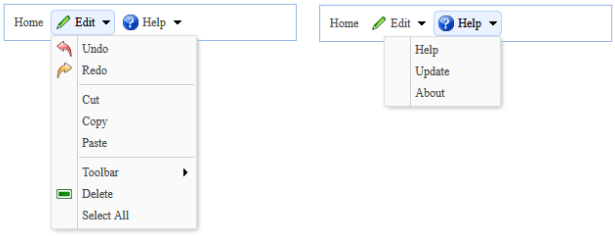
该方法继承自链接按钮（linkbutton），下面是菜单按钮（menubutton）增加的或重写的方法。

名称	参数	描述
options	none	返回选项（options）对象。
disable	none	禁用菜单按钮（menubutton）。
enable	none	启用菜单按钮（menubutton）。
destroy	none	销毁菜单按钮（menubutton）。

5.3.1 基础菜单按钮

菜单按钮的 class=”easyui-menubutton”，通过 menu 属性指定菜单,menu 的值是菜单元素的选择器。

菜单默认是左对齐，通过 menuAlign 可以实现右对齐：

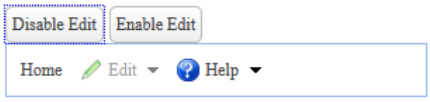



```
<div class="easyui-panel" style="padding:5px;width:300px">
    <a href="#" class="easyui-linkbutton" data-options="plain:true">Home</a>
    <a href="#" class="easyui-menubutton"
data-options="menu:'#mm1',iconCls:'icon-edit'">Edit</a>
    <a href="#" class="easyui-menubutton"
data-options="menu:'#mm2',iconCls:'icon-help',menuAlign: 'right'">Help</a>
</div>
<div id="mm1" style="width:150px;">
    <div data-options="iconCls:'icon-undo'">Undo</div>
    <div data-options="iconCls:'icon-redo'">Redo</div>
    <!--分割线-->
    <div class="menu-sep"></div>
    <div>Cut</div>
    <div>Copy</div>
    <div>Paste</div>
    <div class="menu-sep"></div>
    <div>
```

```
        <span>Toolbar</span>
        <div>
            <div>Address</div>
            <div>Link</div>
            <div class="menu-sep"></div>
            <div>New Toolbar...</div>
        </div>
    </div>
    <div data-options="iconCls:'icon-remove'">Delete</div>
    <div>Select All</div>
</div>
<div id="mm2" style="width:100px;">
    <div>Help</div>
    <div>Update</div>
    <div>About</div>
</div>
```

5.3.2 菜单按钮操作

禁用和启用菜单按钮



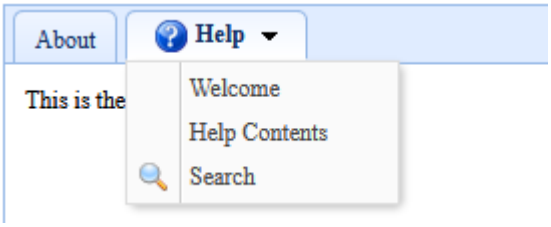

```
<a href="#" class="easyui-linkbutton"
onclick="$('#btn-edit').menubutton('disable')">Disable Edit</a>
<a href="#" class="easyui-linkbutton"
onclick="$('#btn-edit').menubutton('enable')">Enable Edit</a>
<div class="easyui-panel" style="padding:5px;width:300px">
    <a href="#" class="easyui-linkbutton" data-options="plain:true">Home</a>
    <a href="#" id="btn-edit" class="easyui-menubutton"
data-options="menu:'#mm1',iconCls:'icon-edit'">Edit</a>
    <a href="#" class="easyui-menubutton"
data-options="menu:'#mm2',iconCls:'icon-help',menuAlign: 'right'">Help</a>
</div>
```

菜单的操作和菜单 menu 一样。

```
<div id="mm2" style="width:100px;" data-options="onClick:menuHandler">
    <div data-options="name:'help'">Help</div>
    <div data-options="name:'update'">Update</div>
    <div data-options="name:'about'">About</div>
</div>
<script type="text/javascript">
    function menuHandler(item){
        alert(item.name);
```

```
    }
</script>
```

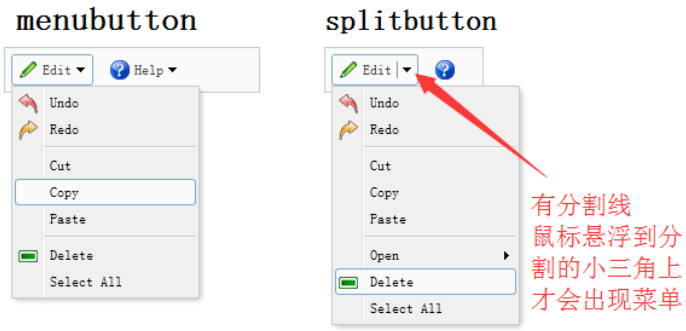
5.3.3 带下拉菜单的选项卡




```
<div id="tt" style="width:700px;height:250px" class="easyui-tabs">
    <div title="About" style="padding:10px">
        <h1>About</h1>
    </div>
    <div title="Help" style="padding:10px" id="menu-tab">
        This is the help content.
    </div>
</div>
<!--声明一个菜单-->
<div id="mm" class="easyui-menu" data-options="onClick:menuHandler">
    <div data-options="name:'welcome'">Welcome</div>
    <div data-options="name:'content'">Help Contents</div>
    <div data-options="name:'serach',iconCls:'icon-search'">Search</div>
</div>

<script>
    $(function () {
        var p = $('#menu-tab');
        var mb = p.panel('options').tab.find('a.tabs-inner');
        mb.menubutton({
            menu: '#mm',
            iconCls: 'icon-help'

        }).click(function () {//选项卡的点击事件
            $('#tt').tabs('select', 1);
        })
    });

    //下拉菜单选项的点击事件
    function menuHandler(item) {
        alert(item.name);
```

```
    }
</script>
```

5.4 分割按钮 splitbutton



与菜单按钮（menubutton）相似，分割按钮（splitbutton）也与链接按钮（linkbutton）及菜单（menu）有关。与菜单按钮（menubutton）不同的是，分割按钮（splitbutton）被分割为两部分。当移动鼠标到分割按钮（splitbutton）上时，将显示一条分割线。只有当移动鼠标在分割按钮的右侧部分时才显示菜单（menu）。

分割按钮的用法与菜单按钮一样。class=”easyui-splitbutton”。其它操作参照 menubutton 的使用。

5.4.1 基础分割按钮

```
<div class="easyui-panel" style="padding:5px;width:300px">
    <a href="#" class="easyui-linkbutton" data-options="plain:true">Home</a>
    <a href="#" class="easyui-splitbutton"
data-options="menu:'#mm1',iconCls:'icon-edit'">Edit</a>
    <a href="#" class="easyui-splitbutton"
data-options="menu:'#mm2',iconCls:'icon-help',menuAlign: 'right'">Help</a>
</div>
<div id="mm1" style="width:150px;">
    <div data-options="iconCls:'icon-undo'">Undo</div>
    <div data-options="iconCls:'icon-redo'">Redo</div>
    <!--分割线-->
    <div class="menu-sep"></div>
    <div>Cut</div>
    <div>Copy</div>
    <div>Paste</div>
    <div class="menu-sep"></div>
    <div>
        <span>Toolbar</span>
```

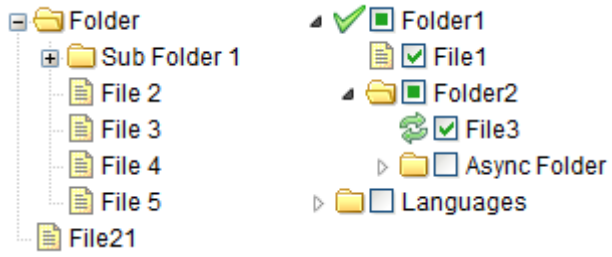
```
        <div>
            <div>Address</div>
            <div>Link</div>
            <div class="menu-sep"></div>
            <div>New Toolbar...</div>
        </div>
    </div>
    <div data-options="iconCls:'icon-remove'">Delete</div>
    <div>Select All</div>
</div>
<div id="mm2" style="width:100px;">
    <div>Help</div>
    <div>Update</div>
    <div>About</div>
</div>
```

6.树形控件

6.1 tree

树（tree）在网页中以树形结构显示分层数据。它向用户提供展开、折叠、拖拽、编辑和异步加载功能。

tree 插件依赖于 draggable 和 droppable。



属性：

名称	类型	描述	默认值
url	string	获取远程数据的 URL 。	null
method	string	检索数据的 http 方法（method）。	post
animate	boolean	定义当节点展开折叠时是否显示动画效果。	FALSE
checkbox	boolean	定义是否在每个节点前边显示复选框。	FALSE
cascadeCheck	boolean	定义是否级联检查。	TRUE
onlyLeafCheck	boolean	定义是否只在叶节点前显示复选框。	FALSE
lines	boolean	定义是否显示树线条。	FALSE
dnd	boolean	定义是否启用拖放。	FALSE

<table>
<tr><td>data</td><td>array</td><td>要加载的节点数据。
<pre>
$('#tt').tree({
    data: [{
        text: 'Item1',
        state: 'closed',
        children: [{
            text: 'Item11'
        },{
            text: 'Item12'
        }]
    },{
        text: 'Item2'
    }]
});
</pre></td><td>null</td></tr>
<tr><td>formatter</td><td>function(node)</td><td>定义如何呈现节点文本。
代码实例：
<pre>
$('#tt').tree({
    formatter:function(node){
        return node.text;
    }
});
</pre></td><td></td></tr>
<tr><td>loader</td><td>function(param,success,error)</td><td>定义如何从远程服务器加载数据。返回 false 则取消该动作。该函数有下列参数：
param：要传递到远程服务器的参数对象。
success(data)：当检索数据成功时调用的回调函数。
error()：当检索数据失败时调用的回调函数。</td><td>json loader</td></tr>
<tr><td>loadFilter</td><td>function(data,parent)</td><td>返回要显示的过滤数据。返回数据时以标准树格式返回的。该函数有下列参数：
data：要加载的原始数据。
parent：DOM 对象，表示父节点。
<pre>
loadFilter: function (data) {
    for (var i = 0; i < data.length; i++) {
        if (data[i].text == "Item2") {
            data[i].iconCls="icon-add";
        }
    }
    return data;
}
</pre></td><td></td></tr>
</table>

事件：

很多事件的回调函数需要 'node' 参数，它包括下列属性：

id：绑定到节点的标识值。

text：要显示的文本。

iconCls：用来显示图标的 css class。
checked：节点是否被选中。
state：节点状态，'open' 或 'closed'。
attributes：绑定到节点的自定义属性。
target：目标的 DOM 对象。

名称	参数	描述

onClick	node	当用户点击一个节点时触发。代码实例：
		`$('#tt').tree({`
		`  onClick: function(node){`
		`  alert(node.text);  // alert node text property when clicked`
		`  }`
		`});`

onDblClick	node	当用户双击一个节点时触发。
onBeforeLoad	node, param	当加载数据的请求发出前触发，返回 false 则取消加载动作。
onLoadSuccess	node, data	当数据加载成功时触发。
onLoadError	arguments	当数据加载失败时触发，arguments 参数与 jQuery.ajax 的 'error' 函数一样。
onBeforeExpand	node	节点展开前触发，返回 false 则取消展开动作。
onExpand	node	当节点展开时触发。
onBeforeCollapse	node	节点折叠前触发，返回 false 则取消折叠动作。
onCollapse	node	当节点折叠时触发。
onBeforeCheck	node, checked	当用户点击复选框前触发，返回 false 则取消该选中动作。该事件自版本 1.3.1 起可用。
onCheck	node, checked	当用户点击复选框时触发。
onBeforeSelect	node	节点被选中前触发，返回 false 则取消选择动作。
onSelect	node	当节点被选中时触发。

onContextMenu	e, node	当右键点击节点时触发。代码实例：
		`// right click node and then display the context menu`
		`$('#tt').tree({`
		`  onContextMenu: function(e, node){`
		`  e.preventDefault();`
		`  // select the node`
		`  $('#tt').tree('select', node.target);`
		`  // display context menu`
		`  $('#mm').menu('show', {`
		`  left: e.pageX,`
		`  top: e.pageY`
		`  });`

		} }); <div id="mm" class="easyui-menu" style="width:120px;"> <div onclick="append()" data-options="iconCls:'icon-add'">Append <div onclick="remove()" data-options="iconCls:'icon-remove'">Remove
onBeforeDrag	node	当节点的拖拽开始时触发，返回 false 则禁止拖拽。该事件自版本 1.3.2 起可用。
onStartDrag	node	当开始拖拽节点时触发。该事件自版本 1.3.2 起可用。
onStopDrag	node	当停止拖拽节点时触发。该事件自版本 1.3.2 起可用。
onDragEnter	target, source	当节点被拖拽进入某个允许放置的目标节点时触发，返回 false 则禁止放置。
		target：被放置的目标节点元素。
		source：被拖拽的源节点。
		该事件自版本 1.3.2 起可用。
onDragOver	target, source	当节点被拖拽到允许放置的目标节点上时触发，返回 false 则禁止放置。
		target：被放置的目标节点元素。
		source：被拖拽的源节点。
		该事件自版本 1.3.2 起可用。
onDragLeave	target, source	当节点被拖拽离开允许放置的目标节点时触发。
		target：被放置的目标节点元素。
		source：被拖拽的源节点。
		该事件自版本 1.3.2 起可用。
onBeforeDrop	target,source,point	节点被放置之前触发，返回 false 则禁止放置。
		target：DOM 对象，放置的目标节点。
		source：源节点。
		point：表示放置操作，可能的值是：'append'、'top' 或 'bottom'。
		该事件自版本 1.3.2 起可用。
onDrop	target,source,point	当节点被放置时触发。 target：DOM 对象，放置的目标节点。
		source：源节点。
		point：表示放置操作，可能的值是：'append'、'top' 或 'bottom'。
onBeforeEdit	node	编辑节点前触发。
onAfterEdit	node	编辑节点后触发。
onCancelEdit	node	当取消编辑动作时触发。

方法：

名称	参数	描述
options	none	返回树的选项（options）。
loadData	data	加载树的数据。
getNode	target	获取指定的节点对象。
getData	target	获取指定的节点数据，包括它的子节点。
reload	target	重新加载树的数据。
getRoot	none	获取根节点，返回节点对象。
getRoots	none	获取根节点，返回节点数组。
getParent	target	获取父节点，target 参数表示节点的 DOM 对象。
getChildren	target	获取子节点， target 参数表示节点的 DOM 对象。

getChecked	state	获取选中的节点。状态可用值有：'checked'、'unchecked'、'indeterminate'。如果状态未分配，则返回 'checked' 节点。
		代码实例：
		`var nodes = $('#tt').tree('getChecked');  // get checked nodes`
		`var nodes = $('#tt').tree('getChecked', 'unchecked'); // get unchecked nodes`
		`var nodes = $('#tt').tree('getChecked', 'indeterminate');`
		`// get indeterminate nodes`
		`var nodes = $('#tt').tree('getChecked',`
		`['checked','indeterminate']); // get checked and indeterminate nodes`

getSelected	none	获取选中的节点并返回它，如果没有选中节点，则返回 null。
isLeaf	target	把指定的节点定义成叶节点，target 参数表示节点的 DOM 对象。

find	id	找到指定的节点并返回该节点对象。代码实例：
		`// find a node and then select it`
		`var node = $('#tt').tree('find', 12);`
		`$('#tt').tree('select', node.target);`

select	target	选中一个节点，target 参数表示节点的 DOM 对象。
check	target	把指定节点设置为勾选。
uncheck	target	把指定节点设置为未勾选。
collapse	target	折叠一个节点，target 参数表示节点的 DOM 对象。
expand	target	展开一个节点，target 参数表示节点的 DOM 对象。当节点关闭且没有子节点时，节点的 id 值（名为 'id' 参数）将被发送至服务器以请求子节点数据。
collapseAll	target	折叠所有的节点。
expandAll	target	展开所有的节点。
expandTo	target	从根部展开一个指定的节点。
scrollTo	target	滚动到指定节点。该方法自版本 1.3.4 起可用。
append	param	追加一些子节点到一个父节点，param 参数有两个属性：

		parent：DOM 对象，要追加到的父节点，如果没有分配，则追加为根节点。
		data：数组，节点的数据。

		代码实例：
		`// append some nodes to the selected node`
		`var selected = $('#tt').tree('getSelected');`
		`$('#tt').tree('append', {`
		`parent: selected.target,`
		`data: [{`
		`id: 23,`
		`text: 'node23'`
		`},{`
		`text: 'node24',`
		`state: 'closed',`
		`children: [{`
		`text: 'node241'`
		`},{`
		`text: 'node242'`
		`}]`
		`}]`
		`});`

toggle	target	切换节点的展开/折叠状态，target 参数表示节点的 DOM 对象。

insert	param	在指定节点的前边或后边插入一个节点，param 参数包括下列属性：
		before：DOM 对象，前边插入的节点。
		after：DOM 对象，后边插入的节点。
		data：对象，节点数据。
		下面的代码演示了如何在选中节点之前插入一个新的节点：
		`var node = $('#tt').tree('getSelected');`
		`if (node){`
		`$('#tt').tree('insert', {`
		`before: node.target,`
		`data: {`
		`id: 21,`
		`text: 'node text'`
		`}`
		`});`
		`}`

remove	target	移除一个节点和它的子节点，target 参数表示节点的 DOM 对象。
pop	target	弹出一个节点和它的子节点，该方法和 remove 一样，但是返回了移除的节点数据。
update	param	更新指定的节点，'param' 参数有下列属性：

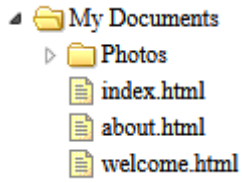
		target（DOM 对象，要被更新的节点）、id、text、iconCls、checked，等等。

		代码实例：
		`// update the selected node text`
		`var node = $('#tt').tree('getSelected');`
		`if (node){`
		`  $('#tt').tree('update', {`
		`  target: node.target,`
		`  text: 'new text'`
		`  });`
		`}`

enableDnd	none	启用拖放功能。
disableDnd	none	禁用拖放功能。
beginEdit	target	开始编辑节点。
endEdit	target	结束编辑节点。
cancelEdit	target	取消编辑节点。

6.1.1 基础树

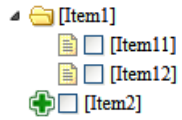
树使用 ul-li 标签，在父节点 ul 上加 class=”easyui-tree”




```
<ul class="easyui-tree">
    <li>
        <span>My Documents</span>
        <ul>
            <li data-options="state:'closed'">
                <span>Photos</span>
                <ul>
                    <li>
                        <span>Friend</span>
                    </li>
                    <li>
                        <span>Wife</span>
                    </li>
                    <li>
                        <span>Company</span>
                    </li>
                </ul>
            </li>
```

```
            <li>index.html</li>
            <li>about.html</li>
            <li>welcome.html</li>
        </ul>
    </li>
</ul>
```

6.1.2 树的基础属性




```
<ul id="tree-ul"></ul>
```

```
<script type="text/javascript">
    $(function () {
        $('#tree-ul').tree({
            animate:true, //展开节点的动画效果
            data: [{//绑定数据
                text: 'Item1',
                state: 'closed',
                children: [{
                    text: 'Item11'
                }, {
                    text: 'Item12'
                }]
            }, {
                text: 'Item2'
            }],
            formatter: function (node) {//格式化输出
                return "[" + node.text + "]";
            },
            loadFilter: function (data) {//过滤节点
                for (var i = 0; i < data.length; i++) {
                    if (data[i].text == "Item2") {
                        data[i].iconCls = "icon-add";
                    }
                }
                return data;
            }
        });
    });
</script>
```

6.1.3 可编辑的树

```
$("#edit-tree").tree({
    onClick: function (node) {//点击节点
        //开始编辑树
        $(this).tree('beginEdit', node.target);
    },
    onAfterEdit: function (node) {//结束编辑
        alert(node.text);
    },
});
```

6.1.4 树的右键菜单

先创建一个菜单

```
<div id="mm" class="easyui-menu" style="width:120px;">
    <div onclick="appendNode()" data-options="iconCls:'icon-add'">Append</div>
    <div onclick="insertNode()" data-options="iconCls:'icon-back'">Insert</div>
    <div onclick="removeNode()" data-options="iconCls:'icon-remove'">Remove</div>
</div>
```

通过树的 onContextMenu 事件设置右键菜单

```
$("#edit-tree").tree({
    onContextMenu: function (e, node) {
        e.preventDefault();
        $(this).tree('select', node.target);
        $('#mm').menu('show', {
            left: e.pageX,
            top: e.pageY
        });
    }
})
```

