

# Tutorial

## Introduction

This tutorial aims to use easyui framework to demonstrate how to create your web page easily.

First of all, you need to include some js and css file:

```
<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 predefines some icon CSS class which can show background image(16x16). If you wish to use it you need to include:

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

## Contents

1. Drag and Drop
  - [Basic Drag and Drop](#)
  - [Building a drag-drop shopping cart](#)
  - [Creating a School Timetable](#)
2. Menu and Button
  - [Create a simple menu](#)
  - [Create a Link Button](#)
  - [Create a Menu Button](#)
  - [Create a Split Button](#)
3. Layout
  - [Build border layout for Web Pages](#)
  - [Complex layout on Panel](#)
  - [Create Accordion](#)
  - [Create Tabs](#)
  - [Dynamically add tabs](#)
  - [Create XP style left panel](#)
4. DataGrid
  - [Convert a HTML table to DataGrid](#)
  - [Add a pagination to DataGrid](#)
  - [Get selected row data from DataGrid](#)
  - [Add a toolbar to DataGrid](#)
  - [Frozen columns for DataGrid](#)
  - [Dynamic change DataGrid columns](#)
  - [Formatting DataGrid columns](#)

- [Add sorting to DataGrid](#)
  - [Create column groups in DataGrid](#)
  - [CheckBox select on DataGrid](#)
  - [Custom DataGrid Paging](#)
  - [Enable DataGrid Inline Editing](#)
5. Window
- [My first window](#)
  - [Custom window tools](#)
  - [Window and Layout](#)
  - [Create Dialog](#)
6. Tree
- [Create Tree from markup](#)
  - [Create Async Tree](#)
  - [Append nodes to tree](#)
  - [Create Tree with CheckBox Nodes](#)
7. Form
- [Submit a form with Ajax](#)
  - [Add ComboTree field to a form](#)
  - [Form Validation](#)

## Basic Drag and Drop

[Tutorial](#) » Basic Drag and Drop

This tutorial show you how to make HTML elements draggable.

For this example, we will create three DIV elements and then enable them drag and drop.



First of all, we create three `<div>` elements:

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

For first `<div>` element, we make it draggable by default.

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

For second `<div>` element, we make it draggable by creating a proxy that clone the original element.

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

For third <div> element, we make it draggable by creating a custom proxy.

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

## Building a drag-drop shopping cart

[Tutorial](#) » Building a drag-drop shopping cart

If you can easily implement drag and drop with your web application, then you know you've got something special. With jQuery EasyUI, we have drag-drop capabilities in web application.

In this tutorial, we will show you how to build a shopping cart page which enables users to drag and drop the products they wish to buy. The shopping basket items and the price will be updated.



Displaying products on the page:

```
<ul class="products">
    <li>
        <a href="#" class="item">
            
            <div>
                <p>Balloon</p>
                <p>Price:$25</p>
```

```

        </div>
    </a>
</li>
<li>
    <a href="#" class="item">
        
        <div>
            <p>Feeling</p>
            <p>Price:$25</p>
        </div>
    </a>
</li>
<!-- other products -->
</ul>

```

As you can see in the code above, we add a <ul> element that contains some <li> elements to display the products. Every product has name and price properties which is contained inside the <p> element.

## Build the cart:

```

<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>

```

We use the datagrid to show the shopping basket items.

## Dragging the cloned product:

```

$($('.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';
    }
});

```

Notice that we set the draggable property 'proxy' to 'clone', so the dragged element will has clone effect.

## Dropping the selected product in the cart

```

$('.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);
}

```

}

When dropping the product, we get the product name and price first, then call 'addProduct' function to update shopping basket.

## Creating a School Timetable

### [Tutorial](#) » Creating a School Timetable

This tutorial will show you how to create a school timetable using jQuery EasyUI. We will build two tables: school subjects on the left and timetable on the right. You could drag school subject and drop it on a timetable cell. The school subject is a `<div class="item">` element while timetable cell is a `<td class="drop">` element.

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

### Displaying school subjects

```
<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>
```

### Displaying timetable

```

<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>

```

Drag the school subject on the left

```

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

```

Drop the school subject on timetable cell

```

$($('.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
        });
    }
}
});

```

As you can see the code above, when users drag the school subject on the left and drop it to the timetable cell, the onDrop callback function will be called. We clone the source element dragged from left and append it on timetable cell. When dragging school subject from timetable cell to another cell, simply move it.

## Create a simple menu

[Tutorial](#) » Create a simple menu

Menu is defined in some DIV markup, just like below:

```

<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>

```

To create the menu you should run following jQuery code:

```

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

```

When menu is created, it is not visible, you can invoke 'show' method to show it or 'hide' method to hide it:

```

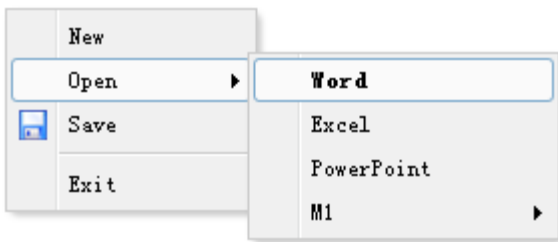
$('#mm').menu('show', {
  left: 200,
  top: 100
});

```

Now we create a menu and show it at postion(200,100).



Run the code and you will see following image:



## Create a Link Button

[Tutorial](#) » Create a Link Button

Normally a button is created using `<button/>` element.

A link button is created using `<a/>` element, so in fact a link button is a `<a/>` element but show a button style.

To create a link button, first create some `<a/>` elements:

