

## ExtJs2.0 学习系列(1)--Ext.MessageBox

---

大家都知道，刚开始搞 extjs 的时候，很是有点困难，所以本人在新手刚入门后准备了这个系列的文章。

个人认为用 extjs 做后台很不错，布局比较完美！

### 1.Ext.MessageBox.alert()方法

有四个参数，为简单起见，主要介绍前面三个参数：

**`alert( title, msg, function(){} )`**

其中 title, msg 为必选参数，function 为可选参数，在关闭弹出窗口后触发。

```
Ext.MessageBox.alert("title","msg");
```

```
Ext.MessageBox.alert("title","msg",function(){alert("关闭对话框后弹出！")});
```

### 2.Ext.MessageBox.confirm()方法

基本上同 alert()方法一模一样。

注意这点：

```
Ext.MessageBox.confirm("title","msg",function(e){alert(e);});
```

这个参数 e 是什么？它是你点击的弹出框的按钮的值，三种值：yes,no,cancel.Alert()方法也是如此，不过只有两种值：ok, cancel.

### 3.Ext.MessageBox.prompt()方法

有六个参数，比前面 alert 方法多一个返回值和是否多行。

```
Ext.MessageBox.prompt("title","msg");
```

```
Ext.MessageBox.prompt("title","msg",function(e,text){alert(e+"-"+text);});
```

```
//输入"qianxudetianxia",点击 ok 按钮，弹出 ok-qianxudetianxia
```

```
Ext.MessageBox.prompt("title","msg",function(e,text){alert(e+"-"+text);},this,true);
```

```
//true 为多行，this 表示作用域
```

#### 4.Ext.MessageBox.show()方法

功能很强大，采用 config 配置形式，比前面的方法使用更方便。

参数很多，在此列举最常用的配置参数：

1.animEl:对话框弹出和关闭时的动画效果,比如设置为"idl",则从 idl 处弹出并产生动画,收缩则相反

2.buttons:弹出框按钮的设置，主要有以下几种：Ext.Msg.OK,

Ext.Msg.OKCANCEL,

Ext.Msg.CANCEL,

Ext.Msg.YESNO,

Ext.Msg.YESNOCANCEL

你也可以自定义按钮上面的字：{"ok","我本来是 ok 的"}。

若设为 false，则不显示任何按钮。

3.closable:如果为 false，则不显示右上角的小叉叉，默认为 true。

4.msg:"消息的内容"

5.title:"标题"

6.fn:关闭弹出框后执行的函数

7.icon: 弹出框内容前面的图标，取值为 Ext.MessageBox.INFO,

Ext.MessageBox.ERROR,

Ext.MessageBox.WARNING,

Ext.MessageBox.QUESTION

8.width:弹出框的宽度，不带单位

9.prompt: 设为 true，则弹出框带有输入框

10.multiline: 设为 true，则弹出框带有多行输入框

11.progress:设为 true，显示进度条，（但是是死的）

12.progressText:显示在进度条上的字

13.wait: 设为 true，动态显示 progress

14.waitConfig:配置参数，以控制显示 progress

example:

```
Ext.MessageBox.show({
    title:"标题",
    msg:"内容的消息",
```

```

        buttons:{"ok":"我不再显示 OK 了"},
        fn:function(e){alert(e);},
        animEl:"test1",
        width:500,
        icon:Ext.MessageBox.INFO,
        closable:false,
        progress:true,
        wait:true,
        progressText:"进度条"
        // prompt:true
        // multiline:true
    });

```

#### 4.Ext.MessageBox.show()中的进度条的使用

首先必须知道例外两个方法 Ext.MessageBox.hide()和 Ext.MessageBox.updateProgress(value,"ProgressText","msg")(三个参数，看名字就知道意思)，

注意 value 为 0-1 之间的数，表示进度条的进度。

第一种：（通过进度的大小控制进度，满进度为 1）

```

Ext.get("btn1").on(
    "click",
    function(){
        Ext.MessageBox.show({
            title:"df",
            msg:"dfd",
            progress:true,
            width:300,
            closable:true
        });
        var f=function(v){
            return function(){
                if(v==12)

```

```

        {
            Ext.MessageBox.hide();

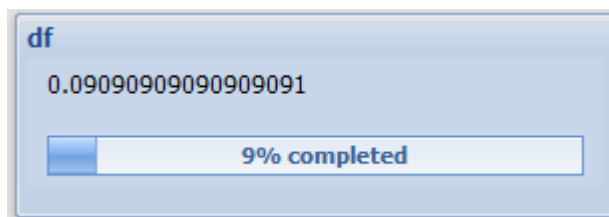
            //alert("加载完成!");
        }
        else
        {
            var i=v/11;

            Ext.MessageBox.updateProgress(i,Math.round(100*i)+
"% completed",i);
        }
    }

    for(var i=1;i<13;i++)
    {
        setTimeout(f(i),i*500);//从点击时就开始计时，所以 500*i 表示
每 500ms 就执行一次
    }
}

);

```



第二种：（通过固定时间控制进度加载）

```

Ext.get("btn1").on(
    "click",
    function(){
        Ext.MessageBox.show({
            title:"时间进度条",
            msg:"5s 后关闭进度框",
            progress:true,