通过右键菜单对数进行增删改：

```
var id = 0;//模拟自增的 id

function appendNode() {//追加子节点
    //获得选中的节点
    var selected = $('#edit-tree').tree('getSelected');
    var node = {
        parent: selected.target,
        data: [{
```

```
            id: ++id,
            text: '',
            parentId: $('#edit-tree').tree("getNode", selected.target).id,
        }]
    }
    $('#edit-tree').tree('append', node);
    node = $('#edit-tree').tree('find', id);
    //添加后开始编辑，输入节点名
    $('#edit-tree').tree('beginEdit', node.target);
}

function insertNode() {//在当前节点后插入一个新节点
    var selected = $('#edit-tree').tree('getSelected');
    var node = {
        id: ++id,
        text: '',
        parentId: $('#edit-tree').tree("getNode", selected.target).parentId,
    }
    $('#edit-tree').tree('insert', {
        after: selected.target,
        data: node
    });
    node = $('#edit-tree').tree('find', id)
    $('#edit-tree').tree('beginEdit', node.target);
}

function removeNode() {//移除一个节点
    var selected = $('#edit-tree').tree('getSelected');
    $('#edit-tree').tree("remove", selected.target)
}
```

6.1.5 复选框树

```
<input type="button" value="选中的"
onclick="alertNodes($('#check-tree').tree('getChecked'))"/>
<input type="button" value="未选中的"
onclick="alertNodes($('#check-tree').tree('getChecked','unchecked'))"/>
<input type="button" value="半选中的"
onclick="alertNodes($('#check-tree').tree('getChecked','indeterminate'))"/>
<input type="button" value="选中和半选中的"
onclick="alertNodes($('#check-tree').tree('getChecked',
['checked','indeterminate']))"/>
```

```
<ul id="check-tree">
    <li>
        <span>My Documents</span>
        <ul>
            <li data-options="state:'closed'">
                <span>Photos</span>
                <ul>
                    <li><span>Friend</span></li>
                    <li><span>Wife</span></li>
                </ul>
            </li>
            <li>index.html</li>
            <li>about.html</li>
            <li>welcome.html</li>
        </ul>
    </li>
</ul>
<script type="text/javascript">
    $(function () {
        $("#check-tree").tree({
            checkbox: true,//复选框
            cascadeCheck: true,//级联选中
            //onlyLeafCheck:true,//只在页节点前显示复选框

        })
    })

    function alertNodes(nodes){
        var texts = new Array();
        //将已选中的节点 id 封装到数组中
        $(nodes).each(function (i, node) {
            texts[i] = node["text"];
        });
        alert(texts.join(","))
    }
</script>
```

6.1.6 可拖动的树

```
<ul id="check-tree">
    <li>
        <span>Photos</span>
        <ul>
```

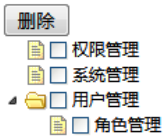
```
            <li><span>Friend</span></li>
            <li><span>Wife</span></li>
        </ul>
    </li>
    <li>index.html</li>
</ul>
<script type="text/javascript">
    $(function () {
        $("#check-tree").tree({
            dnd:true,//开启拖拽
            onDrop: function (target, source, point) {
                alert(source.text);//获取被拖动的节点
                //获取放置的节点
                alert($(this).tree("getNode", target).text);
            }
        })
    })
</script>
```

6.1.7 绑定远程数据

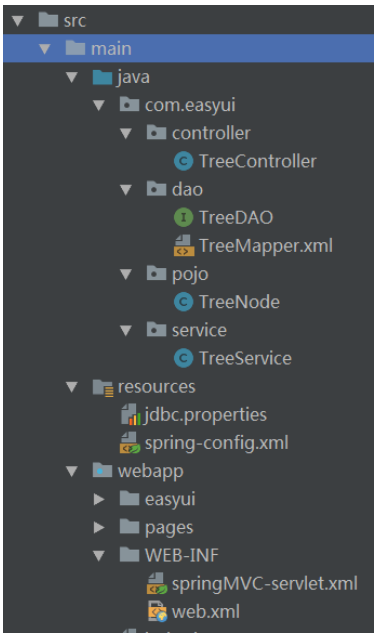
```
<ul id="edit-tree"></ul>
<script type="text/javascript">
    $(function () {
        $("#edit-tree").tree({
            url:'/tree.json',
            method:'post'
        })
    })
</script>
```

6.1.8 动态操作树

结合数据库和 ssm，实现一个可以动态增删改的树。



工程目录：



依赖库：

```
<!--spring 相关包-->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>4.3.11.RELEASE</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.7.4</version>
</dependency>
<!-- Mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.2.8</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.2.2</version>
</dependency>
<!--mysql 驱动-->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.32</version>
```

```
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>4.3.11.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aspects</artifactId>
    <version>4.3.11.RELEASE</version>
</dependency>
```

```
<resources>
    <resource>
        <directory>src/main/java</directory>
        <includes>
            <include>**/*.xml</include>
        </includes>
    </resource>
    <resource>
        <directory>src/main/resources</directory>
        <includes>
            <include>**/*.xml</include>
            <include>**/*.properties</include>
        </includes>
    </resource>
</resources>
```

实现一个树，数据库中至少要有三个字段：
数据表 TREE_NODE

名	类型	长度	小数点	允许空值 (	
ID	int	11	0	☐	1
TEXT	varchar	50	0	☐	
PARENT_ID	int	11	0	☑	

对应的 java 实体类：

```
package com.easyui.pojo;

import java.util.ArrayList;
import java.util.List;

public class TreeNode {
    private Integer id;
    private String text;
    private Integer parentId;
```

```
    private List<TreeNode> children = new ArrayList<>();

    //getter/setter 略
}
```

TreeDAO

```
package com.easyui.dao;

import com.easyui.pojo.TreeNode;

import java.util.List;

public interface TreeDAO {
    List<TreeNode> getTreeNodes();//查询所有树节点
    void addTree(TreeNode node);//添加树节点
    void updateTree(TreeNode node);//修改树节点
    void updateParentId(int parentId);//将父节点置空
    void deleteTree(int id);//删除节点
}
```

TreeMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.easyui.dao.TreeDAO">

    <resultMap id="treeMap" type="TreeNode">
        <id property="id" column="ID"/>
        <result property="text" column="TEXT"/>
        <result property="parentId" column="PARENT_ID"/>
    </resultMap>

    <select id="getTreeNodes" resultMap="treeMap">
        select ID,TEXT,PARENT_ID from TREE_NODE
    </select>

    <insert id="addTree" parameterType="TreeNode" useGeneratedKeys="true"
keyProperty="id">
        insert into TREE_NODE(TEXT,PARENT_ID) values(#{text},#{parentId})
    </insert>

    <update id="updateTree" parameterType="TreeNode">
        update TREE_NODE set TEXT=#{text},PARENT_ID=#{parentId} where ID=#{id}
    </update>

    <update id="updateParentId" parameterType="int">
```

```
        update TREE_NODE set PARENT_ID=null where PARENT_ID=#{parentId}
    </update>

    <delete id="deleteTree" parameterType="int">
        delete from TREE_NODE where ID=#{id}
    </delete>
</mapper>
```

TreeService

```
package com.easyui.service;

import com.easyui.dao.TreeDAO;
import com.easyui.pojo.TreeNode;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Service
public class TreeService {
    @Autowired
    private TreeDAO treeDAO;

    public List<TreeNode> getTree() {//组装树
        List<TreeNode> nodes = treeDAO.getTreeNodes();
        Map<Integer, TreeNode> map = new HashMap<>();
        List<TreeNode> parents = new ArrayList<>();
        for (TreeNode menu : nodes) {
            map.put(menu.getId(), menu);
            if (menu.getParentId() == null) { //代表当前的菜单是父菜单
                parents.add(menu);
            }
        }
        for (TreeNode menu : nodes) {
            //组装子菜单
            TreeNode parent = map.get(menu.getParentId());
            if (parent != null) {
                parent.getChildren().add(menu);
            }
        }
        return parents;
    }
```

```
    public void addTree(TreeNode node) {
        treeDAO.addTree(node);
    }

    public void updateTree(TreeNode node) {
        treeDAO.updateTree(node);
    }

    public void deleteTree(int[] ids) {
        for (int id : ids) {
            treeDAO.deleteTree(id);
            //置空子节点的父 id，当不级联的时候使用
            treeDAO.updateParentId(id);
        }
    }
}
```

TreeController

```
package com.easyui.controller;

import com.easyui.pojo.TreeNode;
import com.easyui.service.TreeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Controller
public class TreeController {
    @Autowired
    private TreeService treeService;

    @RequestMapping("tree.json")//树的列表
    @ResponseBody
    public List<TreeNode> tree() {
        return treeService.getTree();
    }

    @RequestMapping("edit_tree.json")//编辑树节点
```

```
    @ResponseBody
    public Map<String, String> edit(TreeNode treeNode) {
        Map<String, String> result = new HashMap<>();
        try {
            if (treeNode.getId() == null || treeNode.getId() == 0) {
                treeService.addTree(treeNode);
            } else {
                treeService.updateTree(treeNode);
            }
            result.put("status", "true");
            result.put("message", "编辑成功");
        } catch (Exception e) {
            e.printStackTrace();
            result.put("status", "false");
            result.put("message", "编辑失败");
        }
        return result;
    }

    @RequestMapping("delete_tree.json")//批量删除树节点
    @ResponseBody
    public Map<String, String> delete(int[] ids) {
        Map<String, String> result = new HashMap<>();
        try {
            treeService.deleteTree(ids);
            result.put("status", "true");
            result.put("message", "删除成功");
        } catch (Exception e) {
            e.printStackTrace();
            result.put("status", "false");
            result.put("message", "删除失败");
        }
        return result;
    }

}
```

数据库连接 jdbc.properties：

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/test?characterEncoding=utf-8
jdbc.username=root
jdbc.password=
```

spring-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context-4.2.xsd
       http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
       http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.0.xsd">

    <!-- 加载 properties -->
    <context:property-placeholder
            location="classpath:jdbc.properties" ignore-unresolvable="true"/>
    <!--配置数据源-->
    <bean id="dataSource"
          class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="${jdbc.driver}"/>
        <property name="url" value="${jdbc.url}"/>
        <property name="username" value="${jdbc.username}"/>
        <property name="password" value="${jdbc.password}"/>
    </bean>

    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <!--扫描 Mybatis 所有映射文件-->
        <property name="mapperLocations"
                  value="classpath:com/easyui/dao/*.xml"/>
        <property name="typeAliasesPackage" value="com.easyui.pojo"/>
    </bean>
    <!--事务管理器-->
    <bean id="txManager"
          class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource"/>
    </bean>
    <!--定义声明式事务(面向切面)-->
    <tx:advice id="txAdvice" transaction-manager="txManager">
        <tx:attributes>
            <tx:method name="add*"
                       isolation="REPEATABLE_READ" rollback-for="Exception"/>
```

```
            <tx:method name="update*"
                       isolation="REPEATABLE_READ" rollback-for="Exception"/>
            <tx:method name="delete*"
                       isolation="REPEATABLE_READ" rollback-for="Exception"/>
            <tx:method name="get*"
                       isolation="REPEATABLE_READ" read-only="true"/>
            <tx:method name="*"
                       isolation="REPEATABLE_READ" propagation="NOT_SUPPORTED"/>
        </tx:attributes>
    </tx:advice>
    <!--切入点-->
    <aop:config>
        <aop:pointcut
                expression="execution(* com.easyui.service..*(..))" id="point"/>
        <aop:advisor advice-ref="txAdvice" pointcut-ref="point"/>
    </aop:config>
    <context:component-scan base-package="com.easyui.service">
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Service"/>
    </context:component-scan>
    <!--扫描所有的Mybatis的DAO接口，生成代理实现类-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.easyui.dao"/>
    </bean>
</beans>
```

springMVC-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="
  http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
  http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.2.xsd
  http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd">
    <context:component-scan base-package="com.easyui.controller">
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>
    <mvc:annotation-driven
```

```
content-negotiation-manager="contentNegotiationManagerFactoryBean"/>
    <bean id="contentNegotiationManagerFactoryBean"

class="org.springframework.web.accept.ContentNegotiationManagerFactoryBean">
        <!--是否支持后缀匹配-->
        <property name="favorPathExtension" value="false"/>
        <!--是否支持参数匹配-->
        <property name="favorParameter" value="false"/>
        <!--是否 accept-header 匹配-->
        <property name="ignoreAcceptHeader" value="false"/>
        <property name="mediaTypes">
            <map>
                <!--表示.json 结尾的请求返回 json-->
                <entry key="json" value="application/json"/>
            </map>
        </property>
    </bean>

</beans>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
          http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring-config.xml</param-value>
    </context-param>
    <listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-cla
ss>
    </listener>
    <servlet>
        <servlet-name>springMVC</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>springMVC</servlet-name>
        <url-pattern>*.json</url-pattern>
```

```
    </servlet-mapping>

</web-app>
```

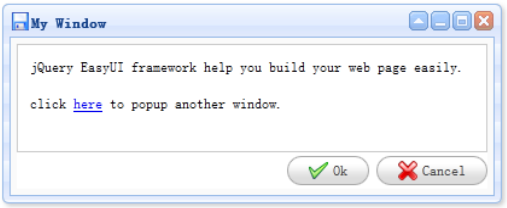
```
<input type="button" value="删除" onclick="deleteNode()" />
<ul id="edit-tree">

</ul>
<div id="mm" class="easyui-menu" style="width:120px;">
    <div onclick="appendNode()" data-options="iconCls:'icon-add'">Append</div>
    <div onclick="insertNode()" data-options="iconCls:'icon-back'">Insert</div>
    <div onclick="removeNode()" data-options="iconCls:'icon-remove'">Remove</div>
</div>
<script type="text/javascript">

    $(function () {
        $("#edit-tree").tree({
            url: '/tree.json',
            method: 'post',
            dnd: true,
            checkbox: true,
            cascadeCheck: false,
            onContextMenu: function (e, node) {
                e.preventDefault();
                // select the node
                $(this).tree('select', node.target);
                // display context menu
                $('#mm').menu('show', {
                    left: e.pageX,
                    top: e.pageY
                });
            },
            onClick: function (node) {
                $(this).tree('beginEdit', node.target);
            },
            onAfterEdit: function (node) {
                //alert(node.text);
                editNode(node.id, node.text, node.parentId)
            },
            onDrop: function (target, source, point) {
                editNode(source.id, source.text, $(this).tree("getNode", target).id)
            }
        })
    });
```

```
    function editNode(id, text, parentId) {
        $.ajax({
            url: "/edit_tree.json",
            method: "post",
            data: {id: id, text: text, parentId: parentId},
            success: function (data) {
                //alert(data.message)
                if (data.status == "true") {
                    $("#edit-tree").tree("reload");
                }
            }
        })
    }

    var id = 0;

    function appendNode() {
        var selected = $('#edit-tree').tree('getSelected');
        var node = {
            parent: selected.target,
            data: [{
                id: id,
                text: '',
                parentId: $('#edit-tree').tree("getNode", selected.target).id,
            }]
        }
        $('#edit-tree').tree('append', node);
        node = $('#edit-tree').tree('find', id)
        $('#edit-tree').tree('beginEdit', node.target);
    }

    function insertNode() {
        var selected = $('#edit-tree').tree('getSelected');
        var node = {
            id: id,
            text: '',
            parentId: $('#edit-tree').tree("getNode", selected.target).parentId,
        }
        $('#edit-tree').tree('insert', {
            after: selected.target,
            data: node
        });
        node = $('#edit-tree').tree('find', id)
```

```
        $('#edit-tree').tree('beginEdit', node.target);
    }

    function deleteNode() {
        var nodes = $('#edit-tree').tree('getChecked');
        var ids = new Array();
        //将已选中的节点 id 封装到数组中
        $(nodes).each(function (i, node) {
            ids[i] = node["id"];
        });
        $.ajax({
            url: '/delete_tree.json',
            type: 'POST',
            data: {ids: ids.join(",")},
            cache: false,
            success: function (data) {
                if (data.status == "true") {
                    $("#edit-tree").tree("reload");
                }
            }
        });
    }
    function removeNode() {
        var selected = $('#edit-tree').tree('getSelected');
        $.ajax({
            url: '/delete_tree.json',
            type: 'POST',
            data: {ids: selected.id},
            cache: false,
            success: function (data) {
                if (data.status == "true") {
                    $("#edit-tree").tree("reload");
                }
            }
        });
    }
</script>
```

7.窗体控件和提示

7.1 窗口 window



窗口（window）是一个浮动的、可拖拽的面板，可以当做应用程序窗口使用。默认情况下，窗口可移动、可调整尺寸、可关闭。它的内容既可以通过静态 html 定义，也可以通过 ajax 动态加载。窗口依赖于 draggable,resizable,panel。

属性：

该属性扩展自面板（panel），下面是为窗口（window）重写和添加的属性。

名称	类型	描述	默认值
title	string	窗口的标题文本。	New Window
collapsible	boolean	定义是否显示折叠按钮。	TRUE
minimizable	boolean	定义是否显示最小化按钮。	TRUE
maximizable	boolean	定义是否显示最大化按钮。	TRUE
closable	boolean	定义是否显示关闭按钮。	TRUE
closed	boolean	定义是否关闭窗口。	FALSE
zIndex	number	从其开始增加的窗口的 z-index。	9000
draggable	boolean	定义窗口是否可拖拽。	TRUE
resizable	boolean	定义窗口是否可调整尺寸。	TRUE
shadow	boolean	如果设置为 true，当窗口能够显示阴影的时候将会显示阴影。	TRUE
inline	boolean	定义如何放置窗口，当设置为 true 时则放在它的父容器里，当设置为 false 时则浮在所有元素的顶部。	FALSE
modal	boolean	定义窗口是不是模态（modal）窗口。	TRUE

事件：

该事件扩展自面板（panel）。

方法：

该方法扩展自面板（panel），下面是为窗口（window）添加的方法。

名称	参数	描述
window	none	返回外部窗口（window）对象。
hcenter	none	水平居中窗口。该方法自版本 1.3.1 起可用。
vcenter	none	垂直居中窗口。该方法自版本 1.3.1 起可用。

center	none	居中窗口。该方法自版本 1.3.1 起可用。

7.1.1 基础窗口

```
<div style="margin:20px 0;">
    <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="$('#w').window('open')">Open</a>
    <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="$('#w').window('close')">Close</a>
</div>
<div id="w" class="easyui-window" title="Basic Window"
data-options="iconCls:'icon-save'"
     style="width:500px;height:200px;padding:10px;">
    The window content.
</div>
```

7.1.2 模态窗口

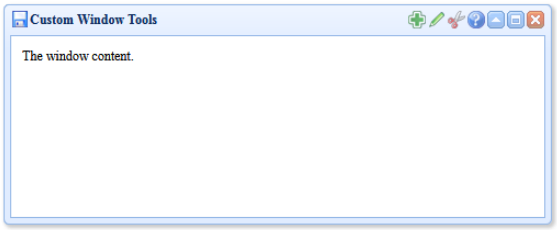
模态窗口打开时有个遮罩层。modal=true

7.1.3 内联窗口

限制一个窗口只能在它的父容器中：inline=true


```
<div class="easyui-panel"
style="position:relative;width:500px;height:300px;overflow:auto;">
    <div class="easyui-window" data-options="title:'Inline Window',inline:true"
style="width:250px;height:150px;padding:10px">
        This window stay inside its parent
    </div>
</div>
```

7.1.4 自定义工具

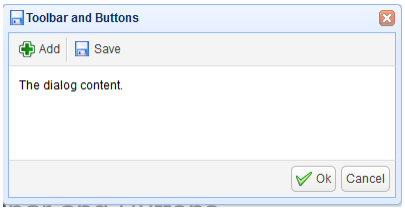



```
<div class="easyui-window" title="Custom Window Tools"
data-options="iconCls:'icon-save',minimizable:false,tools:'#tt'"
style="width:500px;height:200px;padding:10px;">
    The window content.
</div>
<div id="tt">
    <a href="javascript:void(0)" class="icon-add"
onclick="javascript:alert('add')"></a>
    <a href="javascript:void(0)" class="icon-edit"
onclick="javascript:alert('edit')"></a>
    <a href="javascript:void(0)" class="icon-cut"
onclick="javascript:alert('cut')"></a>
    <a href="javascript:void(0)" class="icon-help"
onclick="javascript:alert('help')"></a>
</div>
```

7.1.5 动态加载内容

使用 href 属性。

7.2 对话框 dialog



对话框（dialog）是一个特殊类型的窗口，它在顶部有一个工具栏，在底部有一个按钮栏。默认情况下，对话框（dialog）只有一个显示在头部右侧的关闭工具。用户可以配置对话框行为来显示其他工具（比如：可折叠 collapsible、可最小化 minimizable、可最大化 maximizable，等等）。对话框依赖于 window 和 linkbutton。

属性：

该属性扩展自窗口（window）下面是为对话框（dialog）重写的属性。

<table>
<tr><th>名称</th><th>类型</th><th>描述</th><th>默认值</th></tr>
<tr><td>title</td><td>string</td><td>对话框的标题文本。</td><td>New Dialog</td></tr>
<tr><td>collapsible</td><td>boolean</td><td>定义是否显示折叠按钮。</td><td>FALSE</td></tr>
<tr><td>minimizable</td><td>boolean</td><td>定义是否显示最小化按钮。</td><td>FALSE</td></tr>
<tr><td>maximizable</td><td>boolean</td><td>定义是否显示最大化按钮。</td><td>FALSE</td></tr>
<tr><td>resizable</td><td>boolean</td><td>定义对话框是否可调整尺寸。</td><td>FALSE</td></tr>
<tr><td>toolbar</td><td>array,selector</td><td>对话框的顶部工具栏，可能的值：
1、数组，每个工具的选项都与链接按钮（linkbutton）一样。
2、选择器，指示工具栏。

对话框工具栏可以在 标签中声明：
<div class="easyui-dialog"
style="width:600px;height:300px"
data-options="title:'My
Dialog',toolbar:'#tb',modal:true">
Dialog Content.

