jquery easyui 教程

概述

这个教程的目的是说明如何使用 easyui 框架容易的创建网页。首先，你需要包含一些 js 和 css 文件：

```
<link rel="stylesheet" type="text/css"
href="../themes/default/easyui.css">
<script type="text/javascript"
src="../jquery-1.4.2.min.js"></script>
<script type="text/javascript"
src="../jquery.easyui.min.js"></script>
```

easyui 预定义了一些图标 css，这些 css 类可以显示图片背景（16×16）。使用这些类之前，需要包含：

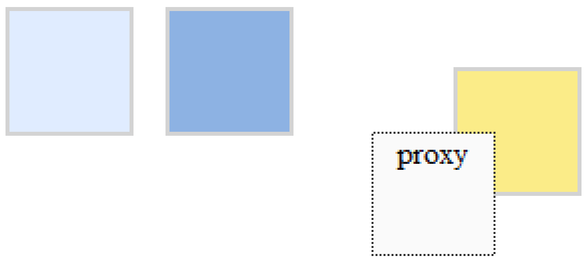
```
<link rel="stylesheet" type="text/css"
href="../themes/icon.css">
```

内容

1. 拖放
   - 基本拖放
   - 创建购物车式拖放
   - 创建课程表

基本拖放

这个教程显示如何使 HTML 元素变得可拖放。这个例子会创建 3 个 DIV 元素然后让它们变得可拖放。



首先，创建三个 DIV 元素：

```
    <div id="dd1" class="dd-demo"></div>
    <div id="dd2" class="dd-demo"></div>
    <div id="dd3" class="dd-demo"></div>
```

让第一个 DIV 元素可拖放，使用默认的拖放样式。

```
$('#dd1').draggable();
```

让第二个 DIV 元素使用 proxy 来拖放，proxy:'clone'表示 proxy 使用原始元素的复制。

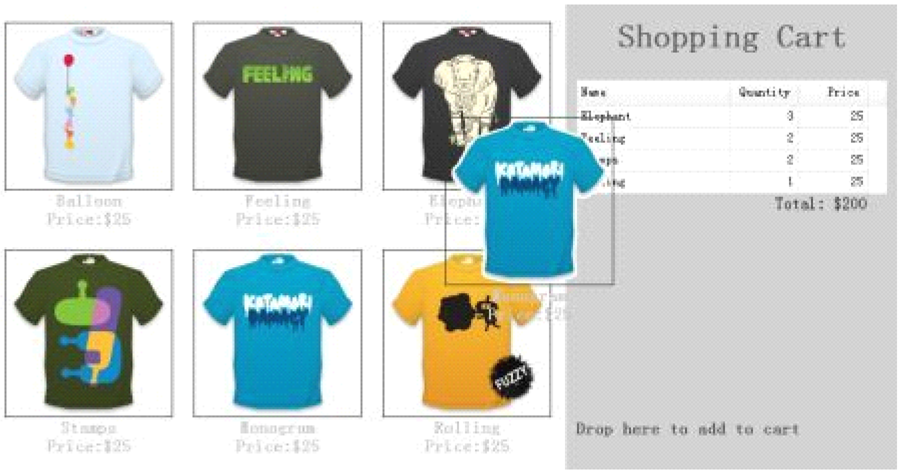
```
$('#dd2').draggable({
    proxy:'clone'
});
```

让第三个 DIV 元素使用自定义 proxy 来拖放

```
$('#dd3').draggable({
    proxy:function(source){
        var p = $('<div class="proxy">proxy</div>');
        p.appendTo('body');
        return p;
    }
});
```

构建购物车型拖放

使用 jQuery easyui，我们在 web 应用中就有了拖放的能力。这个教程显示了如何构建购物车页，它使用户拖放他们希望购买的产品，更新购物篮的物品和价格。



Name	Quantity	Price
Elephant	3	25
Feeling	2	25
mps	2	25
ing	1	25

Total: $200

显示产品页：

```
<ul class="products">
```

```
    <li>
        <a href="#" class="item">
            <img src="shirt1.gif"/>
            <div>
                <p>Balloon</p>
                <p>Price:$25</p>
            </div>
        </a>
    </li>
    <li>
        <a href="#" class="item">
            <img src="shirt2.gif"/>
            <div>
                <p>Feeling</p>
                <p>Price:$25</p>
            </div>
        </a>
    </li>
    <!-- other products -->
</ul>
```

ul 元素包含一些 li 元素以显示产品。每一个产品的名称和单价属性在 P 元素中。

创建购物车：

```
<div class="cart">
    <h1>Shopping Cart</h1>
    <table id="cartcontent"
style="width:300px;height:auto;">
        <thead>
            <tr>
                <th field="name" width=140>Name</th>
                <th field="quantity" width=60
align="right">Quantity</th>
                <th field="price" width=60
align="right">Price</th>
            </tr>
        </thead>
    </table>
    <p class="total">Total: $0</p>
    <h2>Drop here to add to cart</h2>
</div>
```

使用 datagrid 显示购物篮项目。

拖曳产品副本

```
$('.item').draggable({
    revert:true,
    proxy:'clone',
    onStartDrag:function(){
        $(this).draggable('options').cursor = 'not-allowed';
        $(this).draggable('proxy').css('z-index',10);
    },
    onStopDrag:function(){
        $(this).draggable('options').cursor='move';
    }
});
```

我们设置 draggable 属性 proxy 为 clone，所以拖曳元素使用 clone 效果。

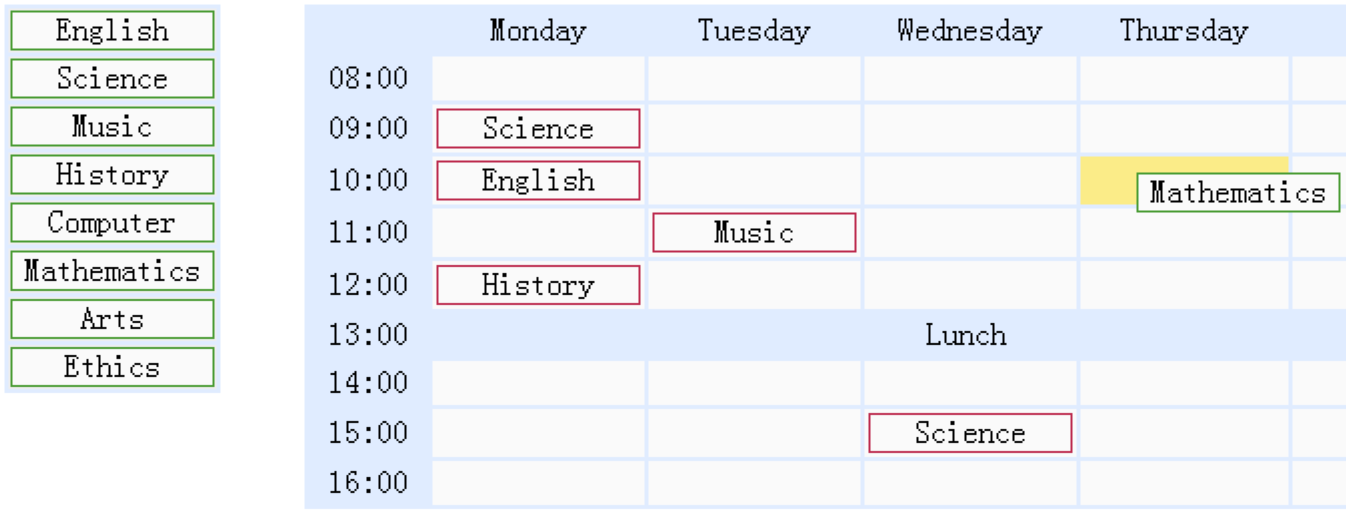
将选择的产品放入购物车

```
$('.cart').droppable({
    onDragEnter:function(e,source){
        $(source).draggable('options').cursor='auto';
    },
    onDragLeave:function(e,source){

$(source).draggable('options').cursor='not-allowed';
    },
    onDrop:function(e,source){
        var name = $(source).find('p:eq(0)').html();
        var price = $(source).find('p:eq(1)').html();
        addProduct(name, parseFloat(price.split('$')[1]));
    }
});
var data = {"total":0,"rows":[]};
var totalCost = 0;
function addProduct(name,price){
    function add(){
        for(var i=0; i<data.total; i++){
            var row = data.rows[i];
            if (row.name == name){
                row.quantity += 1;
                return;
            }
        }
        data.total += 1;
        data.rows.push({
            name:name,
            quantity:1,
            price:price
        });
    }
    add();
    totalCost += price;
    $('#cartcontent').datagrid('loadData', data);
    $('div.cart .total').html('Total: $'+totalCost);
}
```

当放下产品时，我们得到产品的名称和单价，然后调用 addProduct 函数更新购物篮。

创建课程表

本教程显示了如何使用 jQuery easyui 创建课程表。我们创建两个表：在左面的课程列表和右面的时间表。你可以拖课程到时间表的单元格中。课程是<div class='item'>元素，时间格是<td class='drop'>元素。



English
Science
Music
History
Computer
Mathematics
Arts
Ethics

	Monday	Tuesday	Wednesday	Thursday
08:00				
09:00	Science			
10:00	English			Mathematics
11:00		Music		
12:00	History			
13:00			Lunch	
14:00				
15:00			Science	
16:00				

显示课程

```
<div class="left">
    <table>
        <tr>
            <td><div class="item">English</div></td>
        </tr>
        <tr>
            <td><div class="item">Science</div></td>
        </tr>
        <!-- other subjects -->
    </table>
</div>
```

显示时间表

```
<div class="right">
    <table>
        <tr>
            <td class="blank"></td>
            <td class="title">Monday</td>
            <td class="title">Tuesday</td>
            <td class="title">Wednesday</td>
            <td class="title">Thursday</td>
            <td class="title">Friday</td>
        </tr>
        <tr>
            <td class="time">08:00</td>
            <td class="drop"></td>
            <td class="drop"></td>
            <td class="drop"></td>
            <td class="drop"></td>
            <td class="drop"></td>
        </tr>
        <!-- other cells -->
    </table>
</div>
```

拖动左面的课程

```
$('.left .item').draggable({
    revert:true,
    proxy:'clone'
});
```

放置课程到时间表中

```
$('.right td.drop').droppable({
    onDragEnter:function(){
        $(this).addClass('over');
    },
    onDragLeave:function(){
        $(this).removeClass('over');
    },
    onDrop:function(e,source){
        $(this).removeClass('over');
        if ($(source).hasClass('assigned')){
            $(this).append(source);
        } else {
            var c = $(source).clone().addClass('assigned');
            $(this).empty().append(c);
            c.draggable({
                revert:true
            });
        }
    }
});
```

当用户拖动左面的课程到右面的时间表中，onDrop 函数被调用。源元素的副本被从左面拖动并且附加到到时间表的单元格中。当放置课程到时间表的单元格到另一个单元格时，简单的移动它。

2. 菜单和按钮 Menu and Button
   - 建立简单菜单
   - 建立链接按钮
   - 建立菜单按钮
   - 建立分割按钮

创建简单菜单

在 DIV 标记中定义菜单。像这样：

```
<div id="mm" style="width:120px;">
    <div onclick="javascript:alert('new')">New</div>
    <div>
        <span>Open</span>
        <div style="width:150px;">
            <div><b>Word</b></div>
            <div>Excel</div>
            <div>PowerPoint</div>
        </div>
    </div>
    <div icon="icon-save">Save</div>
    <div class="menu-sep"></div>
    <div>Exit</div>
</div>
```

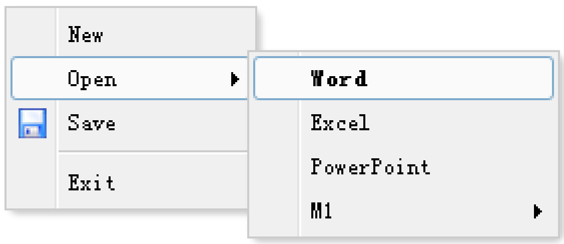
建立菜单，你需要运行下列 jQuery 代码

```
$('#mm').menu();
//或者 $('#mm').menu(options);
```

当菜单被创建时是不可见的，可使用 show 方法显示或者 hide 方法隐藏：

```
$('#mm').menu('show', {
  left: 200,
  top: 100
});
```

现在，我们创建菜单并在（200,100）处显示。运行代码会得到：



创建连接按钮

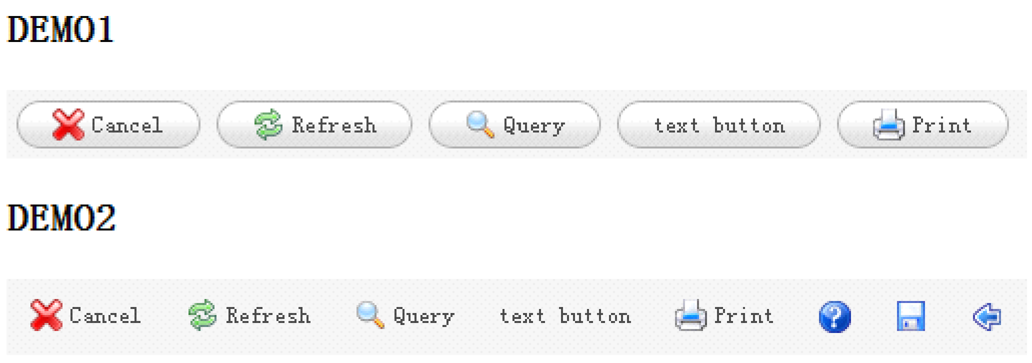
通常使用<button>元素创建按钮。链接按钮使用 A 元素创建，事实上，链接按钮是 A 元素但显示为按钮样式。

创建链接按钮，首先创建 A 元素：

```
<h3>DEMO1</h3>
<div style="padding:5px;background:#efefef;width:500px;">
    <a href="#" class="easyui-linkbutton"
icon="icon-cancel">Cancel</a>
    <a href="#" class="easyui-linkbutton"
icon="icon-reload">Refresh</a>
```

```
    <a href="#" class="easyui-linkbutton"
icon="icon-search">Query</a>
    <a href="#" class="easyui-linkbutton">text button</a>
    <a href="#" class="easyui-linkbutton"
icon="icon-print">Print</a>
</div>

<h3>DEMO2</h3>
<div style="padding:5px;background:#efefef;width:500px;">
    <a href="#" class="easyui-linkbutton" plain="true"
icon="icon-cancel">Cancel</a>
    <a href="#" class="easyui-linkbutton" plain="true"
icon="icon-reload">Refresh</a>
    <a href="#" class="easyui-linkbutton" plain="true"
icon="icon-search">Query</a>
    <a href="#" class="easyui-linkbutton" plain="true">text
button</a>
    <a href="#" class="easyui-linkbutton" plain="true"
icon="icon-print">Print</a>
    <a href="#" class="easyui-linkbutton" plain="true"
icon="icon-help">&nbsp;</a>
    <a href="#" class="easyui-linkbutton" plain="true"
icon="icon-save"></a>
    <a href="#" class="easyui-linkbutton" plain="true"
icon="icon-back"></a>
</div>
```

icon 属性是 icon CSS 类是在按钮上显示的图标。运行代码，出现：



一些时候，你可以决定禁用或者不禁用连接按钮，使用下面的代码可以禁用连接图标：

```
$(selector).linkbutton({disabled:true});
```

建立菜单按钮

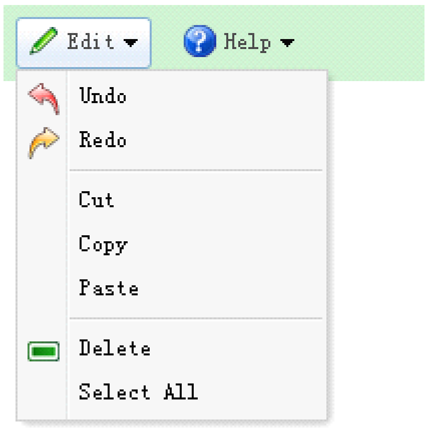
菜单按钮包含按钮和菜单两部分，当点击或者移动鼠标到按钮上的时候，显示相应的菜单。定义菜单按钮，需要定义链接按钮和菜单，像这样：

```
    <div
style="background:#C9EDCC;padding:5px;width:200px;">
        <a href="javascript:void(0)" id="mb1"
icon="icon-edit">Edit</a>
        <a href="javascript:void(0)" id="mb2"
icon="icon-help">Help</a>
    </div>
    <div id="mm1" style="width:150px;">
        <div icon="icon-undo">Undo</div>
        <div icon="icon-redo">Redo</div>
        <div class="menu-sep"></div>
        <div>Cut</div>
        <div>Copy</div>
        <div>Paste</div>
        <div class="menu-sep"></div>
        <div icon="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>
```

使用下列 jQuery 代码：

```
$('#mb1').menubutton({menu:'#mm1'});
$('#mb2').menubutton({menu:'#mm2'});
```

现在，菜单按钮就完成了。



建立拆分按钮

拆分按钮包括链接按钮和菜单。当用户点击或者悬停在下箭头区域时显示相关菜单。这个例子是建立拆分按钮的演示：

首先，创建一个链接按钮和菜单标记：

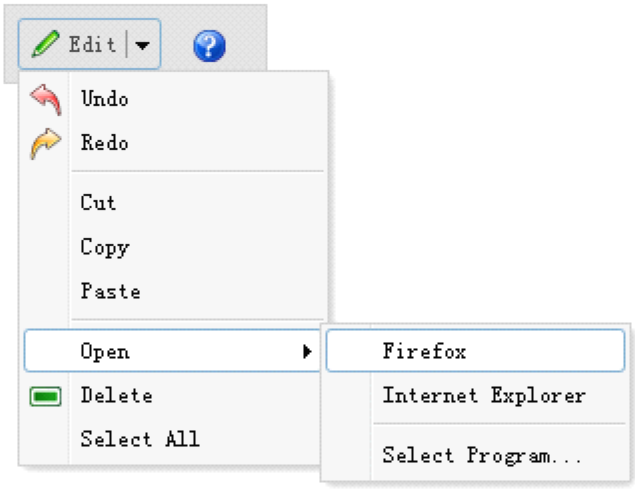
```
<div style="border:1px solid
#ccc;background:#ddd;padding:5px;width:120px;">
   <a href="javascript:void(0)" id="sb"
icon="icon-edit">Edit</a>
   <a href="javascript:void(0)" class="easyui-linkbutton"
plain="true" icon="icon-help"></a>
</div>
<div id="mm" style="width:150px;">
   <div icon="icon-undo">Undo</div>
   <div icon="icon-redo">Redo</div>
   <div class="menu-sep"></div>
   <div>Cut</div>
   <div>Copy</div>
   <div>Paste</div>
   <div class="menu-sep"></div>
   <div>
      <span>Open</span>
      <div style="width:150px;">
         <div>Firefox</div>
         <div>Internet Explorer</div>
         <div class="menu-sep"></div>
         <div>Select Program...</div>
      </div>
   </div>
   <div icon="icon-remove">Delete</div>
   <div>Select All</div>
```

```
</div>
```

jQuery 代码：

```
$('#sb').splitbutton({menu:'#mm'});
```

运行后会出现：



3. 版面
    - 创建边框版面
    - 面板上的复合版面
    - 建立可折叠版面
    - 建立 TABS
    - 动态添加 TABS
    - 建立 XP 样式左面板

创建边框版面网页

边框版面提供 5 个区域：东西南北中（其实就是上下左右中），下面是通常用法：

- 北区可以用于网站 banner
- 南区可以用于版权信息和注释
- 西区可以用于导航菜单
- 东区可以用于推广项目
- 中区可以用于主内容

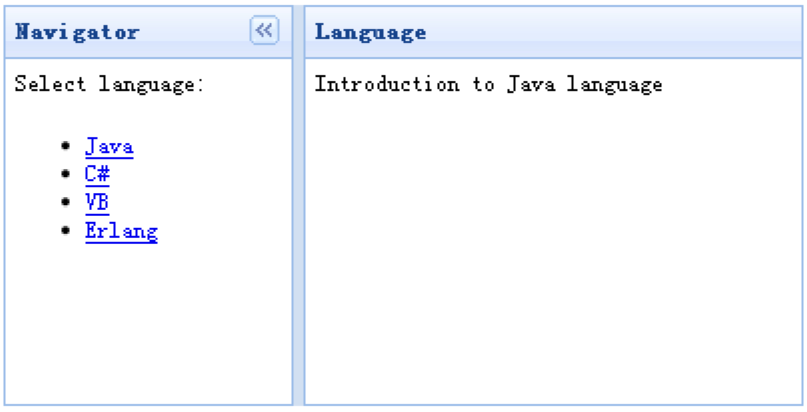
运用版面，需要确认版面容器然后定义一些区域。版面至少要有一个中间区域。下列是版面例子：

```
<div class="easyui-layout"
style="width:400px;height:300px;">
```

```
    <div region="west" split="true" title="Navigator"
style="width:150px;">
        <p style="padding:5px;margin:0;">Select
language:</p>
        <ul>
            <li><a href="javascript:void(0)"
onclick="showpage('java.html')">Java</a></li>
            <li><a href="javascript:void(0)"
onclick="showpage('cshape.html')">C#</a></li>
            <li><a href="javascript:void(0)"
onclick="showpage('vb.html')">VB</a></li>
            <li><a href="javascript:void(0)"
onclick="showpage('erlang.html')">Erlang</a></li>
        </ul>
    </div>
    <div id="content" region="center" title="Language"
href="java.html" style="padding:5px;">
    </div>
</div>
```