```

```

        width:300,

        wait:true,

        waitConfig:{interval:600},//0.6s 进度条自动加载一定长度

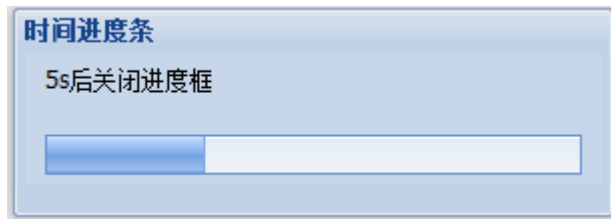
        closable:true

    });

    setTimeout(function(){Ext.MessageBox.hide()},5000);//5 后
    执行关闭窗口函数

}

```



最后关于那个 waitConfig 的参数，在此说明下：

- 1.interval:进度的频率
- 2.duration: 执行进度的持续时间，超过这个时间后，interval 失效，不再产生进度效果，但进度条也不会消失。
- 3.fn:duration 的时间到后执行的函数

所以，上面的通过时间控制进度另外一种写法为：

```

Ext.get("btn1").on(
    "click",
    function(){
        Ext.MessageBox.show({
            title:"时间进度条",
            msg:"5s 后关闭进度框",
            progress:true,
            width:300,
            wait:true,
            waitConfig:{
                interval:600,
                duration:5000,
                fn:function(){

```

```

        Ext.MessageBox.hide();//让进度条消失
    }},
    closable:true
});
//setTimeout(function(){Ext.MessageBox.hide()},5000);
}
);

```

效果一样。

MessageBox 类暂且就说这么多！一起期待下一章...

上一篇文章 ExtJs2.0 学习系列(1)--Ext.MessageBox，受到了大家的褒贬不一，还是有的朋友提出好的建议，在此表示感谢！

今天介绍 extjs 中的 Panel 组件。

//html 代码

```

<div id="container">
    </div>

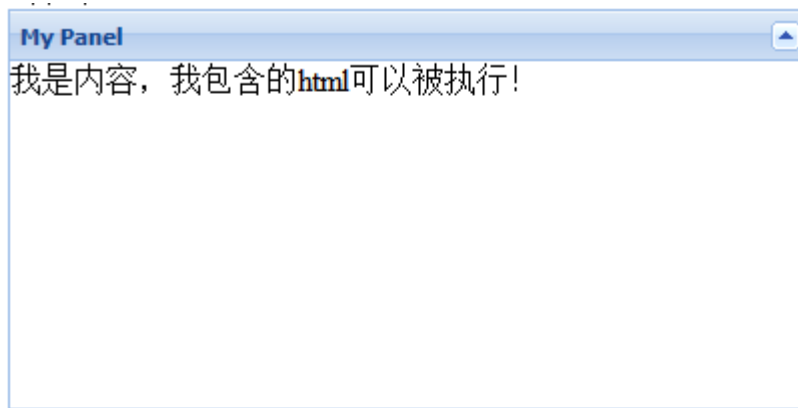
```

//js 代码

```

var p = new Ext.Panel({
    title: 'My Panel',//标题
    collapsible:true,//右上角上的那个收缩按钮，设为 false 则不显示
    renderTo: 'container',//这个 panel 显示在 html 中 id 为 container 的
层中
    width:400,
    height:200,
    html: "<p>我是内容，我包含的 html 可以被执行！</p>"//panel 主体中的内
容，可以执行 html 代码
});

```



因为 panel 组件的子类组件包括 TabPanel,GridPanel,FormPanel,TreePanel 组件，所以非常有必要介绍 Panel 组件的配置参数和相关的属性、方法。

//配置参数(只列举部分常用参数)

1.autoLoad: 有效的 url 字符串，把那个 url 中的 body 中的数据加载显示，但是可能没有样式和 js 控制，只是 html 数据

2.autoScroll: 设为 true 则内容溢出的时候产生滚动条，默认为 false

3.autoShow: 设为 true 显示设为"x-hidden"的元素，很有必要，默认为 false

4.bbar: 底部条，显示在主体内，//代码: `bbar:[{text:'底部工具栏 bottomToolbar'}]`,

5.tbar: 顶部条，显示在主体内，//代码: `tbar:[{text:'顶部工具栏 topToolbar'}]`,

6.buttons: 按钮集合，自动添加到 footer 中（footer 参数，显示在主体外）//代码: `buttons:[{text:"按钮位于 footer"}]`

7.buttonAlign: footer 中按钮的位置，枚举值为: "left","right","center",默认为 right

8.collapsible: 设为 true，显示右上角的收缩按钮，默认为 false

9.draggable: true 则可拖动，但需要你提供操作过程，默认为 false

10.html: 主体的内容

11.id: id 值，通过 id 可以找到这个组件，建议一般加上这个 id 值

12.width: 宽度

13.height: 高度

13.title: 标题

14.titleCollapse: 设为 true, 则点击标题栏的任何地方都能收缩, 默认为 false.