<a href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-edit',plain:true"/a>
<a href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-help',plain:true"/a>

对话框工具栏也可以通过数组定义：
<div class="easyui-dialog"
style="width:600px;height:300px"
data-options="title:'My Dialog',modal:true,
toolbar:[{
text:'Edit',
iconCls:'icon-edit',
handler:function(){alert('edit')}
},{
text:'Help',
iconCls:'icon-help',
handler:function(){alert('help')}
}]">
Dialog Content.
 </td><td>null</td></tr>
</table>

<table>
<tr>
<td>buttons</td>
<td>array,selector</td>
<td>对话框的底部按钮，可能的值：
1、数组，每个按钮的选项都与链接按钮（linkbutton）一样。
2、选择器，指示按钮栏。
按钮可以在 标签中声明：
<pre>
<div class="easyui-dialog"
style="width:600px;height:300px"
    data-options="title:'My
Dialog',buttons:'#bb',modal:true">
    Dialog Content.
</div>
<div id="bb">
    Save
    Close
</div>
</pre>
按钮也可以通过数组定义：
<pre>
<div class="easyui-dialog"
style="width:600px;height:300px"
    data-options="title:'My Dialog',modal:true,
            buttons:[{
                text:'Save',
                handler:function(){...}
            },{
                text:'Close',
                handler:function(){...}
            }]">
    Dialog Content.
</div>
</pre></td>
<td>null</td>
</tr>
</table>

事件：

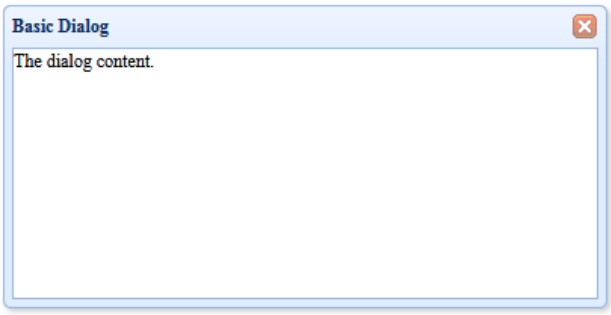
该事件扩展自面板（panel）。

方法：

该方法扩展自窗口（window），下面是为对话框（dialog）添加的方法。

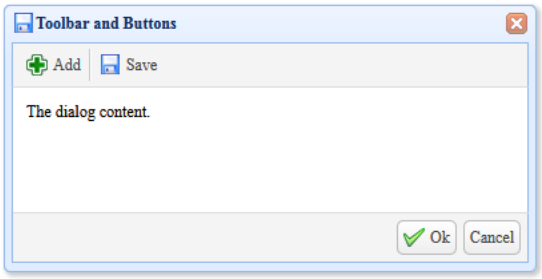
名称	参数	描述
dialog	none	返回外部对话框（dialog）对象。

7.2.1 基本对话框




```
<div class="easyui-dialog" title="Basic Dialog" style="width:400px;height:200px;">
    The dialog content.
</div>
```

7.2.2 工具栏和按钮组




```
<div id="dlg" class="easyui-dialog" title="Toolbar and Buttons"
style="width:400px;height:200px;padding:10px"
    data-options="
            iconCls: 'icon-save',
            toolbar: [{
               text:'Add',
               iconCls:'icon-add',
               handler:function(){
                  alert('add')
               }
            },'-',{
               text:'Save',
               iconCls:'icon-save',
               handler:function(){
                  alert('save')
               }
            }],
            buttons: [{
```

```
                text:'Ok',
                iconCls:'icon-ok',
                handler:function(){
                    alert('ok');
                }
            },{
                text:'Cancel',
                handler:function(){
                    alert('cancel');;
                }
            }]
        ">
    The dialog content.
</div>
```

工具栏也可以引用其它元素：

```
<div id="dlg2" class="easyui-dialog" title="Toolbar and Buttons2"
style="width:400px;height:200px;padding:10px"
     data-options="
            iconCls: 'icon-save',
            toolbar: '#tool',
            buttons: '#btn'
        ">
    The dialog content.
</div>
<div id="tool">
    <a href="javascript:void(0)" class="easyui-linkbutton"
data-options="iconCls:'icon-add',plain:true"
onclick="javascript:alert('add')">Add</a>
    <a href="javascript:void(0)" class="easyui-linkbutton"
data-options="iconCls:'icon-save',plain:true"
onclick="javascript:alert('save')">Save</a>
</div>
<div id="btn" style="padding:5px;">
    <a href="javascript:void(0)" class="easyui-linkbutton"
data-options="iconCls:'icon-ok'" onclick="javascript:alert('ok')"></a>
    <a href="javascript:void(0)" class="easyui-linkbutton"
data-options="iconCls:'icon-cancel'" onclick="javascript:alert('cancel')"></a>
</div>
```

7.2.3 远程加载页面

使用 href 属性。操作和 window 一致。

7.3 消息框 messager

消息框（messager）提供不同样式的消息框，包括警示（alert）、确认（confirm）、提示（prompt）、进展（progress）等等。所有的消息框都是异步的。用户可以在与消息框交互后使用回调函数来完成一些动作。消息框依赖于 window、linkbutton、progressbar。

属性：

名称	类型	描述	默认值
ok	string	确定按钮的文本。	Ok
cancel	string	取消按钮的文本。	Cancel

方法：

名称	参数	描述

$.messager.show	options	在屏幕的右下角显示一个消息窗口，options 参数是一个配置对象：
		showType： 定义消息窗口如何显示。可用的值是：null、slide、fade、show。默认是 slide。
		showSpeed： 定义消息窗口完成显示所需的以毫秒为单位的时间。默认是 600。
		width：定义消息窗口的宽度。默认是 250。
		height：定义消息窗口的高度。默认是 100。
		title：头部面板上显示的标题文本。
		msg：要显示的消息文本。
		style：定义消息窗口的自定义样式。
		timeout：如果定义为 0，除非用户关闭，消息窗口将不会关闭。如果定义为非 0 值，消息窗口将在超时后自动关闭。默认是 4 秒。
		代码实例：
		`$.messager.show({`
		`title:'My Title',`
		`msg:'Message will be closed after 5 seconds.',`
		`timeout:5000,`
		`showType:'slide'`
		`});`
		`// show message window on top center`
		`$.messager.show({`
		`title:'My Title',`
		`msg:'Message will be closed after 4 seconds.',`
		`showType:'show',`
		`style:{`
		`right:'',`
		`top:document.body.scrollTop+document.documentElement.scrollTop,`

		bottom:'' } });
$.messager.alert	title, msg, icon, fn	显示一个警告提示窗口。参数： title：显示在头部面板上的标题文本。 msg：要显示的消息文本。 icon：要显示的图标图片。可用的值是：error、question、info、warning。 fn：当窗口关闭时触发的回调函数。 代码实例： $.messager.alert('My Title','Here is a info message!','info');

$.messager.confirm	title, msg, fn	显示一个带"确定"和"取消"按钮的确认消息窗口。参数：
		title：显示在头部面板上的标题文本。
		msg：要显示的消息文本。
		fn(b)：回调函数，当用户点击确认按钮时传递一个 true 参数给函数，否则给它传递一个 false 参数。
		代码实例：
		$.messager.confirm('Confirm', 'Are you sure to exit this system?', function(r){
		if (r){
		// exit action;
		}
		});

$.messager.prompt	title, msg, fn	显示一个带"确定"和"取消"按钮的消息窗口，提示用户输入一些文本。参数：
		title：显示在头部面板上的标题文本。
		msg：要显示的消息文本。
		fn(val)：带有用户输入的数值参数的回调函数。
		代码实例：
		$.messager.prompt('Prompt', 'Please enter your name:', function(r){
		if (r){
		alert('Your name is:' + r);
		}
		});

$.messager.progress	options or method	显示一个进度的消息窗口。 options 定义如下： title：显示在头部面板上的标题文本，默认值是 '' 。 msg：消息框的主体文本，默认值是 '' 。

		text：显示在进度条里的文本，默认值是 undefined 。
		interval：每次进度更新之间以毫秒为单位的时间长度。默认值是 300。
		方法定义如下： bar：获取进度条（progressbar）对象。
		close：关闭进度窗口。
		代码实例：
		显示进度消息窗口。
		1. $.messager.progress();
		现在关闭消息窗口。
		1. $.messager.progress('close');

7.3.1 基础消息框

```
<div style="margin:20px 0;">
    <a href="#" class="easyui-linkbutton" onclick="show()">Show</a>
    <a href="#" class="easyui-linkbutton" onclick="slide()">Slide</a>
    <a href="#" class="easyui-linkbutton" onclick="fade()">Fade</a>
    <a href="#" class="easyui-linkbutton" onclick="progress()">Progress</a>
</div>
<script type="text/javascript">
    function show(){
        $.messager.show({
            title:'My Title',
            msg:'Message will be closed after 4 seconds.',
            showType:'show'
        });
    }
    function slide(){
        $.messager.show({
            title:'My Title',
            msg:'Message will be closed after 5 seconds.',
            timeout:5000,
            showType:'slide'
        });
    }
    function fade(){
        $.messager.show({
            title:'My Title',
```

```
            msg:'Message never be closed.',
            timeout:0,
            showType:'fade'
        });
    }
    function progress(){
        var win = $.messager.progress({
            title:'Please waiting',
            msg:'Loading data...'
        });
        setTimeout(function(){
            $.messager.progress('close');
        },5000)
    }
</script>
```

7.3.2 消息框位置

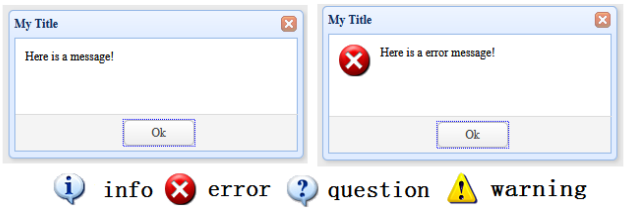
```
<div style="margin:20px 0;">
    <p>
        <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="topLeft();">TopLeft</a>
        <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="topCenter()">TopCenter</a>
        <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="topRight()">TopRight</a>
    </p>
    <p>
        <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="centerLeft()">CenterLeft</a>
        <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="center()">Center</a>
        <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="centerRight()">CenterRight</a>
    </p>
    <p>
        <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="bottomLeft()">BottomLeft</a>
        <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="bottomCenter()">BottomCenter</a>
        <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="bottomRight()">BottomRight</a>
    </p>
```

```
</div>
<script>
    function topLeft(){
        $.messager.show({
            title:'My Title',
            msg:'The message content',
            showType:'show',
            style:{
                right:'',
                left:0,
                top:document.body.scrollTop+document.documentElement.scrollTop,
                bottom:''
            }
        });
    }
    function topCenter(){
        $.messager.show({
            title:'My Title',
            msg:'The message content',
            showType:'slide',
            style:{
                right:'',
                top:document.body.scrollTop+document.documentElement.scrollTop,
                bottom:''
            }
        });
    }
    function topRight(){
        $.messager.show({
            title:'My Title',
            msg:'The message content',
            showType:'show',
            style:{
                left:'',
                right:0,
                top:document.body.scrollTop+document.documentElement.scrollTop,
                bottom:''
            }
        });
    }
    function centerLeft(){
        $.messager.show({
            title:'My Title',
            msg:'The message content',
```

```
            showType:'fade',
            style:{
                left:0,
                right:'',
                bottom:''
            }
        });
    }
    function center(){
        $.messager.show({
            title:'My Title',
            msg:'The message content',
            showType:'fade',
            style:{
                right:'',
                bottom:''
            }
        });
    }
    function centerRight(){
        $.messager.show({
            title:'My Title',
            msg:'The message content',
            showType:'fade',
            style:{
                left:'',
                right:0,
                bottom:''
            }
        });
    }
    function bottomLeft(){
        $.messager.show({
            title:'My Title',
            msg:'The message content',
            showType:'show',
            style:{
                left:0,
                right:'',
                top:'',
                bottom:-document.body.scrollTop-document.documentElement.scrollTop
            }
        });
    }
```

```
    function bottomCenter(){
        $.messager.show({
            title:'My Title',
            msg:'The message content',
            showType:'slide',
            style:{
                right:'',
                top:'',
                bottom:-document.body.scrollTop-document.documentElement.scrollTop
            }
        });
    }
    function bottomRight(){
        $.messager.show({
            title:'My Title',
            msg:'The message content',
            showType:'show'
        });
    }
</script>
```

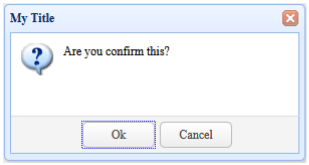
7.3.3 提示消息

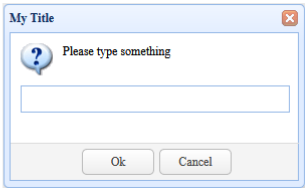



```
<div style="margin:20px 0;">
    <a href="#" class="easyui-linkbutton" onclick="alert1()">Alert</a>
    <a href="#" class="easyui-linkbutton" onclick="alert2()">Error</a>
    <a href="#" class="easyui-linkbutton" onclick="alert3()">Info</a>
    <a href="#" class="easyui-linkbutton" onclick="alert4()">Question</a>
    <a href="#" class="easyui-linkbutton" onclick="alert5()">Warning</a>
</div>
<script type="text/javascript">
    function alert1() {
        $.messager.alert('My Title', 'Here is a message!');
    }
    function alert2() {
        $.messager.alert('My Title', 'Here is a error message!', 'error');
    }
```

```
    function alert3() {
        $.messager.alert('My Title', 'Here is a info message!', 'info');
    }
    function alert4() {
        $.messager.alert('My Title', 'Here is a question message!', 'question');
    }
    function alert5() {
        $.messager.alert('My Title', 'Here is a warning message!', 'warning');
    }
</script>
```

7.3.4 交互式消息






```
<div style="margin:20px 0;">
    <a href="#" class="easyui-linkbutton" onclick="confirm1();">Confirm</a>
    <a href="#" class="easyui-linkbutton" onclick="prompt1()">Prompt</a>
</div>
<script>
    function confirm1(){
        $.messager.confirm('My Title', 'Are you confirm this?', function(r){
            if (r){
                alert('confirmed: '+r);
            }
        });
    }
    function prompt1(){
        $.messager.prompt('My Title', 'Please type something', function(r){
            if (r){
                alert('you type: '+r);
            }
        });
    }
</script>
```

7.4 提示 tooltip

当用户移动鼠标指针在某个元素上时，出现提示信息窗口用来显示额外信息。提示内容可以包含任何来自页面的或者通过 ajax 生成的 html 元素。

属性：

名称	类型	描述	默认值
position	string	提示框（tooltip）位置。可能的值：'left'、'right'、'top'、'bottom'。	bottom
content	string	提示框（tooltip）内容。	null
trackMouse	boolean	如果设置为 true，提示框（tooltip）会随着鼠标移动。	FALSE
deltaX	number	提示框（tooltip）位置的水平距离。	0
deltaY	number	提示框（tooltip）位置的垂直距离。	0
showEvent	string	引发提示框（tooltip）出现的事件。	mouseenter
hideEvent	string	引发提示框（tooltip）消失的事件。	mouseleave
showDelay	number	显示提示框（tooltip）的时间延迟。	200
hideDelay	number	隐藏提示框（tooltip）的时间延迟。	100

事件：

名称	参数	描述
onShow	e	当显示提示框（tooltip）时触发。
onHide	e	当隐藏提示框（tooltip）时触发。
onUpdate	content	当提示框（tooltip）内容更新时触发。
onPosition	left,top	当提示框（tooltip）位置改变时触发。
onDestroy	none	当提示框（tooltip）销毁时触发。

方法：

名称	参数	描述
options	none	返回选项（options）属性（property）。
tip	none	返回提示（tip）对象。
arrow	none	返回箭头（arrow）对象。
show	e	显示提示框（tooltip）。
hide	e	隐藏提示框（tooltip）。
update	content	更新提示框（tooltip）内容。
reposition	none	重置提示框（tooltip）位置。
destroy	none	销毁提示框（tooltip）。

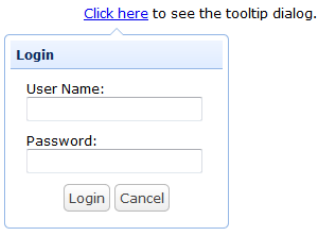
7.4.1 基础提示框

```
<a href="#" title="This is the tooltip message." class="easyui-tooltip">Hover me</a>
<a id="dd" href="javascript:void(0)">Click here</a>
<script type="text/javascript">
    $(function () {
        $('#dd').tooltip({
            position: 'right',
            content: '<span style="color:#fff">This is the tooltip message.</span>',
            onShow: function(){
                $(this).tooltip('tip').css({
                    backgroundColor: '#666',
                    borderColor: '#666'
                });
            }
        });
    })
</script>
```

7.4.2 提示框位置

通过 position 设置提示框位置

7.4.3 提示窗口




```
<div style="padding:10px 200px">
    <p><a id="dd" href="javascript:void(0)">Click here</a> to see the tooltip dialog.
</div>
<div id="dd-form" style="padding:5px;display: none">
    <div style="padding:5px 10px">
        <div>User Name:</div>
        <input style="width:160px">
    </div>
    <div style="padding:5px 10px">
```

```
        <div>Password:</div>
        <input type="password" style="width:160px">
    </div>
    <div style="padding:5px 10px;text-align:center">
        <a href="javascript:void(0)" class="easyui-linkbutton">Login</a>
        <a href="javascript:void(0)" class="easyui-linkbutton">Cancel</a>
    </div>
</div>
<script type="text/javascript">
    $(function(){
        $('#dd').tooltip({
            content: $("#dd-form"),
            showEvent: 'click',
            onUpdate: function(content){
                content.panel({
                    width: 200,
                    border: false,
                    title: 'Login'
                    //href: '_dialog.html'
                });
            },
            onShow: function(){
                var t = $(this);
                t.tooltip('tip').unbind().bind('mouseenter', function(){
                    t.tooltip('show');
                }).bind('mouseleave', function(){
                    t.tooltip('hide');
                });
            }
        });
    });
</script>
```

7.4.4 提示工具栏

Hover me to display toolbar.

```
<div style="padding:10px 200px">
    <p><a id="dd" href="javascript:void(0)" class="easyui-tooltip" data-options="
            hideEvent: 'none',
            content: function(){
                return $('#toolbar');
```

```
            },
            onShow: function(){
                var t = $(this);
                t.tooltip('tip').focus().unbind().bind('blur',function(){
                    t.tooltip('hide');
                });
            }
        ">Hover me</a> to display toolbar.</p>
</div>
<div style="display:none">
    <div id="toolbar">
        <a href="#" class="easyui-linkbutton" title="Add"
            data-options="iconCls:'icon-add',plain:true"></a>
        <a href="#" class="easyui-linkbutton" title="Cut"
            data-options="iconCls:'icon-cut',plain:true"></a>
        <a href="#" class="easyui-linkbutton" title="Remove"
            data-options="iconCls:'icon-remove',plain:true"></a>
    </div>
</div>
```