我们使用 DIV 容器创建边框版面。版面拆分容器为 2 部分，左面是导航菜单右面是主内容。中间区域的面板，我们设置 href 属性以调用出示网页。

运行 layout.html 的结果是：


Navigator

Select language:

- Java
- C#
- VB
- Erlang

Language

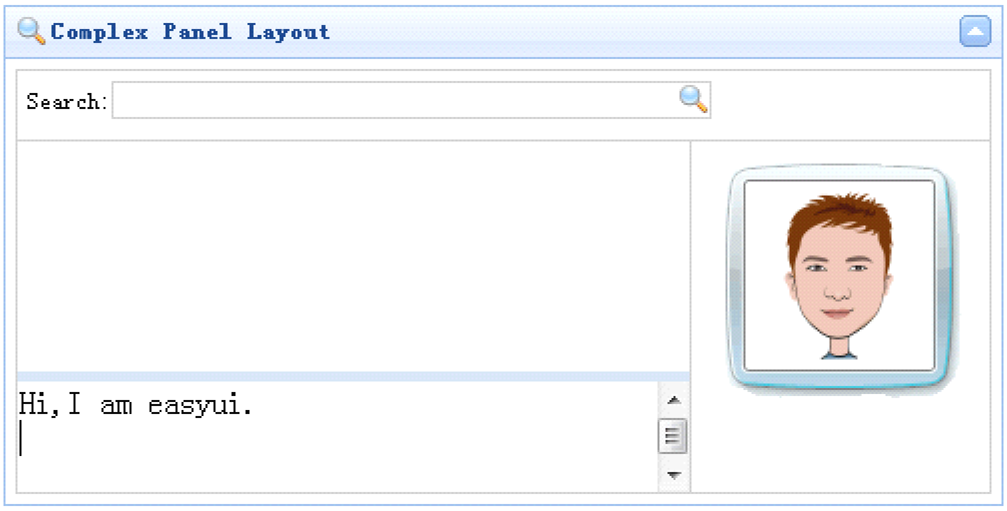
Introduction to Java language

写下 onclick 事件控制函数以获取数据，showpage 函数非常简单：

```
function showpage(url){
    $('#content').load(url);
}
```

面板上的复合版面

面板允许你建立为多用户定制版面。这个例子我们建立 MSN 信息框，通过面板版面插件：



我们使用多种版面在面板区域中。最上面的信息框我们放置搜索 input，也可以放置头像在右面。中间区域我们差分成两部分通过 split 属性为 TRUE，允许用户改变面板上区域的大小：

代码：

```
<div class="easyui-panel" title="Complex Panel Layout"
icon="icon-search" collapsible="true"
style="padding:5px;width:500px;height:250px;">
    <div class="easyui-layout" fit="true">
        <div region="north" border="false" class="p-search">
            <label>Search:</label><input></input>
        </div>
        <div region="center" border="false">
            <div class="easyui-layout" fit="true">
                <div region="east" border="false"
class="p-right">
                    <img src="msn.gif"/>
                </div>
                <div region="center" border="false"
style="border:1px solid #ccc;">
                    <div class="easyui-layout" fit="true">
                        <div region="south" split="true"
border="false" style="height:60px;">
                            <textarea
style="overflow:auto;border:0;width:100%;height:100%;">Hi,
I am easyui.</textarea>
                        </div>
                        <div region="center" border="false">
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

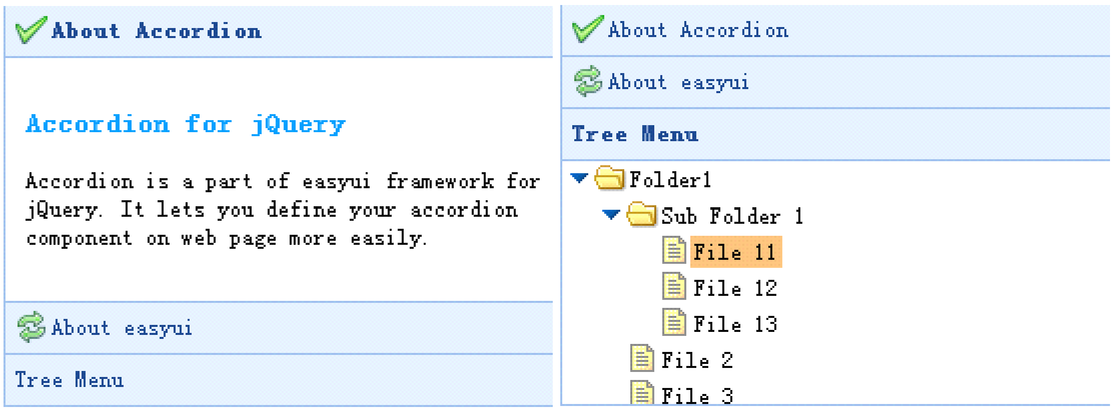
我们不需要编写任何 js 代码，但是拥有强大的用户接口设计的能力。

建立可折叠版面

这个教程中，我们学习关于 easyui 可折叠性。可折叠包括一系列面板。所有面板头是全部可见的，但是在一个时期内只有一个面板的 body 内容是可见的。当用户点击面板头，body 内容变为可见其他面板 body 内容变得不可见。

```
<div class="easyui-accordion"
style="width:300px;height:200px;">
    <div title="About Accordion" icon="icon-ok"
style="overflow:auto;padding:10px;">
        <h3 style="color:#0099FF;">Accordion for jQuery</h3>
        <p>Accordion is a part of easyui framework for jQuery.
It lets you define your accordion component on web page more
easily.</p>
    </div>
    <div title="About easyui" icon="icon-reload"
selected="true" style="padding:10px;">
        easyui help you build your web page easily
    </div>
    <div title="Tree Menu">
        <ul id="tt1" class="easyui-tree">
            <li>
                <span>Folder1</span>
                <ul>
                    <li>
                        <span>Sub Folder 1</span>
                        <ul>
                            <li>
                                <span>File 11</span>
                            </li>
                            <li>
                                <span>File 12</span>
                            </li>
                            <li>
                                <span>File 13</span>
                            </li>
                        </ul>
                    </li>
                    <li>
                        <span>File 2</span>
                    </li>
                    <li>
                        <span>File 3</span>
                    </li>
                </ul>
            </li>
            <li>
                <span>File2</span>
            </li>
        </ul>
    </div>
</div>
```

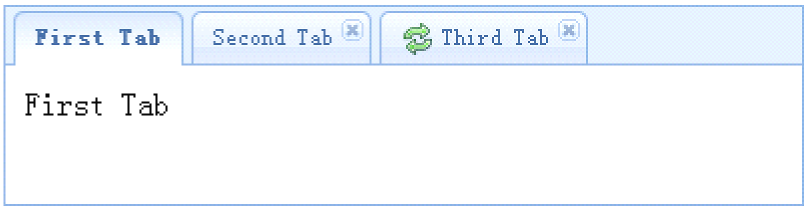
我们建立 3 个面板，第三个面板内容是一个树状菜单。



Accordion is a part of easyui framework for jQuery. It lets you define your accordion component on web page more easily.

建立 TABS

这个教程显示你如何使用 easyui 建立 tabs 组件。tabs 有多个面板，这些面板能被动态的添加或者删除。你可以使用 tabs 来显示不同的实体。在一个时间内只显示一个面板。每一个面板拥有 title，icon 和 close 按钮。当 tabs 被选择时，相关面板的内容被现实。



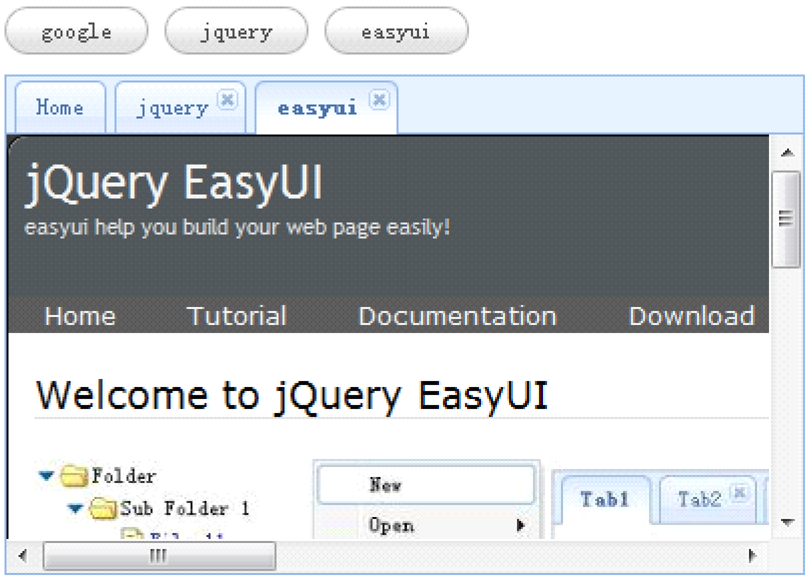
tabs 从 HTML 标记创建，包含 DIV 容器和一些 DIV 面板。

```
<div class="easyui-tabs"
style="width:400px;height:100px;">
   <div title="First Tab" style="padding:10px;">
      First Tab
   </div>
   <div title="Second Tab" closable="true"
style="padding:10px;">
      Second Tab
   </div>
   <div title="Third Tab" icon="icon-reload"
closable="true" style="padding:10px;">
      Third Tab
   </div>
</div>
```

我们创建 3 个面板的 tabs 组件，第二个和第三个面板可以通过点击 close 按钮关闭。

动态添加 tabs

你只需调用 add 方法，就可以使用 jquery easyui 很容易动态添加 tabs。在这个教程中，我们动态的添加显示一个页面使用 iframe。当点击添加 add 按钮，新 tab 被添加。如果 tab 已经存在，被激活。



第一步：建立 tabs

```
<div style="margin-bottom:10px">
```

```
    <a href="#" class="easyui-linkbutton"
onclick="addTab('google','http://www.google.com')">google
</a>
    <a href="#" class="easyui-linkbutton"
onclick="addTab('jquery','http://jquery.com/')">jquery</a
>
    <a href="#" class="easyui-linkbutton"
onclick="addTab('easyui','http://jquery-easyui.wikidot.co
m')">easyui</a>
</div>
<div id="tt" class="easyui-tabs"
style="width:400px;height:250px;">
    <div title="Home">
    </div>
</div>
```

HTML 代码很简单，我们创建 tabs 用一个 tab 面板，名字为 home。记住，我们不需要写任何 js 代码。

第二步：使 **addTab** 函数生效

```
function addTab(title, url){
    if ($('#tt').tabs('exists', title)){
        $('#tt').tabs('select', title);
    } else {
        var content = '<iframe scrolling="auto"
frameborder="0"  src="'+url+'"
style="width:100%;height:100%;"></iframe>';
        $('#tt').tabs('add',{
            title:title,
            content:content,
            closable:true
        });
    }
}
```

我们使用 exists 方法判断 tab 是否存在。如果存在，则激活 tab。调用 add 方法添加新 tab 面板。

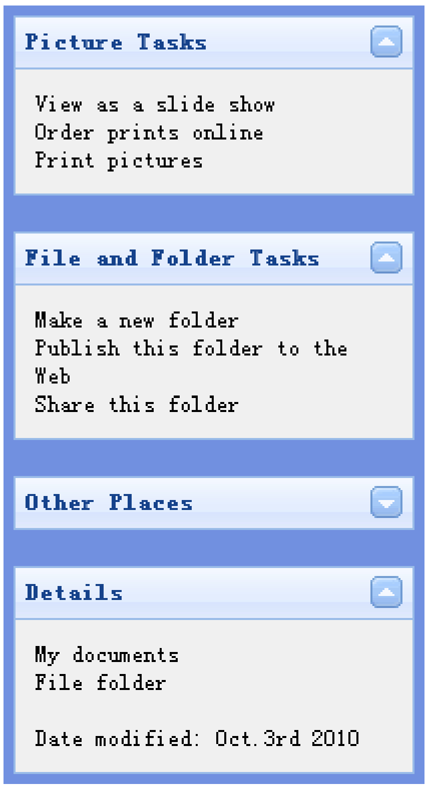
创建 XP 式样左面板

通常，浏览文件夹在 windowsXP 中有左面板，包括常用任务内容。这个教程显示你如何使用 easyui 面板插件建立 XP 左面板。

定义几个面板

我们几个面板显示一些任务，每个面板仅可以折叠和展开工具按钮。代码像这样：

```
<div
style="width:200px;height:auto;background:#7190E0;padding:
5px;">
   <div class="easyui-panel" title="Picture Tasks"
collapsible="true"
style="width:200px;height:auto;padding:10px;">
      View as a slide show<br/>
      Order prints online<br/>
      Print pictures
   </div>
   <br/>
   <div class="easyui-panel" title="File and Folder Tasks"
collapsible="true"
style="width:200px;height:auto;padding:10px;">
      Make a new folder<br/>
      Publish this folder to the Web<br/>
      Share this folder
   </div>
   <br/>
   <div class="easyui-panel" title="Other Places"
collapsible="true" collapsed="true"
style="width:200px;height:auto;padding:10px;">
      New York<br/>
      My Pictures<br/>
      My Computer<br/>
      My Network Places
   </div>
   <br/>
   <div class="easyui-panel" title="Details"
collapsible="true"
style="width:200px;height:auto;padding:10px;">
      My documents<br/>
      File folder<br/><br/>
      Date modified: Oct.3rd 2010
   </div>
</div>
```

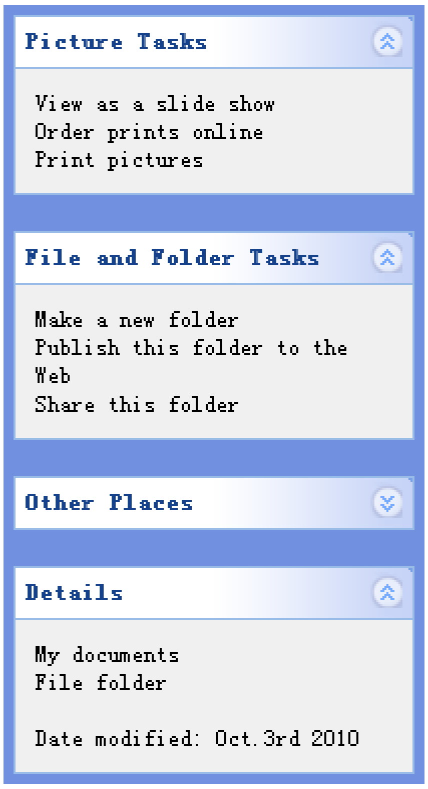


视图效果是不是我们想要的，我们必须改变面板 header 背景图片和收缩按钮 icon。

定制面板外观效果

做到这一点并不难，我们需要做的是重新定义一些 CSS。

```
.panel-header{
    background:#fff url('panel_header_bg.gif') no-repeat top right;
}
.panel-body{
    background:#f0f0f0;
}
.panel-tool-collapse{
    background:url('arrow_up.gif') no-repeat 0px -3px;
}
.panel-tool-expand{
    background:url('arrow_down.gif') no-repeat 0px -3px;
}
```



当使用 easyui 定义用户接口时是很简单的。

4. DataGrid 数据格
    - 转换 HTML 表格到 DataGrid
    - 给 DataGrid 添加分页
    - 从 DataGrid 中获得选定行的数据
    - 添加工具栏到 DataGrid
    - DataGrid 冻结列
    - 动态改变 DataGrid 列
    - 格式化 DataGrid 列
    - 添加 DataGrid 的分类
    - 在 DataGrid 中建立列组
    - 在 DataGrid 中选择复选框
    - 定制 DataGrid 页面
    - 使 DataGrid 能行嫩编辑
    - 合并 DataGrid 单元格

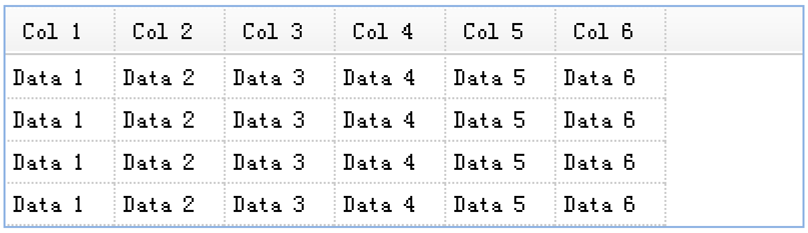
转换 HTML 表格到 DataGrid

这个例子显示如何转换表格到 DataGrid。DataGrid 在 thead 标记中定义列，在 tbody 标记中定义数据。确定给每一个数据列设置字段名，看这个例子：

```html
<table id="tt" class="easyui-datagrid"
style="width:400px;height:auto;">
   <thead>
      <tr>
         <th field="name1" width="50">Col 1</th>
         <th field="name2" width="50">Col 2</th>
         <th field="name3" width="50">Col 3</th>
         <th field="name4" width="50">Col 4</th>
         <th field="name5" width="50">Col 5</th>
         <th field="name6" width="50">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>
      <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>
```

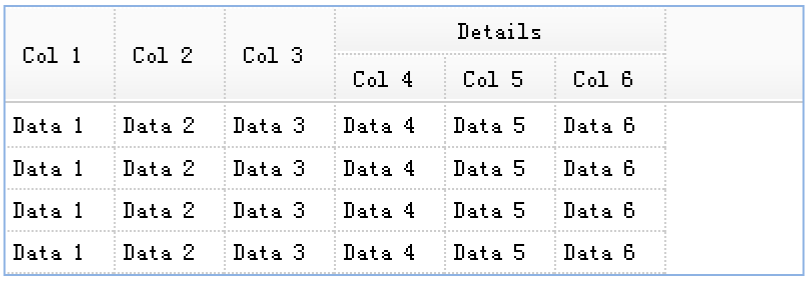
不需要 js 代码就能看到这个效果：



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

当然，你也可以定义复合表头，像这样：

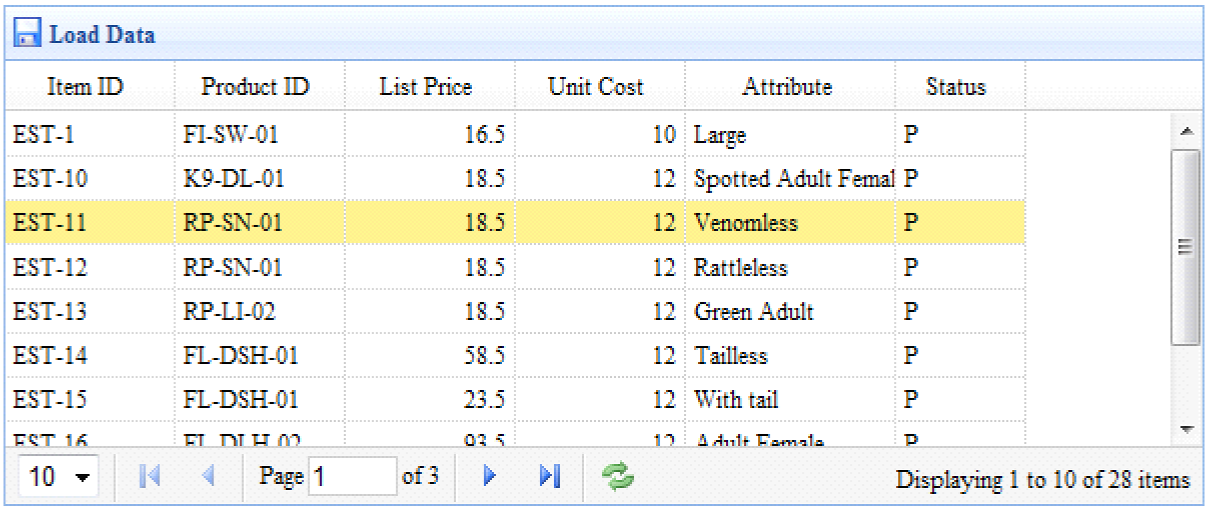
```
<thead>
    <tr>
        <th field="name1" width="50" rowspan="2">Col 1</th>
        <th field="name2" width="50" rowspan="2">Col 2</th>
        <th field="name3" width="50" rowspan="2">Col 3</th>
        <th colspan="3">Details</th>
    </tr>
    <tr>
        <th field="name4" width="50">Col 4</th>
        <th field="name5" width="50">Col 5</th>
        <th field="name6" width="50">Col 6</th>
    </tr>
</thead>
```



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

给 DataGrid 添加分页

这个例子显示如何能从服务器中调用数据，如何添加分页到 DataGrid 中。


Load Data

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Femal	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P

从远程服务器中调用数据，你必须设置 url 属性，服务器应该返回 JSON 格式数据。获得更多数据格式，请参考 DataGrid 文档。

建立<table>标记

首先，我们在网页上定义标记。