15.applyTo: (id) 呈现在哪个 html 元素里面

16.contentEl: (id) 呈现哪个 html 元素里面, 把 el 内的内容呈现

17.renderTo: (id) 呈现在哪个 html 元素里面

//关于这三个参数的区别(个人认为: applyTo 和 RenderTo 强调 to 到 html 元素中, contentEl 则是 html 元素到 ext 组件中去):

英文如下(本人英语 poor, 不敢乱翻译):

contentEl - This config option is used to take existing content and place it in the body of a new panel. It is not going to be the actual panel itself. (It will actually copy the innerHTML of the el and use it for the body). You should add either the x-hidden or the x-hide-display CSS class to prevent a brief flicker of the content before it is rendered to the panel.

applyTo - This config option allows you to use pre-defined markup to create an entire Panel. By entire, I mean you can include the header, tbar, body, footer, etc. These elements must be in the correct order/hierarchy. Any components which are not found and need to be created will be autogenerated.

renderTo - This config option allows you to render a Panel as its created. This would be the same as saying myPanel.render(ELEMENT\_TO\_RENDER\_TO);

哪位大人帮忙翻译下...

考虑到入门, 方法事件会在以后的文章中以实例穿插。

## 1.可拖动的 panel 实例

下面我们做个可拖动 panel 例子来熟悉下 panel 这个最基本的组件.

//html 代码

..无..

//下面创建一个允许拖动的 panel, 但是拖动的结果不能保存

```
var p=new Ext.Panel({  
    title: 'Drag me',
```

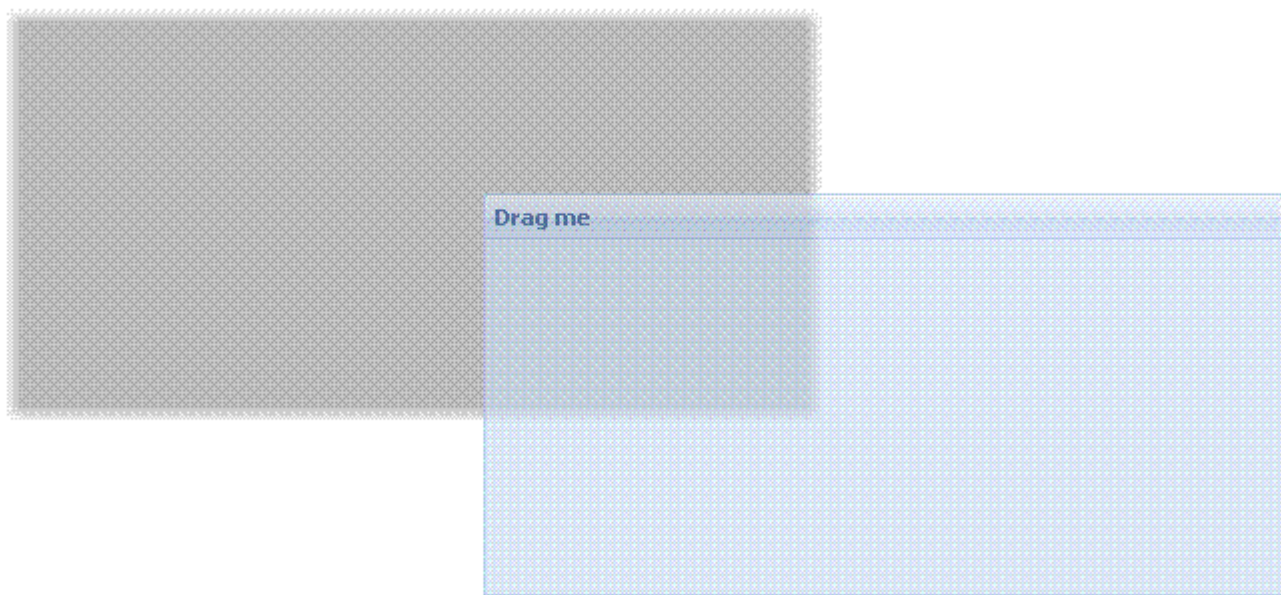


```
x: 100,  
y: 100,  
renderTo: Ext.getBody(), //x,y,renderTo:Ext.getBody() 初始化 panel 的  
位置  
floating: true, //true  
frame: true, //圆角边框  
width: 400,  
height: 200,  
draggable: true  
}).show(); //在这里也可以不 show ()
```

但是还不能拖到其他地方，我们需要改写 draggable:

```
draggable: {  
    insertProxy: false, //拖动时不虚线显示原始位置  
  
    onDrag : function(e){  
        var pel = this.proxy.getEl();  
        this.x = pel.getLeft(true);  
        this.y = pel.getTop(true); //获取拖动时 panel 的坐标  
    },  
  
    endDrag : function(e){  
        this.panel.setPosition(this.x, this.y); //移动到最终位置  
    }  
}
```

实现了可保存的拖动，如图：



拖动的时候阴影还在原位置，我们再在 draggable 中的 onDrag 事件中添加代码：

```
var s = this.panel.getEl().shadow;

    if (s) {
        s.realign(this.x, this.y, pel.getWidth(), pel.getHeight());
    }

//shadow 的 realign 方法的四个参数，改变 shadow 的位置大小属性
```