7.4.5 自定义样式

```
$('#pp1').tooltip({
    position: 'right',
    content: '<span style="color:#fff">This is the tooltip message.</span>',
    onShow: function(){
        $(this).tooltip('tip').css({
            backgroundColor: '#666',
            borderColor: '#666'
        });
    }
});
```

8.表单控件

8.1 验证框 validatebox

验证框（validatebox）是为了验证表单输入字段而设计的。如果用户输入无效的值，它将改变背景颜色，显示警告图标和提示消息。验证框（validatebox）可与表单（form）插件集成，防止提交无效的字段。验证框依赖于 tooltip。

属性：

名称	类型	描述	默认值
required	boolean	定义是否字段应被输入。	FALSE
validType	string,array	定义字段的验证类型，比如 email、url，等等。可能的值： 1、验证类型字符串，应用单个验证规则。 2、验证类型数组，应用多个验证规则。单个字段上的多个验证规则自版本 1.3.2 起可用。 代码实例： <input class="easyui-validatebox" data-options="required:true,validType:'url'"> <input class="easyui-validatebox" data-options=" required:true, validType:['email','length[0,20]'] ">	null
delay	number	延迟验证最后的输入值。该属性自版本 1.3.2 起可用。	200
missingMessage	string	当文本框为空时出现的提示文本。	该字段是必需的。
invalidMessage	string	当文本框的内容无效时出现的提示文本。	null
tipPosition	string	定义当文本框的内容无效时提示消息的位置。可能的值：'left'、'right'。该属性自版本 1.3.2 起可用。	right
deltaX	number	在 X 方向的提示偏移。该属性自版本 1.3.3 起可用。	0
novalidate	boolean	当设置为 true 时，则禁用验证。该属性自版本 1.3.4 起可用。	FALSE

方法：

名称	参数	描述
destroy	none	移除并销毁该组件。
validate	none	进行验证以判定文本框的内容是否有效。
isValid	none	调用 validate 方法并且返回验证结果，true 或者 false。
enableValidation	none	启用验证。该方法自版本 1.3.4 起可用。
disableValidation	none	禁用验证。该方法自版本 1.3.4 起可用。

8.1.1 基本验证框

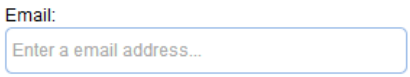
```
<div>
    必填<input class="easyui-validatebox"
data-options="required:true,validateOnCreate:false,validType:'length[3,10]'"
```

```
name="username"/>
</div>
<div>
    邮箱<input class="easyui-validatebox"
data-options="required:true,validType:'email'" name="email"/>
</div>
<div>
    网址<input class="easyui-validatebox"
data-options="required:true,validType:'url'" name="url"/>
</div>
<div>
    js<input id="vv"/>
</div>
<div>
    密码<input id="pwd" class="easyui-validatebox" data-options="required:true"/>
</div>
<div>
    确认密码<input class="easyui-validatebox" data-options="required:true"
validType="equals['#pwd']"/>
</div>
<script type="text/javascript">
    $(function () {
        $("#vv").validatebox({
            required:true,
            validType:'email'
        });
    })
</script>
```

8.1.2 自定义验证框

```
$.extend($.fn.validatebox.defaults.rules,{
    equals:{
        validator:function(value,param){
            return value==$(param[0]).val()
        },
        message:'field do not match'
    }
})
```

8.2 文本框 textbox



8.2.1 基础文本框

```
<input type="button" value="取值" onclick="alert($('#tt').textbox('getValue'))"/>
<input type="button" value="赋值" onclick="$('#tt').textbox('setValue','123')"/>
<input class="easyui-textbox" data-options="prompt:'please input
email',validType:'email'" style="width: 10%">
```

8.2.2 多行文本框

```
<input class="easyui-textbox" data-options="multiline:true" id="tt">
```

8.2.3 带图标的文本框

```
<input class="easyui-textbox" data-options="icons:[{
    iconCls:'icon-add',
    handler:function(e){
       alert('add')
    }
},
{
    iconCls:'icon-remove',
    handler:function(e){
       alert('remove')
    }
}]" style="width:400px">
```

8.3 数字框 numberbox

数字框（numberbox）用于让用户仅能输入数字的值。它可以把输入元素转换为不同类型的输入（比如：数字 numeric、百分比 percentage、货币 currency，等等）。更多的输入类型依赖 'formatter' 和 'parser' 函数来定义。数字框依赖于 validatebox。

属性：

该属性扩展自验证框（validatebox），下面是为数字框（numberbox）添加的属性。

名称	类型	描述	默认值
disabled	boolean	定义是否禁用该字段。	FALSE
value	number	默认值。	
min	number	允许的最小值。	null
max	number	允许的最大值。	null
precision	number	显示在小数点后面的最大精度。	0
decimalSeparator	string	分隔数字的整数部分和小数部分的分隔字符	.
groupSeparator	string	分隔整数组合的字符。	
prefix	string	前缀字符串。	
suffix	string	后缀字符串。	
filter	function(e)	定义如何过滤被按下的键，返回 true 则接受输入字符。该属性自版本 1.3.3 起可用。	
formatter	function(value)	用来格式数字框（numberbox）值的函数。返回显示在框中的字符串值。	
parser	function(s)	用来解析字符串的函数。返回数字框（numberbox）值。	

事件：

名称	参数	描述
onChange	newValue,oldValue	当字段值改变时触发。

方法：

该方法扩展自验证框（validatebox），下面是为数字框（numberbox）添加或重写的方法。

名称	参数	描述
options	none	返回选项（options）对象。
destroy	none	销毁数字框（numberbox）对象。
disable	none	禁用该域。
enable	none	启用该域。
fix	none	把值固定为有效的值。
setValue	none	设置数字框（numberbox）的值。 代码实例： `$('#nn').numberbox('setValue', 206.12);`

getValue	none	获取数字框（numberbox）的值。
		代码实例：
		`var v = $('#nn').numberbox('getValue');`
		`alert(v);`

clear	none	清除数字框（numberbox）的值。
reset	none	重置数字框（numberbox）的值。该方法自版本 1.3.2 起可用。

8.3.1 基础数值框

```
<input class="easyui-numberbox"/>
```

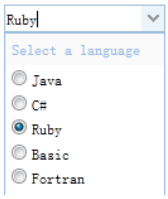
8.3.2 数值范围

```
<input class="easyui-numberbox"
data-options="min:10,max:90,precision:2,required:true"/>
```

8.3.3 格式化数字框

```
<input class="easyui-numberbox" value="12345678.09"
data-options="precision:2,groupSeparator:',',prefix:'$'"/>
    <input class="easyui-numberbox" value="12345678.09"
data-options="precision:2,groupSeparator:',',suffix:'$'"/>
    <input id="num"/>
<script type="text/javascript">
    $(function () {
        $("#num").numberbox({
            precision:2,
            groupSeparator:','
        })
    })

</script>
```

8.4 组合 combo



组合（combo）是在 html 页面上显示一个可编辑的文本框和下拉面板。它是用于创建其他复杂的组合组件（比如：组合框 combobox、组合树 combotree、组合网格 combogrid，等等）的基础组件。组合依赖于 validatebox 和 panel。

属性：

该属性扩展自验证框（validatebox），下面是为组合（combo）添加的属性。

名称	类型	描述	默认值
width	number	组件的宽度。	auto
height	number	组件的高度。该属性自版本 1.3.2 起可用。	22
panelWidth	number	下拉面板的宽度。	null
panelHeight	number	下拉面板的高度。	200
multiple	boolean	定义是否支持多选。	FALSE
selectOnNavigation	boolean	定义当通过键盘导航项目时是否选择项目。该属性自版本 1.3.3 起可用。	TRUE
separator	string	多选时文本的分隔符。	,
editable	boolean	定义用户是否可以往文本域中直接输入文字。	TRUE
disabled	boolean	定义是否禁用文本域。	FALSE
readonly	boolean	定义组件是否只读。该属性自版本 1.3.3 起可用。	FALSE
hasDownArrow	boolean	定义是否显示向下箭头的按钮。	TRUE
value	string	默认值。	
delay	number	从最后一个键的输入事件起，延迟进行搜索。	200
keyHandler	object	当用户按键后调用的函数。默认的 keyHandler 定义如下： `keyHandler: { up: function(){}, down: function(){}, enter: function(){}, query: function(q){} }`	

事件：

名称	参数	描述
onShowPanel	none	当下拉面板显示的时候触发。
onHidePanel	none	当下拉面板隐藏的时候触发。
onChange	newValue,	当文本域的值改变的时候触发。

	oldValue	

方法：

该方法扩展自验证框（validatebox），下面是为组合（combo）添加的方法。

名称	参数	描述
options	none	返回选项（options）对象。
panel	none	返回下拉面板对象。
textbox	none	返回文本框对象。
destroy	none	销毁组件。
resize	width	调整组件的宽度。
showPanel	none	显示下拉面板。
hidePanel	none	隐藏下拉面板。
disable	none	禁用组件。
enable	none	启用组件。

readonly	mode	启用/禁用只读模式。该方法自版本 1.3.3 起可用。
		用法实例：
		`$('#cc').combo('readonly');  // enable readonly mode`
		`$('#cc').combo('readonly', true);  // enable readonly mode`
		`$('#cc').combo('readonly', false); // disable readonly mode`

validate	none	验证输入的值。
isValid	none	返回验证结果。
clear	none	清除组件的值。
reset	none	重置组件的值。该方法自版本 1.3.2 起可用。
getText	none	获取输入的文本。
setText	none	设置文本值。
getValues	none	获取组件的值的数组。
setValues	none	设置组件的值的数组。
getValue	none	获取组件的值。
setValue	none	设置组件的值。

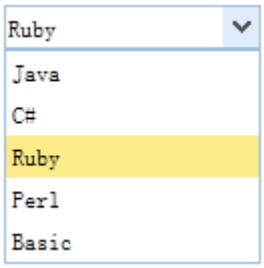
8.4.1 基础组合

```
<div>
    <select id="cc" style="width: 150px"></select>
</div>
<div id="sp">
    <input type="radio" name="lang" value="01"><span>java</span><br/>
    <input type="radio" name="lang" value="02"><span>python</span><br/>
    <input type="radio" name="lang" value="03"><span>php</span><br/>
    <input type="radio" name="lang" value="04"><span>c#</span><br/>
</div>
<script type="text/javascript">
    $(function(){
```

```
        $("#cc").combo({
            required:true,
            editable:false,
            multiple:true
        });
        $("#sp").appendTo($("#cc").combo("panel"));
        $("#sp input").click(function(){
            var v=$(this).val();
            var s=$(this).next("span").text();
            $("#cc").combo("setValue",v).combo("setText",s).combo("hidePanel")
        })
    })
</script>
```

8.5 组合框 combobox

组合框（combobox）显示一个可编辑的文本框和下拉列表，用户可以从下拉列表中选择一个或多个值。用户可以直接输入文本到列表的顶部，或者从列表中选择一个或多个现成的值。组合框依赖 combo



该属性扩展自组合（combo），下面是为组合框（combobox）添加的属性。

名称	类型	描述	默认值
valueField	string	绑定到该组合框（ComboBox）的 value 上的基础数据的名称。	value
textField	string	绑定到该组合框（ComboBox）的 text 上的基础数据的名称。	text
groupField	string	指示要被分组的字段。该属性自版本 1.3.4 起可用。	null

groupFormatter	function(group)	返回要显示在分组项目上的分组文本。该属性自版本 1.3.4 起可用。
		代码实例：
		`$('#cc').combobox({`
		`groupFormatter: function(group){`
		`return '<span style="color:red">' + group + '</span>';`	

		} });	
mode	string	定义在文本改变时如何加载列表数据。如果组合框（combobox）从服务器加载就设置为'remote'。当设置为'remote'模式时，用户输入的值将会被作为名为'q'的 http 请求参数发送到服务器，以获取新的数据。	local
url	string	从远程加载列表数据的 URL 。	null
method	string	用来检索数据的 http 方法。	post

data	array	被加载的列表数据。
		代码实例：
		<input class="easyui-combobox"
		data-options="
		valueField: 'label',
		textField: 'value',
		data: [{
		label: 'java',
		value: 'Java'
		},{
		label: 'perl',
		value: 'Perl'
		},{
		label: 'ruby',
		value: 'Ruby'
		}]" />
			null

filter	function	定义当'mode'设置为'local'时如何过滤本地数据。该函数有两个参数：
		q：用户输入的文本。
		row：列表中的行数据。
		返回 true 则允许显示该行。
		代码实例：
		$('#cc').combobox({
		filter: function(q, row){
		var opts =
		$(this).combobox('options');
		return
		row[opts.textField].indexOf(q) == 0;
		}
		});	

formatter	function	定义如何呈现行。该函数有一个参数：row。
		代码实例：
		$('#cc').combobox({
		formatter: function(row){
		var opts =	

		$(this).combobox('options'); return row[opts.textField]; } });	
loader	function(param,success,error)	定义如何从远程服务器加载数据。返回 false 则取消该动作。该函数有下列参数： param：要传到远程服务器的参数对象。 success(data)：当获取数据成功时将被调用的回调函数。 error()：当获取数据失败时将被调用的回调函数。	json loader
loadFilter	function(data)	返回要显示的过滤数据。该属性自版本 1.3.3 起可用。	

事件：

该事件扩展自组合（combo），下面是为组合框（combobox）添加的事件。

名称	参数	描述

onBeforeLoad	param	在请求加载数据之前触发，返回 false 则取消加载动作。
		代码实例：
		// change the http request parameters before load data from server
		$('#cc').combobox({
		onBeforeLoad: function(param){
		param.id = 2;
		param.language = 'js';
		}
		});

onLoadSuccess	none	当远程数据加载成功时触发。
onLoadError	none	当远程数据加载失败时触发。
onSelect	record	当用户选择一个列表项时触发。
onUnselect	record	当用户取消选择一个列表项时触发。

方法：

该方法扩展自组合（combo），下面是为组合框（combobox）添加或重写的方法。

名称	参数	描述
options	none	返回选项（options）对象。
getData	none	返回加载的数据。
loadData	data	加载本地列表数据。

reload	url	请求远程的列表数据。传 'url' 参数来重写原始的 URL 值。
		代码实例：
		$('#cc').combobox('reload'); // reload list data using old URL
		$('#cc').combobox('reload','get_data.php'); // reload list

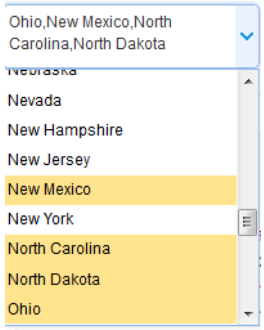
		data using new URL
setValues	values	设置组合框（combobox）值的数组。 代码实例： $('#cc').combobox('setValues', ['001','002']);
setValue	value	设置组合框（combobox）的值。 代码实例： $('#cc').combobox('setValue', '001');
clear	none	清除组合框（combobox）的值。
select	value	选择指定的选项。
unselect	value	取消选择指定的选项。

8.5.1 基础组合框

一个普通的 select 标签，设置 class="easyui-combobox"

```
<select class="easyui-combobox" style="width:200px;">
    <option value="AL">Alabama</option>
    <option value="AK">Alaska</option>
    <option value="CA">California</option>
    <option value="CT">Connecticut</option>
    <option value="DE">Delaware</option>
</select>
```

8.5.2 多行和多选组合框

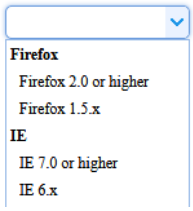


设置 multiple=true 可以多选，multiline 可以换行

```
<select class="easyui-combobox" name="state"
data-options="multiple:true,multiline:true"
        style="width:200px;height:50px">
    <option value="AL">Alabama</option>
    <option value="AK">Alaska</option>
    <option value="CA">California</option>
    <option value="CO">Colorado</option>
```

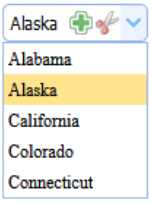
```
    <option value="CT">Connecticut</option>
</select>
```

8.5.3 组合框组




```
<input id="group"/>
<script type="text/javascript">
    $(function () {
        var data = [{
            "value": "f20",
            "text": "Firefox 2.0 or higher",
            "group": "Firefox"
        }, {
            "value": "f15",
            "text": "Firefox 1.5.x",
            "group": "Firefox"
        }, {
            "value": "ie7",
            "text": " IE7.0 or higher",
            "group": "IE"
        }, {
            "value": "ie6",
            "text": " IE 6.x",
            "group": "IE"
        }];
        $("#group").combobox({
            data:data,
            groupField:"group"
        })
    })
</script>
```

8.5.4 带图标的组合框




```
<select class="easyui-combobox" data-options="panelHeight:'auto',
                icons:[{
                    iconCls:'icon-add'
                },{
                    iconCls:'icon-cut'
                }]" style="width: 100px">
    <option value="AL" data-options="">Alabama</option>
    <option value="AK">Alaska</option>
    <option value="CA">California</option>
    <option value="CO">Colorado</option>
    <option value="CT">Connecticut</option>
</select>
```

8.5.5 动态加载组合框

```
<input id="cbb" class="easyui-combobox"/>
<input  type="button" value="加载"
        onclick='$("#cbb").combobox("loadData",
        [{
            "value": "f20",
            "text": "Firefox 2.0 or higher"
            }, {
            "value": "f15",
            "text": "Firefox 1.5.x"
        }])'/>
```

8.5.6 绑定远程数据

```
public class Category {
    private Integer id;
    private String name;
```

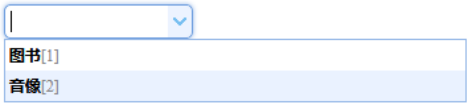
```
    //getter/setter 及构造方法略
}
```

controller 方法：

```
@RequestMapping("combobox.json")
@ResponseBody
public List<Category> data() {
    List<Category> list = new ArrayList<>();
    list.add(new Category(1, "图书"));
    list.add(new Category(2, "音像"));
    return list;
}
```

```
<input class="easyui-combobox"
       data-options="url:'/combobox.json',
                method:'post',
                valueField:'id',
                textField:'name'">
```

8.5.7 自定义样式




```
<input class="easyui-combobox"
       data-options="url:'/combobox.json',
                method:'post',
                valueField:'id',
                textField:'name',
                 panelWidth: 350,
                 panelHeight: 'auto',
                 formatter: formatItem">
<script type="text/javascript">
    function formatItem(row){
        var s = '<span style="font-weight:bold">' + row.name + '</span>' +
            '<span style="color:#888">[' + row.id + ']</span>';
        return s;
    }
</script>
```

8.5.8 远程 jsonp

注：本地模拟数据的话可以配个 host 模拟跨域。

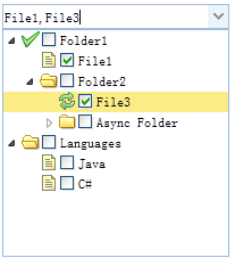
```
@RequestMapping("combobox_p.json")
@ResponseBody
public String regist(String callback) throws Exception {
    List<Category> list = new ArrayList<>();
    list.add(new Category(1, "图书"));
    list.add(new Category(2, "音像"));
    ObjectMapper mapper = new ObjectMapper();
    String jsonStr = mapper.writeValueAsString(list);
    return callback + "(" + jsonStr + ")";
}
```

注意 SpringMVC 中的 String 类型默认编码是 iso-8859-1，需要转成 utf-8 否则会乱码：

```
<mvc:annotation-driven>
    <!--String 返回值默认编码是 ISO-8859-1，需要-->
    <mvc:message-converters>
        <bean class="org.springframework.http.converter.StringHttpMessageConverter">
            <constructor-arg value="UTF-8" />
        </bean>
    </mvc:message-converters>
</mvc:annotation-driven>
```

```
<input class="easyui-combobox"
       data-options="loader: myloader,
                mode: 'remote',
               valueField:'id',
               textField:'name'
                ">
<script type="text/javascript">
    var myloader = function(param,success,error){
        $.ajax({
            url: 'http://www.myeasyui.com:8080/combobox_p.json',
            dataType: 'jsonp',
            success: function(data){
                success(data);
            },
            error: function(){
                error.apply(this, arguments);
            }
        });
    }
</script>
```

8.6 组合树 combotree



组合树（combotree）把选择控件和下拉树结合起来。它与组合框（combobox）相似，不同的是把列表替换成树组件。组合树（combotree）支持带有用于多选的树状态复选框的树。组合树依赖于 combo 和 tree。

8.6.1 基础组合树

```
<input class="easyui-combotree"
       data-options="url:'/tree.json',method:'post'" style="width:200px;">
```