```
<table id="tt"></table>
```

jQuery 代码

然后，写一些 jQuery 代码建立 DataGrid 组件

```
$('#tt').datagrid({
    title:'Load Data',
    iconCls:'icon-save',
    width:600,
    height:250,
    url:'/demo3/data/getItems',
    columns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'listprice',title:'List
Price',width:80,align:'right'},
        {field:'unitcost',title:'Unit
Cost',width:80,align:'right'},
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]],
    pagination:true
});
```

我们定义 DataGrid 列并且设置 pagination 属性为 true，这样可以在 DataGrid 上产生分页栏按钮。分页发送 2 个参数到服务器。

- page：页号，从 1 开始。
- rows：每页的列数。

我们使用 etmvc framework 编写后台服务代码，所以，url 被映射到 DataController 类和 getItems 方法。

定义数据模型的例子

```
@Table(name="item")
public class Item extends ActiveRecordBase{
    @Id public String itemid;
    @Column public String productid;
    @Column public java.math.BigDecimal listprice;
    @Column public java.math.BigDecimal unitcost;
    @Column public String attr1;
    @Column public String status;
}
```

编写控制代码

```
public class DataController extends ApplicationController{
    /**
     * get item data
     * @param page page index
     * @param rows rows per page
     * @return JSON format string
     * @throws Exception
     */
    public View getItems(int page, int rows) throws Exception{
        long total = Item.count(Item.class, null, null);
        List<Item> items = Item.findAll(Item.class, null,
null, null, rows, (page-1)*rows);
        Map<String, Object> result = new HashMap<String,
Object>();
        result.put("total", total);
        result.put("rows", items);
        return new JsonView(result);
    }
}
```

数据库配置实例

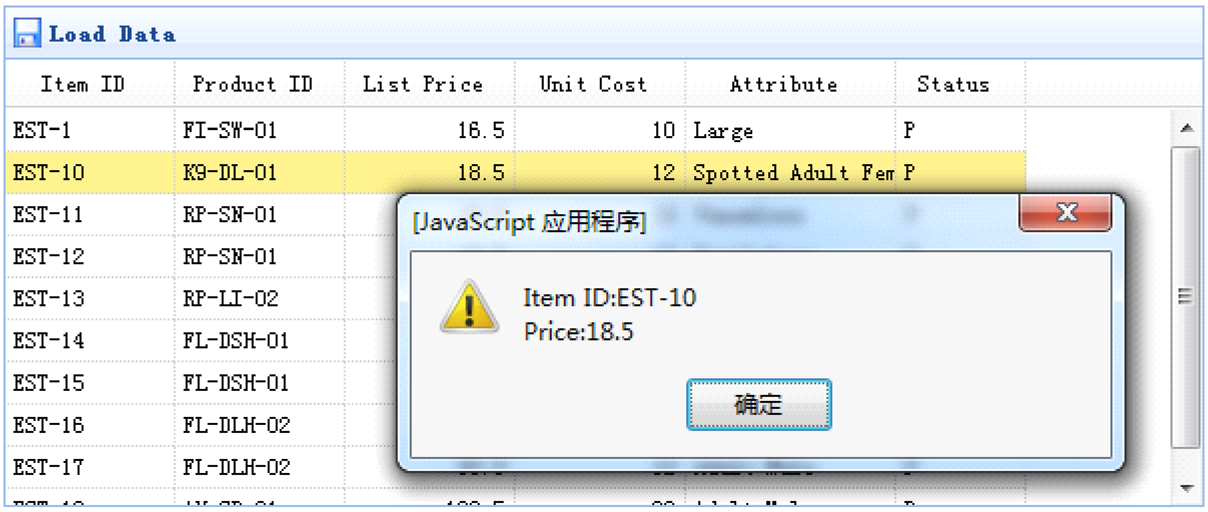
```
domain_base_class=com.et.ar.ActiveRecordBase

com.et.ar.ActiveRecordBase.adapter_class=com.et.ar.adapters.MySqlAdapter
com.et.ar.ActiveRecordBase.driver_class=com.mysql.jdbc.Driver
com.et.ar.ActiveRecordBase.url=jdbc:mysql://localhost/jpetstore
com.et.ar.ActiveRecordBase.username=root
com.et.ar.ActiveRecordBase.password=soft123456
com.et.ar.ActiveRecordBase.pool_size=0
```

部署

- 建立 MySQL 数据库
- 从'/db/item.sql'导入测试表数据，表名是'item'.
- 按需要改变数据库配置，配置文件在/WEB-INF/classes/activerecord.properties 中。
- 运行程序

得到 DataGrid 选择行

这个例子显示了如何得到选择行的数据。



Load Data

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Fem	P
EST-11	RP-SN-01				
EST-12	RP-SN-01				
EST-13	RP-LI-02				
EST-14	FL-DSH-01				
EST-15	FL-DSH-01				
EST-16	FL-DLH-02				
EST-17	FL-DLH-02				

[JavaScript 应用程序]

Item ID:EST-10
Price:18.5

DataGrid 组件包括 2 个方法检索选择行数据：

- getSelected：得到第一个选择行的数据，如果没有选择行则返回 null 否则返回该记录
- getSelections:得到全部的选择行的数据，如果元素是记录的话，返回数组数据

创建标记

```
<table id="tt"></table>
```

创建 **datagrid**

```
$('#tt').datagrid({
    title:'Load Data',
    iconCls:'icon-save',
    width:600,
    height:250,
    url:'datagrid_data.json',
    columns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'listprice',title:'List
Price',width:80,align:'right'},
        {field:'unitcost',title:'Unit
Cost',width:80,align:'right'},
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]]
});
```

用法演示

得到选择行数据：

```
var row = $('#tt').datagrid('getSelected');
if (row){
    alert('Item ID:'+row.itemid+"\nPrice:"+row.listprice);
}
```

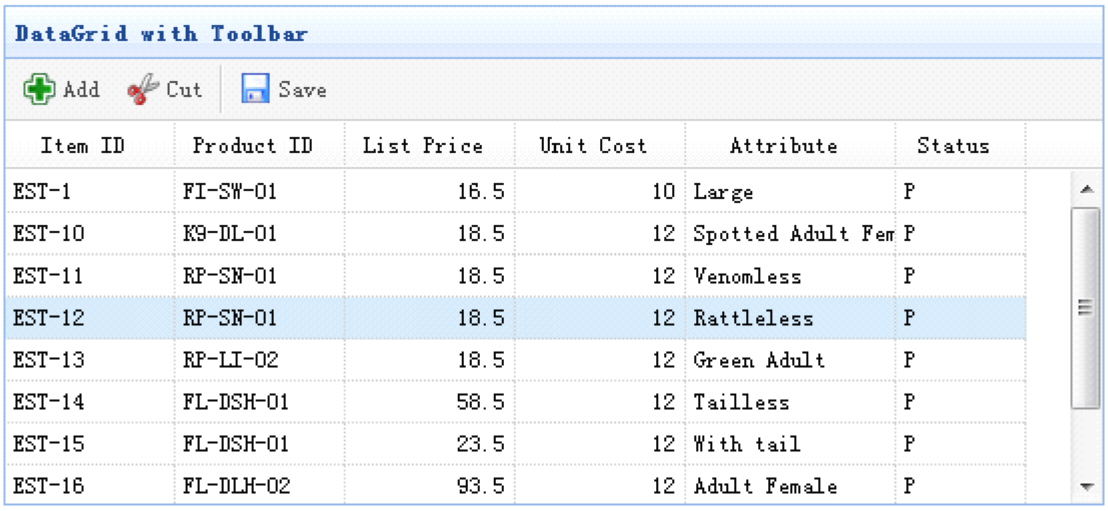
得到全部选择行的 itemid:

```
var ids = [];
var rows = $('#tt').datagrid('getSelections');
```

```
for(var i=0; i<rows.length; i++){
    ids.push(rows[i].itemid);
}
alert(ids.join('\n'));
```

添加工具栏到 DataGrid

这个例子显示了如何添加工具栏：



Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Fem	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P

DataGrid 插件有工具栏属性，这个属性可以定义工具栏。工具栏包括定义了下列属性的按钮：

- text：在按钮上显示的文本
- iconCls：定义背景图标显示在按钮的左面的 CSS 类。
- handler：当用户按下按钮时，处理一些事情的函数

标记

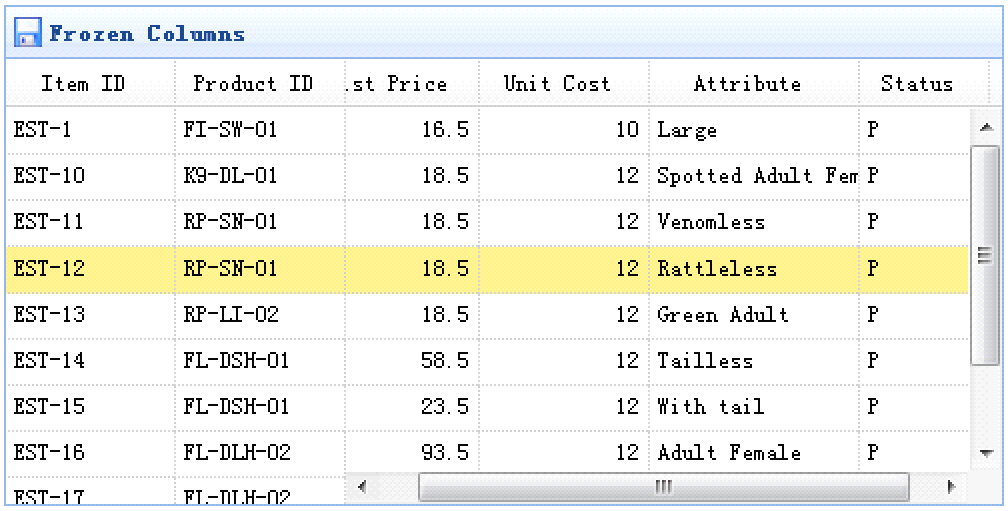
```
<table id="tt"></table>
```

jQuery

```
$('#tt').datagrid({
    title:'DataGrid with Toolbar',
    width:550,
    height:250,
    url:'datagrid_data.json',
    columns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'listprice',title:'List
Price',width:80,align:'right'},
        {field:'unitcost',title:'Unit
Cost',width:80,align:'right'},
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]],
    toolbar:[{
        text:'Add',
        iconCls:'icon-add',
        handler:function(){
            alert('add')
        }
    },{
        text:'Cut',
        iconCls:'icon-cut',
        handler:function(){
            alert('cut')
        }
    },'-',{
        text:'Save',
        iconCls:'icon-save',
        handler:function(){
            alert('save')
        }
    }]
});
```

DataGrid 冻结列

这个例子演示了如何冻结列。当用户水平滚动的时候，冻结列不能滚动出视图。



Frozen Columns

Item ID	Product ID	st Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Fem	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P
EST-17	FL-DLH-02				

冻结列，你应该定义 frozenColumns 属性，这个属性和 columns 属性相似。

```
<table id="tt"></table>
$('#tt').datagrid({
    title:'Frozen Columns',
    iconCls:'icon-save',
    width:500,
    height:250,
    url:'datagrid_data.json',
    frozenColumns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
    ]],
    columns:[[
        {field:'listprice',title:'List
Price',width:80,align:'right'},
        {field:'unitcost',title:'Unit
Cost',width:80,align:'right'},
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]]
});
```

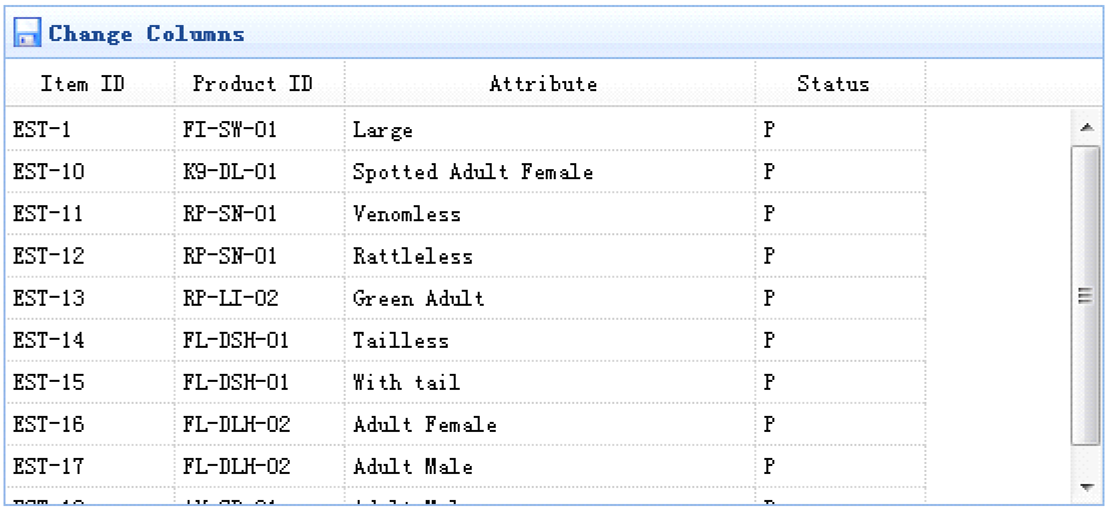
动态改变 DataGrid 列

DataGrid 列可以使用 columns 属性定义。如果你想动态改变列，也没问题。改变列你可以重新调用 DataGrid 方法平且传递新 columns 属性。

下面定义 DataGrid 组件

```
<table id="tt"></table>
$('#tt').datagrid({
    title:'Change Columns',
    iconCls:'icon-save',
    width:550,
    height:250,
    url:'datagrid_data.json',
    columns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'attr1',title:'Attribute',width:200},
        {field:'status',title:'Status',width:80}
    ]]
});
```

运行网页，我们看到：

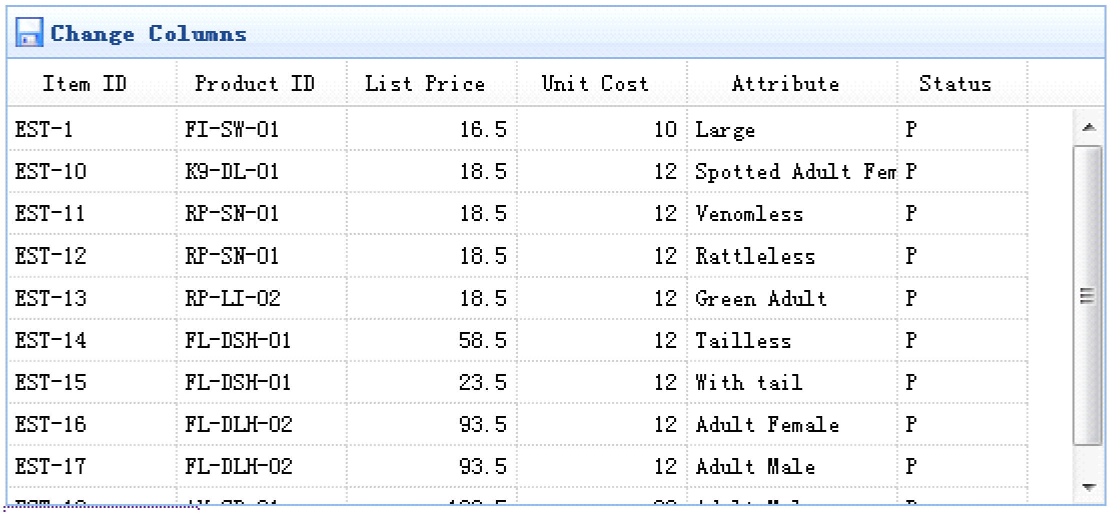

Change Columns

Item ID	Product ID	Attribute	Status
EST-1	FI-SW-01	Large	P
EST-10	K9-DL-01	Spotted Adult Female	P
EST-11	RP-SN-01	Venomless	P
EST-12	RP-SN-01	Rattleless	P
EST-13	RP-LI-02	Green Adult	P
EST-14	FL-DSH-01	Tailless	P
EST-15	FL-DSH-01	With tail	P
EST-16	FL-DLH-02	Adult Female	P
EST-17	FL-DLH-02	Adult Male	P

通常，我们想改变列，你可以写这些代码：

```
$('#tt').datagrid({
    columns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'listprice',title:'List
Price',width:80,align:'right'},
        {field:'unitcost',title:'Unit
Cost',width:80,align:'right'},
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]]
});
```

记住，我们已经定义其他属性，比如：url，width，height 等，我们不需要重复定义他们，我们定义我们想改变的。

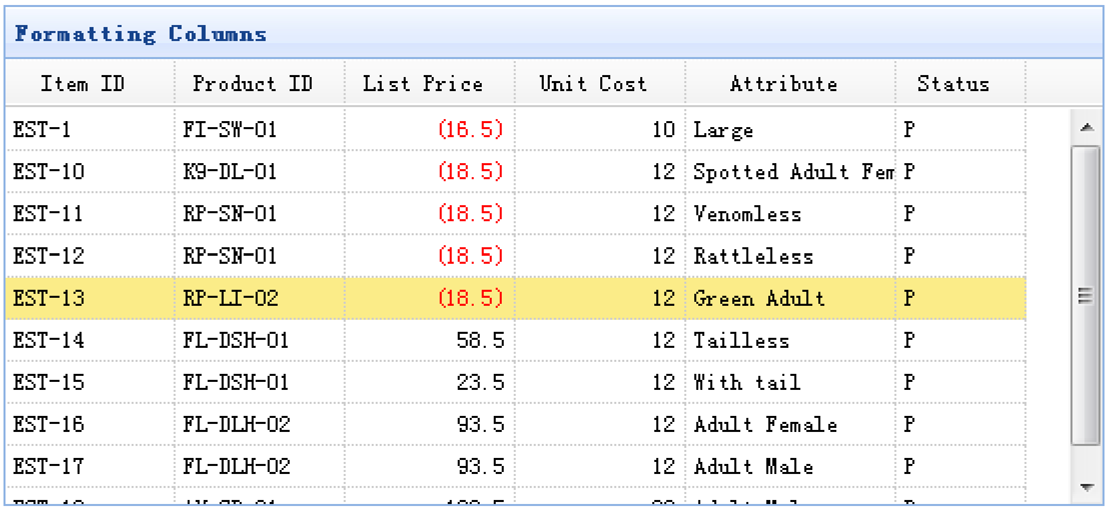


Change Columns

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Fem	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P
EST-17	FL-DLH-02	93.5	12	Adult Male	P

格式化 DataGrid 列

下面的例子是在 easyui DataGrid 中格式化列，如果单价低于 20，则使用定义列 formatter 为红色文本。



Formatting Columns

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	(16.5)	10	Large	P
EST-10	K9-DL-01	(18.5)	12	Spotted Adult Fem	P
EST-11	RP-SN-01	(18.5)	12	Venomless	P
EST-12	RP-SN-01	(18.5)	12	Rattleless	P
EST-13	RP-LI-02	(18.5)	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P
EST-17	FL-DLH-02	93.5	12	Adult Male	P

格式化 DataGrid 列，我们应该设置 formatter 属性，这个属性是一个函数。格式化函数包括两个参数：

- value：显示字段当前列的值
- record：当前行记录数据

Markup

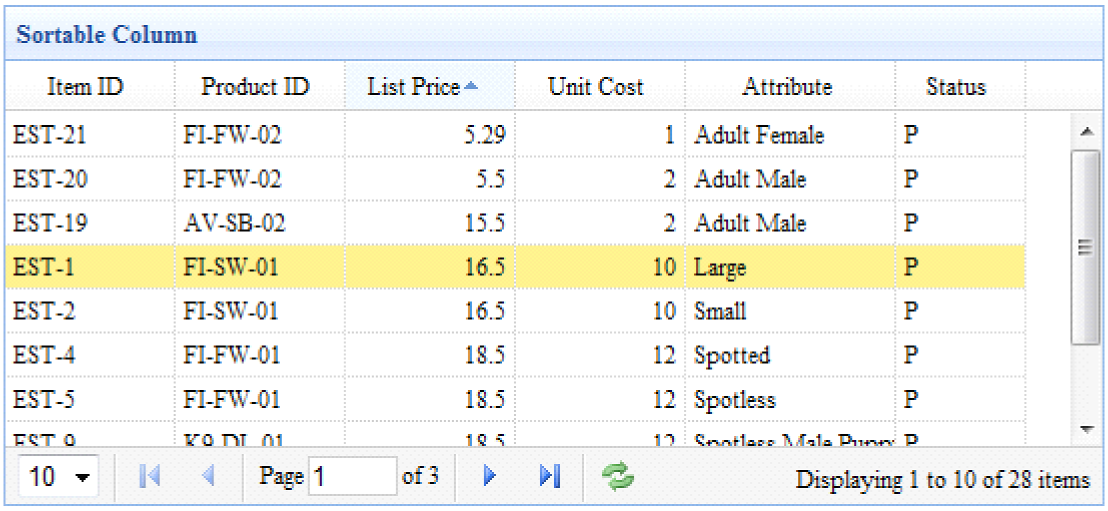
```
<table id="tt"></table>
```

jQuery