最后这个可拖动的 panel 的代码为：

```
var p=new Ext.Panel({
    title: 'Drag me',
    x: 100,
    y: 100,
    renderTo: Ext.getBody(),
    floating: true,
    frame: true,
    width: 400,
    height: 200,
```

```

draggable: {
    insertProxy: false,

    onDrag : function(e){
        var pel = this.proxy.getEl();
        this.x = pel.getLeft(true);
        this.y = pel.getTop(true);

        var s = this.panel.getEl().shadow;
        if (s) {
            s.realign(this.x, this.y, pel.getWidth(), pel.getHeig
ht());
        }
    },
    endDrag : function(e){
        this.panel.setPosition(this.x, this.y);
    }
}
}))
//效果图片我就不贴出来了

```

## 2.带顶部，底部，脚部工具栏的 panel

```

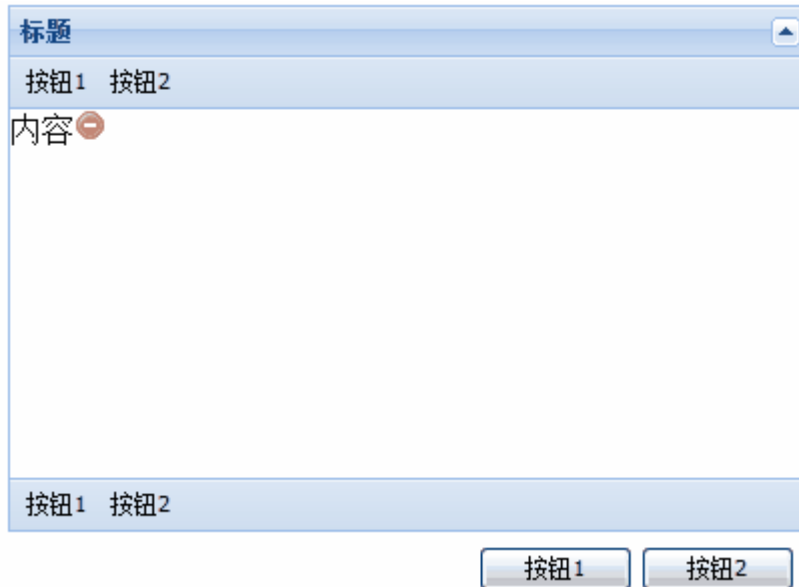
var p=new Ext.Panel({
    id:"panell1",
    title:"标题",
    collapsible:true,
    renderTo:"container",
    closable:true,
    width:400,
    height:300,
    tbar:[{text:"按钮 1"},{text:"按钮 2"}], //顶部工具栏
    bbar:[{text:"按钮 1"},{text:"按钮 2"}], //底部工具栏

```

```

        html: "内容",
        buttons: [{text: "按钮 1"}, {text: "按钮 2"}] //footer 部工具栏
    });

```



我们已经在各种工具栏上添加了按钮，但是却没有激发事件，下面我们来添加按钮事件代码：

```

tbar: [{text: "按钮 1", handler: function() {Ext.MessageBox.alert("我是按钮 1", "我是通过按钮 1 激发出来的弹出框！")}}, {text: "按钮 2"}],
//改写 tbar，添加 handler 句柄，点击顶部工具栏上按钮 1，弹出提示框，效果图大家想象下，就不贴出来了

```

当然，一般情况下，我们只要一个工具栏，这里只是为了演示！

### 3.panel 工具栏

```

//添加下面的代码到 panel 配置参数中
tools: [{id: "save"}, {id: "help"}, {id: "up"}, {id: "close", handler: function() {Ext.MessageBox.alert("工具栏按钮", "工具栏上的关闭按钮时间被激发了")}}],
//id 控制按钮，handler 控制相应的事件
//id 的枚举值为：
toggle (collapsible 为 true 时的默认值)
close
minimize
maximize
restore

```

```
gear
pin
unpin
right
left
up
down
refresh
minus
plus
help
search
save
print
```

标题



关于 panel 今天就讨论到这里，欢迎批评！一起期待下一篇文章.

前言：关于 extjs，为了照顾还没有入门的新手，我给一点提示，有一个网站\*\*\*视频在线里面请了个老师录制了 extjs 的介绍入门的视频，环境可能不同，但原理和使用方法是一样的，绝对值得一看（如果你想入门的话），希望没有广告的意思。

主站：\*\*\*视频在线

extjs 介绍及应用举例：ExtJS 视频教程 第 1 讲 ExtJS 介绍及应用举例

我不知道是不是要注册，觉的好的话，就值！

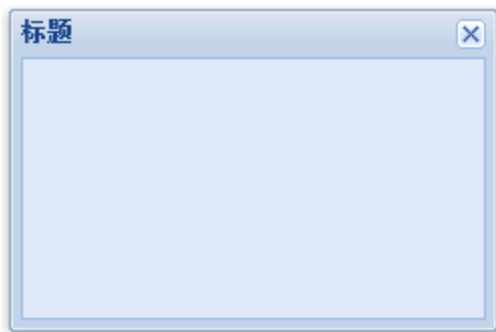
前面介绍了 panel 组件--ExtJs2.0 学习系列(2)--Ext.Panel，今天将介绍 window 组件，它继承自 panel。