```
<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>
```

As you see, the icon attribute is a icon CSS class which to show a icon image on button.  
Now run the code, you will see following pretty buttons:

## DEMO1



## DEMO2



Sometime you decide to disable a link button or enable it, below code demonstrate how to disable a link button:

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

## Create a Menu Button

[Tutorial](#) » Create a Menu Button

Menu button contains a button and a menu component, when click or move mouse over the button, a corresponding menu will show.

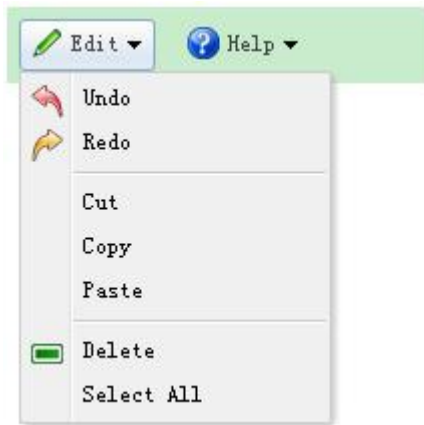
To define a menu button you should define a link button and a menu, below is an example:

```
<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>
```

Write some jQuery code:

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

Now a menu button is finished.



## Create a Split Button

[Tutorial](#) » Create a Split Button

SplitButton contains a linkbutton and a menu. When user click or hovering on down arrow area a corresponding menu will show. This example demonstrates how to create and use a splitbutton.

First of all, we create a linkbutton and menu markup:

```
<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>
  </div>
```

```

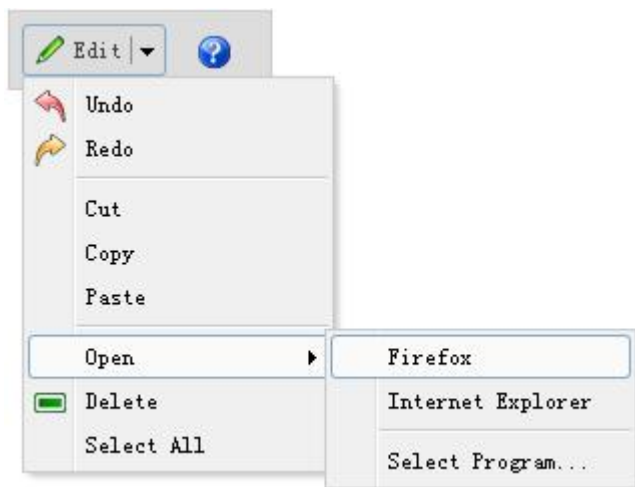
        <div class="menu-sep"></div>
        <div>Select Program...</div>
    </div>
</div>
<div icon="icon-remove">Delete</div>
<div>Select All</div>
</div>

```

Second, write jQuery code:

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

Finally, run the example and you will see:



## Build border layout for web pages

[Tutorial](#) » Build border layout for web pages

The border layout provides five regions: east,west,north,south,center. Below is some normal usage:

- The north region can be used for a web site banner.
- The south region can be used for copyright and other notes.
- The west region can be used for navigation menu.
- The east region can be used for promotion items.
- The center region can be used for main content.

To apply a layout you should confirm a layout container and then defines some region. The layout must has at least a center region. Below is a layout example:

```

<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>

```

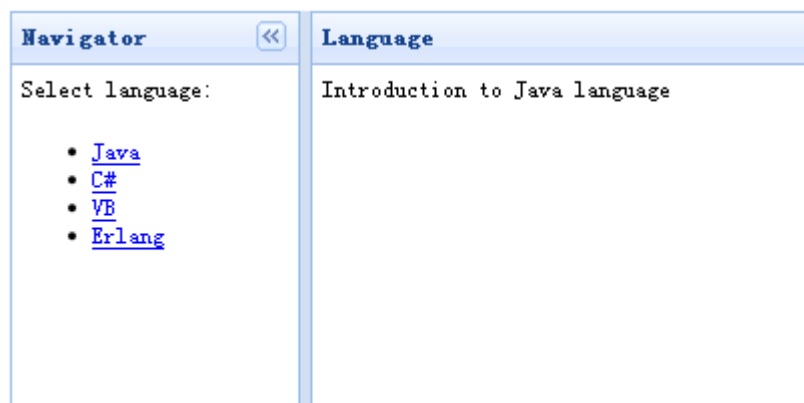
```

        <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>

```

We build a border layout in a <div> container. The layout split a container into two part, the left is a navigation menu and right is a main content. In the center region panel we set a 'href' attribute to load a initialize page.

Run the 'layout.html' page and you will see:



Finally we write an onclick event handle function to retrieve data, the 'showpage' function is very simple:

```

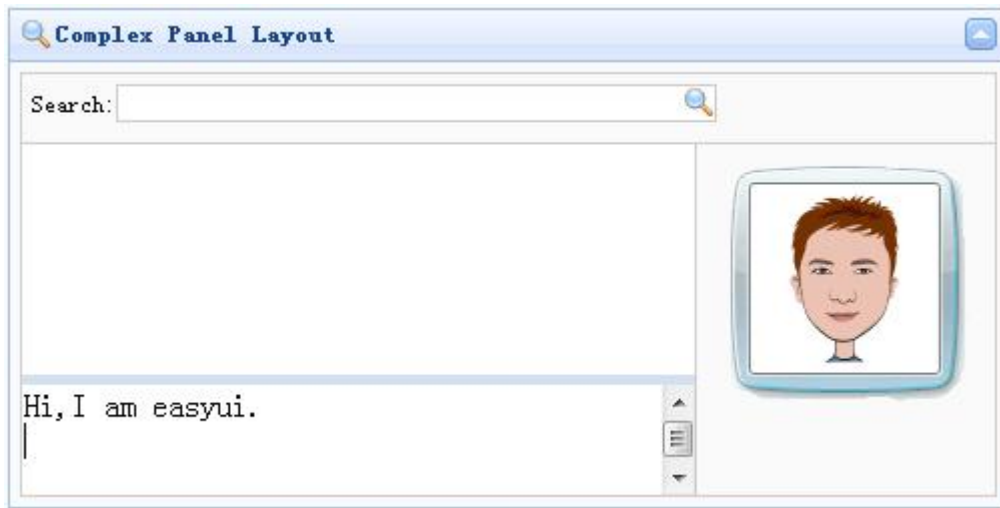
function showpage(url){
    $('#content').load(url);
}

```

## Complex layout on Panel

[Tutorial](#) » Complex layout on Panel

Panel allows you to create customized layouts for multiple uses. In this example we create a msn message box by using panel and layout plugins.



We use multiple layout in region panel. On top of message box we place a searching input, we also place a man image on the right. In center region we split it into two parts by set split attribute to true, which allows user to change the size of region panel.

Below is all the code:

```
<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">
          
        </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>
</div>
```

```

    </div>
</div>

```

We don't need to write any js code but own powerful ability of designing user-interface.

## Create Accordion

### [Tutorial](#) » Create Accordion

In this tutorial, you will learn about the easyui Accordion. Accordion contains a set of panels. All panel headers are all visible, but only one panel's body content is visible at a time. When user click the header of panel, the body content of that panel will become visible and other panel's body contents will become invisible.

```

<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="ttl1" 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>
                </ul>
            </li>
            <li>
                <span>File 2</span>
            </li>
        </ul>
    </div>

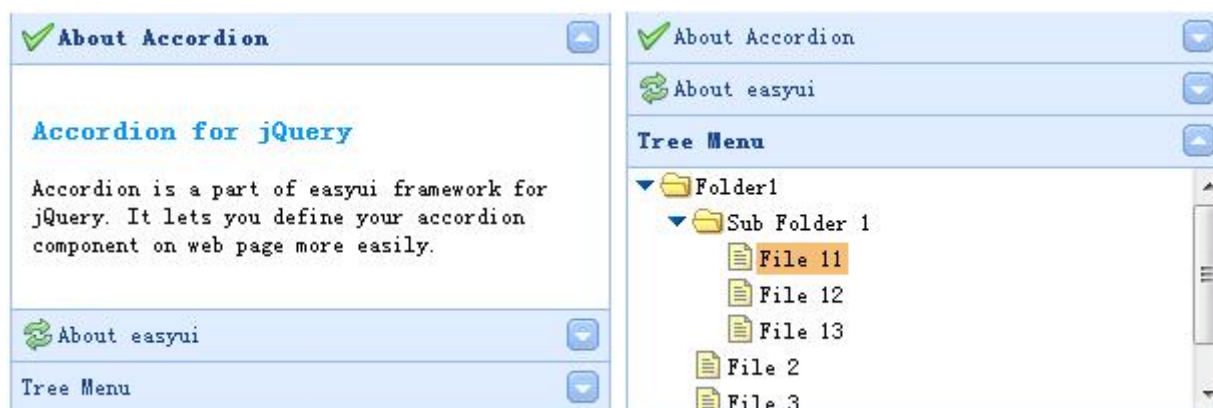
```

```

        </li>
        <li>
            <span>File 3</span>
        </li>
    </ul>
</li>
<li>
    <span>File2</span>
</li>
</ul>
</div>
</div>

```

We create three panels, the third panel contains a tree menu. Run the demo page and you will see:



## Create Tabs

### [Tutorial](#) » Create Tabs

This tutorial will show you how to create a tabs component using easyui. Tabs has multiple panels which can be added or removed dynamically. You can use tabs to show different entities on the same page.

Tabs display only one panel at a time, each panel has title, icon and close button. When tabs is selected the content of the corresponding panel shows.



Tabs created from HTML markup, including a DIV container and some DIV panels.