```
$('#tt').datagrid({
    title:'Formatting Columns',
    width:550,
    height:250,
    url:'datagrid_data.json',
    columns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'listprice',title:'List
Price',width:80,align:'right',
            formatter:function(val,rec){
                if (val < 20){
                    return '<span
style="color:red;">('+val+')</span>';
                } else {
                    return val;
                }
            }
        },
        {field:'unitcost',title:'Unit
Cost',width:80,align:'right'},
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]]
});
```

添加排序到 DataGrid

这个事例演示了如何在点击列头的时候排序


Sortable Column

Item ID	Product ID	List Price ▲	Unit Cost	Attribute	Status
EST-21	FI-FW-02	5.29	1	Adult Female	P
EST-20	FI-FW-02	5.5	2	Adult Male	P
EST-19	AV-SB-02	15.5	2	Adult Male	P
EST-1	FI-SW-01	16.5	10	Large	P
EST-2	FI-SW-01	16.5	10	Small	P
EST-4	FI-FW-01	18.5	12	Spotted	P
EST-5	FI-FW-01	18.5	12	Spotless	P
EST-9	K9-DL-01	18.5	12	Spotless Male Pupp	P

Displaying 1 to 10 of 28 items

DataGrid 中全部的列可以通过点击列头被排序。你可以定义可以被排序的列。默认的，列不能被排序除非你设置 sortable 属性为 TRUE，下面是例子：

标记

```
<table id="tt"></table>
```

jQuery

```
$('#tt').datagrid({
    title:'Sortable Column',
    width:550,
    height:250,
    url:'/demo4/data/getItems',
    columns:[[
        {field:'itemid',title:'Item 
ID',width:80,sortable:true},
        {field:'productid',title:'Product 
ID',width:80,sortable:true},
        {field:'listprice',title:'List 
Price',width:80,align:'right',sortable:true},
        {field:'unitcost',title:'Unit 
Cost',width:80,align:'right',sortable:true},
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]],
    pagination:true,
    sortName:'itemid',
    sortOrder:'asc'
});
```

我们定义一些可排序的列，包括 itemid，productid，listprice，unitcost 等。attr1 列和 status 列不能被排序。我们设置默认排序列：itemid，按 asc（升序）排序。

当排序时， DataGrid 发送两个参数到服务器：

- sort: 排序列字段名
- order: 排序次序： 'asc' 或 'desc', 默认为'asc'.

我们使用 etmvc framework 写后台服务器代码，首先定义数据模型

```
@Table(name="item")
public class Item extends ActiveRecordBase{
    @Id public String itemid;
    @Column public String productid;
    @Column public java.math.BigDecimal listprice;
    @Column public java.math.BigDecimal unitcost;
    @Column public String attr1;
    @Column public String status;
}
```

写控制代码：

```
public class DataController extends ApplicationController{
    /**
     * get item data
     * @param page page number
     * @param rows page size
     * @param sort sort column field name
     * @param order sort order, can be 'asc' or 'desc'
     * @return JSON format string
     * @throws Exception
     */
    public View getItems(int page, int rows, String sort,
String order) throws Exception{
        long total = Item.count(Item.class, null, null);
        List<Item> items = Item.findAll(Item.class, null,
null, sort+" "+order, rows, (page-1)*rows);
        Map<String, Object> result = new HashMap<String,
Object>();
        result.put("total", total);
        result.put("rows", items);
        return new JsonView(result);
    }
}
```

我们使用 MySQL 数据库存储演示数据，下面是配置实例：

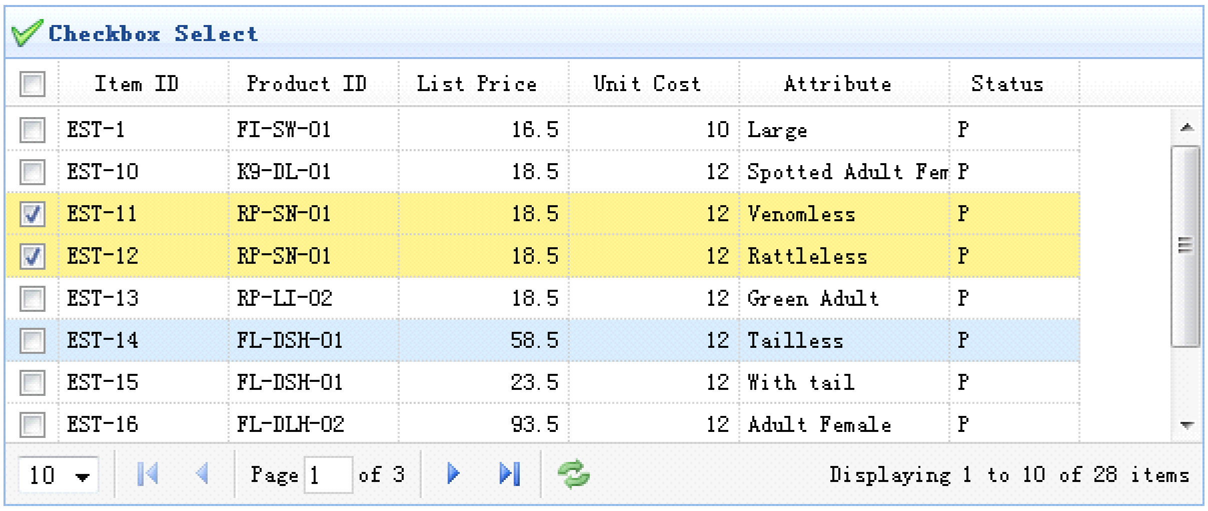
```
domain_base_class=com.et.ar.ActiveRecordBase

com.et.ar.ActiveRecordBase.adapter_class=com.et.ar.adapters.MySqlAdapter
com.et.ar.ActiveRecordBase.driver_class=com.mysql.jdbc.Driver
com.et.ar.ActiveRecordBase.url=jdbc:mysql://localhost/jpetstore
com.et.ar.ActiveRecordBase.username=root
com.et.ar.ActiveRecordBase.password=soft123456
com.et.ar.ActiveRecordBase.pool_size=0
```

部署

- 建立 MySQL 数据库
- 从'/db/item.sql'导入测试表数据，表名是'item'.
- 按需要改变数据库配置，配置文件在/WEB-INF/classes/activerecord.properties 中。
- 运行程序

在 DataGrid 上的复选框

本教程显示了你如何放置 checkbox 列。使用 checkbox，用户可以选定/取消数据行。


Checkbox Select

	Item ID	Product ID	List Price	Unit Cost	Attribute	Status
☐	EST-1	FI-SW-01	16.5	10	Large	P
☐	EST-10	K9-DL-01	18.5	12	Spotted Adult Fem	P
✓	EST-11	RP-SN-01	18.5	12	Venomless	P
✓	EST-12	RP-SN-01	18.5	12	Rattleless	P
☐	EST-13	RP-LI-02	18.5	12	Green Adult	P
☐	EST-14	FL-DSH-01	58.5	12	Tailless	P
☐	EST-15	FL-DSH-01	23.5	12	With tail	P
☐	EST-16	FL-DLH-02	93.5	12	Adult Female	P

10 Page 1 of 3 Displaying 1 to 10 of 28 items

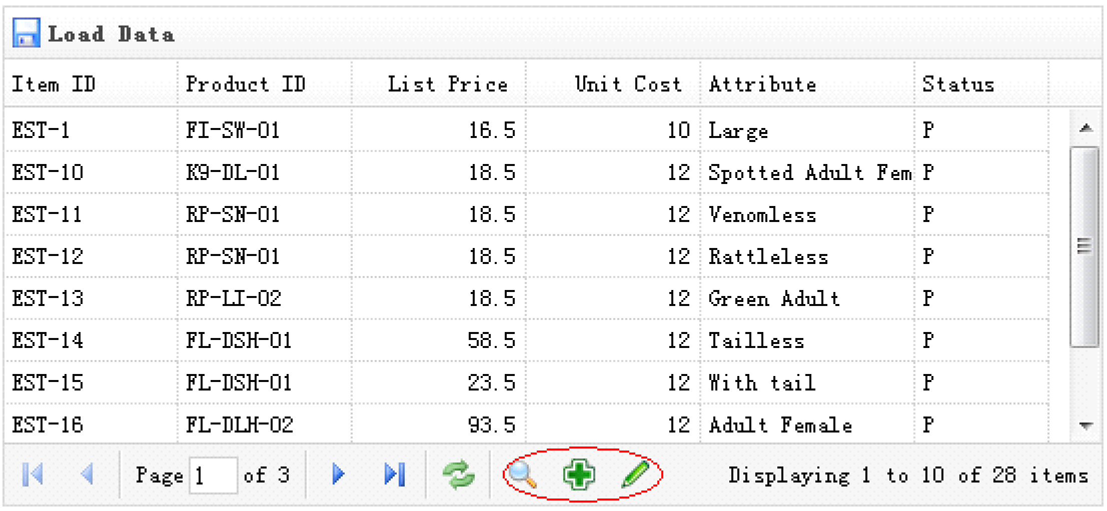
添加 checkbox 列，我们简单的添加列的 checkbox 属性，并且设置为 true。代码像这样：

```
<table id="tt"></table>
$('#tt').datagrid({
    title:'Checkbox Select',
    iconCls:'icon-ok',
    width:600,
    height:250,
    url:'datagrid_data.json',
    idField:'itemid',
    columns:[[
        {field:'ck',checkbox:true},
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'listprice',title:'List
Price',width:80,align:'right'},
        {field:'unitcost',title:'Unit
Cost',width:80,align:'right'},
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]],
    pagination:true
});
```

上面的代码，我们可以添加列的 checkbox 属性，然后他就会出现选择列。如果 idField 属性被设置，DataGrid 的选择会被不同的页保持。

自定义 DataGrid 分页

DataGrid 内建分页能力是强大的，它比自定义相对容易。在这个教程，我们将要创建 DataGrid 并且在页面工具栏中添加一些自定义按钮。


Load Data

Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	FI-SW-01	16.5	10	Large	P
EST-10	K9-DL-01	18.5	12	Spotted Adult Fem	P
EST-11	RP-SN-01	18.5	12	Venomless	P
EST-12	RP-SN-01	18.5	12	Rattleless	P
EST-13	RP-LI-02	18.5	12	Green Adult	P
EST-14	FL-DSH-01	58.5	12	Tailless	P
EST-15	FL-DSH-01	23.5	12	With tail	P
EST-16	FL-DLH-02	93.5	12	Adult Female	P

标记

```
<table id="tt"></table>
```

创建 DataGrid

```
$('#tt').datagrid({
    title:'Load Data',
    iconCls:'icon-save',
    width:550,
    height:250,
    pagination:true,
    url:'datagrid_data.json',
    columns:[[
        {field:'itemid',title:'Item ID',width:80},
        {field:'productid',title:'Product ID',width:80},
        {field:'listprice',title:'List
Price',width:80,align:'right'},
        {field:'unitcost',title:'Unit
Cost',width:80,align:'right'},
        {field:'attr1',title:'Attribute',width:100},
        {field:'status',title:'Status',width:60}
    ]]
});
```

记住设置 pagination 属性为 true 产生页面工具栏。

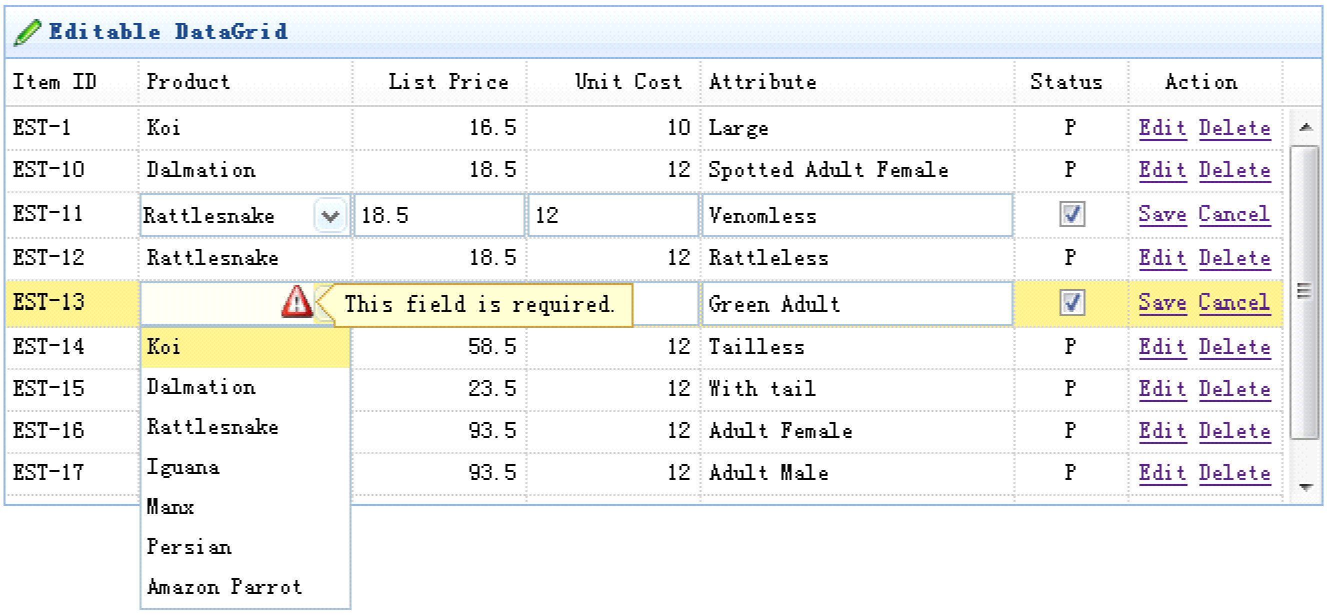
自定义页面工具栏

```
var pager = $('#tt').datagrid('getPager');    //得到 DataGrid 页面
pager.pagination({
    showPageList:false,
    buttons:[{
        iconCls:'icon-search',
        handler:function(){
            alert('search');
        }
    },{
        iconCls:'icon-add',
        handler:function(){
            alert('add');
        }
    },{
        iconCls:'icon-edit',
        handler:function(){
            alert('edit');
        }
    }],
    onBeforeRefresh:function(){
        alert('before refresh');
        return true;
    }
});
```

我们得到 DataGrid 页，然后重新构建页面。我们隐藏页列表然后添加新按钮。

使 DataGrid 能编辑

可编辑特征是最近添加的。它能让用户添加新行。用户也可以更新一行或多行。这个教程显示了如何创建使用行内编辑的 DataGrid。



Editable DataGrid

Item ID	Product	List Price	Unit Cost	Attribute	Status	Action
EST-1	Koi	16.5	10	Large	P	Edit Delete
EST-10	Dalmation	18.5	12	Spotted Adult Female	P	Edit Delete
EST-11	Rattlesnake	18.5	12	Venomless	✓	Save Cancel
EST-12	Rattlesnake	18.5	12	Rattleless	P	Edit Delete
EST-13	This field is required.			Green Adult	✓	Save Cancel
EST-14		58.5	12	Tailless	P	Edit Delete
EST-15		23.5	12	With tail	P	Edit Delete
EST-16		93.5	12	Adult Female	P	Edit Delete
EST-17		93.5	12	Adult Male	P	Edit Delete

Koi
Dalmation
Rattlesnake
Iguana
Manx
Persian
Amazon Parrot

创建 DataGrid

```
<table id="tt"></table>
$('#tt').datagrid({
    title:'Editable DataGrid',
    iconCls:'icon-edit',
    width:660,
    height:250,
    singleSelect:true,
    idField:'itemid',
    url:'datagrid_data.json',
    columns:[[
        {field:'itemid',title:'Item ID',width:60},
        {field:'productid',title:'Product',width:100,
            formatter:function(value){
                for(var i=0; i<products.length; i++){
                    if (products[i].productid == value) return
products[i].name;
                }
                return value;
            },
            editor:{
                type:'combobox',
                options:{
                    valueField:'productid',
                    textField:'name',
                    data:products,
                    required:true
                }
            }
        },
        {field:'listprice',title:'List
Price',width:80,align:'right',editor:{type:'numberbox',op
tions:{precision:1}}},
        {field:'unitcost',title:'Unit
Cost',width:80,align:'right',editor:'numberbox'},

{field:'attr1',title:'Attribute',width:150,editor:'text'}
,

{field:'status',title:'Status',width:50,align:'center',
            editor:{
                type:'checkbox',
                options:{
                    on: 'P',
                    off: ''
                }
            }
        },

{field:'action',title:'Action',width:70,align:'center',
            formatter:function(value,row,index){
                if (row.editing){
                    var s = '<a href="#"
onclick="saverow('+index+')">Save</a> ';
                    var c = '<a href="#"
onclick="cancelrow('+index+')">Cancel</a>';
                    return s+c;
                } else {
                    var e = '<a href="#"
onclick="editrow('+index+')">Edit</a> ';
                    var d = '<a href="#"
onclick="deleterow('+index+')">Delete</a>';
                    return e+d;
                }
            }
        }
    ]],
    onBeforeEdit:function(index,row){
        row.editing = true;
        $('#tt').datagrid('refreshRow', index);
    },
    onAfterEdit:function(index,row){
        row.editing = false;
        $('#tt').datagrid('refreshRow', index);
    },
    onCancelEdit:function(index,row){
        row.editing = false;
        $('#tt').datagrid('refreshRow', index);
    }
});
```

使 DataGrid 可编辑，你应该添加 editor 属性到列中。editor 告诉 DataGrid 如何编辑字和如何存储值。我们定义了三个 editor：text，combobox，checkbox。

添加编辑功能