先介绍个最简单例子

//html 代码

```
<div id="win" class="x-hidden">
    </div>
```

```
//js 代码
var w=new Ext.Window({
    contentEl:"win",//主体显示的 html 元素，也可以写为 el:"win"
    width:300,
    height:200,
    title:"标题"
});
w.show();
```



#### 参数介绍:

因为前面已经介绍了 panel 组件，下面只介绍 window 组件的几个其他特别的配置参数

//几个前面没有介绍的 window 自己的配置参数

- 1.closeAction: 枚举值为: close(默认值), 当点击关闭后, 关闭 window 窗口  
hide, 关闭后, 只是 hidden 窗口
- 2.closable:true 在右上角显示小叉叉的关闭按钮, 默认为 true
- 3.constrain: true 则强制此 window 控制在 viewport, 默认为 false
- 4.modal:true 为模式窗口, 后面的内容都不能操作, 默认为 false
- 5.plain: //true 则主体背景透明, false 则主体有小差别的背景色, 默认为 false

//需要显示下 show () 方法, 默认的窗口是可拖动的, 可关闭的, 可拖动大小的  
w.show ()

#### 实例介绍:

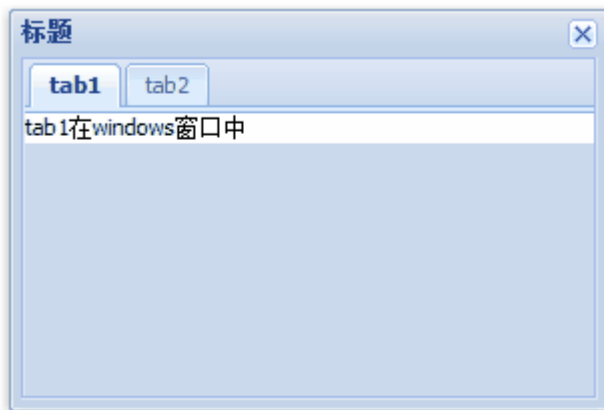
##### 1.嵌套了 tabpanel 的 window

```
var w=new Ext.Window({
    contentEl:"win",
```

```

width:300,
height:200,
items:new Ext.TabPanel({
    activeTab:0,//当前标签为第 1 个 tab (从 0 开始索引)
    border:false,
    items:[{title:"tab1",html:"tab1 在 windows 窗口中"},{title:"tab2",html:"tab2 在 windows 窗口中"}]//TabPanel 中的标签页，以后再深入讨论
}),
plain:true,//true 则主体背景透明，false 则主体有小差别的背景色，默认为 false
title:"标题"
});
w.show();

```



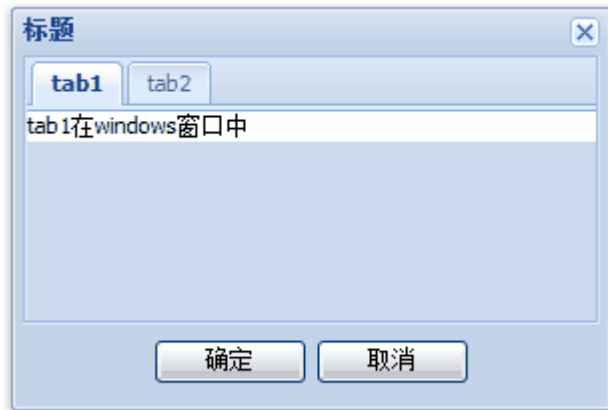
我们通过 items 把 TabPanel 组件嵌套在 window 的主体中去了。

我们在添加工具栏看看

```

// bbar:[{text:"确定"},{text:"取消",handler:function(){w.close();}},//
//bottom 部
    buttons:[{text:"确定"},{text:"取消",handler:function(){w.close
();}},//footer 部
    buttonAlign:"center",//footer 部按钮排列位置,这里是中间
// collapsible:true,//右上角的收缩按钮

```



其他工具栏方法一样。

关于 window 中嵌套复杂的布局，等我们谈完了 extjs 布局再举例说明。

今天的内容比较少，请高手 pp，让我们期待下篇文章！

上篇中我们讨论了 Ext.window 的简单使用，今天我们要看看强大的 FormPanel,也是继承 panel 组件的使用。

首先弄清楚这个问题，创建的时候：

//查看源代码便知，两种方法是一样的

```
Ext.form.FormPanel = Ext.FormPanel;
```

我们还是从最简单的代码实例开始吧：

```
<!--html 代码-->
<body>
<div id="form1"></div>
</body>
```

//js 代码

```
var form1 = new Ext.form.FormPanel({
    width:350,
    frame:true,//圆角和浅蓝色背景
    renderTo:"form1",//呈现
    title:"FormPanel",
    bodyStyle:"padding:5px 5px 0",
    items:[
        {
            fieldLabel:"UserName",//文本框标题
            xtype:"textfield",//表单文本框
```



```

        name: "user",
        id: "user",
        width: 200
    },
    {
        fieldLabel: "PassWord",
        xtype: "textfield",
        name: "pass",
        id: "pass",
        width: 200
    }
],
    buttons: [{text: "确定"}, {text: "取消", handler: function() {alert("
事件! ");}}]
});