```

<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>

```

We create a tabs component with three panels, the second and third panel can be closed by clicking close button.

## Dynamically add tabs

[Tutorial](#) » Dynamically add tabs

When working with jQuery EasyUI it is easy to dynamically add tabs. You just call 'add' method.

In this tutorial we will dynamically add tabs that display one page using iframe. When clicking the add button, a new tab will be added. If the tab already exists it will become activated.



### Step 1: Create the 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.com')">easyui</a>
</div>
<div id="tt" class="easyui-tabs" style="width:400px;height:250px;">
  <div title="Home">
  </div>
</div>

```

The html code is simple, we create the tabs with one tab panel named 'Home'. Remember we don't need write any js code.

## Step 2: Implement the 'addTab' function

```

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
    });
  }
}

```

We use the 'exists' method to determine whether the tab is exists, if so active the tab. Calling the 'add' method to add a new tab panel.

## Create XP style left panel

[Tutorial](#) » Create XP style left panel

Normally, Explorer folders in Windows XP has left panel that contains common tasks. This tutorial show you how to create the XP left panel with panel plugin of easyui.

### Define several panels

We defines several panels to show some tasks, each panel should has only collapse/expand tool button.

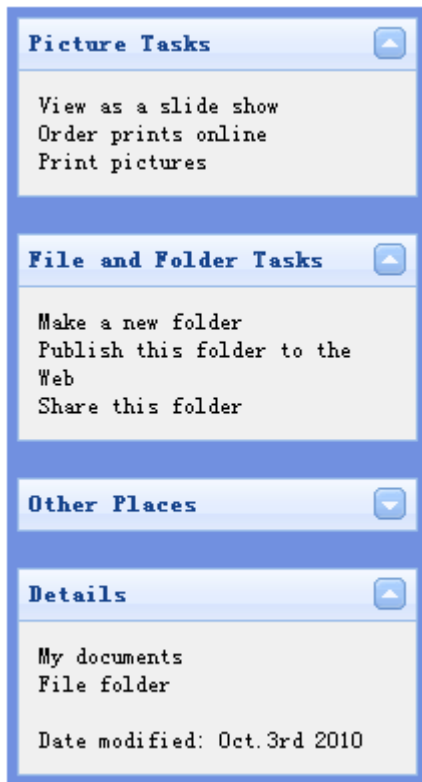
The code looks just like this:

```

<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>
```



Notice that the view appearance effect is not what we want, we must change the header background image of panel

and the collapse/expand button icon.

## Custom appearance effect of panel

It's not difficult to do this, What we should do is to redefine some 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;
}
```



As you can see, it's very easy while using easyui to define user-interface.

## Convert a HTML table to DataGrid

This example demonstrates how we can convert a table into datagrid.

The datagrid columns is defined in `<thead>` markup and data is defined in `<tbody>` markup. Make sure to set a field name for every data column, see below example:

```
<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>

```

We don't write any js code and will see the result:

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	

Sure, you can defines a complex table header such as:

```

<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	

## Add a pagination to DataGrid

[Tutorial](#) » Add a pagination to DataGrid

This example show how we can load data from server and how to add a pagination to datagrid.

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	

To load data from remote server, you should set 'url' peoperty, where server will return JSON format data. see [datagrid](#) document for more about the data format.

## Create a <table> markup

First of all, we defines a markup on web page:

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

## jQuery code

And write some jQuery code to create datagrid component:

```
$('#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
});
```

We defines datagrid columns and set 'pagination' property to true, which will generate a pagination bar on datagrid bottom. The pagination will send two parameters to server:

- page: The page number, start with 1.
- rows: The page rows per page.

We will use [etmvc framework](#) to write the background server code. So the url will be mapped to 'DataController' class and 'getItems' method.

For the example we defines data model:

```
@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;
}
```

Then write controller code:

```
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);
    }
}

```

## Database configuration example

```

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

```

## Deploy the example

- Create a MySQL database.
- Import the test table data from path '/db/item.sql', the table is named 'item'.
- Change the database configuration if needed, the configuration file is placed in /WEB-INF/classes/activerecord.properties.
- Run the program <http://localhost:8080/demo3/>

## Get selected row from DataGrid

[Tutorial](#) » Get selected row from DataGrid

This example show how to get the selected row data.

Load Data					
Item ID	Product ID	List Price	Unit Cost	Attribute	Status
EST-1	PI-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				



The datagrid component contains two methods to retrieve selected row data:

- `getSelected`: Get the first selected row data, if no row selected return null else return the record.
- `getSelections`: Get all selected row data, return array data which element is the record.

## Create markup

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

## Create 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}
    ]]
});
```

## Usage demos

To get the selected row data:

```
var row = $('#tt').datagrid('getSelected');
if (row){
```

```

    alert('Item ID:'+row.itemid+"\nPrice:"+row.listprice);
}

```

To get all selected 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'));

```

## Add a toolbar to DataGrid

[Tutorial](#) » Add a toolbar to DataGrid

This example show how to add a toolbar to datagrid.

DataGrid with Toolbar						
<div>  Add            Cut            Save         </div>						
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 plugin has a toolbar property which can defines a toolbar. A toolbar contains many button which is defined with following properties:

- text: The text show on button.
- iconCls: A CSS class which defines a background icon to show on left of button.
- handler: A function to process some things when user click the button.