8.6.2 多选组合树

```
<input class="easyui-combotree"

data-options="url:'/tree.json',method:'post',multiple:true,cascadeCheck:false"
style="width:200px;">
```

8.6.3 组合树的操作

```
<div style="margin:20px 0">
    <a href="javascript:void(0)"
onclick="alert($('#ct').combotree('getValues'))">GetValue</a>
    <a href="javascript:void(0)"
onclick="$('#ct').combotree('setValues',[1,2])">SetValue</a>
    <a href="javascript:void(0)" onclick="$('#ct').combotree('disable')">Disable</a>
    <a href="javascript:void(0)" onclick="$('#ct').combotree('enable')">Enable</a>
</div>
<input id="ct" class="easyui-combotree"
```

```
data-options="url:'/tree.json',method:'post',multiple:true,cascadeCheck:false"
style="width:200px;">
```

8.7 日历 calender

日历（calendar）显示允许用户选择日期的一个月份日历，并允许移动到上一月和下一页。默认情况下，每星期的第一天设置为星期日。这可以通过设置 'firstDay' 属性的值来改变。

属性

名称	类型	描述	默认值
width	number	日历（calendar）组件的宽度。	180
height	number	日历（calendar）组件的高度。	180
fit	boolean	当设置为 true 时，则设置日历的尺寸以适应它的父容器。	FALSE
border	boolean	定义是否显示边框。	TRUE
firstDay	number	定义每星期的第一天。星期日（Sunday）是 0，星期一（Monday）是 1，...	0
weeks	array	显示星期的列表。	['S','M','T','W','T','F','S']
months	array	显示月份的列表。	['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
year	number	日历的年。下面的实例演示如何使用指定的年和月来创建一个日历。 `<div class="easyui-calendar" data-options="year:2012,month:6" />`	当前年份（4 位）
month	number	日历的月。	当前月份（从 1 开始）
current	Date	当前的日期。	当前日期

事件：

名称	参数	描述

onSelect	date	当用户选择一个日期时触发。
		代码实例：
		`$('#cc').calendar({`
		`onSelect: function(date){`
		
		`alert(date.getFullYear()+":"+(date.getMonth()+1)+":"+date.getDate());`
		`}`
		`});`

方法：

名称	参数	描述
options	none	返回选项（options）对象。
resize	none	调整日历的尺寸。
moveTo	date	移动日历到一个指定的日期。 代码实例： `$('#cc').calendar('moveTo', new Date(2012, 6, 1));`

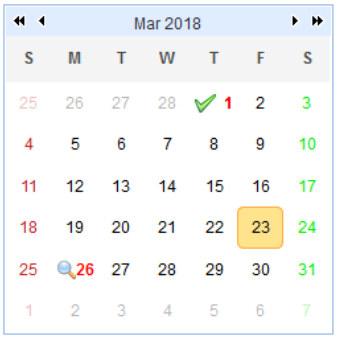
8.7.1 基本日历

```
<div class="easyui-calendar" style="width:180px;height: 180px"></div>
```

8.7.2 禁用日历日期

```
<div class="easyui-calendar" style="width:250px;height:250px;" data-options="
        validator: function(date){
            if (date.getDay() == 1){
                return true;
            } else {
                return false;
            }
        }
        "></div>
```

8.7.3 自定义日历




```
var d1 = Math.floor((Math.random()*30)+1);
var d2 = Math.floor((Math.random()*30)+1);
function formatDay(date){
    var m = date.getMonth()+1;
    var d = date.getDate();
    var opts = $(this).calendar('options');
    if (opts.month == m && d == d1){
        return '<div class="icon-ok md">' + d + '</div>';
    } else if (opts.month == m && d == d2){
        return '<div class="icon-search md">' + d + '</div>';
    }
    return d;
}
```

8.8 日期框

日期框（datebox）把可编辑的文本框和下拉日历面板结合起来，用户可以从下拉日历面板中选择日期。在文本框中输入的字符串可悲转换为有效日期。被选择的日期也可以被转换为期望的格式。依赖 combo 和 calendar。

属性：

该属性扩展自组合（combo），下面是为日期框（datebox）添加的属性。

名称	类型	描述	默认值
panelWidth	number	下拉日历面板的宽度。	180
panelHeight	number	下拉日历面板的高度。	auto
currentText	string	当前日期按钮上显示的文本。	Today
closeText	string	关闭按钮上显示的文本。	Close
okText	string	确定按钮上显示的文本。	Ok
disabled	boolean	设置为 true 时禁用该域。	FALSE

buttons	array	日历下面的按钮。该属性自版本 1.3.5 起可用。
		代码实例：
		`var buttons = $.extend([],`
		`$.fn.datebox.defaults.buttons);`
		`buttons.splice(1, 0, {`
		`text: 'MyBtn',`
		`handler: function(target){`
		`alert('click MyBtn');`
		`}`
		`});`
		`$('#dd').datebox({`
		`buttons: buttons`
		`});`	

sharedCalendar	string,selector	多个日期框（datebox）组件使用的共享日历。该属性自版本 1.3.5 起可用。
		代码实例：
		`<input class="easyui-datebox"`
		`sharedCalendar="#sc">`
		`<input class="easyui-datebox"`
		`sharedCalendar="#sc">`
		`<div id="sc" class="easyui-calendar"></div>`
			null

formatter	function	格式化日期的函数，该函数有一个 'date' 参数，并返回一个字符串值。下面的实例演示如何重写默认的格式化（formatter）函数。
		`$.fn.datebox.defaults.formatter = function(date){`
		`var y = date.getFullYear();`
		`var m = date.getMonth()+1;`
		`var d = date.getDate();`
		`return m+'/'+d+'/'+y;`
		`}`	

parser	function	解析日期字符串的函数，该函数有一个 'date' 字符串，并返回一个日期值。下面的实例演示如何重写默认的解析（parser）函数。	

		$.fn.datebox.defaults.parser = function(s){
		var t = Date.parse(s);
		if (!isNaN(t)){
		return new Date(t);
		} else {
		return new Date();
		}
		}	

事件：

名称	参数	描述

onSelect	date	当用户选择一个日期时触发。
		代码实例：
		$('#dd').datebox({
		onSelect: function(date){
		
		alert(date.getFullYear()+":"+(date.getMonth()+1)+":"+date.getDate());
		}
		});

方法：

该方法扩展自组合（combo），下面是为日期框（datebox）重写的方法。

名称	参数	描述
options	none	返回选项（options）对象。

calendar	none	获取日历（calendar）对象。下面的实例演示如何获取日历（calendar）对象，然后重现它。
		// get the calendar object
		var c = $('#dd').datebox('calendar');
		// set the first day of week to monday
		c.calendar({
		firstDay: 1
		});

setValue	value	设置日期框（datebox）的值。
		代码实例：
		$('#dd').datebox('setValue', '6/1/2012'); // set datebox value
		var v = $('#dd').datebox('getValue'); // get datebox value

8.8.1 基础日期框

```
<input class="easyui-datebox" type="text"/>
```

8.8.2 日期格式

```
$("#dd").datebox({
    required: true,
    formatter: function (date) {
        return date.getFullYear() + "-" + (date.getMonth() + 1) + "-" + date.getDate();
    },
    parser: function (s) {
        if (!s) return new Date();
        var ss = (s.split("-"));
        var y = parseInt(ss[0], 10);
        var m = parseInt(ss[1], 10);
        var d = parseInt(ss[2], 10);
        if (!isNaN(y) && !isNaN(m) && !isNaN(d)) {
            return new Date(y, m - 1, d);
        } else {
            return new Date();
        }
    }
});
```

8.8.3 限制时间范围

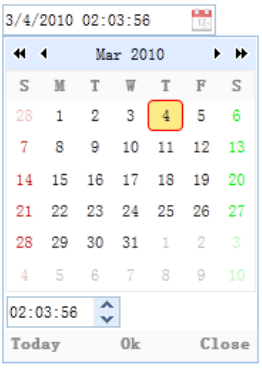
```
$("#dd").datebox().datebox("calendar").calendar({
    validator: function (date) {
        var now = new Date();
        return date > now;
    }
})
```

8.8.4 共享日历

```
<table>
    <tr>
        <td>Start Date:</td>
        <td>
            <input class="easyui-datebox" data-options="sharedCalendar:'#cc'">
        </td>
        <td>End Date:</td>
        <td>
```

```
            <input class="easyui-datebox" data-options="sharedCalendar:'#cc'">
        </td>
    </tr>
</table>
<div id="cc" class="easyui-calendar"></div>
```

8.9 日期时间框 datetimebox



与日期框（datebox）相似，日期时间框（datetimebox）允许用户选择要显示的带有指定格式的日期和时间的日期和时间。它向下拉面板添加一个时间微调器（timespinner）组件。依赖 datebox 和 timespinner。

属性：

该属性扩展自日期框（datebox），下面是为日期时间框（datetimebox）添加的属性。

名称	类型	描述	默认值
showSeconds	boolean	定义是否显示秒的信息。	TRUE
timeSeparator	string	时分秒之间的时间分隔符。该属性自版本 1.3 起可用。	:

方法：

该方法扩展自日期框（datebox），下面是为日期时间框（datetimebox）重写的方法。

名称	参数	描述
options	none	返回选项（options）对象。
spinner	none	返回时间微调器（timespinner）对象。

setValue	value	设置日期时间框（datetimebox）的值。
		代码实例：
		`$('#dt').datetimebox('setValue', '6/1/2012 12:30:56');`
		`// set datetimebox value`
		`var v = $('#dt').datetimebox('getValue');  // get datetimebox value`
		`alert(v);`

8.9.1 基础日期时间框

```
<input class="easyui-datetimebox" style="width: 200px" />
```

8.9.2 初始值和格式化

```
$("#dd").datetimebox({
    value: "2018-09-10 10:09:10",
    formatter: function (date) {
        return date.getFullYear() + "-" + (date.getMonth() + 1) + "-" + date.getDate()
+ " "
            + date.getHours() + ":" + date.getMinutes() + ":" + date.getSeconds();
    },
    parser: function (s) {
        if (!s) return new Date();
        var arr = s.split(" ");
        var ss = (arr[0].split("-"));
        var y = parseInt(ss[0], 10);
        var m = parseInt(ss[1], 10);
        var d = parseInt(ss[2], 10);
        var ss2 = arr[1].split(":");
        var h = parseInt(ss2[0], 10)
        var mm = parseInt(ss2[1], 10)
        var second = parseInt(ss2[2], 10)
        if (!isNaN(y) && !isNaN(m) && !isNaN(d) && !isNaN(h) && !isNaN(mm)
&& !isNaN(second)) {
            return new Date(y, m - 1, d, h, mm, second);
        } else {
            return new Date();
        }
    }
});
```

8.10 微调器 spinner

微调器（spinner）把可编辑的文本框和两个小按钮结合起来，允许用户从某个范围的值中进行选择。与组合框（combobox）相似，微调器（spinner）允许用户输入一个值，但是它买 i 有下拉列表。微调器（spinner）是创建其他微调器组件（比如：数值微调器 numberspinner、时间微调器 timespinner，等等）的基础组件。依赖 validatebox。

属性：

该属性扩展自验证框（validatebox），下面是为微调器（spinner）添加的属性。

名称	类型	描述	默认值
width	number	该组件的宽度。	auto
height	number	该组件的高度。该属性自版本 1.3.2 起可用。	22
value	string	初始值。	
min	string	允许的最小值。	null
max	string	允许的最大值。	null
increment	number	点击微调器按钮时的增量值。	1
editable	boolean	定义用户是否可以往文本域中直接输入值。	TRUE
disabled	boolean	定义是否禁用文本域。	FALSE
spin	function(down)	当用户点击微调按钮时调用的函数。'down' 参数指示用户是否点击了向下微调按钮。	

事件：

名称	参数	描述
onSpinUp	none	当用户点击向上微调按钮时触发。
onSpinDown	none	当用户点击向下微调按钮时触发。

方法:

该方法扩展自验证框（validatebox），下面是为微调器（spinner）添加的方法。

名称	参数	描述
options	none	返回选项（options）对象。
destroy	none	销毁微调器（spinner）组件。

resize	width	重置组件的宽度。通过传递 'width' 参数来重写初始宽度。
		代码实例：
		`$('#ss').spinner('resize'); // resize with original width`
		`$('#ss').spinner('resize', 200); // resize with new width`

enable	none	启用组件。
disable	none	禁用组件。
getValue	none	获取组件的值。
setValue	value	设置组件的值。
clear	none	清除组件的值。
reset	none	重置组件的值。该方法自版本 1.3.2 起可用。

8.10.1基础微调器

```
<input id="ss" value="2">
```

```
$('#ss').spinner({
    required:true,
```

```
    increment:10
});
```

8.11 时间微调 timespinner

时间微调器（timespinner）是基于微调器（spinner）创建的。它与数值微调器（numberspinner）相似，但是它只显示时间值。时间微调器（timespinner）允许用户通过点击组件右侧的小微调按钮来增加或减少时间。依赖 spinner。

属性：

该属性扩展自微调器（spinner），下面是为时间微调器（timespinner）添加的属性。

名称	类型	描述	默认值
separator	string	时分秒之间的分隔符。	:
showSeconds	boolean	定义是否显示秒的信息。	FALSE
highlight	number	初始高亮的域，0 = 时，1 = 分，...	0

事件：

该事件扩展自微调器（spinner）。

方法：

该方法扩展自微调器（spinner），下面是为时间微调器（timespinner）重写的方法。

名称	参数	描述
options	none	返回选项（options）对象。

setValue	value	设置时间微调器（timespinner）的值。
		代码实例：
		`$('#ss').timespinner('setValue', '17:45');  // set timespinner value`
		`var v = $('#ss').timespinner('getValue');  // get timespinner value`
		`alert(v);`

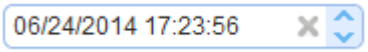
getHours	none	获取当前的时钟的值。
getMinutes	none	获取当前的分钟的值。
getSeconds	none	获取当前的秒钟的值。

8.11.1基础时间微调

```
<input type="button" value="赋值"
onclick="$('#dd').timespinner('setValue','09:45')"/>
<input type="button" value="取值" onclick="alert($('#dd').timespinner('getValue'))"/
<input type="button" value="禁用 " onclick="$('#dd').timespinner('disable')"/>
```

```
<input type="button" value="启用" onclick="$('#dd').timespinner('enable')"/>
<input id="dd" class="easyui-timespinner" style="width: 20%"
 data-options="max:'18:00',min:'09:00'"
/>
```

8.12 时间日期微调 datetimespinner




```
<input class="easyui-datetimespinner" style="width: 200px" data-options="
icons:[{
iconCls:'icon-clear',
handler:function(e){
$(e.data.target).datetimespinner('clear')
}
}]
"/>
```

8.13 数字调节器 numberspinner

数值微调器（numberspinner）是基于微调器（spinner）和数字框（numberbox）创建的。它可以把输入值转换为不同类型（比如：数字 numeric、百分比 percentage、货币 currency，等等）。它允许用户使用向上/向下微调按钮滚动到一个期望值。依赖 spinner 和 numberbox。

属性：

该属性扩展自微调器（spinner）和数字框（numberbox）。

事件：

该事件扩展自微调器（spinner）。

方法：

该方法扩展自微调器（spinner），下面是为数值微调器（numberspinner）重写的方法。

名称	参数	描述
options	none	返回选项（options）对象。

setValue	value	设置数值微调器（numberspinner）的值。
		代码实例：
		`$('#ss').numberspinner('setValue', 8234725);  // set value`
		`var v = $('#ss').numberspinner('getValue');  // get value`
		`alert(v)`

```
<input class="easyui-numberspinner" style="width: 80px"
data-options="min:0,max:100,increment:10,editable:false"
/>
```

8.14 slider 滑块

滑块（slider）允许用户从一个有限的范围内选择一个数值。当沿着轨道移动滑块控件时，将显示一个表示当前值的提示框，用户可通过设置它的属性来自定义滑块。依赖 draggable。

属性：

名称	类型	描述	默认值
width	number	滑块（slider）的宽度。	auto
height	number	滑块（slider）的高度。	auto
mode	string	只是滑块（slider）的类型。可能的值：'h'(horizontal)、'v'(vertical)。	h
reversed	boolean	当设置为 true 时，将会对调最小值和最大值的位置。该属性自版本 1.3.2 起可用。	FALSE
showTip	boolean	定义是否显示值信息提示框。	FALSE
disabled	boolean	定义是否禁用滑块（slider）。	FALSE
value	number	默认值。	0
min	number	允许的最小值。	0
max	number	允许的最大值。	100
step	number	增加或减少的值。	1
rule	array	在滑块旁边显示标签，'|' — 值显示线。	[]
tipFormatter	function	格式化滑块值的函数。返回作为提示框显示的字符串值。	

事件：

名称	参数	描述
onChange	newValue,oldValue	当文本域的值改变时触发。
onSlideStart	value	当开始拖拽滑块时触发。
onSlideEnd	value	当停止拖拽滑块时触发。
onComplete	value	当滑块的值被用户改变时触发，无论是通过拖拽滑块改变还是通过点击滑块改变都会触发。该事件自版本 1.3.4 起可用。

方法：

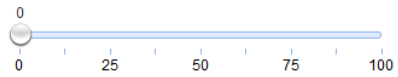
名称	参数	描述

options	none	返回滑块（slider）选项（options）。
destroy	none	销毁滑块（slider）对象。
resize	param	设置滑块尺寸。’param’ 参数包含下列属性：
		width：新的滑块宽度
		height：新的滑块高度
getValue	none	获取滑块（slider）的值。
setValue	value	设置滑块（slider）的值。
clear	none	清除滑块（slider）的值。该方法自版本 1.3.5 起可用。
reset	none	重置滑块（slider）的值。该方法自版本 1.3.5 起可用。
enable	none	启用滑块（slider）组件。
disable	none	禁用滑块（slider）组件。

8.14.1基础滑块

```
<input class="easyui-slider" style="width:300px" data-options="showTip:true">
```

8.14.2滑块规则




```
<input id="ss" class="easyui-slider" value="12" style="height:200px;"

data-options="showTip:true,rule:[0,'|',25,'|',50,'|',75,,'|',100],step:10,mode:'v',
tipFormatter:function(value){
return value+'%'
}"/>
```

8.14.3垂直滑块

```
mode:'v'
```

8.14.4滑块操作

```
<input type="button" value="取值" onclick="alert($('#ss').slider('getValue'))"/>
<input type="button" value="禁用" onclick="$('#ss').slider('disable')"/>
<input type="button" value="启用" onclick="$('#ss').slider('enable')"/>
```

8.15 表单 form

表单（form）提供多种方法来执行带有表单字段的动作，比如 ajax 提交、加载、清除，等等。当提交表单时，调用 'validate' 方法来检查表单是否有效。

属性：

名称	类型	描述	默认值
url	string	要提交的表单动作 URL。	null

事件：

名称	参数	描述
onSubmit	param	提交前触发，返回 false 来阻止提交动作。
success	data	当表单提交成功时触发。
onBeforeLoad	param	发出请求加载数据之前触发。返回 false 就取消这个动作。
onLoadSuccess	data	当表单数据加载时触发。
onLoadError	none	加载表单数据时发生某些错误的时候触发。

方法：

名称	参数	描述
submit	options	做提交动作，options 参数是一个对象，它包含下列属性： url：动作的 URL onSubmit：提交之前的回调函数 success：提交成功之后的回调函数 下面的实例演示如何提交一个有效表单，避免重复提交表单。
load	data	加载记录来填充表单。data 参数可以是一个字符串或者对象类型，字符串作为一个远程 URL，否则作为一个本地记录。

```
$.messager.progress();  // display the progress bar
$('#ff').form('submit', {
    url: "",
    onSubmit:
        function () {
            var isValid = $(this).form('validate');
            if (!isValid) {
                $.messager.progress('close');  // hide
progress bar while the form is invalid
            }
            return isValid;    // return false will stop the
form submission
        },
    success: function () {
        $.messager.progress('close');  // hide progress bar
while submit successfully
    }
});
```

		代码实例：
		`$('#ff').form('load','get_data.php');  // load from URL`
		
		`$('#ff').form('load',{`
		`  name:'name2',`
		`  email:'mymail@gmail.com',`
		`  subject:'subject2',`
		`  message:'message2',`
		`  language:5`
		`});`

clear	none	清除表单数据。
reset	none	重置表单数据。该方法自版本 1.3.2 起可用。
validate	none	进行表单字段验证，当全部字段都有效时返回 true 。该方法和 validatebox 插件一起使用。
enableValidation	none	启用验证。该方法自版本 1.3.4 起可用。
disableValidation	none	禁用验证。该方法自版本 1.3.4 起可用。