```
function editrow(index){
    $('#tt').datagrid('beginEdit', index);
}
function deleterow(index){
    $.messager.confirm('Confirm','Are you
sure?',function(r){
        if (r){
            $('#tt').datagrid('deleteRow', index);
        }
    });
}
function saverow(index){
    $('#tt').datagrid('endEdit', index);
}
function cancelrow(index){
    $('#tt').datagrid('cancelEdit', index);
}
```

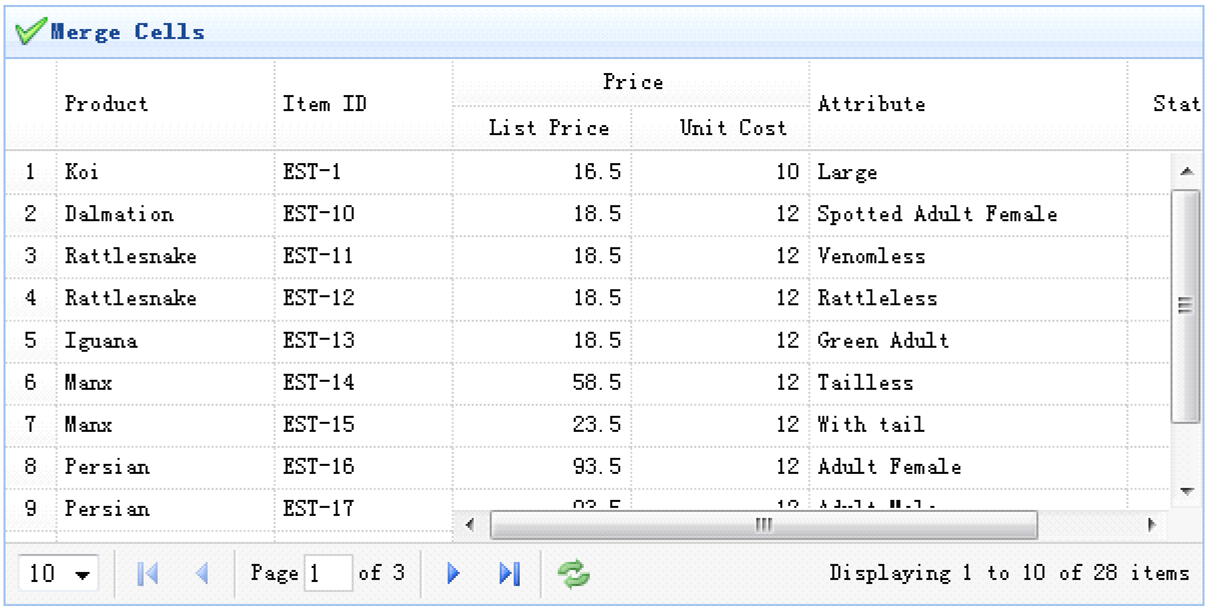
DataGrid 中合并单元格

合并一些单元格经常是必要的，这个教程显示了你如何合并单元格：

合并单元格，简单的调用 mergeCells 方法并传递信息参数就能告诉 DataGrid 如何合并单元格了。当单元格合并时，每种东西在合并单元格中，除了第一个单元格，都会被隐藏。

创建 DataGrid

```
<table id="tt"></table>
$('#tt').datagrid({
    title:'Merge Cells',
    iconCls:'icon-ok',
    width:600,
    height:300,
    singleSelect:true,
    rownumbers:true,
    idField:'itemid',
    url:'datagrid_data.json',
    pagination:true,
    frozenColumns:[[
        {field:'productid',title:'Product',width:100,
            formatter:function(value){
                for(var i=0; i<products.length; i++){
                    if (products[i].productid == value) return
products[i].name;
                }
                return value;
            }
        },
        {field:'itemid',title:'Item ID',width:80}
    ]],
    columns:[[
        {title:'Price',colspan:2},

{field:'attr1',title:'Attribute',width:150,rowspan:2},

{field:'status',title:'Status',width:60,align:'center',ro
wspan:2}
    ],[
        {field:'listprice',title:'List
Price',width:80,align:'right'},
        {field:'unitcost',title:'Unit
Cost',width:80,align:'right'}
    ]]
});
```


Merge Cells

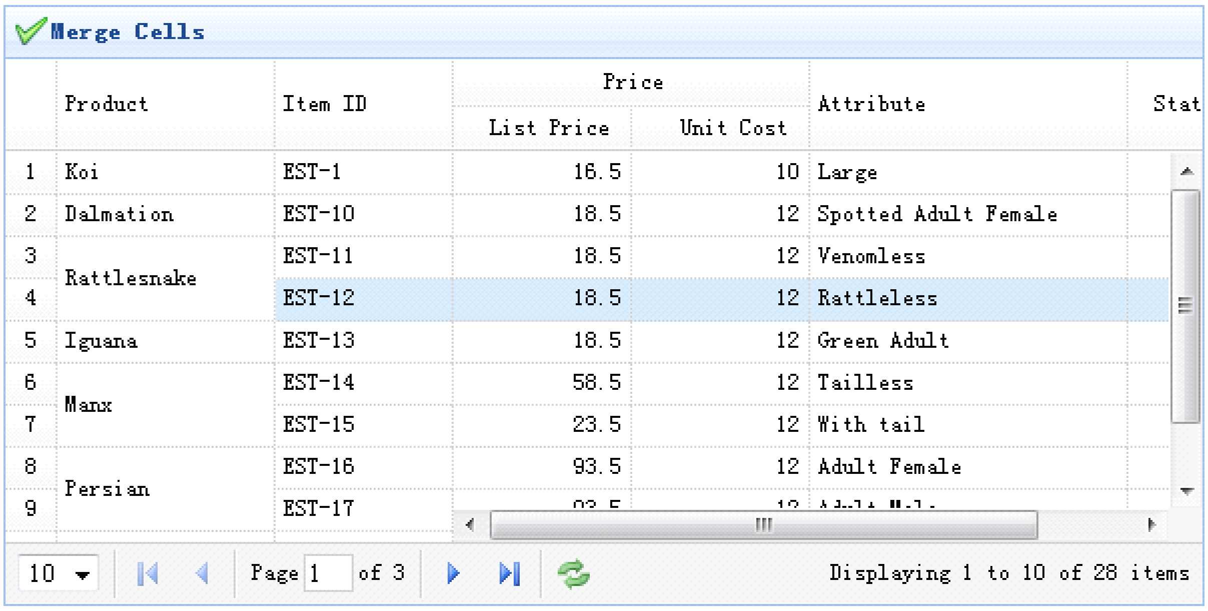
	Product	Item ID	Price		Attribute	Stat
			List Price	Unit Cost		
1	Koi	EST-1	16.5	10	Large	
2	Dalmation	EST-10	18.5	12	Spotted Adult Female	
3	Rattlesnake	EST-11	18.5	12	Venomless	
4	Rattlesnake	EST-12	18.5	12	Rattleless	
5	Iguana	EST-13	18.5	12	Green Adult	
6	Manx	EST-14	58.5	12	Tailless	
7	Manx	EST-15	23.5	12	With tail	
8	Persian	EST-16	93.5	12	Adult Female	
9	Persian	EST-17				

10 Page 1 of 3 Displaying 1 to 10 of 28 items

合并单元格

当数据被载入，我们在 DataGrid 中合并一些单元格，所以放置下列代码在 onLoadSuccess 函数中。

```
var merges = [{
    index:2,
    rowspan:2
},{
    index:5,
    rowspan:2
},{
    index:7,
    rowspan:2
}];
for(var i=0; i<merges.length; i++)
    $('#tt').datagrid('mergeCells',{
        index:merges[i].index,
        field:'productid',
        rowspan:merges[i].rowspan
    });
```


Merge Cells

	Product	Item ID	Price		Attribute	Stat
			List Price	Unit Cost		
1	Koi	EST-1	16.5	10	Large	
2	Dalmation	EST-10	18.5	12	Spotted Adult Female	
3	Rattlesnake	EST-11	18.5	12	Venomless	
4		EST-12	18.5	12	Rattleless	
5	Iguana	EST-13	18.5	12	Green Adult	
6	Manx	EST-14	58.5	12	Tailless	
7		EST-15	23.5	12	With tail	
8	Persian	EST-16	93.5	12	Adult Female	
9		EST-17	[illegible]	[illegible]	[illegible]	

10 Page 1 of 3 Displaying 1 to 10 of 28 items

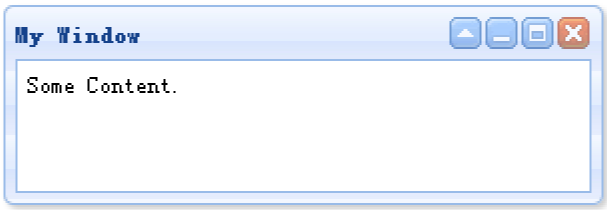
5. 窗口
    - 我第一个窗口
    - 自定义窗口工具
    - 窗口和版面
    - 创建对话框

我第一个窗口

建立窗口时很简单的，我们建立 DIV 标记：

```
<div id="win" class="easyui-window" title="My Window"
style="width:300px;height:100px;padding:5px;">
    Some Content.
</div>
```

然后测试就出出现一个窗口，我们不用写任何 js 代码


My Window

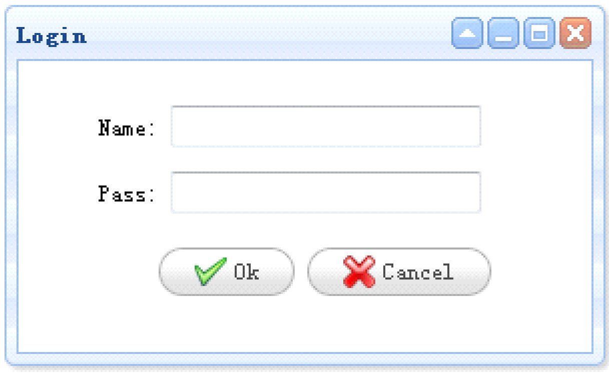
Some Content.

如果你想建立看不见的窗口，记住设置 closed 属性为 true，你能调用 open 方法打开窗口：

```
<div id="win" class="easyui-window" title="My Window"
closed="true"
style="width:300px;height:100px;padding:5px;">
    Some Content.
</div>
$('#win').window('open');
```

这个演示，我们创建一个登陆窗口

```
<div id="win" class="easyui-window" title="Login"
style="width:300px;height:180px;">
    <form style="padding:10px 20px 10px 40px;">
        <p>Name: <input type="text"></p>
        <p>Pass: <input type="password"></p>
        <div style="padding:5px;text-align:center;">
            <a href="#" class="easyui-linkbutton"
icon="icon-ok">Ok</a>
            <a href="#" class="easyui-linkbutton"
icon="icon-cancel">Cancel</a>
        </div>
    </form>
</div>
```

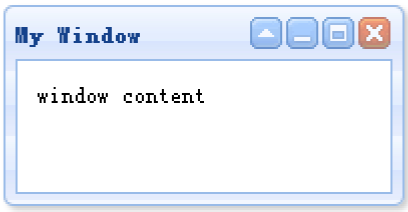


自定义窗口工具

默认的窗口有 4 个工具：collapsible（可折叠）,minimizable（最小化）,maximizable（最大化）和 closable（关闭），例如，我们定义下列窗口：

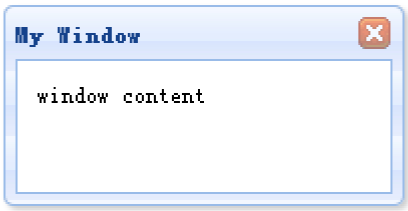
```
<div id="win" class="easyui-window" title="My Window"
style="padding:10px;width:200px;height:100px;">
```

```
    window content
</div>
```



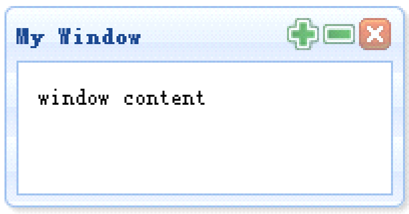
自定义工具，设置工具为 true 或者 false。例如，我们希望窗口只有一个 closeable 工具，可以设置任何其他工具为 false。我们可以定义工具属性在标记中或者 jquery 代码中。现在我们使用 jquery 代码来定义窗口：

```
$('#win').window({
    collapsible:false,
    minimizable:false,
    maximizable:false
});
```



如果你想添加自定义工具到窗口，我们可以使用 tools 属性，下面演示了我们添加自己的两个工具：

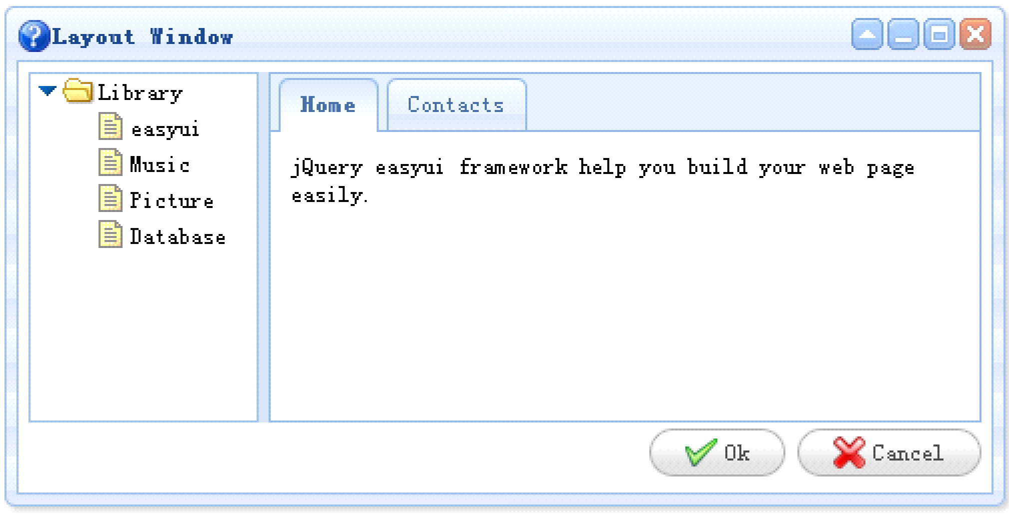
```
$('#win').window({
    collapsible:false,
    minimizable:false,
    maximizable:false,
    tools:[{
        iconCls:'icon-add',
        handler:function(){
            alert('add');
        }
    },{
        iconCls:'icon-remove',
        handler:function(){
            alert('remove');
        }
    }]
});
```



window content

Window 和 Layout

版式组件可以嵌套在窗口。我们可以创建复合版面窗口和事件而不用写任何 JS 代码.jquery-easyui 框架在后台帮助我们进行渲染和改变工作。这个例子我们创建窗口，这个窗口有左右两部分。在左窗口，我们建立 tree，在右窗口，我们建立 tabs 内容。



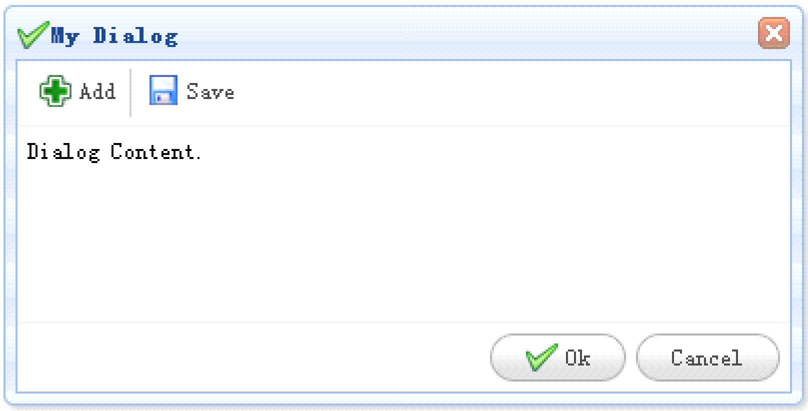
jQuery easyui framework help you build your web page easily.

```
<div class="easyui-window" title="Layout Window"
icon="icon-help"
style="width:500px;height:250px;padding:5px;background:
#fafafa;">
    <div class="easyui-layout" fit="true">
        <div region="west" split="true"
style="width:120px;">
            <ul class="easyui-tree">
                <li>
                    <span>Library</span>
                    <ul>
                        <li><span>easyui</span></li>
                        <li><span>Music</span></li>
                        <li><span>Picture</span></li>
                        <li><span>Database</span></li>
                    </ul>
                </li>
            </ul>
        </div>
        <div region="center" border="false" border="false">
            <div class="easyui-tabs" fit="true">
                <div title="Home" style="padding:10px;">
                    jQuery easyui framework help you build your
web page easily.
                </div>
                <div title="Contacts">
                    No contact data.
                </div>
            </div>
        </div>
        <div region="south" border="false"
style="text-align:right;height:30px;line-height:30px;">
            <a class="easyui-linkbutton" icon="icon-ok"
href="javascript:void(0)">Ok</a>
            <a class="easyui-linkbutton" icon="icon-cancel"
href="javascript:void(0)">Cancel</a>
        </div>
    </div>
</div>
```

看上面的代码，我们只需使用 HTML 标记，然后复合版面和 window 就会显示。这个 jquery-easyui 框架，是容易和强大的。

创建对话框

对话框是特殊的窗口，它能包括上面的工具栏和下面的按钮。默认对话框不能改变大小，但是用户可以设置 resizeable 属性为 true 来使它可以被改变大小：



Dialog Content.

对话框非常简单，可以使用 DIV 标记创建：

```
<div id="dd"
style="padding:5px;width:400px;height:200px;">
    Dialog Content.
</div>
$('#dd').dialog({
    title:'My Dialog',
    iconCls:'icon-ok',
    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(){
            $('#dd').dialog('close');
        }
    }]
});
```

上面的代码创一个有工具栏和按钮的对话框。这是对话框、工具栏、内容和按钮的标准设置。

6. Tree
    - 从标记创建 tree
    - 创建异步 tree

- 添加 tree 节点
- 创建 checkbox 节点的 tree

从标记创建 tree

tree 可以被从标记创建。easyui tree 应该定义在 ul 元素中。无序列表 ul 元素提供了基本 tree 结构。每一个 li 元素被产生一个 tree 节点，子 ul 元素产生父 tree 节点。

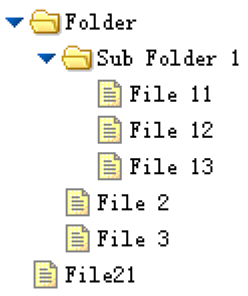
例子：

```
    <ul id="tt">
        <li>
            <span>Folder</span>
            <ul>
                <li>
                    <span>Sub Folder 1</span>
                    <ul>
                        <li><span>File 11</span></li>
                        <li><span>File 12</span></li>
                        <li><span>File 13</span></li>
                    </ul>
                </li>
                <li><span>File 2</span></li>
                <li><span>File 3</span></li>
            </ul>
        </li>
        <li><span>File21</span></li>
    </ul>
```

创建 tree:

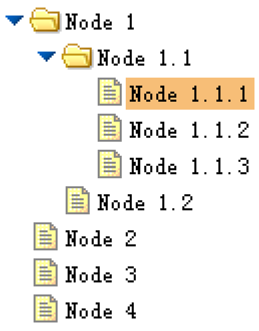
```
$('#tt').tree();
```

显示：



创建异步 Tree

创建异步 tree，每一个 tree 节点必须有 id 属性，这个属性被传递到检索子节点数据。我们这里例子使用 etmvc framework 返回 json 数据。



创建 **HTML** 标记

```
<ul id="tt"></ul>
```

创建 **jQuery** 代码

我们使用 url 属性来指向远程数据

```
$('#tt').tree({
   url:'/demo2/node/getNodes'    // The url will be mapped to
NodeController class and getNodes method
});
```

数据模型

```
@Table(name="nodes")
public class Node extends ActiveRecordBase{
   @Id public Integer id;
   @Column public Integer parentId;
   @Column public String name;

   public boolean hasChildren() throws Exception{
       long count = count(Node.class, "parentId=?", new
Object[]{id});
       return count > 0;
```

```
    }
}
```

写控制代码

如果 node 是子，记住设置 node 状态为 closed。

```
public class NodeController extends ApplicationController{
    /**
     * get nodes, if the 'id' parameter equals 0 then load the 
first level nodes,
     * otherwise load the children nodes
     * @param id the parent node id value
     * @return the tree required node json format
     * @throws Exception
     */
    public View getNodes(int id) throws Exception{
        List<Node> nodes = null;

        if (id == 0){    // return the first level nodes
            nodes = Node.findAll(Node.class, "parentId=0 or 
parentId is null", null);
        } else {    // return the children nodes
            nodes = Node.findAll(Node.class, "parentId=?", new 
Object[]{id});
        }

        List<Map<String,Object>> items = new 
ArrayList<Map<String,Object>>();
        for(Node node: nodes){
            Map<String,Object> item = new 
HashMap<String,Object>();
            item.put("id", node.id);
            item.put("text", node.name);

            // the node has children,
            // set the state to 'closed' so the node can 
asynchronous load children nodes
            if (node.hasChildren()){
                item.put("state", "closed");
            }
            items.add(item);
        }
        return new JsonView(items);
    }
}
```

数据配置实例