## Markup

```
<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')
        }
    }
    ]
});

```

## Frozen Columns for DataGrid

[Tutorial](#) » Frozen Columns for DataGrid

This example demonstrates how to freeze some columns, The frozen columns do not scroll out of view when users moving horizontally across the grid.

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	FI-DLH-02				

To freeze columns you should define the frozenColumns property. The frozenColumn property is same as columns property.

Below is an example:

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

## Dynamic change DataGrid columns

[Tutorial](#) » Dynamic change DataGrid columns

The DataGrid columns can be defined using 'columns' property easily. If you want to dynamically change the columns, that's no problem at all. To change columns you can recall the datagrid method and pass a new columns property.

Below we define a datagrid component:

```

<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}
    ]]
});

```

Run the web page and you will see:

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	

Sometimes you want to change the columns, so you can write some code:

```

$('#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}
    ]]
});

```

Remember we have defined other properties such as url,width,height,etc. We don't need to defines them again, we define what we want to change.

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

## Formatting DataGrid columns

[Tutorial](#) » Formatting DataGrid columns

The following example formats a column in easyui DataGrid and uses a custom column formatter to color the text red if a price is below than 20.

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

To format a DataGrid column, we should set the formatter property which is a function. The format function contains two parameters:

- value: The current column value responding to field.
- record: The current row record data.

### 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}
    ]
});

```

## Add sorting to DataGrid

[Tutorial](#) » Add sorting to DataGrid

This example demonstrates how to sort the DataGrid by clicking on header of a column.

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-6	KO-DL-01	18.5	12	Spotless Male Puppy	P	
<div> 10 ▼   Page 1 of 3    Displaying 1 to 10 of 28 items </div>						

All columns in DataGrid can be sorted by clicking on header of column. You can define which column can be sorted. By default the column cannot be sorted unless you set the sortable property to true.

Below is an example:



## Markup

```
<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'
});
```

We defines some sortable columns including itemid,productid,listprice,unitcost etc. The 'attr1' column and 'status' column cannot be sorted. We also set the default sort column to 'itemid' and sort order to 'asc'.

When sorting the DataGrid will send two parameters to server:

- sort: The sort column field name.
- order: The sort order, can be 'asc' or 'desc', default is 'asc'.

We use [etmvc framework](#) to write background server code. First of all we define data models:

```
@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;
}
```

Then write controller code:

```

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);
    }
}

```

We use mysql database to store the demo data, below is the configuration example:

```

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

```

## Deploy the example

- Create a MySql database.
- Import the test table data from path '/db/item.sql', the table is named 'item'.
- Change the database configuration if needed, the configuration file is placed in /WEB-INF/classes/activerecord.properties.
- Run the program <http://localhost:8080/demo4/>

## Create column groups in DataGrid

[Tutorial](#) » Create column groups in DataGrid

The easyui DataGrid has ability to group columns, as the following example shows:

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

In this example, we use flat data to populate the DataGrid data, and group the listprice,unitcost,addr1,status columns under a single column.

To create column groups you should defines the columns property of datagrid plugin. Each element of columns is a definition of group which can use rowspan or colspan property to combine cells together.

The following code implements above example:

```
<table id="tt"></table>
$('#tt').datagrid({
    title:'Column Group',
    width:560,
    height:250,
    url:'datagrid_data.json',
    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}
    ]],
    rownumbers:true
});
```

## CheckBox select on DataGrid

### [Tutorial](#) » CheckBox select on DataGrid

This tutorial shows you how to place a checkbox column on DataGrid. With the checkbox users will have the option to select/deselect grid rows all at once.

✓ CheckBox Select							
<input type="checkbox"/>	Item ID	Product ID	List Price	Unit Cost	Attribute	Status	
<input type="checkbox"/>	EST-1	FI-SW-01	16.5	10	Large	P	
<input type="checkbox"/>	EST-10	K9-DL-01	18.5	12	Spotted Adult Fem	P	
<input checked="" type="checkbox"/>	EST-11	RP-SN-01	18.5	12	Venomless	P	
<input checked="" type="checkbox"/>	EST-12	RP-SN-01	18.5	12	Rattleless	P	
<input type="checkbox"/>	EST-13	RP-LI-02	18.5	12	Green Adult	P	
<input type="checkbox"/>	EST-14	FL-DSM-01	58.5	12	Tailless	P	
<input type="checkbox"/>	EST-15	FL-DSM-01	23.5	12	With tail	P	
<input type="checkbox"/>	EST-16	FL-DLM-02	93.5	12	Adult Female	P	

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

To add a checkbox column we simply add a column with checkbox property and set it to true. The code looks something like this:

```
<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
});
```

The code above we add a column that has a checkbox property so it will become a checkbox column. If the idField property is setted the selections of DataGrid will be maintained in different pages.

## Custom DataGrid Paging

[Tutorial](#) » Custom DataGrid Paging

The datagrid's built-in paging capability is a great feature, and it's relatively easy to custom. In this tutorial, we will create a datagrid and add some custom buttons on the pager bar.

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	

Page 1 of 3
 Displaying 1 to 10 of 28 items

### Markup

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

### Create 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}
    ]
});
```

Remember to set the 'pagination' property to true and the pager bar will be generated.