8.15.1基础表单

```
<form id="ff" method="post" action="/form.json">
    <table cellpadding="5">
        <tr>
            <td>Name:</td>
            <td><input class="easyui-textbox" type="text" name="name"
data-options="required:true"/></td>
        </tr>
        <tr>
            <td>Birthday:</td>
            <td><input class="easyui-datebox" name="birthday"/>
            </td>
        </tr>
        <tr>
            <td>Language:</td>
            <td>
                <select class="easyui-combobox" name="language">
                    <option value="ar">Arabic</option>
                    <option value="bg">Bulgarian</option>
                    <option value="ca">Catalan</option>
                </select>
```

```
            </td>
        </tr>
    </table>
</form>
<div>
    <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="submitForm()">Submit</a>
    <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="$('#ff').form('clear')">Clear</a>
    <a href="javascript:void(0)" class="easyui-linkbutton"
onclick="$('#ff').form('reset')">Reset</a>
</div>
```

```
<script type="text/javascript">
    function submitForm() {
        // $('#ff').submit();
        //$('#ff').form('submit');//ajax 提交
    }
</script>
```

8.15.2验证表单提交

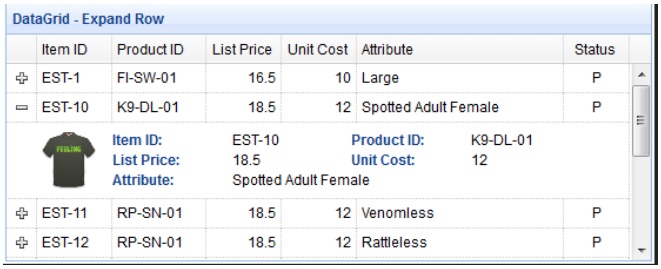
```
$('#ff').form('submit', {
    onSubmit: function () {
        //获得表单的验证状态
        return $(this).form('enableValidation').form('validate');
    }
});
```

8.15.3加载表单数据

```
$(function () {
    $('#ff').form('load', {
        name: 'name2',
        language: "ca",
        birthday: "03/8/2018"
    });
})
```

9.表格

9.1 数据表格 datagrid



数据网格（datagrid）以表格格式显示数据，并为选择、排序、分组和编辑数据提供了丰富的支持。数据网格（datagrid）的设计目的是为了减少开发时间，且不要求开发人员具备指定的知识。它是轻量级的，但是功能丰富。它的特性包括单元格合并，多列页眉，冻结列和页脚，等等。表格依赖于 panel,resizable,linkbutton,pagination。

属性：

该属性扩展自面板（panel），下面是为数据网格（datagrid）添加的属性。

名称	类型	描述	默认值
columns	array	数据网格（datagrid）的列（column）的配置对象，更多细节请参见列（column）属性。	undefined
frozenColumns	array	和列（column）属性一样，但是这些列将被冻结在左边。	undefined
fitColumns	boolean	设置为 true,则会自动扩大或缩小列的尺寸以适应网格的宽度并且防止水平滚动。	FALSE
resizeHandle	string	调整列的位置,可用的值有:'left'、'right'、'both'。当设置为 'right' 时,用户可通过拖拽列头部的右边缘来调整列。 该属性自版本 1.3.2 起可用。	right
autoRowHeight	boolean	定义是否设置基于该行内容的行高度。设置为 false，则可以提高加载性能。	TRUE

toolbar	array,selector	数据网格（datagrid）面板的头部工具栏。可能的值：
		1、数组，每个工具选项与链接按钮（linkbutton）一样。
		2、选择器，只是工具栏。
		在 标签内定义工具栏：
		`$('#dg').datagrid({`
		`  toolbar: '#tb'`
		`});`
			null

		
		
		
		通过数组定义工具栏：
		$('#dg').datagrid({
		  toolbar: [{
		  iconCls: 'icon-edit',
		  handler: function(){alert('edit')}
		  },'-',{
		  iconCls: 'icon-help',
		  handler: function(){alert('help')}
		  }]
		});	

striped	boolean	设置为 true，则把行条纹化。（即奇偶行使用不同背景色）	FALSE
method	string	请求远程数据的方法（method）类型。	post
nowrap	boolean	设置为 true，则把数据显示在一行里。设置为 true 可提高加载性能。	TRUE
idField	string	指示哪个字段是标识字段。	null
url	string	从远程站点请求数据的 URL。	null

data	array,object	要加载的数据。该属性自版本 1.3.2 起可用。
		代码实例：
		$('#dg').datagrid({
		  data: [
		  {f1:'value11', f2:'value12'},
		  {f1:'value21', f2:'value22'}
		  ]
		});
			null

loadMsg	string	当从远程站点加载数据时，显示的提示消息。	Processing, please wait …
pagination	boolean	设置为 true，则在数据网格（datagrid）底部显示分页工具栏。	FALSE
rownumbers	boolean	设置为 true，则显示带有行号的列。	FALSE
singleSelect	boolean	设置为 true，则只允许选中一行。	FALSE
checkOnSelect	boolean	如果设置为 true，当用户点击某一行时，则会选中/取消选中复选框。如果设置为 false 时，只有当用户点击了复选框时，才会选中/取消选中复选框。	TRUE

		该属性自版本 1.3 起可用。	
selectOnCheck	boolean	如果设置为 true，点击复选框将会选中该行。如果设置为 false，选中该行将不会选中复选框。 该属性自版本 1.3 起可用。	TRUE
pagePosition	string	定义分页栏的位置。可用的值有：'top'、'bottom'、'both'。 该属性自版本 1.3 起可用。	bottom
pageNumber	number	当设置了 pagination 属性时，初始化页码。	1
pageSize	number	当设置了 pagination 属性时，初始化页面尺寸。	10
pageList	array	当设置了 pagination 属性时，初始化页面尺寸的选择列表。	[10,20,30,40,50]

queryParams	object	当请求远程数据时，发送的额外参数。
		代码实例：
		`$('#dg').datagrid({`
		`queryParams: {`
		`name: 'easyui',`
		`subject: 'datagrid'`
		`}`
		`});`
			{}

sortName	string	定义可以排序的列。	null
sortOrder	string	定义列的排序顺序，只能用 'asc' 或 'desc'。	asc
multiSort	boolean	定义是否启用多列排序。该属性自版本 1.3.4 起可用。	FALSE
remoteSort	boolean	定义是否从服务器排序数据。	TRUE
showHeader	boolean	定义是否显示行的头部。	TRUE
showFooter	boolean	定义是否显示行的底部。	FALSE
scrollbarSize	number	滚动条宽度（当滚动条是垂直的时候）或者滚动条的高度（当滚动条是水平的时候）。	18

rowStyler	function	返回例如 'background:red' 的样式。该函数需要两个参数：
		rowIndex：行的索引，从 0 开始。
		rowData：该行相应的记录。
		代码实例：
		`$('#dg').datagrid({`
		`rowStyler: function(index,row){`
		`if (row.listprice>80){`
		`return 'background-color:#6293BB;color:#fff;'; // return inline style`
		`// the function can return predefined css class and inline style`
		`// return {class:'r1', style:{'color:#fff'}};`
		`}`	

		} });	
loader	function	定义如何从远程服务器加载数据。返回 false 则取消该动作。该函数有下列参数： param：要传递到远程服务器的参数对象。 success(data)：当检索数据成功时调用的回调函数。 error()：当检索数据失败时调用的回调函数。	json loader

loadFilter	function	返回要显示的过滤数据。该函数有一个参数 'data'，表示原始数据。您可以把原始数据变成标准数据格式。该函数必须返回标准数据对象，含有 'total' 和 'rows' 属性。
		代码实例：
		// removing 'd' object from asp.net web service json output
		$('#dg').datagrid({
		loadFilter: function(data){
		if (data.d){
		return data.d;
		} else {
		return data;
		}
		}
		});	

editors	object	定义编辑行时的编辑器。	predefined editors
view	object	定义数据网格（datagrid）的视图。	default view

列属性：

数据网格（DataGrid） 的列（Column）是一个数组对象，它的每个元素也是一个数组。元素数组的元素是一个配置对象，它定义了每个列的字段。

名称	类型	描述	默认值
title	string	列的标题文本。	undefined
field	string	列的字段名。	undefined
width	number	列的宽度。如果未定义，则宽度会自动扩展以适应它的内容。没有定义宽度将会降低性能。	undefined
rowspan	number	指示一个单元格占据多少行。	undefined
colspan	number	指示一个单元格占据多少列。	undefined
align	string	指示如何对齐该列的数据，可以用 'left'、'right'、'center'。	undefined
halign	string	指示如何对齐该列的头部，可能的值：'left'、'right'、'center'。如果没有分配值，则头部对齐方式将与通过 'align' 属性定义的数据对齐方式一致。该属性自版本 1.3.2 起可用。	undefined

sortable	boolean	设置为 true，则允许该列被排序。	undefined
order	string	默认的排序顺序，只能用 'asc' 或 'desc'。该属性自版本 1.3.2 起可用。	undefined
resizable	boolean	设置为 true，则允许该列可调整尺寸。	undefined
fixed	boolean	设置为 true，则当 'fitColumns' 设置为 true 时放置调整宽度。	undefined
hidden	boolean	设置为 true，则隐藏该列。	undefined
checkbox	boolean	设置为 true，则显示复选框。复选框有固定宽度。	undefined

formatter	function	单元格的格式化函数，需要三个参数：
		value：字段的值。
		rowData：行的记录数据。
		rowIndex：行的索引。
		代码实例：
		`$('#dg').datagrid({`
		`columns:[[`
		`{field:'userId',title:'User', width:80,`
		`formatter:`
		`function(value,row,index){`
		`if (row.user){`
		`return row.user.name;`
		`} else {`
		`return value;`
		`}`
		`}`
		`}`
		`]]`
		`});`
			undefined

styler	function	单元格的样式函数，返回样式字符串来自定义该单元格的样式，例如 'background:red' 。该函数需要三个参数：
		value：字段的值。
		rowData：行的记录数据。
		rowIndex：行的索引。
		代码实例：
		`$('#dg').datagrid({`
		`columns:[[`
		`{field:'listprice',title:'List Price',`
		`width:80, align:'right',`
		`styler: function(value,row,index){`
		`if (value < 20){`
		`return`
		`'background-color:#ffee00;color:red;';`
		`// the function can return predefined css class`
		`and inline style`
			undefined

		// return {class:'c1',style:'color:red'}
		}
		}
		}
		]]
		});	

sorter	function	用于本地排序的自定义字段的排序函数，需要两个参数：
		a：第一个字段值。
		b：第二个字段值。
		代码实例：
		$('#dg').datagrid({
		remoteSort: false,
		columns: [[
		
		{field:'date',title:'Date',width:80,sortable:true,align:'center',
		sorter:function(a,b){
		a = a.split('/');
		b = b.split('/');
		if (a[2] == b[2]){
		if (a[0] == b[0]){
		return (a[1]>b[1]?1:-1);
		} else {
		return (a[0]>b[0]?1:-1);
		}
		} else {
		return (a[2]>b[2]?1:-1);
		}
		}
		}
		]]
		});
			undefined

editor	string,object	指示编辑类型。当是字符串（string）时则指编辑类型，当是对象（object）时则包含两个属性： type：字符串，编辑类型，可能的类型：text、textarea、checkbox、numberbox、validatebox、datebox、combobox、combotree。 options：对象，编辑类型对应的编辑器选项。	undefined

编辑器：

通过 $.fn.datagrid.defaults.editors 重写默认的 defaults。

每个编辑器有下列行为：

名称	参数	描述
init	container, options	初始化编辑器并且返回目标对象。
destroy	target	如果必要就销毁编辑器。
getValue	target	从编辑器的文本获取值。
setValue	target , value	给编辑器设置值。
resize	target , width	如果必要就调整编辑器的尺寸。

数据网格视图：

通过 $.fn.datagrid.defaults.view 重写默认的 defaults。

视图（view）是一个对象，它告诉数据网格（datagrid）如何呈现行。该对象必须定义下列函数：

名称	参数	描述
render	target, container, frozen	当数据加载时调用。
		target：DOM 对象，数据网格（datagrid）对象。
		container：行的容器。
		frozen：指示是否呈现冻结容器。
renderFooter	target, container, frozen	这是呈现行脚的选项函数。
renderRow	target, fields, frozen, rowIndex, rowData	这是将会被 render 函数调用的选项函数。
refreshRow	target, rowIndex	定义如何刷新指定的行。
onBeforeRender	target, rows	视图被呈现前触发。
onAfterRender	target	视图被呈现后触发。

事件：

该事件扩展自面板（panel），下面是为数据网格（datagrid）添加的事件。

名称	参数	描述
onLoadSuccess	data	当数据加载成功时触发。
onLoadError	none	加载远程数据发生某些错误时触发。
onBeforeLoad	param	发送加载数据的请求前触发，如果返回 false 加载动作就会取消。
onClickRow	rowIndex, rowData	当用户点击一行时触发，参数包括：
		rowIndex：被点击行的索引，从 0 开始
		rowData：被点击行对应的记录
onDblClickRow	rowIndex, rowData	当用户双击一行时触发，参数包括：
		rowIndex：被双击行的索引，从 0 开始
		rowData：被双击行对应的记录
onClickCell	rowIndex, field, value	当用户单击一个单元格时触发。
onDblClickCell	rowIndex, field, value	当用户双击一个单元格时触发。
		代码实例：
		// when double click a cell, begin editing and make the

		editor get focus
		$('#dg').datagrid({
		onDblClickCell: function(index,field,value){
		$(this).datagrid('beginEdit', index);
		var ed = $(this).datagrid('getEditor',
		{index:index,field:field});
		$(ed.target).focus();
		}
		});

onSortColumn	sort, order	当用户对一列进行排序时触发，参数包括： sort：排序的列的字段名 order：排序的列的顺序
onResizeColumn	field, width	当用户调整列的尺寸时触发。
onSelect	rowIndex, rowData	当用户选中一行时触发，参数包括： rowIndex：选中行的索引，从 0 开始 rowData：选中行对应的记录
onUnselect	rowIndex, rowData	当用户取消选中一行时触发，参数包括： rowIndex：取消选中行的索引，从 0 开始 rowData：取消选中行对应的记录
onSelectAll	rows	当用户选中全部行时触发。
onUnselectAll	rows	当用户取消选中全部行时触发。
onCheck	rowIndex, rowData	当用户勾选一行时触发，参数包括： rowIndex：勾选行的索引，从 0 开始 rowData：勾选行对应的记录 该事件自版本 1.3 起可用。
onUncheck	rowIndex, rowData	当用户取消勾选一行时触发，参数包括： rowIndex：取消勾选行的索引，从 0 开始 rowData：取消勾选行对应的记录 该事件自版本 1.3 起可用。
onCheckAll	rows	当用户勾选全部行时触发。该事件自版本 1.3 起可用。
onUncheckAll	rows	当用户取消勾选全部行时触发。该事件自版本 1.3 起可用。
onBeforeEdit	rowIndex, rowData	当用户开始编辑一行时触发，参数包括： rowIndex：编辑行的索引，从 0 开始 rowData：编辑行对应的记录
onAfterEdit	rowIndex, rowData, changes	当用户完成编辑一行时触发，参数包括： rowIndex：编辑行的索引，从 0 开始 rowData：编辑行对应的记录 changes：更改的字段/值对
onCancelEdit	rowIndex, rowData	当用户取消编辑一行时触发，参数包括： rowIndex：编辑行的索引，从 0 开始 rowData：编辑行对应的记录
onHeaderContex	e, field	当数据网格（datagrid）的头部被右键单击时触发。

tMenu		
onRowContextMenu	e, rowIndex, rowData	当右键点击行时触发。

方法：

名称	参数	描述
options	none	返回选项（options）对象。
getPager	none	返回分页（pager）对象。
getPanel	none	返回面板（panel）对象。

getColumnFields	frozen	返回列的字段，如果 frozen 设置为 true，则冻结列的字段被返回。
		代码实例：
		`var opts = $('#dg').datagrid('getColumnFields');  // get unfrozen columns`
		`var opts = $('#dg').datagrid('getColumnFields', true); // get frozen columns`

getColumnOption	field	返回指定列的选项。
resize	param	调整尺寸和布局。

load	param	加载并显示第一页的行，如果指定 'param' 参数，它将替换 queryParams 属性。通常情况下，通过传递一些从参数进行查询，该方法被调用来从服务器加载新数据。
		`$('#dg').datagrid('load',{`
		`  code:'01',`
		`  name:'name01'`
		`});`

reload	param	重新加载行，就像 load 方法一样，但是保持在当前页。

reloadFooter	footer	重新加载底部的行。代码实例：
		`// update footer row values and then refresh`
		`var rows = $('#dg').datagrid('getFooterRows');`
		`rows[0]['name'] = 'new name';`
		`rows[0]['salary'] = 60000;`
		`$('#dg').datagrid('reloadFooter');`
		
		`// update footer rows with new data`
		`$('#dg').datagrid('reloadFooter',[`
		`  {name:'name1', salary:60000},`
		`  {name:'name2', salary:65000}`
		`]);`

loading	none	显示正在加载状态。
loaded	none	隐藏正在加载状态。
fitColumns	none	使列自动展开/折叠以适应数据网格（datagrid）的宽度。
fixColumnSize	field	固定列的尺寸。如果 'field' 参数未设置，所有的列的尺寸将是固定的。

		代码实例：
		`$('#dg').datagrid('fixColumnSize', 'name'); // fix the 'name' column size`
		`$('#dg').datagrid('fixColumnSize'); // fix all columns size`

fixRowHeight	index	固定指定行的高度。如果 'index' 参数未设置，所有的行的高度将是固定的。
freezeRow	index	冻结指定的行，以便数据网格（datagrid）向下滚动时这些冻结行总是被显示在顶部。该方法自版本 1.3.2 起可用。
autoSizeColumn	field	调整列的宽度以适应内容。该方法自版本 1.3 起可用。
loadData	data	加载本地数据，旧的行会被移除。
getData	none	返回加载的数据。
getRows	none	返回当前页的行。
getFooterRows	none	返回底部的行。
getRowIndex	row	返回指定行的索引，row 参数可以是一个行记录或者一个 id 字段的值。
getChecked	none	返回复选框选中的所有行。该方法自版本 1.3 起可用。
getSelected	none	返回第一个选中的行或者 null。
getSelections	none	返回所有选中的行，当没有选中的记录时，将返回空数组。
clearSelections	none	清除所有的选择。
clearChecked	none	清除所有勾选的行。该方法自版本 1.3.2 起可用。
scrollTo	index	滚动到指定行。该方法自版本 1.3.3 起可用。
highlightRow	index	高亮显示一行。该方法自版本 1.3.3 起可用。
selectAll	none	选中当前页所有的行。
unselectAll	none	取消选中当前页所有的行。
selectRow	index	选中一行，行索引从 0 开始。
selectRecord	idValue	通过传递 id 的值做参数选中一行。
unselectRow	index	取消选中一行。
checkAll	none	勾选当前页所有的行。该方法自版本 1.3 起可用。
uncheckAll	none	取消勾选当前页所有的行。该方法自版本 1.3 起可用。
checkRow	index	勾选一行，行索引从 0 开始。该方法自版本 1.3 起可用。
uncheckRow	index	取消勾选一行，行索引从 0 开始。该方法自版本 1.3 起可用。
beginEdit	index	开始对一行进行编辑。
endEdit	index	结束对一行进行编辑。
cancelEdit	index	取消对一行进行编辑。
getEditors	index	获取指定行的编辑器。每个编辑器有下列属性： actions：编辑器能做的动作，与编辑器定义相同。 target：目标编辑器的 jQuery 对象。 field：字段名。 type：编辑器的类型，比如：'text'、'combobox'、'datebox'，等等。
getEditor	option	获取指定的编辑器， options 参数包含两个属性：

	s	index：行的索引。
		field：字段名。
		代码实例：

		`// get the datebox editor and change its value`
		`var ed = $('#dg').datagrid('getEditor',`
		`{index:1,field:'birthday'});`
		`$(ed.target).datebox('setValue', '5/4/2012');`

refreshRow	index	刷新一行。
validateRow	index	验证指定的行，有效时返回 true。
updateRow	param	更新指定的行， param 参数包括下列属性：
		index：要更新的行的索引。
		row：新的行数据。
		代码实例：

		`$('#dg').datagrid('updateRow',{`
		`index: 2,`
		`row: {`
		`name: 'new name',`
		`note: 'new note message'`
		`}`
		`});`

appendRow	row	追加一个新行。新的行将被添加在最后的位置：

		`$('#dg').datagrid('appendRow',{`
		`name: 'new name',`
		`age: 30,`
		`note: 'some messages'`
		`});`

insertRow	param	插入一个新行， param 参数包括下列属性：
		index：插入进去的行的索引，如果没有定义，就追加该新行。
		row：行的数据。
		代码实例：