```
domain_base_class=com.et.ar.ActiveRecordBase

com.et.ar.ActiveRecordBase.adapter_class=com.et.ar.adapters.MySqlAdapter
com.et.ar.ActiveRecordBase.driver_class=com.mysql.jdbc.Driver
com.et.ar.ActiveRecordBase.url=jdbc:mysql://localhost/mydb
com.et.ar.ActiveRecordBase.username=root
com.et.ar.ActiveRecordBase.password=soft123456
com.et.ar.ActiveRecordBase.pool_size=0
```

部署

- 建立 MySQL 数据库
- 从'/db/item.sql'导入测试表数据，表名是'item'.
- 按需要改变数据库配置，配置文件在/WEB-INF/classes/activerecord.properties 中。
- 运行程序

添加节点

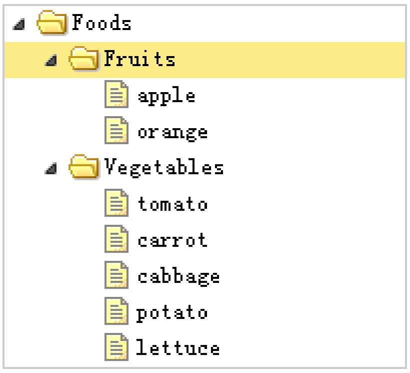
本教程显示了如何添加节点。我们建立 foods tree，这个 tree 包括 vegetable、fruit 节点。然后添加一些 fruits 到存在的 fruit 节点。

创建 foods tree

首先，我们创建 foods tree，代码像这样：

```
<div style="width:200px;height:auto;border:1px solid #ccc;">
    <ul id="tt" class="easyui-tree" url="tree_data.json"></ul>
</div>
```

注意，tree 组件被定义在 UL 标记，tree 节点数据载入 tree_data.json。



得到父节点

我们点击节点以选择 fruit 节点，我们添加一些 fruits 数据。调用 getSelected 方法来得到节点 handle。

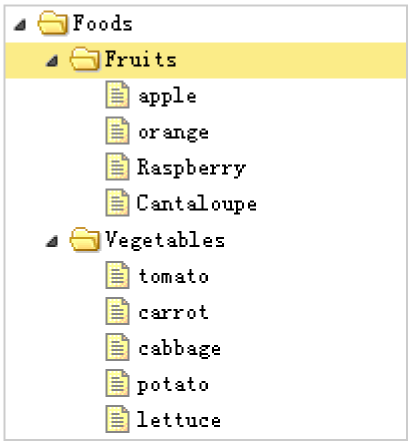
```
var node = $('#tt').tree('getSelected');
```

getSelect 方法的返回值是一个 js 对象，包括 id，text，attributes 和 target 属性。target 属性是 DOM 对象，引用了被选择的节点，使用 append 方法添加节点。

添加节点：

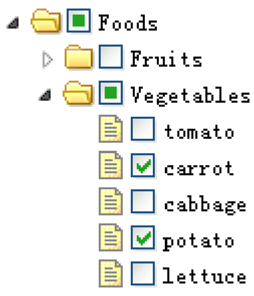
```
var node = $('#tt').tree('getSelected');
if (node){
    var nodes = [{
        "id":13,
        "text":"Raspberry"
    },{
        "id":14,
        "text":"Cantaloupe"
    }];
    $('#tt').tree('append', {
        parent:node.target,
        data:nodes
    });
}
```

当我们添加一些 fruits，可以看到：



创建带有 checkbox 节点的 tree

tree 插件允许你创建 checkbox tree，如果你点击节点的 checkbox，被点击的节点信息得到下和上的继承。例如，点击 tomato 节点的 checkbox，你可以看到 vegetables 节点现在只被选择一部分。



创建 tree 标记

```
<ul id="tt"></ul>
```

创建 checkbox tree

```
using('tree', function(){
    $('#tt').tree({
        url:'tree_data.json',
        checkbox:true
    });
});
```

我们使用 easyloader 以动态的载入 tree 插件。这个特征允许我们载入网页快一点。

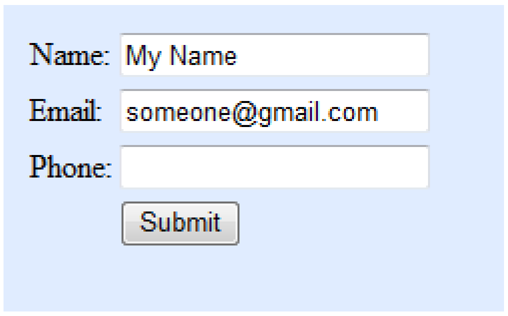
7. 表单
    - Ajax 方式发送表单
    - 添加复合 tree 到表单
    - 表单检验

Ajax 方式发送表单

这个教程显示如何发送表单。我们创建一个例子表单：name，email 和 phone 字段。使用 easyui 表单插件，我们可以将表单变成 ajax 表单。表单发送所有的字段到后台处理服务，服务处理和发送一些数据返回前台网页。我们收到返回的数据后显示他。

创建 form

```
<div style="width:230px;background:#E0ECFF;padding:10px;">
    <form id="ff" action="/demo5/ProcessServlet"
method="post">
        <table>
            <tr>
                <td>Name:</td>
                <td><input name="name"
type="text"></input></td>
            </tr>
            <tr>
                <td>Email:</td>
                <td><input name="email"
type="text"></input></td>
            </tr>
            <tr>
                <td>Phone:</td>
                <td><input name="phone"
type="text"></input></td>
            </tr>
            <tr>
                <td></td>
                <td><input type="submit"
value="Submit"></input></td>
            </tr>
        </table>
    </form>
</div>
```



Name: My Name

Email: someone@gmail.com

Phone:

转换成 Ajax 表单

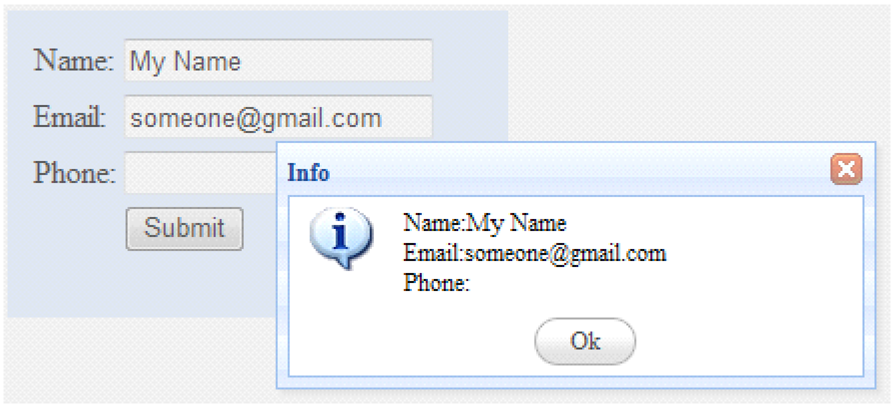
我们写一些 jquery 代码使表单以 ajax 方式发送。注意，当数据返回时，form 插件的 success 函数激发，所以我们可以处理一点事情。

```
$('#ff').form({
    success:function(data){
        $.messager.alert('Info', data, 'info');
    }
});
```

服务处理：

```
protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    // TODO Auto-generated method stub
    String name = request.getParameter("name");
    String email = request.getParameter("email");
    String phone = request.getParameter("phone");
    System.out.println(name+":"+email+":"+phone);
    PrintWriter out = response.getWriter();

out.print("Name:"+name+"<br/>Email:"+email+"<br/>Phone:"+
phone);
    out.flush();
    out.close();
}
```

当我们点击发送按钮时，可以看到；



Name:My Name
Email:someone@gmail.com
Phone:

给表单添加复合 tree 字段

复合 tree 是一种复选框和下拉 tree。它能像表单字段一样传递到服务端。在这个教程中，我们建立注册表单，这个表单有 name，address，city 字段。city 字段是一个复合 tree 字段，用户可以下拉 tree 面板并选择指定 city。

第一步：创建 **HTML** 标记

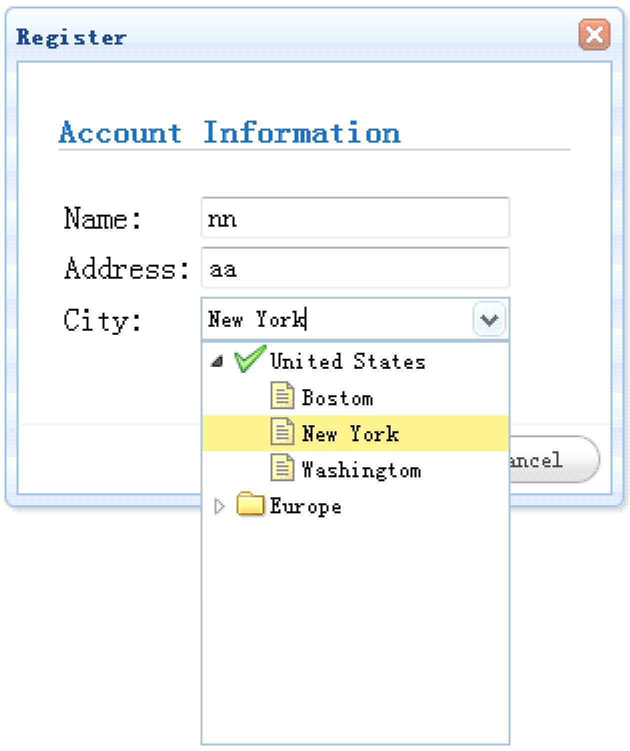
```
<div id="dlg" style="padding:20px;">
    <h2>Account Information</h2>
    <form id="ff" action="/demo6/ProcessServlet"
method="post">
        <table>
            <tr>
                <td>Name:</td>
                <td><input type="text" name="name" /></td>
            </tr>
            <tr>
                <td>Address:</td>
                <td><input type="text" name="address" /></td>
            </tr>
            <tr>
                <td>City:</td>
                <td><select class="easyui-combotree"
url="city_data.json" name="city"
style="width:155px;"/></td>
            </tr>
        </table>
    </form>
</div>
```

我们设置复合 tree 的 url 属性，这个字段可以被从服务器端检索 tree。注意，字段的 class 名应该是 easyui-combotree，所以我们不需要些任何 js 代码，复合 tree 字段就会自动生成。

第二步，创建对话框

我们在对话框中放置表单，这个对话框有发送和取消两个按钮。

```
$('#dlg').dialog({
    title:'Register',
    width:310,
    height:250,
    buttons:[{
        text:'Submit',
        iconCls:'icon-ok',
        handler:function(){
            $('#ff').form('submit',{
                success:function(data){
                    $.messager.alert('Info',data,'info');
                }
            });
        }
    },{
        text:'Cancel',
        iconCls:'icon-cancel',
        handler:function(){
            $('#dlg').dialog('close');
        }
    }]
});
```

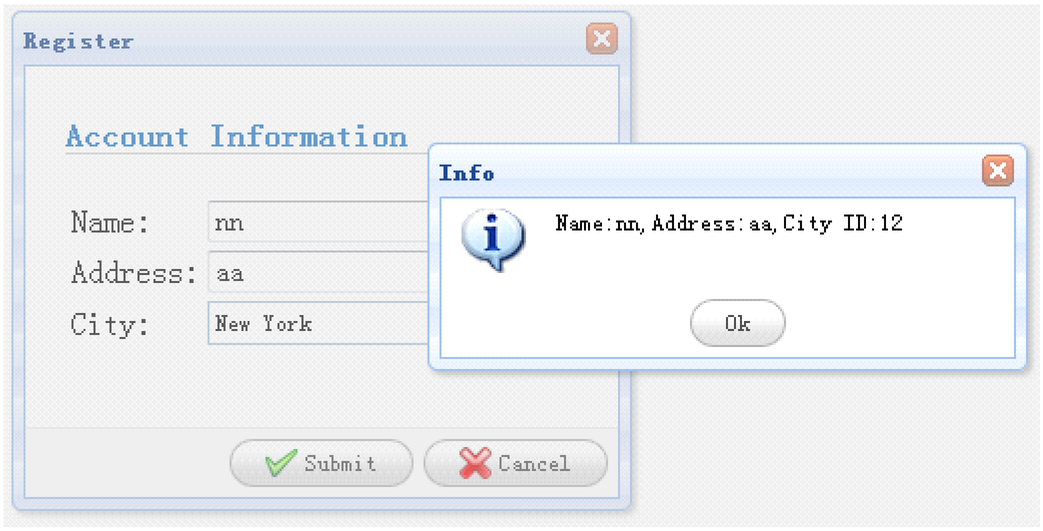


第三部，写服务程序

服务代码接受表单数据并返回：

```
public class ProcessServlet extends HttpServlet {
   protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
       String name = request.getParameter("name");
       String address = request.getParameter("address");
       String city = request.getParameter("city");
       System.out.println(name);
       System.out.println(address);
       System.out.println(city);
       PrintWriter out = response.getWriter();
       out.print("Name:"+name+",Address:"+address+",City
ID:"+city);
       out.flush();
       out.close();
   }
}
```

现在我们点击发送按钮，得到一个信息框，显示一些数据。



Name:nn,Address:aa,City ID:12

复合 tree 是非常简单的。我们做设置 url 属性以检索 tree 数据。

验证表单

本教程将要显示你如何验证表单。easyui 框架提供了 validatebox 插件以验证表单。在这个教程中，我们将要构建联系表单并且应用 validatebox 插件验证表单。你可以修改它适应自己的要求。

构建表单

让我们构建简单的内容的表单： name, email, subject 和 message 字段：

```
<div
style="background:#fafafa;padding:10px;width:300px;height:
300px;">
   <form id="ff" method="post">
      <div>
         <label for="name">Name:</label>
         <input class="easyui-validatebox" type="text"
name="name" required="true"></input>
      </div>
      <div>
         <label for="email">Email:</label>
         <input class="easyui-validatebox" type="text"
name="email" required="true" validType="email"></input>
      </div>
      <div>
         <label for="subject">Subject:</label>
```

```
            <input class="easyui-validatebox" type="text"
name="subject" required="true"></input>
        </div>
        <div>
            <label for="message">Message:</label>
            <textarea name="message"
style="height:60px;"></textarea>
        </div>
        <div>
            <input type="submit" value="Submit">
        </div>
    </form>
</div>
```

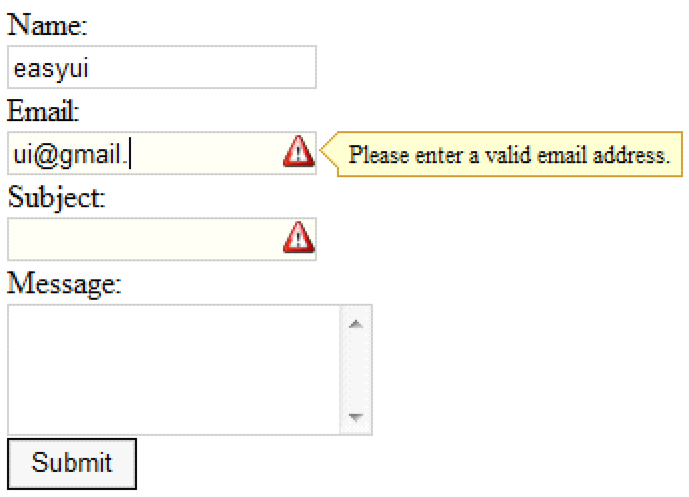
我们添加 class 名为 easyui-validatebox 到 input 标记，所以 input 标记应用验证依照 validType 属性。

当出现错误的时候阻止表单发送

当用户点击发送按钮，我们应该阻止有错误的表单发送。

```
$('#ff').form({
    url:'/demo7/ProcessServlet',
    onSubmit:function(){
        return $(this).form('validate');
    },
    success:function(data){
        alert(data);
    }
});
```

如果表单不可以，出现提示：



easyui

ui@gmail.

Please enter a valid email address.

编写处理代码

最后，我们编写后台处理服务代码，这个代码显示在控制台上的接收参数并发送简单信息到前台页面。

```
public class ProcessServlet extends HttpServlet {
   protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
       String name = request.getParameter("name");
       String email = request.getParameter("email");
       String subject = request.getParameter("subject");
       String message = request.getParameter("message");
       System.out.println("Name:"+name);
       System.out.println("Email:"+email);
       System.out.println("Subject:"+subject);
       System.out.println("Message:"+message);

       PrintWriter out = response.getWriter();
       out.println("ok");
       out.close();
   }
```

Documentation

EasyLoader

Usage

```
easyloader.base = '../';    // set the easyui base directory
easyloader.load('messager', function(){        // load the specified module
    $.messager.alert('Title', 'load ok');
});
```

Options

Properties

Name	Type	Description	Default
modules	object	Predefined modules.	
locales	object	Predefined locales.	
base	string	The easyui base directory, must end with '/'.	The base directory will be auto setted relative to easyload.js
theme	string	The name of theme that defined in 'themes' directory	default
css	boolean	Defines if loading css file when loading module	true
locale	string	The locale name	null
timeout	number	Timeout value in milliseconds. Fires if a timeout occurs.	2000

Defined locales

- af
- bg
- ca
- cs
- da
- de
- en
- fr
- nl
- zh_CN
- zh_TW

Events

Name	Parameters	Description
onProgress	name	Fires when a module is loaded successfully.
onLoad	name	Fires when a module and it's dependencies are loaded successfully.

Methods

Name	Parameter	Description
load	module, callback	Load the specified module. When load success a callback function will be called. The module parameter valid type are: a single module name an module array a css file that end with '.css' a js file that end with '.js'

Draggable

Usage

Markup

```
<div id="dd" style="width:100px;height:100px;border:1px
solid #ccc;">
   <div id="title" style="background:#ccc;">title</div>
</div>
```

jQuery

```
$('#dd').draggable(options);
```

Options

Override defaults with $.fn.draggable.defaults.

Properties

Name	Type	Description	Default
proxy	string,function	A proxy element to be used when dragging, when set to 'clone', a clone element is used as proxy. If a function is specified, it must return a jQuery object.	null
revert	boolean	If set to true, the element will return to its start position when dragging stops.	false
cursor	string	The css cursor when dragging.	move
deltaX	number	The dragged element position x corresponding to current cursor	null
deltaY	number	The dragged element position y corresponding to current cursor	null

handle	selector	The handle that start the draggable.	null
disabled	boolean	True to stop draggable.	false
edge	number	The drag width in which can start draggable.	0
axis	string	Defines the axis which the dragged elements moves on, available value is 'v' or 'h', when set to null will move across 'v' and 'h' direction.	null

Events

Name	Parameters	Description
onStartDrag	e	Fires when the target object start dragging.
onDrag	e	Fires during dragging. Return false will not do dragging actually.
onStopDrag	e	Fires when the dragging stops.

Methods

Name	Parameter	Description
options	none	Return the options property.
proxy	none	Return the drag proxy if the proxy property is setted.
enable	none	Enable the drag action.

disable	none	Disable the drag action.

Droppable

Usage

Markup

```
<div id="dd" style="width:100px;height:100px;border:1px
solid #ccc;"></div>
```

jQuery

```
$('#dd').droppable(options);
```

Options

Override defaults with $.fn.droppable.defaults

Properties

Name	Type	Description	Default
accept	selector	Determine which draggable element will be accepted	null

Events

Name	Parameters	Description

onDragEnter	e,source	Fires when the draggable element is dragged enter. The source parameter indicate the dragged DOM element.
onDragOver	e,source	Fires when the draggable element is dragged over. The source parameter indicate the dragged DOM element.
onDragLeave	e,source	Fires when the draggable element is dragged leave. The source parameter indicate the dragged DOM element.
onDrop	e,source	Fires when the draggable element is dropped. The source parameter indicate the dragged DOM element.

Resizable

Usage

Markup

```
<div id="rr" style="width:100px;height:100px;border:1px
solid #ccc;"></div>
```

jQuery

```
$('#rr').resizable(options);
```

Options

Override defaults with $.fn.resizable.defaults.

Properties

Name	Type	Description	Default
disabled	boolean	True to disable resizing.	false
handles	string	Indicate the direction of resizable,'n' is the north,'e' is the east,etc.	n, e, s, w, ne, se, sw, nw, all
minWidth	number	The minimum width when resizing.	10
minHeight	number	The minimum height when resizing.	10
maxWidth	number	The maximum width when resizing.	10000
maxHeight	number	The maximum height when resizing.	10000
edge	number	The edge of border to be resized.	5

Events

Name	Parameters	Description
onStartResize	e	Fires when start resizing.
onResize	e	Fires during resizing. When return false, the DOM element will not acts actual resize action.
onStopResize	e	Firest when stop resizing.

Panel

Usage

Markup

Many panel properties can be defined in <div/> markup.