## Custom the pager bar


```
var pager = $('#tt').datagrid('getPager');    // get the pager of datagrid
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;
    }
});
```

As you can see above, we get the pager of datagrid first and then rebuild the pagination. We hide the page list and add three new buttons.

## Enable DataGrid Inline Editing

[Tutorial](#) » Enable DataGrid Inline Editing

The editable feature was recently added to the datagrid. It enable the user to add a new row to the datagrid. The user may also update one or more rows. This tutorial shows how to create a datagrid with inline editing.

✎ Editable DataGrid						
Item ID	Product	List Price	Unit Cost	Attribute	Status	Action
EST-1	Koi	16.5	10	Large	P	<a href="#">Edit</a> <a href="#">Delete</a>
EST-10	Dalmation	18.5	12	Spotted Adult Female	P	<a href="#">Edit</a> <a href="#">Delete</a>
EST-11	Rattlesnake ▼	18.5	12	Venomless	<input checked="" type="checkbox"/>	<a href="#">Save</a> <a href="#">Cancel</a>
EST-12	Rattlesnake	18.5	12	Rattleless	P	<a href="#">Edit</a> <a href="#">Delete</a>
EST-13	 This field is required.			Green Adult	<input checked="" type="checkbox"/>	<a href="#">Save</a> <a href="#">Cancel</a>
EST-14	Koi	58.5	12	Tailless	P	<a href="#">Edit</a> <a href="#">Delete</a>
EST-15	Dalmation	23.5	12	With tail	P	<a href="#">Edit</a> <a href="#">Delete</a>
EST-16	Rattlesnake	93.5	12	Adult Female	P	<a href="#">Edit</a> <a href="#">Delete</a>
EST-17	Iguana	93.5	12	Adult Male	P	<a href="#">Edit</a> <a href="#">Delete</a>
	Manx					
	Persian					
	Amazon Parrot					

## Create 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',options:{precis
ion: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);
    }

```



```
    }
  });
}
```

To enable editing in datagrid, you should add a editor property to the columns. The editor tells datagrid how to edit the field and how to save the field value. As you can see we define three editor: text, combobox and checkbox.

## Adding Editing Functionality

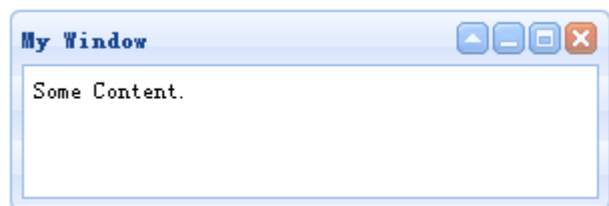
```
function editrow(index){
    $('#tt').datagrid('beginEdit', index);
}
function deleterow(index){
    $.messager.confirm('Confirm','Are you sure?',function(r){
        if (r){
            $('#tt').datagrid('deleteRow', index);
        }
    });
}
function saverow(index){
    $('#tt').datagrid('endEdit',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

## My first window

Create a window is very simple. We create a DIV markup:

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

Now run your test page and you will see a window show on screen. We donot need to write any js code.



If you wish to create a invisible window, remember to set a 'closed' attribute to 'true' value and you can invoke a 'open' method to open the window:

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

As a demonstration we create a login window in finally:

```
<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>
```

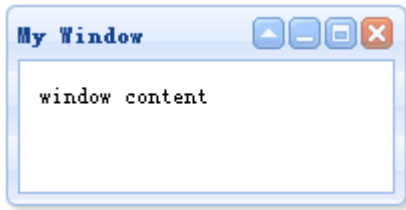


## Custom window tools

[Tutorial](#) » Custom window tools

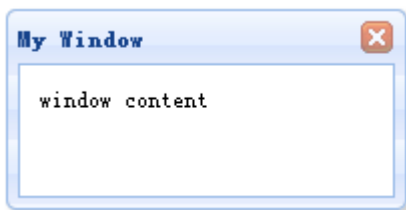
By default the window has four tools:collapsible,minimizable,maximizable and closable. For example we defines below window:

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



To custom the tools set the tools to true or false. For example we wish to define a window which has only one closable tool. You should set any other tool to false. We can defines tools property in markup or by jQuery code. Now we use the jQuery code to defines the window:

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



If we wish to add our custom tools to window, we can use tools property, as a demonstration we add our two tools to window:

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



## Window and Layout

[Tutorial](#) » Window and Layout

Layout components can be nested in window. We can build a complex layout window and even don't write any js code. The jquery-easyui framework helps us do rendering and resizing works in background.

As an example we build a window which contains two parts, one placed in left and another placed in right. In left of window we create a tree and in right of window create a tabs container.



```
<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>
</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
    </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>

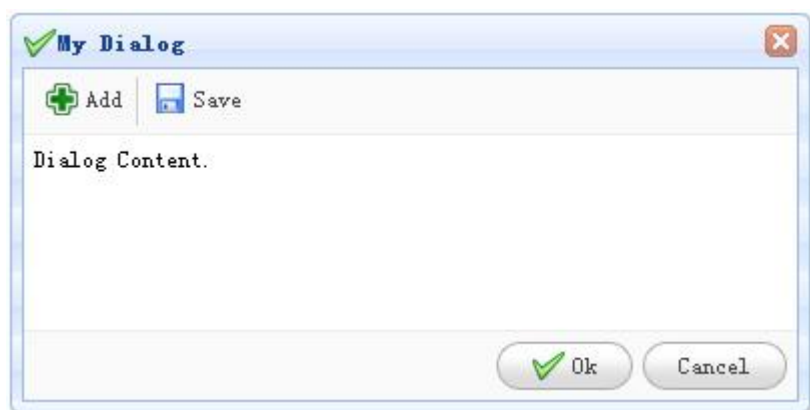
```

See above code, we only use HTML markup and the complex layout window show. This is jquery-easyui framework, easy and powerful.