		`// insert a new row at second row position`
		`$('#dg').datagrid('insertRow',{`
		`index: 1, // index start with 0`
		`row: {`
		`name: 'new name',`
		`age: 30,`
		`note: 'some messages'`
		`}`
		`});`

deleteRow	index	删除一行。
getChanges	type	获取最后一次提交以来更改的行，type 参数表示更改的行的类型，可能的值是：inserted、deleted、updated，等等。当 type 参数没有分配时，返回所有改变的行。
acceptChanges	none	提交自从被加载以来或最后一次调用 acceptChanges 以来所有

		更改的数据。
rejectChanges	none	回滚自从创建以来或最后一次调用 acceptChanges 以来所有更改的数据。
mergeCells	options	把一些单元格合并为一个单元格，options 参数包括下列特性：
		index：列的索引。
		field：字段名。
		rowspan：合并跨越的行数。
		colspan：合并跨越的列数。
showColumn	field	显示指定的列。
hideColumn	field	隐藏指定的列。

9.1.1 HTML 表格转 datagrid

将一个 html 表格转换成 datagrid,只需要在 table 上添加 class=”easyui-datagrid”。注意 datagrid 每一列需要有一个 field 属性，区分每一列的值。

```
<table class="easyui-datagrid" style="width:400px;height:auto;">
    <thead>
    <tr>
        <th field="name1">Col 1</th><th field="name2">Col 2</th>
        <th field="name3">Col 3</th><th field="name4">Col 4</th>
        <th field="name5">Col 5</th><th field="name6">Col 6</th>
    </tr>
    </thead>
    <tbody>
    <tr>
        <td>Data 1</td><td>Data 2</td><td>Data 3</td>
        <td>Data 4</td><td>Data 5</td><td>Data 6</td>
    </tr>
    <tr>
        <td>Data 1</td><td>Data 2</td><td>Data 3</td>
        <td>Data 4</td><td>Data 5</td><td>Data 6</td>
    </tr>
    </tbody>
</table>
```

运行效果如图：

Col 1	Col 2	Col 3	Col 4	Col 5	Col 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6

也可以定义复杂的表头：

Col 1	Col 2	Col 3	Details		
			Col 4	Col 5	Col 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
Data 1	Data 2	Data 3	Data 4	Data 5	Data 6

```
<thead>
<tr>
    <th field="name1" rowspan="2">Col 1</th>
    <th field="name2" rowspan="2">Col 2</th>
    <th field="name3" rowspan="2">Col 3</th>
    <th colspan="3">Details</th>
</tr>
<tr>
    <th field="name4">Col 4</th>
    <th field="name5">Col 5</th>
    <th field="name6">Col 6</th>
</tr>
</thead>
```

9.1.2 绑定动态数据

后台代码：

```
@RequestMapping("category.json")
@ResponseBody
public List<Category> category() {
    List<Category> list = new ArrayList<>();
    list.add(new Category(1, "小学"));
    list.add(new Category(2, "初中"));
    return list;
}
```

返回的数据格式：

```
[{
    "id": 1,
    "name": "小学"
},{
    "id": 2,
    "name": "初中"
}]
```

页面上使用 url 属性设置请求的地址，表头的 field 属性对应 json 数据中的属性名。

```
<table class="easyui-datagrid" data-options="url:'/category.json',method:'post'"
style="width:100px">
    <thead>
    <tr>
```

```
        <th field="id">ID</th>
        <th field="name">分类</th>
    </tr>
    </thead>
</table>
```

使用 js 初始化表格：

```
<table id="dg" style="width:100px">
</table>
<script type="text/javascript">
    $(function () {
        $("#dg").datagrid({
            url: '/category.json',
            method: 'post',
            columns:[[
                {field:'id',title:'ID'},
                {field:'name',title:'分类'}
            ]]
        });
    })
</script>
```

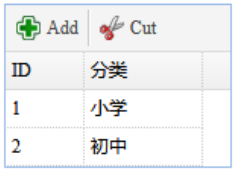
9.1.3 格式化列

ID	分类
1	小学[1]
2	初中[2]

使用 columns 的 formatter 属性可以格式化列，可以返回 html 标签，如显示<img>，row 是当前 json 数据中的某个对象，value 是对应着 field 属性的从 json 对象中读取出的值。

```
$("#dg").datagrid({
    url: '/category.json',
    method: 'post',
    columns: [[
        {field: 'id', title: 'ID'},
        {
            field: 'name', title: '分类', formatter: function (value, row, index) {
            return value + '<span style="color:red">[' + row.id + ']</span>'
        }
        }
    ]]
});
```

9.1.4 工具栏



通过数组定义工具栏：

```
<table class="easyui-datagrid"
data-options="url:'/category.json',method:'post',toolbar: [{
                text: 'Add',
                iconCls: 'icon-add',
                handler: function () {
                    alert('add')
                }
            },'-', {
                text: 'Cut',
                iconCls: 'icon-cut',
                handler: function () {
                    alert('cut')
                }
            }]" style="width:150px">
    <thead>
    <tr>
        <th field="id" width="40%">ID</th>
        <th field="name" width="60%">分类</th>
    </tr>
    </thead>
</table>
```

通过引用标签定义工具栏：

```
<table class="easyui-datagrid"
data-options="url:'/category.json',method:'post',toolbar:'#tb' " style="width:150px"
    <thead>
    <tr>
        <th field="id" width="40%">ID</th>
        <th field="name" width="60%">分类</th>
    </tr>
    </thead>
</table>
<div id="tb">
    <a href="#" class="easyui-linkbutton"
```

```
data-options="iconCls:'icon-add',plain:true">Add</a>
    <a href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-cut',plain:true">Cut</a>
</div>
```

9.1.5 添加复选框

	ID	分类
	1	小学
	2	初中

```
<table class="easyui-datagrid" data-options="url:'/category.json',method:'post'"
style="width:170px">
    <thead>
    <tr>
        <th field="ck" checkbox="true"></th>
        <th field="id" width="40%">ID</th>
        <th field="name" width="50%">分类</th>
    </tr>
    </thead>
</table>
```

table 上配合使用的有两个属性：

checkOnSelect：如果设置为 true，当用户点击某一行时，则会选中/取消选中复选框。如果设置为 false 时，只有当用户点击了复选框时，才会选中/取消选中复选框。

selectOnCheck：如果设置为 true，点击复选框将会选中该行。如果设置为 false，选中该行将不会选中复选框。

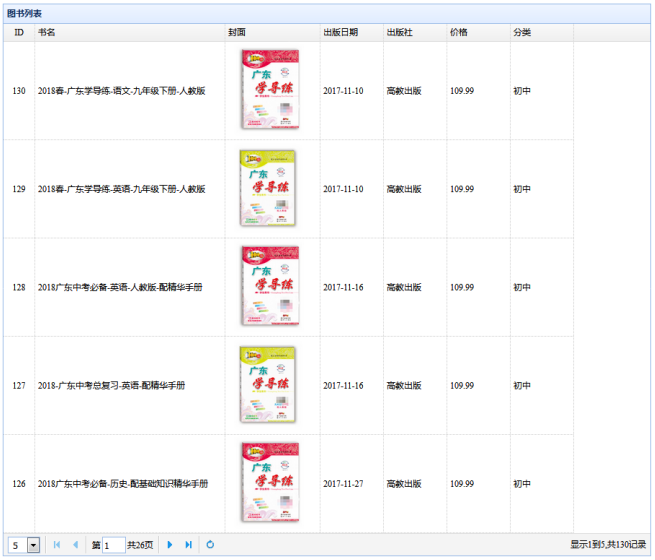
9.1.6 获得选中行

```
//表格的 singleSelect=true 是单选
var row = $('#dg').datagrid('getSelected');

//表格的 singleSelect=false 是多选，rows 是数组
var rows = $('#dg').datagrid('getSelections');
```

9.1.7 复杂数据绑定和分页

datagrid 分页时需要返回的数据包括两部分，total 是总条数，rows 是当前页的数据列表。数据格式如下所示。请求服务端分页数据时，datagrid 传递的页码参数名是 page，每页显示

几条的参数是 rows。本示例演示 json 数据中普通字段、嵌套对象(如 category)和图片的显示。




```
{
    "total": 130,
    "rows": [{
        "id": 5,
        "name": "2018广东中考-考前押题卷-英语",
        "author": "人民教育出版社",
        "publishDate": "2018-02-27",
        "category": {
            "id": 2,
            "name": "初中"
        },
        "summary": null,
        "cover": "default_cover1.png",
        "price": 36.0
    }, {
        "id": 4,
        "name": "2018春-教材全解-英语-三年级下册-粤人民版",
        "author": "人民教育出版社",
        "publishDate": "2018-03-01",
        "category": {
            "id": 1,
            "name": "小学"
        },
        "summary": null,
        "cover": "default_cover2.png",
        "price": 36.0
    }]
}
```


分类表 category：

名	类型	长度	小数点	允许空值 (	
ID	int	11	0	☐	1
NAME	varchar	20	0	☐	

图书表 book：

名	类型	长度	小数点	允许空值 (	
ID	int	11	0	☐	1
NAME	varchar	30	0	☐	
AUTHOR	varchar	30	0	☐	
PUBLISH_DATE	date	0	0	☐	
PRICE	decimal	10	2	☐	
COVER	varchar	150	0	☐	
CATEGORY_ID	int	11	0	☐	
SUMMARY	varchar	300	0	☐	

测试数据略。

```
package com.easyui.pojo;

public class Category {
    private Integer id;
    private String name;
    //getter/setter 略
}
```

```
package com.easyui.pojo;

import com.fasterxml.jackson.annotation.JsonFormat;
import org.springframework.format.annotation.DateTimeFormat;

import java.util.Date;

public class Book {
```

```
    private Integer id;
    private String name;
    private String author;
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "GMT+8")
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date publishDate;
    private Category category;
    private String summary;
    private String cover;
    private Float price;
    //getter/setter 略
}
```

BookMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.easyui.dao.BookDAO">

    <resultMap id="bookMap" type="Book">
        <id property="id" column="ID"/>
        <result property="author" column="AUTHOR"/>
        <result property="cover" column="COVER"/>
        <result property="name" column="NAME"/>
        <result property="price" column="PRICE"/>
        <result property="publishDate" column="PUBLISH_DATE"/>
        <result property="summary" column="SUMMARY"/>
        <result property="category.id" column="CATEGORY_ID"/>
        <result property="category.name" column="CATEGORY_NAME"/>
    </resultMap>

    <select id="getBookList" parameterType="Book" resultMap="bookMap">
        select
B.ID,B.AUTHOR,B.COVER,B.NAME,B.PRICE,B.PUBLISH_DATE,B.CATEGORY_ID,C.NAME
CATEGORY_NAME
        from BOOK B
        inner join CATEGORY C ON B.CATEGORY_ID =C.ID
        <where>
            <if test="id!=null">and B.ID=#{id}</if>
            <if test="category!=null and category.id !=null">and
B.CATEGORY_ID=#{category.id}</if>
            <if test="name!=null and name !=''">and B.NAME like
concat('%',#{name},'%')</if>
        </where>
```

```
    </select>

</mapper>
```

BookDAO

```
package com.easyui.dao;

import com.easyui.pojo.Book;

import java.util.List;

public interface BookDAO {
    List<Book> getBookList(Book criteria);

}
```

分页使用 Mybatis 的 PageHelper 插件 pom.xml：

```
<!--mybatis 分页插件-->
<dependency>
    <groupId>com.github.pagehelper</groupId>
    <artifactId>pagehelper</artifactId>
    <version>4.2.1</version>
</dependency>
```

spring-config.xml 中修改 sqlSession 的配置，加入插件信息：

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <!--扫描 Mybatis 所有映射文件-->
    <property name="mapperLocations"
              value="classpath:com/easyui/dao/*.xml"/>
    <property name="typeAliasesPackage" value="com.easyui.pojo"/>
    <property name="plugins">
        <array>
            <bean class="com.github.pagehelper.PageHelper">
                <property name="properties">
                    <value>
                        dialect=mysql
                        reasonable=true
                    </value>
                </property>
            </bean>
        </array>
    </property>
</bean>
```

BookService

```
package com.easyui.service;

import com.easyui.dao.BookDAO;
import com.easyui.pojo.Book;
import com.github.pagehelper.PageHelper;
import com.github.pagehelper.PageInfo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class BookService {
    @Autowired
    private BookDAO bookDAO;

    public PageInfo<Book> getBookList(Book criteria, int pageNum, int pageSize) {
        PageHelper.startPage(pageNum, pageSize, "id desc");
        List<Book> list = bookDAO.getBookList(criteria);
        PageInfo<Book> pageInfo = new PageInfo<>(list);
        return pageInfo;
    }
}
```

BookController

```
package com.easyui.controller;

import com.easyui.pojo.Book;
import com.easyui.service.BookService;
import com.github.pagehelper.PageInfo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import java.util.HashMap;
import java.util.Map;

@Controller
public class BookController {

    @Autowired
```

```
    private BookService bookService;

    @RequestMapping("book_list.json")
    @ResponseBody
    public Map<String, Object> bookList(Book book, Integer page, Integer rows) {
        PageInfo<Book> pageInfo = bookService.getBookList(book, page, rows);
        Map<String, Object> result = new HashMap<>();
        result.put("total", pageInfo.getTotal());//总条数
        result.put("rows", pageInfo.getList());//返回的 list 数据
        return result;
    }
}
```

页面：

```
<table id="dg" style="width: 900px;height:400px">
</table>
<script type="text/javascript">
    $(function () {
        $("#dg").datagrid({
            title: "图书列表",
            url: '/book_list.json',
            method: 'post',
            pageSize: 5,//初始页码
            pagination: true,//启用分页
            singleSelect: true,//限制单选
            pageList: [5, 10, 15, 20, 25],//页码选择的下拉列表
            columns: [[
                {field: 'id', title: "ID", width: '5%', align: 'center'},
                {
                    field: 'name', title: "书名", width: '30%'
                },
                {
                    field: 'cover', title: "封面", width: '15%',
                    formatter: function (value, row, index) {
                        return '<img src="' + value + '" style="max-height: 130px"/>';
                    }
                },
                {field: 'publishDate', title: "出版日期", width: '10%'},
                {field: 'author', title: "出版社", width: '10%'},
                {field: 'price', title: "价格", width: '10%'},
                {
                    field: 'category', title: "分类", width: '10%',
                    formatter: function (value, row, index) {
                        return row.category.name;
```

```
                    }
                }
            ]]
        });
    })

</script>
```

查询条件封装：

```
var queryParams = $('#dg').datagrid('options').queryParams;
queryParams.id = $("#search-id").val();
queryParams.name = $("#search-name").val();
queryParams["category.id"] = $("#search-category").combobox("getValue");
//重新加载 datagrid
$('#dg').datagrid('options').pageNumber=1;
$("#dg").datagrid('reload');
```

9.1.8 冻结行和列

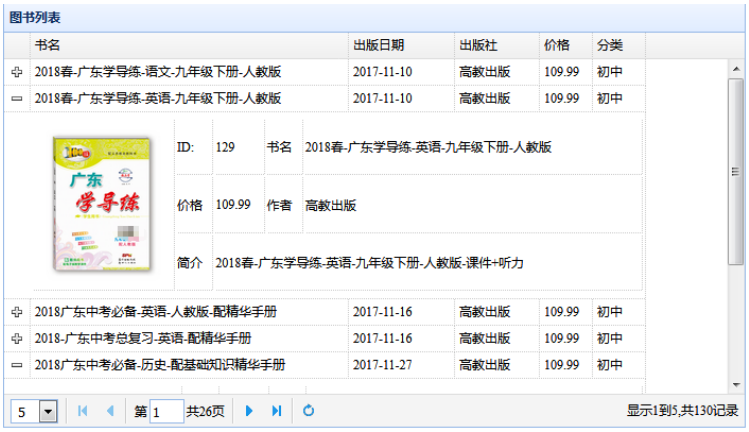
冻结列和 columns 写法一样，不过此列会被固定在左边。注意表格列宽度不要设置成百分比，否则看不到效果

```
frozenColumns:[[{field: 'id', title: "ID", width: 50, align: 'center'}]],
```

冻结行，在表格加载完成后设置冻结的行，freezeRow 设置冻结行的索引，从 0 开始：

```
onLoadSuccess: function(){
    $(this).datagrid('freezeRow',0);
}
```

9.1.9 扩展行显示细节



展开行需要引入额外的插件：

```
<script type="text/javascript" src="/easyui/datagrid-detailview.js"></script>
```

关键代码

```
view: detailview,//设置详情视图
detailFormatter: function (index, row) {//详情的格式
    return '<div class="ddv" style="padding:5px 0"></div>';
},onExpandRow: function (index, row) {
    var ddv = $(this).datagrid('getRowDetail', index).find('div.ddv');
    ddv.panel({
        border: false,
        cache: false,
        href: '/book_detail.json?id=' + row.id,
        onLoad: function () {
            $('#dg').datagrid('fixDetailRowHeight', index);
        }
    });
    $('#dg').datagrid('fixDetailRowHeight', index);
}
```

详情页 detail.jsp：

```
<table class="dv-table" border="0" style="width:100%;">
    <tr>
        <td rowspan="3" style="width:60px">
            <img src="${book.cover}"/>
        </td>
        <td class="dv-label">ID: </td>
        <td>${book.id}</td>
        <td class="dv-label">书名</td>
        <td>${book.name}</td>
    </tr>
```

```
    <tr>
        <td class="dv-label">价格 </td>
        <td>${book.price}</td>
        <td class="dv-label">作者</td>
        <td>${book.author}</td>
    </tr>
    <tr>
        <td class="dv-label">简介 </td>
        <td colspan="3">${book.summary}</td>
    </tr>
</table>
```

后台代码：

BookMapper.xml

```
<select id="getBookById" parameterType="int" resultMap="bookMap">
    select ID,AUTHOR,COVER,NAME,PRICE,PUBLISH_DATE,SUMMARY,CATEGORY_ID
    from BOOK
    where ID=#{id}
</select>
```

BookDAO

```
Book getBookById(Integer id);
```

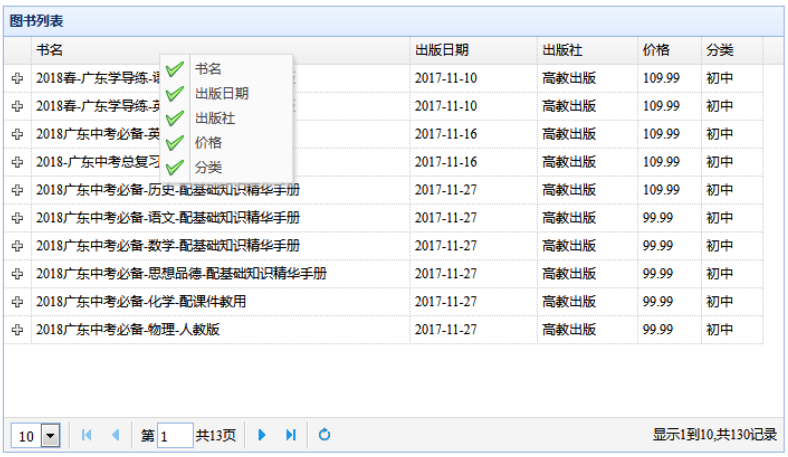
BookService

```
public Book getBookById(Integer id) {
    return bookDAO.getBookById(id);
}
```

BookController

```
@RequestMapping("book_detail.json")
public ModelAndView detail(Integer id) {
    Book book = bookService.getBookById(id);
    return new ModelAndView("/pages/datagrid/detail.jsp", "book", book);
}
```

9.1.10右键菜单



在表头上点击右键，控制显示的列。

关键代码：

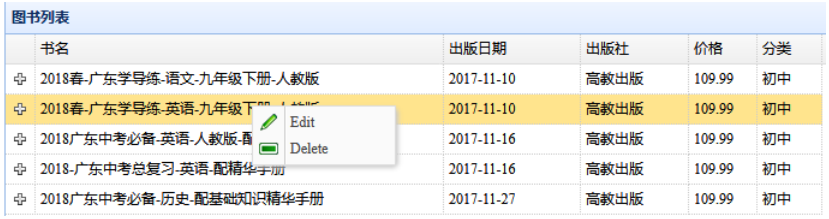
```
fitColumns: true,
onHeaderContextMenu: function (e, field) {
    e.preventDefault();
    if (!cmenu) {
        createColumnMenu();
    }
    cmenu.menu('show', {
        left: e.pageX,
        top: e.pageY
    });
}
```