```
<div id="p" title="My Panel" collapsible="true"
style="padding:10px;">
  Panel Content
</div>
```

jQuery

To create a panel

```
$('#p').panel(options);
```

To create a panel with custom tools

```
$('#p').panel({
  title: 'My Panel',
  tools: [{
    iconCls:'icon-new',
    handler:function(){alert('new')}
  },{
    iconCls:'icon-save'
    handler:function(){alert('save')}
  }]
});
```

To move panel to other position

```
$('#p').panel('move',{
  left:100,
  top:100
});
```

Dependencies

none

Options

Override defaults with $.fn.panel.defaults.

Properties

Name	Type	Description	Default
title	string	The title text to display in panel header.	null
iconCls	string	A CSS class to display a 16x16 icon in panel.	null
width	number	Set the panel width.	auto
height	number	Set the panel height.	auto
left	number	Set the panel left position.	null
top	number	Set the panel top position.	null
cls	string	Add a CSS class to the panel.	null
headerCls	string	Add a CSS class to the panel header.	null
bodyCls	string	Add a CSS class to the panel body.	null
style	object	Add a custom specification style to the panel.	{}
fit	boolean	When true to set the panel size fit it's parent container.	false
border	boolean	Defines if to show panel border.	true
doSize	boolean	If set to true,the panel will be resize and do layout when created.	true
noheader	boolean	If set to true, the panel header will not be created.	false
content	string	The panel body content.	null
collapsible	boolean	Defines if to show collapsible button.	false

minimizable	boolean	Defines if to show minimizable button.	false
maximizable	boolean	Defines if to show maximizable button.	false
closable	boolean	Defines if to show closable button.	false
tools	array	Custom tools, every tool can contain two properties: iconCls and handler	[]
collapsed	boolean	Defines if the panel is collapsed at initialization.	false
minimized	boolean	Defines if the panel is minimized at initialization.	false
maximized	boolean	Defines if the panel is maximized at initialization.	false
closed	boolean	Defines if the panel is closed at initialization.	false
href	string	A URL to load remote data and then display in the panel.	null
cache	boolean	True to cache the panel content that loaded from href.	true
loadingMessage	string	When loading remote data show a message in the panel.	Loading…

Events

Name	Parameters	Description
onLoad	none	Fires when remote data is loaded.

onBeforeOpen	none	Fires before panel is opened, return false to stop the open.
onOpen	none	Fires after panel is opened.
onBeforeClose	none	Fires before panel is closed, return false to cancel the close.
onClose	none	Fires after panel is closed.
onBeforeDestroy	none	Fires before panel is destroyed, return false to cancel the destroy.
onDestroy	none	Fires after panel is destroyed.
onBeforeCollapse	none	Fires before panel is collapsed, return false to stop the collapse.
onCollapse	none	Fires after panel is collpased.
onBeforeExpand	none	Fires before panel is expanded, return false to stop the expand.
onExpand	none	Fires after panel is expanded.
onResize	width, height	Fires after panel is resized. width: the new outer width height: the new outer height
onMove	left,top	Fires after panel is moved. left: the new left postion top: the new top position
onMaximize	none	Fires after the window has been maximized.
onRestore	none	Fires after the window has been restored to its original size.
onMinimize	none	Fires after the window has been minimized.

Methods

Name	Parameter	Description
options	none	Return options property.
panel	none	Return the panel object.
header	none	Return the panel header object.
body	none	Return the panel body object.
setTitle	title	Set the title text of header.
open	forceOpen	When forceOpen parameter set to true, the panel is opened bypass the onBeforeOpen callback.
close	forceClose	When forceClose parameter set to true, the panel is closed bypass the onBeforeClose callback.
destroy	forceDestroy	When forceDestroy parameter set to true, the panel is destroyed bypass the onBeforeDestroy callback.
refresh	none	Refresh the panel to load remote data when href property is setted.
resize	options	Set panel size and do layout. The options object contains following properties: width: the new panel width height: the new panel height left: the new panel left position top: the new panel top position
move	options	Move the panel to a new position. The options object contains following properties: left: the new panel left position

		top: the new panel top position
maximize	none	Fits the panel winthin its container.
minimize	none	Minimizing the panel.
restore	none	Restores the maximized panel back to its original size and position.
collapse	forceCollapse	Collapses the panel body. When forceCollapse is setted to true, the onBeforeCollapse event will not be triggered.
expand	forceExpand	Expand the panel body. When forceExpand is setted to true, the onBeforeExpand event will not be triggered.

tabs

Usage

Markup

```
<div id="tt" style="width:500px;height:250px;">
    <div title="Tab1" style="padding:20px;display:none;">
        tab1
    </div>
    <div title="Tab2" closable="true"
style="overflow:auto;padding:20px;display:none;">
        tab2
    </div>
    <div title="Tab3" icon="icon-reload" closable="true"
style="padding:20px;display:none;">
        tab3
    </div>
</div>
```

jQuery

To create a tabs container

```
$('#tt').tabs(options);
```

To add a tab panel:

```
$('#tt').tabs('add',{
    title:'New Tab',
    content:'Tab Body',
    closable:true
});
```

To get the selected tab panel and its tab object:

```
var pp = $('#tt').tabs('getSelected');
var tab = pp.panel('options').tab;    // the corresponding
tab object
```

Dependencies

- panel

Options

Tabs Container

Override defaults with $.fn.tabs.defaults.

Properties

Name	Type	Description	Default
width	number	The width of tabs container.	auto
height	number	The height of tabs container.	auto
plain	boolean	True to render the tab strip without a background container image.	false
fit	boolean	True to set the size of tabs container to fit	false

		it's parent container.	
border	boolean	True to show tabs container border.	true
scrollIncrement	number	The number of pixels to scroll each time a tab scroll button is pressed.	100
scrollDuration	number	The number of milliseconds that each scroll animation should last.	400

Events

Name	Parameters	Description
onLoad	panel	Fires when an ajax tab panel finish loading remote data.
onSelect	title	Fires when user select a tab panel.
onBeforeClose	title	Fires before the tab panel is closed, return false to cancel this close action.
onClose	title	Fires when user close a tab panel.
onAdd	title	Fires when a new tab panel is added.
onUpdate	title	Fires when a tab panel is updated.

Methods

Name	Parameter	Description
options	none	Return the tabs options.
tabs	none	Return all tab panels.

resize	none	Resize the tabs container and do layout.
add	options	Add a new tab panel, the options parameter is a config object, see tab panel properties for more details.
close	title	Close a tab panel, title parameter indicate which panel to be closed.
getTab	title	Get the specified tab panel.
getSelected	none	Get the selected tab panel.
select	title	Select a tab panel.
exists	title	Indicate if the special panel is exists.
update	param	Update the specified tab panel, the param parameter contains two properties: tab: the tab panel to be updated. options: the panel options.

Tab Panel

Properties

Name	Type	Description	Default
title	string	The tab panel title text.	
content	string	The tab panel content.	
href	string	A URL to load remote content to fill the tab panel.	null
cache	boolean	True to cache the tab panel, valid when href property is setted.	true

iconCls	string	An icon CSS class to show on tab panel title.	null
closable	boolean	When set to true, the tab panel will show a closable button which can click to close the tab panel.	false
selected	boolean	When set to true, tab tab panel will be selected.	false
width	number	The width of tab panel.	auto
height	number	The height of tab panel.	auto

accordion

Usage

Markup

```
<div id="aa" style="width:300px;height:200px;">
    <div title="Title1" icon="icon-save"
style="overflow:auto;padding:10px;">
        <h3 style="color:#0099FF;">Accordion for jQuery</h3>
        <p>Accordion is a part of easyui framework for jQuery.
It lets you define your accordion component on web page more
easily.</p>
    </div>
    <div title="Title2" icon="icon-reload" selected="true"
style="padding:10px;">
        content2
    </div>
    <div title="Title3">
        content3
    </div>
</div>
```

jQuery

```
$('#aa').accordion(options);
```

Dependencies

- panel

Options

Container Options

Name	Type	Description	Default
width	number	The width of accordion container.	auto
height	number	The height of accordion container.	auto
fit	boolean	Set to true to set the accordion container size fit it's parent container.	false
border	boolean	Defines if to show the border.	true
animate	boolean	Defines if to show animation effect when expand or collapse panel.	true

Panel Options

The accordion panel options is inhirited from panel, many properties is defined in <div/> markup.
Bellow is the addition properties:

Name	Type	Description	Default
selected	boolean	Set to true to expand the panel.	false

Events

Name	Parameters	Description
onSelect	title	Fires when a panel is selected.
onAdd	title	Fires when a new panel is added.
onBeforeRemove	title	Fires before a panel is removed, return false to cancel the remove action.
onRemove	title	Fires when a panel is removed.

Methods

Name	Parameter	Description
options	none	Return the options of accordion.
panels	none	Get all panels.
resize	none	Resize the accordion.
getSelected	none	Get the selected panel.
getPanel	title	Get the specified panel.
select	title	Select the specified panel.
add	options	Add a new panel.
remove	title	Remove the specified panel.

layout

Usage

Markup

The layout panel must has one 'center' panel.

```
<div id="cc" style="width:600px;height:400px;">
   <div region="north" title="North Title" split="true"
style="height:100px;"></div>
   <div region="south" title="South Title" split="true"
style="height:100px;"></div>
   <div region="east" icon="icon-reload" title="East"
split="true" style="width:100px;"></div>
   <div region="west" split="true" title="West"
style="width:100px;"></div>
   <div region="center" title="center title"
style="padding:5px;background:#eee;"></div>
</div>
```

jQuery

```
$('#cc').layout(options);
```

Dependencies

- panel
- resizable

Options

Layout Panel Options

All the properties is defined on <div/> markup, which the layout panel is created from it.

Name	Type	Description	Default
title	string	The layout panel title text.	null
region	string	Defines the layout panel position, the value is one of following: north, south, east, west, center.	

border	boolean	True to show layout panel border.	true
split	boolean	True to show a split bar which user can change the panel size.	false
icon	string	An icon CSS class to show a icon on panel header.	null
href	string	An URL to load data from remote site.	null

Methods

Name	Parameter	Description
resize	none	Set the layout size.
panel	region	Return the specified panel, the 'region' parameter possible values:'north','south','east','west','center'.
collapse	region	Collapse the specified panel, the 'region' parameter possible values:'north','south','east','west'.
expand	region	Expand the specified panel, the 'region' parameter possible values:'north','south','east','west'.

menu

Usage

Markup

```
<div id="mm" style="width:120px;">
        <div>New</div>
        <div>
                <span>Open</span>
                <div style="width:150px;">
```

```
                    <div><b>Word</b></div>
                    <div>Excel</div>
                    <div>PowerPoint</div>
             </div>
      </div>
      <div icon="icon-save">Save</div>
      <div class="menu-sep"></div>
      <div>Exit</div>
</div>
```

jQuery

To create a menu:

```
$('#mm').menu(options);
```

To show a menu on special position:

```
$('#mm').menu('show', {
  left: 200,
  top: 100
});
```

Dependencies

none

Options

Override defaults with $.fn.menu.defaults.

Properties

Name	Type	Description	Default
zIndex	number	Menu z-index style,increase from it.	110000
left	number	Menu left position.	0

top	number	Menu top position.	0
href	string	Indicate a different page URL that can be displayed in the current browser window when clicked menu item.	null

Events

Name	Parameters	Description
onShow	none	Fires after menu is showed.
onHide	none	Fires after menu is hidden.

Methods

Name	Parameter	Description
show	pos	Show a menu on special position. pos parameter have two properties: left: the new left position. top: the new top position.
hide	none	Hide a menu.

linkbutton

Usage

Markup

```
<a href="#" id="btn" icon="icon-search">easyui</a>
```

jQuery

```
$('#btn').linkbutton(options);
```

Dependencies

none

Options

Override defaults with $.fn.linkbutton.defaults.

Properties

Name	Type	Description	Default
id	string	The id attribute of this component.	null
disabled	boolean	True to disable the button	false
plain	boolean	True to show a plain effect.	false
text	string	The button text.	''
iconCls	string	A CSS class to display a 16x16 icon on left.	null

Methods

Name	Parameter	Description
options	none	Return options property.

disable	none	Disable the button
enable	none	Enable the button

MenuButton

Usage

Markup

```
<a href="javascript:void(0)" id="mb"
icon="icon-edit">Edit</a>

<div id="mm" style="width:150px;">
    <div icon="icon-undo">Undo</div>
    <div icon="icon-redo">Redo</div>
    <div class="menu-sep"></div>
    <div>Cut</div>
    <div>Copy</div>
    <div>Paste</div>
    <div class="menu-sep"></div>
    <div icon="icon-remove">Delete</div>
    <div>Select All</div>
</div>
```

jQuery

```
$('#mb').menubutton({
    menu: '#mm'
});
```

Dependencies

- menu
- linkbutton

Options

Override default with $.fn.menubutton.defaults.

Name	Type	Description	Default
disabled	boolean	True to disable the button.	false
plain	boolean	True to show plain effect.	false
menu	string	A selector to create a corresponding menu.	null
duration	number	Defines duration time in milliseconds to show menu when hovering the button.	100

SplitButton

Usage

Markup

```
<a href="javascript:void(0)" id="sb" icon="icon-ok"
onclick="javascript:alert('ok')">Ok</a>

<div id="mm" style="width:100px;">
    <div icon="icon-ok">Ok</div>
    <div icon="icon-cancel">Cancel</div>
</div>
```

jQuery

```
$('#sb').splitbutton({
    menu:'#mm'
});
```

Dependencies

- menu
- linkbutton

Options

Override default with $.fn.splitbutton.defaults.

Name	Type	Description	Default
disabled	boolean	True to disable the button.	false
plain	boolean	True to show plain effect.	false
menu	string	A selector to create a corresponding menu.	null
duration	number	Defines duration time in milliseconds to show menu when hovering the button.	100

form

Usage

Markup

```
<form id="ff" method="post">
...
</form>
```

jQuery

To make the form become ajax submit form

```
$('#ff').form({
	url:...,
	onSubmit: function(){
		// do some check
		// return false to prevent submit;
	},
	success:function(data){
		alert(data)
	}
});
```

To do a submit action

```
$('#ff').form('submit', {
	url:...,
	onSubmit: function(){
		// do some check
		// return false to prevent submit;
	},
	success:function(data){
		alert(data)
	}
});
```

Dependencies

none

Options

Properties

Name	Type	Description	Default
url	string	The form action URL to submit	null

Events

Name	Parameters	Description
onSubmit	none	Fires before submit, return false to prevent submit action.
success	data	Fires when the form is submitted successfuly.

onBeforeLoad	param	Fires before a request is made to load data. Return false to cancel this action.
onLoadSuccess	data	Fires when the form data is loaded.
onLoadError	none	Fires when some errors occur while loading form data.

Methods

Name	Parameter	Description
submit	options	Do the submit action, the options parameter is a object which contains following properties: url: the action URL onSubmit: callback function before submit submit: callback function after submit successfuly
load	data	Load records to fill the form. The data parameter can be a string or a object type, when string acts as a remote URL, otherwise acts as a local record
clear	none	Clear the form data
validate	none	Do the form fields validation, return true when all fields is valid. The method is used with the validatebox plugin.

combobox

Usage

Markup

```
<select id="cc" name="dept" style="width:200px;">
    <option value="aa">aitem1</option>
    <option>bitem2</option>
    <option>bitem3</option>
    <option>ditem4</option>
    <option>eitem5</option>
</select>
```

jQuery

```
$('#cc').combobox(options);
```

To create from remote data:

```
$('#cc').combobox({
    url:'combobox_data.json',
    valueField:'id',
    textField:'text'
});
```

The remote data format sample:

```
[{
    "id":1,
    "text":"text1"
},{
    "id":2,
    "text":"text2"
},{
    "id":3,
    "text":"text3",
    "selected":true
},{
    "id":4,
    "text":"text4"
},{
    "id":5,
    "text":"text5"
}]
```

Dependencies

- validatebox

Options

Override defaults with $.fn.combobox.defaults.

Properties

Name	Type	Description	Default
width	number	The width of the component.	auto
listWidth	number	The width of the drop down list.	null
listHeight	number	The height of the drop down list.	null
valueField	string	The underlying data value name to bind to this ComboBox.	value
textField	string	The underlying data field name to bind to this ComboBox.	text
editable	boolean	Defines if user can type text directly into the field.	true
disabled	boolean	Defines if to disable the field.	false
url	string	A URL to load list data from remote.	null
data	array	The list data to be loaded.	null
required	boolean	Defines if the field should be inputed.	false
missingMessage	string	Tooltip text that appears when the text box is empty.	This field is required.

Events

Name	Parameters	Description
onLoadSuccess	none	Fires when remote data is loaded successfully.
onLoadError	none	Fires when remote data load error.
onSelect	record	Fires when user select a list item.
onChange	newValue, oldValue	Fires when the field value is changed.

Methods

Name	Parameter	Description
destroy	none	Destroy the component.
resize	width	Resize the component width.
select	value	Select an item in the dropdown list.
clear	none	Clear the component value.
setValue	param	Set the specified value into the field. The 'param' parameter can be a value string or a javascript object that contains two properties corresponding to valueField and textField property.
getValue	none	Get the field value.
getText	none	Get the field text.

loadData	data	Load the locale list data.
reload	url	Request the remote list data.
disable	none	Disable the field.
enable	none	Enable the field.

combotree

Usage

Markup

```
<select id="cc" style="width:200px;"></select>
```

jQuery

```
$('#cc').combotree({
   url:'tree_data.json'
});
```

Dependencies

- tree
- validatebox

Options

Override defaults with $.fn.combotree.defaults.

Properties

Name	**Type**	**Description**	**Default**

width	number	The width of the component.	auto
treeWidth	number	The width of the tree list.	null
treeHeight	number	The height of the tree list.	200
url	string	A URL to load remote tree data.	null
data	array	The data to be loaded.	null
disabled	boolean	Defines if to disable the field.	false
required	boolean	Defines if the field should be inputed.	false
missingMessage	string	Tooltip text that appears when the text box is empty.	This field is required.

Events

Name	Parameters	Description
onBeforeSelect	node	Fires before a tree node is selected, return false to cancel the selection.
onSelect	node	Fires when user select a tree node.
onChange	newValue, oldValue	Fires when the field value is changed.

Methods

Name	Parameter	Description

destroy	none	Destroy the component.
resize	width	Resize the component width.
tree	none	Get the tree.
clear	none	Clear the component value.
setValue	param	Set the specified value into the field. The 'param' parameter can be a tree node id value or a javascript object that contains two properties: id and text.
getValue	none	Get the field value.
getText	none	Get the field text.
loadData	data	Load the locale tree data.
reload	url	Request the remote tree data again.
disable	none	Disable the field.
enable	none	Enable the field.

NumberBox

Usage

Markup

```
<input type="text" id="nn"></input>
```

jQuery

```
$('#nn').numberbox(options);
```

Dependencies

- validatebox

Options

Override defaults with $.fn.numberbox.defaults.

Properties

Name	Type	Description	Default
disabled	boolean	Defines if to disable the field.	false
min	number	The minimum allowed value.	null
max	number	The maximum allowed value.	null
precision	number	The maximum precision to display after the decimal separator.	0

Methods

validatebox

Usage

Markup

```
<input id="vv" required="true" validType="email">
```

jQuery

```
$('#vv').validatebox(options)
```

Validate Rule

The validate rule is defined by using required and validType property, here are the rules already implemented:

- email: Match email regex rule.
- url: Match URL regex rule.
- length[0,100]: Between x and x characters allowed.

To custom validate rule, override $.fn.validatebox.defaults.rules that defines a validator function and invalid message. For example, to define a minLength valid type:

```
$.extend($.fn.validatebox.defaults.rules, {
    minLength: {
        validator: function(value, param){
            return value.length >= param[0];
        },
        message: 'Please enter at least {0} characters.'
    }
});
```

Now you can use the minLength validtype:

```
<input class="easyui-validatebox"
validType="minLength[5]">
```

In the above code, we define a input box that should be inputed at least 5 characters.

Dependencies

none

Options

Override defaults with $.fn.validatebox.defaults

Properties

Name	Type	Description	Default

required	boolean	Defines if the field should be inputed.	false
validType	string	Defines the field valid type, such as email, url, etc.	null
missingMessage	string	Tooltip text that appears when the text box is empty.	This field is required.
invalidMessage	string	Tooltip text that appears when the content of text box is invalid.	null

Methods

Name	Parameter	Description
destroy	none	Remove and destroy the component.
validate	none	Do the validation to determine whether the content of text box is valid.
isValid	none	call validate method and return the validation result, true or false.

datebox

Usage

Markup

```
<input id="dd" type="text"></input>
```

jQuery

```
$('#dd').datebox(options);
```

Dependencies

- calendar
- validatebox

Options

Override defaults with $.fn.datebox.defaults

Properties

Name	Type	Description	Default
currentText	string	The text to display for the current day button.	Today
closeText	string	The text to display for the close button.	Close
disabled	boolean	When true to disable the field.	false
required	boolean	Defines if the field should be inputed.	false
missingMessage	string	Tooltip text that appears when the text box is empty.	This field is required.
formatter	function	A function to format the date, the function take a 'date' parameter and return a string value.	
parser	function	A function to parse a date string, the function take a 'date' string and return a date value.	

Events

Name	Parameters	Description
onSelect	date	Fires when user select a date.

Methods

Name	Parameter	Description
destroy	none	Destroy the component.
disable	none	Disable the field.
enable	none	Enable the field.

calendar

Usage

Markup

```
<div id="cc" style="width:180px;height:180px;"></div>
```

jQuery

```
$('#cc').calendar(options);
```

Options

Override defaults with $.fn.calendar.defaults

Properties

Name	Type	Description	Default
width	number	The width of calendar component.	180
height	number	The height of calendar component.	180
fit	boolean	When true to set the calendar size fit it's parent container.	false
border	boolean	Defines if to show the border.	true
weeks	array	The list of week to be showed.	['S','M','T','W','T','F','S']
months	array	The list of month to be showed.	['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
year	number	The year of calendar.	current year(four digits)
month	number	The month of calendar.	current month, start with 1
current	Date	The current date.	current date

Events

Name	Parameters	Description
onSelect	date	Fires when user select a date.

window

Usage

Markup

Many window properties can be defined in markup, such as icon, title, etc.