## Create Dialog

[Tutorial](#) » Create Dialog

Dialog is a specified window, it can contains toolbar on top and buttons on bottom. By default dialog can not be resized but user can set resizable property to true to make it resizable.



Dialog is very simple, it can be created from DIV markup, just like this:

```

<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');
        }
    }
    ]
});

```

The code above we create a dialog with toolbar and buttons. This is standard configuration for dialog, toolbar, content and buttons.

## Create Tree from markup

A tree can be created from markup. easyui tree should be defined in `<ul>` element. The unordered lists `<ul>` element provides a basic tree structure.

Each `<li>` element will produce a tree node and sub `<ul>` element will produce a parent tree node.

Below is an example:

```

<ul id="tt">
    <li>
        <span>Folder</span>
        <ul>

```

```

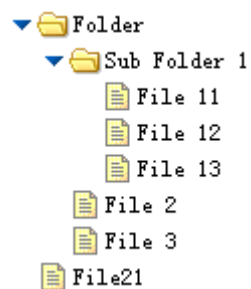
<li>
  <span>Sub Folder 1</span>
  <ul>
    <li><span>File 11</span></li>
    <li><span>File 12</span></li>
    <li><span>File 13</span></li>
  </ul>
</li>
<li><span>File 2</span></li>
<li><span>File 3</span></li>
</ul>
</li>
<li><span>File21</span></li>
</ul>

```

Create tree:

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

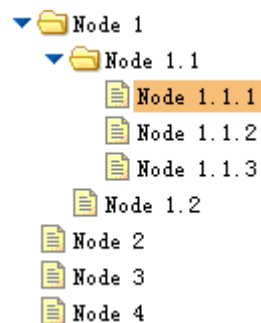
And you will see:



## Create Async Tree

To create Async Tree, every tree node must has a 'id' attribute which will post to back to retrieve children nodes data.

We will build an example using [etmvc framework](#) to return the nodes JSON data.



## Create a HTML Markup

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

## Build jQuery Code

We use the url property for retrieving remote data.

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

## Data Model

```
@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;
    }
}
```

## Writing Controller Code

If a node has children, remember to set the node state to '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);
    }
}

```

## Database Configuration Example

```

domain_base_class=com.et.ar.ActiveRecordBase

com.et.ar.ActiveRecordBase.adapter_class=com.et.ar.adapters.MySqlAdapt
er
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

```

## Deploy the example

- Create a MySql database.
- Import the test table data from path '/db/nodes.sql', the table is named 'nodes'.
- Change the database configuration if needed, the configuration file is placed in /WEB-INF/classes/activerecord.properties.
- Run the program <http://localhost:8080/demo2/>

## Append nodes to tree

[Tutorial](#) » Append nodes to tree

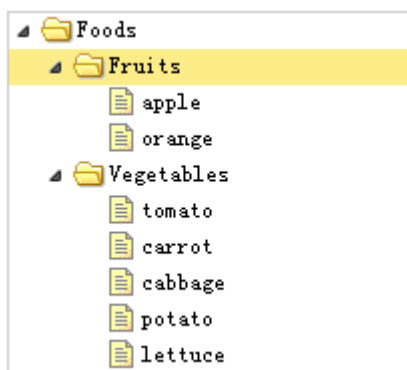
This tutorial show you how to append nodes to tree. We will create a Foods tree that contains fruit and vegetable node, and then add some other fruits to the existing fruit node.

## Create foods tree

First of all, we create the foods tree, the code looks like this:

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

Notice that the tree component is defined in `<UL>` markup and the tree node data is loaded from URL "tree\_data.json".



## Get parent node

Then we select the fruit node by clicking the node, to which we will append some other fruits data. Invoke the `getSelected` method to get the handle on node:

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

The return result of `getSelected` method is a javascript object that has `id`, `text`, `attributes` and `target` properties. The `target` property is a DOM object that reference to the selected node, with which the `append` method will use to append children nodes.

## Append nodes

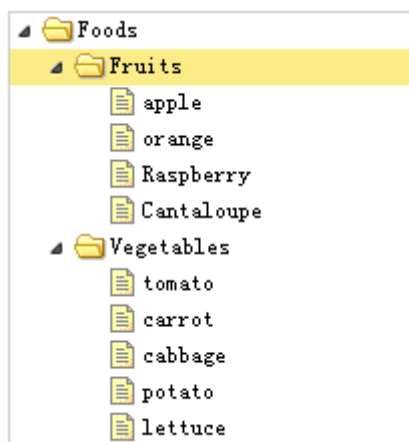
```
var node = $('#tt').tree('getSelected');
if (node){
  var nodes = [{
    id:13,
    text:"Raspberry"
  },{
    id:14,
    text:"Cantaloupe"
  }];
  $('#tt').tree('append', {
    parent:node.target,
    data:nodes
  });
}
```