```

都是通过 items 属性参数把表单元素添加到这个表单中。

我们发现两个文本框的类型和宽度是一样的，我们还可以把 items 里面的相同项提取出来，以简化代码：

```

//简化代码，和上面的代码效果一样
var form1 = new Ext.form.FormPanel({
    width: 350,
    frame: true,
    renderTo: "form1",
    title: "FormPanel",
    bodyStyle: "padding: 5px 5px 0",
    defaults: {width: 200, xtype: "textfield"}, //*****简化*****//
    items: [

```

```

        {
            fieldLabel: "UserName",
            // xtype: "textfield",
            name: "user",
            id: "user",
            // width: 200
        },
        {
            fieldLabel: "PassWord",
            // xtype: "textfield",
            name: "pass",
            id: "pass",
            inputType: "password",
            // width: 200
        }
    ],
    buttons: [{ text: "确定" }, { text: "取消", handler: function() { alert
(); } } ],
    });

```

关于 inputType, 参数如下:

```

//input 的各种类型 (这个大家都知道, 就只列了几个典型的)
radio
check
text (默认)
file
password 等等

```

关于 FormPanel 的配置参数, 请主要参考 panel 的参数, 这里列举另外两个:

1. labelAlign: fieldlabel 的排列位置, 默认为 "left", 其他两个枚举值是 "center", "right"
2. labelWidth: fieldlabel 的占位宽
3. method: "get" 或 "post"
4. url: "提交的地址"

5.submit:提交函数 //稍后我们一起详细分析

因为内容太多，我们先看一个例子。

### 1.FormPanel 的 fieldset 应用

//把前面代码重写 items 属性

```
items:[
    {
        xtype:'fieldset',
        title: '个人信息',
        collapsible: true,
        autoHeight:true,
        width:330,
        defaults: {width: 150},
        defaultType: 'textfield',
        items :[{
            fieldLabel: '爱好',
            name: 'hobby',
            value: 'www.cnblogs.com'
        },{
            xtype:"combo",
            name: 'sex',
            store:["男","女","保密"],//数据源为一数组
            fieldLabel:"性别",
            emptyText:'请选择性别.'
        }]
    }
]
```

这里的 combox 组件只是简单的演示，具体还是要深入了解，我们会在以后的内容中详细探讨。

**2.关于 xtype 的类型，在 extjs 的 form 表单（其他的请参考 api）中已经定义的有：**

Form components

form	Ext.FormPanel
checkbox	Ext.form.Checkbox
combo	Ext.form.ComboBox
datefield	Ext.form.DateField
field	Ext.form.Field
fieldset	Ext.form.FieldSet
hidden	Ext.form.Hidden
htmleditor	Ext.form.HtmlEditor
label	Ext.form.Label
numberfield	Ext.form.NumberField
radio	Ext.form.Radio
textarea	Ext.form.TextArea
textfield	Ext.form.TextField
timefield	Ext.form.TimeField
trigger	Ext.form.TriggerField

不早了，FormPanel 还有很多的东西要了解，但是今天不能再讲了，太多了，大家都没有兴致看下去,明天继续。

extjs 的东西很庞大，但是还没有一个人用中文系统的分析写文章，所以资料都很可贵，现在我想做这件事，请大家多多支持，我才有动力，才有激情，才能写出更好的下一篇文章，让我们期待更精彩的 next 吧！

前言：关于 extjs，为了照顾还没有入门的新手，我给一点提示，有一个网站\*\*\*视频在线里面请了个老师录制了 extjs 的介绍入门的视频，环境可能不同，但原理和使用方法是一样的，绝对

值得一看（如果你想入门的话），希望没有广告的意思。

主站：\*\*\*视频在线

extjs 介绍及应用举例：ExtJS 视频教程 第 1 讲 ExtJS 介绍及应用举例

我不知道是不是要注册，觉的好的话，就值！

前面介绍了 panel 组件--ExtJs2.0 学习系列(2)--Ext.Panel，今天将介绍 window 组件，它继承自 panel。

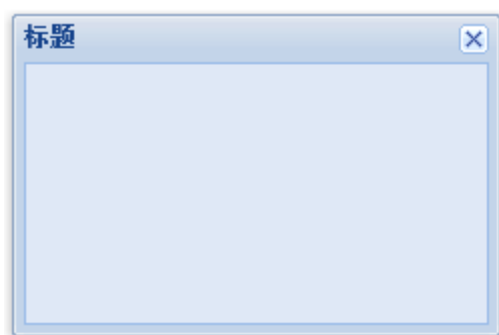
先介绍个最简单例子

//html 代码

```
<div id="win" class="x-hidden">
    </div>
```

//js 代码

```
var w=new Ext.Window({
    contentEl:"win", //主体显示的 html 元素，也可以写为 el:"win"
    width:300,
    height:200,
    title:"标题"
});
w.show();
```