```
<div id="win" icon="icon-save" title="My Window">
  Window Content
</div>
```

jQuery

To create a window:

```
$('#win').window(options);
```

To open a window:

```
$('#win').window('open');
```

Dependencies

- draggable
- resizable
- panel

Options

Override defaults with $.fn.window.defaults.

Properties

Many window properties can inhirit from panel, below is the window private properties.

Name	Type	Description	Default
zIndex	number	Window z-index,increase from it.	9000

draggable	boolean	Defines if window can be dragged.	true
resizable	boolean	Defines if window can be resized.	true
shadow	boolean	If set to true,when window show the shadow will show also.	true
modal	boolean	Defines if window is a modal window.	true

Window override some panel properties.

Name	**Type**	**Description**	**Default**
title	string	The window title text.	New Window
collapsible	boolean	Defines if to show collapsible button.	true
minimizable	boolean	Defines if to show minimizable button.	true
maximizable	boolean	Defines if to show maximizable button.	true
closable	boolean	Defines if to show closable button.	true

Events

Window events is same as panel events, see panel events for more information.

Methods

Window methods is same as panel methods, except the 'header' and 'body' method.

dialog

Usage

Markup

```
<div id="dd" title="My Dialog"
style="width:400px;height:200px;">
   Dialog Content.
</div>
```

jQuery

```
$('#dd').dialog(options);
```

Dependencies

- draggable
- resizable
- panel
- window
- linkbutton

Options

Override defaults with $.fn.dialog.defaults.

Properties

Many properties can inhirit from window, below is the dialog private properties:

Name	Type	Description	Default
title	string	The dialog title text.	New Dialog
collapsible	boolean	Defines if to show collapsible button.	false
minimizable	boolean	Defines if to show minimizable button.	false
maximizable	boolean	Defines if to show maximizable button.	false

resizable	boolean	Defined if the dialog can be resized.	false
toolbar	array	The top toolbar of dialog, each tool contains: text, iconCls, disabled, handler etc.	null
buttons	array	The bottom buttons of dialog, each button contains: text, iconCls, handler etc.	null

Events

Dialog events is same as window events, see window events for more information.

Methods

Dialog methods is same as window methods, see window methods for more information.

Messager

Dependencies

- draggable
- resizable
- panel
- window
- linkbutton

Options

Override defaults with $.messager.defaults.

Name	**Type**	**Description**	**Default**
ok	string	The Ok button text.	Ok
cancel	string	The Cancel button text.	Cancel

Methods

Name	Parameters	Description
$.messager.show	options	Show a message window on right bottom of screen. The options parameter is a configuration object: showType: Defines how the message window to be showed. Available values are: null,slide,fade,show. Defaults to slide. showSpeed: Defines the time in milliseconds message window finishs show. Defaults to 600. width: Defines the width of message window. Defaults to 250. height: Defines the height of message window. Defaults to 100. msg: The message text to be showed. title: The title text to be showed on header panel. timeout: If defines to 0, the message window will not close unless user close it. Defines to unzero, the message window will be auto closed when timeout.
$.messager.alert	title, msg, icon, fn	Show an alert window. Parameters: title: The title text to be showed on header panel. msg: The message text to be showed. icon: The icon image to be showed. Available value are: error,question,info,warning. fn: The callback function triggered when window closed.
$.messager.confirm	title, msg, fn	Show a confirmation message window with Ok and Cancel buttons. Parameters: title: The title text to be showed on header panel.

		msg: The message text to be showed. fn(b): The callback function, when user click Ok button, pass a true value to function, otherwise pass a false to it.
$.messager.prompt	title, msg, fn	Show a message window with Ok and Cancel buttons prompting user to enter some text. Parameters: title: The title text to be showed on header panel. msg: The message text to be showed. fn(val): The callback function with a value parameter user entered.

Pagination

Usage

Markup

```
<div id="pp" style="background:#efefef;border:1px solid
#ccc;"></div>
```

jQuery

```
$('#pp').pagination(options);
```

Dependencies

- linkbutton

Options

Override defaults with $.fn.pagination.defaults.

Properties

Name	Type	Description	Default
total	number	The total records, which should be setted when pagination is created.	1
pageSize	number	The page size.	10
pageNumber	number	Show the page number when pagination is created.	1
pageList	array	User can change the page size. The pageList property defines how many size can be changed.	[10,20,30,50]
loading	boolean	Defines if data is loading.	false
buttons	array	Defines custom buttons, each button contains two properties: iconCls: the CSS class which will show a background image handler: a handler function when button is clicked	null
showPageList	boolean	Defines if to show page list.	true
showRefresh	boolean	Defines if to show refresh button.	true
beforePageText	string	Show a label before the input component.	Page
afterPageText	string	Show a label after the input component.	of {pages}
displayMsg	string	Display a page information.	Displaying {from} to {to} of {total}

			items

Events

Name	Parameters	Description
onSelectPage	pageNumber, pageSize	Fires when user select a new page. callback function contains two parameter: pageNumber: the new page number pageSize: the new page size
onBeforeRefresh	pageNumber, pageSize	Fires before the refresh button is clicked, return false to cancel the refresh action.
onRefresh	pageNumber, pageSize	Fires after refresh.
onChangePageSize	pageSize	Fires when user change the page size.

datagrid

Usage

Markup

```
<table id="tt"></table>
```

jQuery

```
$('#tt').datagrid(options);
```

The DataGrid data format sample

```
{"total":28,"rows":[{"productid":"FI-SW-01","unitcost":10.
00,"status":"P","listprice":16.50,"attr1":"Large","itemid
":"EST-1"},
{"productid":"K9-DL-01","unitcost":12.00,"status":"P","li
stprice":18.50,"attr1":"Spotted Adult
Female","itemid":"EST-10"},
{"productid":"RP-SN-01","unitcost":12.00,"status":"P","li
stprice":18.50,"attr1":"Venomless","itemid":"EST-11"},
{"productid":"RP-SN-01","unitcost":12.00,"status":"P","li
stprice":18.50,"attr1":"Rattleless","itemid":"EST-12"},
{"productid":"RP-LI-02","unitcost":12.00,"status":"P","li
stprice":18.50,"attr1":"Green Adult","itemid":"EST-13"},
{"productid":"FL-DSH-01","unitcost":12.00,"status":"P","l
istprice":58.50,"attr1":"Tailless","itemid":"EST-14"},
{"productid":"FL-DSH-01","unitcost":12.00,"status":"P","l
istprice":23.50,"attr1":"With tail","itemid":"EST-15"},
{"productid":"FL-DLH-02","unitcost":12.00,"status":"P","l
istprice":93.50,"attr1":"Adult Female","itemid":"EST-16"},
{"productid":"FL-DLH-02","unitcost":12.00,"status":"P","l
istprice":93.50,"attr1":"Adult Male","itemid":"EST-17"},
{"productid":"AV-CB-01","unitcost":92.00,"status":"P","li
stprice":193.50,"attr1":"Adult Male","itemid":"EST-18"}]}
```

Dependencies

- panel
- resizable
- linkbutton
- pagination

Options

DataGrid Properties

Override default with $.fn.datagrid.defaults.

Name	Type	Description	Default
title	string	The datagrid panel title text.	null

iconCls	string	A CSS class that will provide a background image to be used as the header icon.	null
border	boolean	True to show datagrid panel border.	true
width	number	The width of datagrid width.	auto
height	number	The height of datagrid height.	auto
columns	array	The datagrid columns config object, see column properties for more details.	null
frozenColumns	array	Same as the columns property, but the these columns will be frozen on left.	null
striped	boolean	True to stripe the rows.	false
method	string	The method type to request remote data.	post
nowrap	boolean	True to display data in one line.	true
idField	string	Indicate which field is an identity field.	null
url	string	A URL to request data from remote site.	null
loadMsg	string	When loading data from remote site, show a prompt message.	Processing, please wait …
pagination	boolean	True to show a pagination toolbar on datagrid bottom.	false

rownumbers	boolean	True to show a row number column.	false
singleSelect	boolean	True to allow selecting only one row.	false
fit	boolean	True to set size to fit it's parent container.	false
pageNumber	number	When set pagination property, initialize the page number.	1
pageSize	number	When set pagination property, initialize the page size.	10
pageList	array	When set pagination property, initialize the page size selecting list.	[10,20,30,40,50]
queryParams	object	When request remote data, sending additional parameters also.	{}
sortName	string	Defines which column can be sorted.	null
sortOrder	string	Defines the column sort order, can only be 'asc' or 'desc'.	asc
remoteSort	boolean	Defines if to sort data from server.	true
editors	object	Defines the editor when editing a row.	predefined editors

Column Properties

The DataGrid Columns is an array object, which element is an array too.
The element of element array is a config object, which defines every column field.
code example:

```
columns:[[
   {field:'itemid',title:'Item
ID',rowspan:2,width:80,sortable:true},
   {field:'productid',title:'Product
ID',rowspan:2,width:80,sortable:true},
   {title:'Item Details',colspan:4}
],[
   {field:'listprice',title:'List
Price',width:80,align:'right',sortable:true},
   {field:'unitcost',title:'Unit
Cost',width:80,align:'right',sortable:true},
   {field:'attr1',title:'Attribute',width:100},
   {field:'status',title:'Status',width:60}
]]
```

Name	Type	Description	Default
title	string	The column title text.	undefined
field	string	The column field name.	undefined
width	number	The width of column.	undefined
rowspan	number	Indicate how many rows a cell should take up.	undefined
colspan	number	Indicate how many columns a cell should take up.	undefined
align	string	Indicate how to align the column data. 'left','right','center' can be used.	undefined
sortable	boolean	True to allow the column can be sorted.	undefined

checkbox	boolean	True to show a checkbox.	undefined
formatter	function	The cell formatter function. take three parameter: value: the field value. rowData: the row record data. rowIndex: the row index.	undefined
sorter	function	The custom field sort function, take three parameter: a: the first field value. b: the second field value. order: the sort order, 'asc' or 'desc'.	undefined
editor	string,object	Indicate the edit type. When string indicates the edit type, when object contains two properties: type: string, the edit type, possible type is: text,textarea,checkbox,numberbox,validatebox,datebox,combobox,combotree. options: object, the editor options corresponding to the edit type.	undefined

Editor

Override default with $.fn.datagrid.defaults.editors.
Every editor has following functions:

Name	Parameters	Description
init	container, options	Initialize and create the editor.
destroy	elem	Destroy the editor if necessary.
getValue	elem	Get value from editor text.
setValue	elem, value	Set value for editor.

resize	elem, width	Resize the editor if necessary.

For example, the text editor is defined as following:

```
$.extend($.fn.datagrid.defaults.editors, {
    text: {
        init: function(container, options){
            var input = $('<input type="text"
class="datagrid-editable-input">').appendTo(container);
            return input;
        },
        getValue: function(elem){
            return $(elem).val();
        },
        setValue: function(elem, value){
            $(elem).val(value);
        },
        resize: function(elem, width){
            var input = $(elem);
            if ($.boxModel == true){
                input.width(width - (input.outerWidth() -
input.width()));
            } else {
                input.width(width);
            }
        }
    }
});
```

Events

Name	Parameters	Description
onLoadSuccess	data	Fires when data is loaded successfully.
onLoadError	none	Fires when some error occur to load remote data.
onBeforeLoad	param	Fires before a request is made to load data.

		If return false the load action will be canceled.
onClickRow	rowIndex, rowData	Fires when user click a row, the parameters contains: rowIndex: the clicked row index, start with 0 rowData: the record corresponding to the clicked row
onDblClickRow	rowIndex, rowData	Fires when user dblclick a row, the parameters contains: rowIndex: the clicked row index, start with 0 rowData: the record corresponding to the clicked row
onSortColumn	sort, order	Fires when user sort a column, the parameters contains: sort: the sort column field name order: the sort column order
onSelect	rowIndex, rowData	Fires when user select a row, the parameters contains: rowIndex: the selected row index, start with 0 rowData: the record corresponding to the selected row
onUnselect	rowIndex, rowData	Fires when user unselect a row, the parameters contains: rowIndex: the unselected row index, start with 0 rowData: the record corresponding to the unselected row
onBeforeEdit	rowIndex, rowData	Fires when user start editing a row, the parameters contains: rowIndex: the editing row index, start with 0 rowData: the record corresponding to the

		editing row
onAfterEdit	rowIndex, rowData, changes	Fires when user finish editing, the parameters contains: rowIndex: the editing row index, start with 0 rowData: the record corresponding to the editing row changes: the changed field/value pairs
onCancelEdit	rowIndex, rowData	Fires when user cancel editing a row, the parameters contains: rowIndex: the editing row index, start with 0 rowData: the record corresponding to the editing row

Methods

Name	Parameter	Description
options	none	Return the options object.
getPager	none	Return the pager object.
getPanel	none	Return the panel object.
resize	param	Do resize and do layout.
reload	param	Reload the rows. If the 'param' is specified, it will replace with the queryParams property.
fixColumnSize	none	fix columns size.
loadData	data	Load local data, the old rows will be removed.
getData	none	Return the loaded data.

getRows	none	Return the current page rows.
getRowIndex	row	Return the specified row index, the row parameter can be a row record or an id field value.
getSelected	none	Return the first selected row record or null.
getSelections	none	Return all selected rows, when no record selected, am empty array will return.
clearSelections	none	Clear all selections.
selectAll	none	Select all current page rows.
selectRow	index	Select a row, the row index start with 0.
selectRecord	idValue	Select a row by passing id value parameter.
unselectRow	index	Unselect a row.
beginEdit	index	Begin editing a row.
endEdit	index	End editing a row.
cancelEdit	index	Cancel editing a row.
refreshRow	index	Refresh a row.
validateRow	index	validate the specified row, return true when valid.
appendRow	row	Append a new row.
deleteRow	index	Delete a row.
getChanges	type	Get changed rows since the last commit. The type parameter indicate which type changed rows, possible value is: inserted,deleted,updated,etc. When the type parameter is not assigned, return

		all changed rows.
acceptChanges	none	Commits all the changes data since it was loaded or since the last time acceptChanges was called.
rejectChanges	none	Rolls back all the changes data since it was created, or since the last time acceptChanges was called.
mergeCells	options	Merge some cells to one cell, the options contains following properties: index: the row index. field: the field name. rowspan: the rowspan count to be merged. colspan: the colspan count to be merged.

tree

Usage

Markup

Tree can be definded in <ul/> element. The markup can defines leaf and children, bellow is an example:

```
<ul id="tt">
    <li>
        <span>Folder</span>
        <ul>
            <li>
                <span>Sub Folder 1</span>
                <ul>
                    <li>
                        <span><a href="#">File 11</a></span>
                    </li>
                    <li>
                        <span>File 12</span>
                    </li>
                    <li>
                        <span>File 13</span>
                    </li>
                </ul>
            </li>
            <li>
                <span>File 2</span>
            </li>
            <li>
                <span>File 3</span>
            </li>
        </ul>
    </li>
    <li>
        <span>File21</span>
    </li>
</ul>
```

Tree can also be defined in an empty <ul/> element:

```
<ul id="tt"></ul>
```

jQuery

```
$('#tt').tree(options);
```

Tree data format

Every node can contains following properties:

- id: node id, which is important to load remote data
- text: node text to show
- state: node state, 'open' or 'closed', default is 'open'. When set to 'closed', the node have children nodes and will load them from remote site
- checked: Indicate whether the node is checked selected.
- attributes: custom attributes can be added to a node
- children: an array nodes defines some children nodes

Some example:

```
[{
    "id":1,
    "text":"Folder1",
    "iconCls":"icon-save",
    "children":[{
        "text":"File1",
        "checked":true
    },{
        "text":"Books",
        "state":"open",
        "attributes":{
            "url":"/demo/book/abc",
            "price":100
        },
        "children":[{
            "text":"PhotoShop",
            "checked":true
        },{
            "id": 8,
            "text":"Sub Bookds",
            "state":"closed"
        }]
    }]
},{
    "text":"Languages",
    "state":"closed",
    "children":[{
        "text":"Java"
    },{
        "text":"C#"
    }]
}]
```

Dependencies

none

Options

Override defaults with $.fn.tree.defaults.

Tree Node is a javascript object which contains following properties:

- id: An identity value bind to the node.
- text: Text to be showed.
- checked: Whether the node is checked.
- attributes: Custom attributes bind to the node.
- target: Target DOM object.

Properties

Name	Type	Description	Default
url	string	a URL to retrive remote data.	null
animate	boolean	Defines if to show animation effect when node expand or collapse.	false
checkbox	boolean	Defines if to show the checkbox before every node.	false
data	array	The node data to be loaded.	null

Events

Name	Parameters	Description
onClick	node	Fires when user click a node, the node parameter contains following properties: id: the node id text: the node text checked: Whether the node is checked attributes: the node custom attributes target: the target clicked DOM object
onDblClick	node	Fires when user dblclick a node.
onBeforeLoad	node, param	Fires before a request is made to load data, return false to cancel this load action.
onLoadSuccess	node, data	Fires when data loaded successfully.
onLoadError	arguments	Fires when data loaded fail, the arguments parameter is same as the 'error' function of jQuery.ajax.

onBeforeExpand	node	Fires before node is expanded, return false to cancel this expand action.
onExpand	node	Fires when node is expanded.
onBeforeCollapse	node	Fires before node is collapsed, return false to cancel this collapse action.
onCollapse	node	Fires when node is collapsed.

Methods

Name	Parameter	Description
options	none	Return the options of tree.
loadData	data	Load the tree data.
reload	none	Reload tree data.
getRoot	none	Get the root node, return node object
getRoots	none	Get the root nodes, return node array.
getParent	target	Get the parent node, the target parameter indicate the node DOM object.
getChildren	target	Get the children nodes, the target parameter indicate the node DOM object.
getChecked	none	Get all checked nodes.
getSelected	none	Get the selected node and return it, if no node selected return null.
isLeaf	target	Determine the specified node is leaf, the target

		parameter indicate the node DOM object.
find	id	Find the specifed node and return the node object.
select	target	Select a node, the target parameter indicate the node DOM object.
check	target	Set the specified node to checked.
uncheck	target	Set the specified node to unchecked.
collapse	target	Collapse a node, the target parameter indicate the node DOM object.
expand	target	Expand a node, the target parameter indicate the node DOM object.
collapseAll	none	Collapse all nodes.
expandAll	none	Expand all nodes.
expandTo	target	Expand from root to specified node.
append	param	Append some children nodes to a parent node. param parameter has two properties: parent: DOM object, the parent node to append to, if not assigned, append as root nodes. data: array, the nodes data.
toggle	target	Toggles expanded/collapsed state of the node, the target parameter indicate the node DOM object.
remove	target	Remove a node and it's children nodes, the target parameter indicate the node DOM object.
pop	target	Pop a node and it's children nodes, the method is same as remove but return the removed node data.
update	param	Update the specified node. param has following

		properties: target(DOM object, the node to be updated),id,text,iconCls,checked,etc.