```

    } ) ;
}

```

When append some fruits, you will see:



As you can see, using tree plugin of easyui to append node is not difficult.

## Create Tree with CheckBox Nodes

[Tutorial](#) » Create Tree with CheckBox Nodes

The tree plugin of easyui allows you to build a checkbox tree. If you click the checkbox of a node, the clicked node information gets inherited down and up. For example, click the checkbox of 'tomato' node and you can see 'Vegetables' node is now only checked partly.

### Create the tree markup

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

### Build the checkbox tree

```

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

```

As you can see, we use the easyloader to dynamic load the tree plugin. This feature allows us to load the page more quickly.

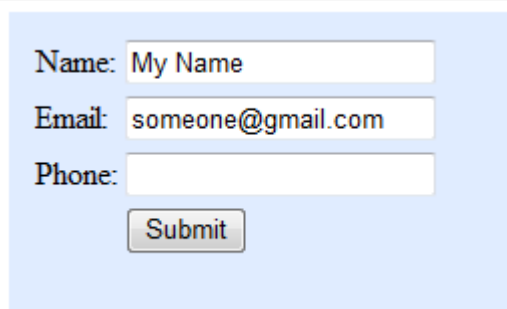
## Submit a form with Ajax

[Tutorial](#) » Submit a form with Ajax

This tutorial will show you how to submit a form with easyui. We create an example form with name, email and phone field. By using easyui form plugin we change the form to Ajax form. The form submits all the fields to the background process servlet, the servlet processes and sends some data back to the front page. We receive the back data and show it out.

### Build 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:

Submit

## Change to Ajax form

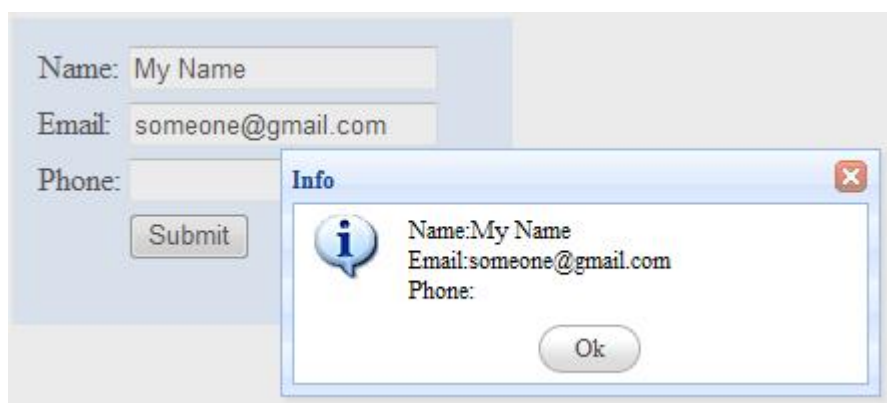
We write some jQuery code, and make the form can be submitted with Ajax. Notice that success function in the form plugin, when callback data received it will triggered so we can do something with this data.

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

## Process Servlet

```
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();
}
```

When click the submit button of form you will see:



## Add ComboTree field to a form

[Tutorial](#) » Add ComboTree field to a form

ComboTree is a ComboBox with a drop-down tree. It can be used as a form field that can be posted to remote server.

In this tutorial we will create a register form that has name,address,city fields. The city field is a combotree field in which user can drop down a tree panel and select a specified city.

## Step 1 - Create a HTML markup

```
<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>
```

See the code above, we set a url attribute for the combotree field named 'city', with which the field can retrieve tree data from remote server. Notice that the field has a class named 'easyui-combotree', so we don't need to write any js code, the combotree field will be rendered automatically.

## Step 2 - Create a dialog

We place the form in a dialog that has a submit button and cancel button.

```
$( '#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');
    }
  }
]
});

```



### Step 3 - Write Process Servlet

The server code process the request from form page and send some data back:

```

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();
    }
}

```

```
}
}
```

Now you can click a submit button and you will get a message box that show some data retrieved from remote server.



The ComboTree is very simple, what we do is to set a url attribute to retrieve tree data.

## Form Validation

[Tutorial](#) » Form Validation

This tutorial will show you how to validate a form. The easyui framework provides a validatebox plugin to validate a form. In this tutorial we will build a contact form and apply the validatebox plugin to validate the form. You can then adapt this form to your own requirements.

### Build form

Let's build a simple contact form with name, email, subject and message field:

```
<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>

```

We add a class named easyui-validatebox to input markup so the input markup will be applied the validation according the validType attribute.

## Prevent the form to submit when invalid

When users click the submit button of form, we should prevent the form to submit if the form is invalid.


```

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

```

If the form is invalid, a tooltip message will show.

Name:

Email:  
  Please enter a valid email address.

Subject:

Message:

## Write Process Servlet

Finally we write a background process servlet that show the request parameter on console and send a simple message to front page.

```
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();
    }
}
```