参数介绍：

因为前面已经介绍了 panel 组件，下面只介绍 window 组件的几个其他特别的配置参数

//几个前面没有介绍的 window 自己的配置参数

1.closeAction:枚举值为: close(默认值)，当点击关闭后，关闭 window 窗口

hide, 关闭后，只是 hidden 窗口

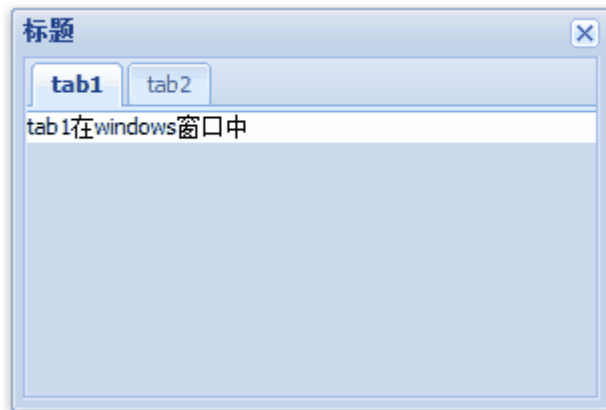
- 2.closable:true 在右上角显示小叉叉的关闭按钮，默认为 true
- 3.constrain: true 则强制此 window 控制在 viewport，默认为 false
- 4.modal:true 为模式窗口，后面的内容都不能操作，默认为 false
- 5.plain: //true 则主体背景透明，false 则主体有小差别的背景色，默认为 false

//需要显示下 show () 方法,默认的窗口是可拖动的,可关闭的,可拖动大小的  
w.show ()

实例介绍:

### 1.嵌套了 tabpanel 的 window

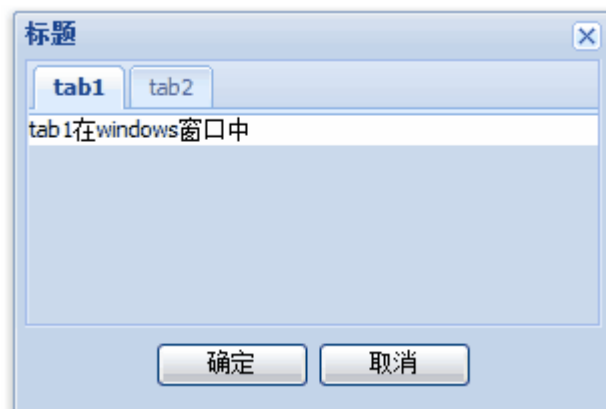
```
var w=new Ext.Window({
    contentEl:"win",
    width:300,
    height:200,
    items:new Ext.TabPanel({
        activeTab:0,//当前标签为第 1 个 tab (从 0 开始索引)
        border:false,
        items:[{title:"tab1",html:"tab1 在 windows 窗口中"},{title:"tab2",html:"tab2 在 windows 窗口中"}]//TabPanel 中的标签页，以后再深入讨论
    }),
    plain:true,//true 则主体背景透明，false 则主体有小差别的背景色，默认为 false
    title:"标题"
});
w.show();
```



我们通过 items 把 TabPanel 组件嵌套在 window 的主体中去了。

我们在添加工具栏看看

```
// bbar:[{text:"确定"},{text:"取消",handler:function(){w.close();}},/  
/bottom 部  
  
    buttons:[{text:"确定"},{text:"取消",handler:function(){w.close  
();}},],//footer 部  
  
    buttonAlign:"center",//footer 部按钮排列位置,这里是中间  
// collapsible:true,//右上角的收缩按钮
```



其他工具栏方法一样。

关于 window 中嵌套复杂的布局，等我们谈完了 extjs 布局再举例说明。

今天的内容比较少，请高手 pp，让我们期待下篇文章！

上篇中我们讨论了 Ext.window 的简单使用，今天我们要看看强大的 FormPanel,也是继承 panel 组件的使用。

首先弄清楚这个问题，创建的时候：

```
//查看源代码便知，两种方法是一样的  
  
Ext.form.FormPanel = Ext.FormPanel;
```

我们还是从最简单的代码实例开始吧：

```
<!--html 代码-->

<body>

<div id="form1"></div>

</body>
```

```
//js 代码

var form1 = new Ext.form.FormPanel({
    width:350,
    frame:true,//圆角和浅蓝色背景
    renderTo:"form1",//呈现
    title:"FormPanel",
    bodyStyle:"padding:5px 5px 0",
    items:[
        {
            fieldLabel:"UserName",//文本框标题
            xtype:"textfield",//表单文本框
            name:"user",
            id:"user",
            width:200
        },
        {
            fieldLabel:"PassWord",
            xtype:"textfield",
            name:"pass",
            id:"pass",
            width:200
        }
    ],
    buttons:[{text:"确定"},{text:"取消",handler:function(){alert("
事件! ");}}]
});
```