创建菜单：

```
var cmenu;

function createColumnMenu() {
    cmenu = $('<div/>').appendTo('body');
    cmenu.menu({
        onClick: function (item) {
            if (item.iconCls == 'icon-ok') {
                $('#dg').datagrid('hideColumn', item.name);
                cmenu.menu('setIcon', {
                    target: item.target,
                    iconCls: 'icon-empty'
                });
            } else {
                $('#dg').datagrid('showColumn', item.name);
                cmenu.menu('setIcon', {
```

```
                    target: item.target,
                    iconCls: 'icon-ok'
                });
            }
        }
    });
    var fields = $('#dg').datagrid('getColumnFields');
    for (var i = 0; i < fields.length; i++) {
        var field = fields[i];
        var col = $('#dg').datagrid('getColumnOption', field);
        cmenu.menu('appendItem', {
            text: col.title,
            name: field,
            iconCls: 'icon-ok'
        });
    }
}
```

行右键菜单：

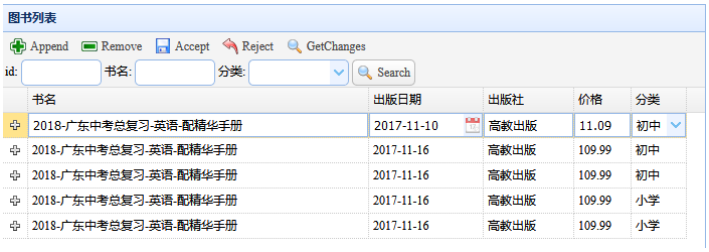



```
<div id="mm" class="easyui-menu" data-options="onClick:menuHandler"
style="width:120px;">
    <div data-options="name:'edit',iconCls:'icon-edit'">Edit</div>
    <div data-options="name:'delete',iconCls:'icon-remove'">Delete</div>
</div>
```

```
function menuHandler(){
    var row =$("#dg").datagrid("getSelected");
    alert(row.name)
}
```

```
onRowContextMenu: function (e, rowIndex, rowData) {
    e.preventDefault();//阻止默认事件
    $(this).datagrid("selectRow",rowIndex);
    $('#mm').menu('show', {
        left: e.pageX,
        top: e.pageY
    });
}
```

9.1.11数据表格编辑器




```
columns: [[
    {
        field: 'name', title: "书名", width: 300,editor:'textbox'
    },
    {field: 'publishDate', title: "出版日期", width:
100,editor:{type:'datebox',options:{formatter: function (date) {
        return date.getFullYear() + "-" + (date.getMonth() + 1) + "-" + date.getDate();
    },
        parser: function (s) {
            if (!s) return new Date();
            var ss = (s.split("-"));
            var y = parseInt(ss[0], 10);
            var m = parseInt(ss[1], 10);
            var d = parseInt(ss[2], 10);
            if (!isNaN(y) && !isNaN(m) && !isNaN(d)) {
                return new Date(y, m - 1, d);
            } else {
                return new Date();
            }
        }}}},
    {field: 'author', title: "出版社", width: 80,editor:'textbox'},
    {field: 'price', title: "价格", width:
50,editor:{type:'numberbox',options:{precision:2}}},
    {
        field: 'category', title: "分类", width: 50,
        formatter: function (value, row, index) {
            return row.category['name'];
        }, editor: {
        type: 'combobox',
        options: {
            valueField: 'id',
            textField: 'name',
            method: 'get',
            url: '/category.json',
```

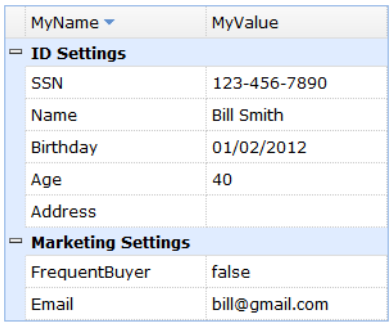
```
            required: true
        }
    }
    }
]],
```

```
onClickRow: function (index) {
    if(editIndex != index){
        $('#dg').datagrid("rejectChanges");
    }
    editIndex = index;
    $('#dg').datagrid('selectRow', index)
        .datagrid('beginEdit', index);
    $('#dg').datagrid('selectRow', editIndex);
}
```

```
var editIndex = undefined;

function accept() {
    if (!endEditing()) {
        return;
    }
    var row = $('#dg').datagrid('getRows')[editIndex];
    var ed = $('#dg').datagrid('getEditor', {index: editIndex, field: 'category'});
    var categoryId = $(ed.target).combobox('getValue');
    $('#dg').datagrid('acceptChanges');
    var book = {
        id: row.id,
        name: row.name,
        author: row.author,
        publishDate: row.publishDate,
        "category.id": categoryId,
        price: row.price
    }
    $.ajax({
        url: "/book_update.json",
        method: "post",
        data: book,
        success: function (data) {
            if (data.status == "true") {
                editIndex = undefined;
                $('#dg').datagrid('reload');
            } else {
                alert(data.message);
```

```
            }
        }
    })
}

function reject() {
    $('#dg').datagrid('rejectChanges');
    editIndex = undefined;
}

function endEditing() {
    if (editIndex == undefined) {
        return true
    }
    if ($('#dg').datagrid('validateRow', editIndex)) {
        return true;
    } else {
        return false;
    }
}
```

9.2 属性网格 propertygrid



MyName	MyValue
ID Settings	
SSN	123-456-7890
Name	Bill Smith
Birthday	01/02/2012
Age	40
Address	
Marketing Settings	
FrequentBuyer	false
Email	bill@gmail.com

属性网格是内联编辑的数据网格。它相当容易使用。用户可以很容易就创建一个可编辑属性的分层列表和表示任何数据类型的项目。属性网格带有内置的排序和分组特征。属性表格依赖 datagrid

属性网格（propertygrid）扩展自数据网格（datagrid）。它的行数据格式与数据网格（datagrid）相同。作为一个属性行，下列字段是必需的:

name：字段名。

value：要被编辑的字段值。

group：组的字段值。

editor：编辑属性值的编辑器。

属性：

该属性扩展自数据网格（datagrid）。下面是为属性网格（propertygrid）添加的属性。

名称	类型	描述	默认值

showGroup	boolean	定义是否显示属性组。	FALSE
groupField	string	定义组的字段名。	group
groupFormatter	function(group,rows)	定义如何格式化组的值。该函数包括下列参数：	
		group：组的字段名。	
		rows：属于改组的行。	

方法：

该方法扩展自数据网格（datagrid）。下面是为属性网格（propertygrid）添加的方法。

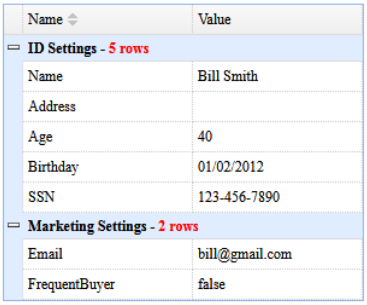
名称	参数	描述
expandGroup	groupIndex	展开指定的组。如果 'groupIndex' 参数未分配，则展开所有的组。
collapseGroup	groupIndex	折叠指定的组。如果 'groupIndex' 参数未分配，则折叠所有的组。

9.2.1 基础数据网格

```
<table id="pg" style="width:300px">
</table>
<script type="text/javascript">
    var data={"total":7,"rows":[
        {"name":"Name","value":"Bill Smith","group":"ID Settings","editor":"text"},
        {"name":"Address","value":"","group":"ID Settings","editor":"text"},
        {"name":"Age","value":"40","group":"ID Settings","editor":"numberbox"},
        {"name":"Birthday","value":"01/02/2012","group":"ID
Settings","editor":"datebox"},
        {"name":"SSN","value":"123-456-7890","group":"ID Settings","editor":"text"},
        {"name":"Email","value":"bill@gmail.com","group":"Marketing
Settings","editor":{
            "type":"validatebox",
            "options":{
                "validType":"email"
            }
        }},
        {"name":"FrequentBuyer","value":"false","group":"Marketing
Settings","editor":{
            "type":"checkbox",
            "options":{
                "on":true,
                "off":false
            }
        }}
    ]};
    $(function () {
```

```
        $("#pg").propertygrid({
            data:data,
            showGroup:true,
            scrollbarSize:0
        })
    })
</script>
```

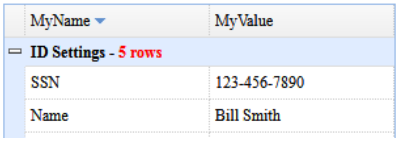
9.2.2 格式组



Name	Value
ID Settings - 5 rows	
Name	Bill Smith
Address	
Age	40
Birthday	01/02/2012
SSN	123-456-7890
Marketing Settings - 2 rows	
Email	bill@gmail.com
FrequentBuyer	false

```
groupFormatter: function (fvalue, rows) {
    return fvalue + ' - <span style="color:red">' + rows.length + ' rows</span>';
}
```

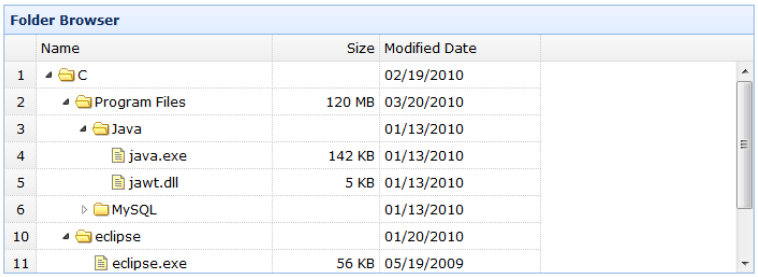
9.2.3 自定义栏目



MyName	MyValue
ID Settings - 5 rows	
SSN	123-456-7890
Name	Bill Smith

```
columns: [[
    {field:'name',title:'MyName',width:100,sortable:true},
    {field:'value',title:'MyValue',width:100,resizable:false}
]]
```

9.3 树形网格 treegrid



树形网格（treegrid）用于以网格形式显示分层数据。它是基于数据网格（datagrid）的，并结合了树视图（treeview）和可编辑网格。树形网格（treegrid）允许您创建可定制的、可异步展开的行，并以多列形式显示分层数据。

属性：

该属性扩展自数据网格（datagrid），下面是为树形网格（treegrid）添加的属性。

名称	类型	描述	默认值
idField	string	定义标识树节点的键名字段。必需。	null
treeField	string	定义树节点的字段。必需。	null
animate	boolean	定义当节点展开或折叠时是否显示动画效果。	FALSE
loader	function(param,success,error)	定义如何从远程服务器加载数据。返回 false 则取消该动作。该函数有下列参数：	json loader
		param：要传递到远程服务器的参数对象。	
		success(data)：当检索数据成功时调用的回调函数。	
		error()：当检索数据失败时调用的回调函数。	
loadFilter	function(data,parentId)	返回要显示的过滤数据。	

事件：

该事件扩展自数据网格（datagrid），下面是为树形网格（treegrid）添加的事件。

名称	参数	描述
onClickRow	row	当用户点击一个节点时触发。
onDblClickRow	row	当用户双击一个节点时触发。
onClickCell	field,row	当用户点击一个单元格时触发。
onDblClickCell	field,row	当用户双击一个单元格时触发。
onBeforeLoad	row, param	当加载数据的请求发出前触发，返回 false 则取消加载动作。
onLoadSuccess	row, data	当数据加载成功时触发。
onLoadError	arguments	当数据加载失败时触发，arguments 参数和 jQuery.ajax 的 'error' 方法一样。
onBeforeExpand	row	节点展开前触发，返回 false 则取消展开动作。
onExpand	row	当节点展开时触发。
onBeforeCollapse	row	节点折叠前触发，返回 false 则取消折叠动作。
onCollapse	row	当节点折叠时触发。
onContextMenu	e, row	当右键点击节点时触发。

onBeforeEdit	row	当用户开始编辑节点时触发。
onAfterEdit	row,changes	当用户完成编辑时触发。
onCancelEdit	row	当用户取消编辑节点时触发。

方法：

很多方法需要一个名为 'id' 的参数，该参数表示树节点的值。

名称	参数	描述
options	none	返回树形网格（treegrid）的选项（options）。
resize	options	设置树形网格（treegrid）的尺寸， options 参数包含两个属性： width：树形网格（treegrid）的新宽度。 height：树形网格（treegrid）的新高度。
fixRowHeight	id	固定指定行的高度。
loadData	data	加载树形网格（treegrid）的数据。

load	param	加载并显示第一页。该方法自版本 1.3.4 起可用。
		代码实例：
		`// load and send some request parameters`
		`$('#tg').treegrid('load', {`
		`q: 'abc',`
		`name: 'name1'`
		`});`

reload	id	重新加载树形网格（treegrid）的数据。如果传递了 'id' 参数，则重新加载树的指定行，否则重新加载树的所有行。
		代码实例：
		`$('#tt').treegrid('reload', 2); // reload the row which value is equals to 2`
		`$('#tt').treegrid('reload');  // reload the all rows`
		`$('#tt').treegrid('reload', {id:2, q:'abc'});  // reload the specified row with 'q' parameter passing to server`

reloadFooter	footer	重新加载底部数据。
getData	none	获取加载的数据。
getFooterRows	none	获取底部数据。
getRoot	none	获取根节点，返回节点对象。
getRoots	none	获取根节点，返回节点数组。
getParent	id	获取父节点。
getChildren	id	获取子节点。
getSelected	none	获取选中的节点并返回它，如果没有选中节点则返回 null。
getSelections	none	获取所有选中的节点。
getLevel	id	获取指定节点的层级。
find	id	找到指定节点并返回该节点数据。
select	id	选择节点。
unselect	id	取消选择节点。
selectAll	none	选择所有节点。
unselectAll	none	取消选择所有节点。
collapse	id	折叠节点。

<table>
<tr><td>expand</td><td>id</td><td>展开节点。</td></tr>
<tr><td>collapseAll</td><td>id</td><td>折叠所有的节点。</td></tr>
<tr><td>expandAll</td><td>id</td><td>展开所有的节点。</td></tr>
<tr><td>expandTo</td><td>id</td><td>从根部展开一个指定的节点。</td></tr>
<tr><td>toggle</td><td>id</td><td>切换节点的展开/折叠状态。</td></tr>
<tr><td rowspan="6">append</td><td rowspan="6">param</td><td>追加一些子节点到一个父节点，’param’ 参数包括下列属性：</td></tr>
<tr><td>parent：父节点的 id，如果没有分配，则追加为根节点。</td></tr>
<tr><td>data：数组，节点的数据。</td></tr>
<tr><td>代码实例：</td></tr>
<tr><td><pre>
// append some nodes to the selected row
var node = $('#tt').treegrid('getSelected');
$('#tt').treegrid('append',{
    parent: node.id, // the node has a 'id' value that
defined through 'idField' property
    data: [{
        id: '073',
        name: 'name73'
    }]
});
</pre></td></tr>
<tr><td rowspan="6">insert</td><td rowspan="6">param</td><td>在指定节点的前边或后边插入一个节点，’param’ 参数包括下列属性：</td></tr>
<tr><td>before：前边插入的节点的 id 值。</td></tr>
<tr><td>after：后边插入的节点的 id 值。</td></tr>
<tr><td>data：新的节点数据。</td></tr>
<tr><td>代码实例：</td></tr>
<tr><td><pre>
// insert a new node before the selected node
var node = $('#tt').treegrid('getSelected');
if (node){
    $('#tt').treegrid('insert', {
        before: node.id,
        data: {
            id: 38,
            name: 'name38'
        }
    });
}
</pre></td></tr>
<tr><td>remove</td><td>id</td><td>移除节点和它的子节点。</td></tr>
<tr><td>pop</td><td>id</td><td>弹出节点并在移除该节点后返回包含其子节点的节点数据。该方法自版本 1.3.1 起可用。</td></tr>
<tr><td>refresh</td><td>id</td><td>刷新指定的节点。</td></tr>
<tr><td rowspan="2">update</td><td rowspan="2">param</td><td>更新指定的节点。’param’ 参数包括下列属性：</td></tr>
<tr><td>id：表示要被更新的节点的 id。</td></tr>
</table>

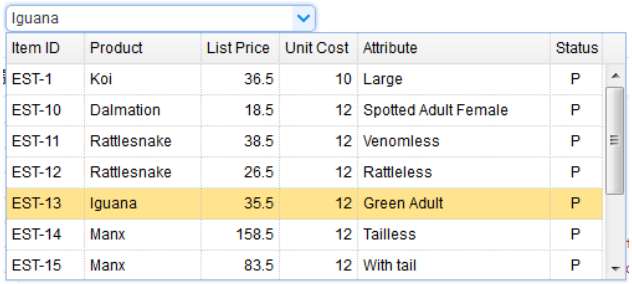
		row：新的行数据。

		代码实例：
		`$('#tt').treegrid('update',{`
		`id: 2,`
		`row: {`
		`name: 'new name',`
		`iconCls: 'icon-save'`
		`}`
		`});`

		该方法自版本 1.3.1 起可用。
beginEdit	id	开始编辑节点。
endEdit	id	结束编辑节点。
cancelEdit	id	取消编辑节点。
getEditors	id	获取指定行的编辑器。每个编辑器有下列属性： actions：编辑器可以做的动作。 target：目标编辑器的 jQuery 对象。 field：字段名。 type：编辑器的类型。
getEditor	param	获取指定的编辑器，param 参数包含两个属性： id：行节点的 id。 field：字段名。

```
<table title="Folder Browser" class="easyui-treegrid"
style="width:700px;height:250px"
        data-options="
            url: '/tree.json',
            method: 'get',
            rownumbers: true,
            idField: 'id',
            treeField: 'text'
        ">
    <thead>
    <tr>
        <th data-options="field:'text'" width="220">名称</th>
        <th data-options="field:'parentId'" width="100" align="right">父节点</th>
    </tr>
    </thead>
</table>
```

9.4 组合网格 combogrid



组合网格（combogrid）把可编辑的文本框和下拉数据网格面板结合起来，用户可以从下拉数据网格面板中快速查找和选择。组合网格（combogrid）提供了选择某个项目的键盘导航支持。组合网格依赖于 combo 和 datagrid。

属性：

该属性扩展自组合（combo）和数据网格（datagrid），下面是为组合网格（combogrid）添加的属性。

名称	类型	描述	默认值
loadMsg	string	当数据网格(datagrid)加载远程数据时显示的消息。	null
idField	string	id 的字段名。	null
textField	string	显示在文本框中的 text 字段名。	null
mode	string	定义当文本改变时如何加载数据网格（datagrid）数据。如果组合网格（combogrid）从服务器加载就设置为 'remote'。当设置为 'remote' 模式时，用户输入的值将会被作为名为 'q' 的 http 请求参数发送到服务器，以获取新的数据。	local

filter	function(q, row)	定义当 'mode' 设置为 'local' 时如何选择本地数据。返回 true 则选择该行。
		代码实例：
		`$('#cc').combogrid({`
		`  filter: function(q, row){`
		`  var opts = $(this).combogrid('options');`
		`  return row[opts.textField].indexOf(q) == 0;`
		`  }`
		`});`
			10

事件：

该事件扩展自组合（combo）和数据网格（datagrid）。

方法：

该方法扩展自组合（combo），下面是为组合网格（combogrid）添加或重写的方法。

名称	参数	描述
options	none	返回选项（options）对象。

grid	none	返回数据网格（datagrid）对象。下面的实例显示如何取得选中的行：
		`var g = $('#cc').combogrid('grid'); // get datagrid object`
		`var r = g.datagrid('getSelected'); // get the selected row`

		alert(r.name);
setValues	values	设置组件值的数组。 代码实例： $('#cc').combogrid('setValues', ['001','007']);
setValue	value	设置组件的值。 代码实例： $('#cc').combogrid('setValue', '002');
clear	none	清除组件的值。

```
<select id="cg" style="width: 300px"></select>
<script type="text/javascript">
    $(function () {
        $("#cg").combogrid({
            panelWidth: 500,
            fitColumns: true,
            idField: 'id',
            textField: 'name',
            url: '/book_list.json',
            method: 'get',
            pageSize: 5,//初始页码
            pagination: true,//启用分页
            singleSelect: true,//限制单选
            pageList: [5, 10, 15, 20, 25],//页码选择的下拉列表
            columns: [[
                {field: 'name', title: "书名", width: 300},
                {field: 'publishDate', title: "出版日期", width: 100},
                {field: 'author', title: "出版社", width: 80},
                {field: 'price', title: "价格", width: 50},
                {
                    field: 'category', title: "分类", width: 30,
                    formatter: function (value, row, index) {
                        return row.category.name;
                    }
                }
            ]]
        })
    })
</script>
```