都是通过 items 属性参数把表单元素添加到这个表单中。

我们发现两个文本框的类型和狂度是一样的，我们还可以把 items 里面的相同项提取出来，以简化代码：

//简化代码，和上面的代码效果一样

```
var form1 = new Ext.form.FormPanel({
    width:350,
    frame:true,
    renderTo:"form1",
    title:"FormPanel",
    bodyStyle:"padding:5px 5px 0",
    defaults:{width:200,xtype:"textfield"},//*****简化****//
    items:[
        {
            fieldLabel:"UserName",
            //xtype:"textfield",
            name:"user",
            id:"user",
            //width:200
        },
        {
            fieldLabel:"PassWord",
            //xtype:"textfield",
            name:"pass",
            id:"pass",
            inputType:"password",
            //width:200
        }
    ]
})
```

```

        ],
        buttons:[{text:"确定"},{text:"取消",handler:function(){alert
();}}]
    });
};

```

关于 `inputType`, 参数如下:

```

//input 的各种类型 (这个大家都知道, 就只列了几个典型的)
radio
check
text (默认)
file
password 等等

```

关于 `FormPanel` 的配置参数, 请主要参考 `panel` 的参数, 这里列举另外两个:

1. `labelAlign: fieldlabel` 的排列位置, 默认为 "left", 其他两个枚举值是 "center", "right"
2. `labelWidth: fieldlabel` 的占位宽
3. `method: "get" 或 "post"`
4. `url: "提交的地址"`
5. `submit: 提交函数` //稍后我们一起详细分析

因为内容太多, 我们先看一个例子。

## 1. `FormPanel` 的 `fieldset` 应用

```

//把前面代码重写 items 属性
items:[
    {
        xtype:'fieldset',
        title: '个人信息',
        collapsible: true,
        autoHeight:true,
        width:330,
        defaults: {width: 150},
        defaultType: 'textfield',
        items :[{

```

```

        fieldLabel: '爱好',
        name: 'hobby',
        value: 'www.cnblogs.com'
    }, {
        xtype: "combo",
        name: 'sex',
        store: ["男", "女", "保密"], //数据源为一数组
        fieldLabel: "性别",
        emptyText: '请选择性别.'
    }
    ]
}
]

```

这里的 combo 组件只是简单的演示, 具体还是要深入了解, 我们会在以后的内容中详细探讨。

**2.关于 xtype 的类型, 在 extjs 的 form 表单** (其他的请参考 api) 中已经定义的有:

Form components

form	Ext.FormPanel
checkbox	Ext.form.Checkbox
combo	Ext.form.ComboBox
datefield	Ext.form.DateField
field	Ext.form.Field
fieldset	Ext.form.FieldSet
hidden	Ext.form.Hidden
htmleditor	Ext.form.HtmlEditor
label	Ext.form.Label

numberfield	Ext.form.NumberField
radio	Ext.form.Radio
textarea	Ext.form.TextArea
textfield	Ext.form.TextField
timefield	Ext.form.TimeField
trigger	Ext.form.TriggerField

不早了, FormPanel 还有很多的东西要了解, 但是今天不能再讲了, 太多了, 大家都没有兴致看下去, 明天继续。

extjs 的东西很庞大, 但是还没有一个人用中文系统的分析写文章, 所以资料都很可贵, 现在我想做这件事, 请大家多多支持, 我才有动力, 才有激情, 才能写出更好的下一篇文章, 让我们期待更精彩的 next 吧!

上篇中我们简单的谈到了 FormPanel 中的 fieldset 和 ComboBox, 今天我们继续把这个话题说下去, 说全一点, 说深一点。

### 3. 可选的 fieldset 实例

其实就是带个 checkbox, 有点像论坛注册时有一部分是选填信息的那种效果, 主要知识点:

// 因为觉得这个参数特别, 特举一例以说明

1. checkboxToggle: true // true 则呈现一个带 checkbox 的 fieldset, 选中则展开, 否则相反, 默认为 false

2. checkboxName: string // 当上面为 true 时, 作为 checkbox 的 name, 方便表单操作

这里我把 js 核心代码贴出来 (html 代码与上一篇中完全相同, 不列出):

// 在上一节的基础代码的 items 里面添加如下配置

```
{
  xtype: "fieldset",
  checkboxToggle: true, // 关键参数, 其他和以前的一样
  checkboxName: "dfdf",
  title: "选填信息",
  defaultType: 'textfield',
  width: 330,
  autoHeight: true, // 使自适应展开排版
  items: [{
    fieldLabel: "UserName",
    name: "user",
    anchor: "95%" // 330px - labelWidth 剩下的宽度的 95%, 留下 5% 作为后面提到的验
```

证错误提示

```
    },  
    {  
      fieldLabel: "PassWord",  
      inputType: "password", //密码文本  
      name: "pass",  
      anchor: "95%"  
    }  
  ]  
}
```

The image shows a 'FormPanel' dialog box with a blue header. It contains two sections: '选填信息' (Optional Information) with a checked checkbox, containing 'UserName:' and 'PassWord:' text labels next to empty input fields; and '个人信息' (Personal Information) with an unchecked checkbox, containing '爱好:' (Hobby) with the value 'www.cnblogs.com' and '性别:' (Gender) with a dropdown menu showing '请选择性别....'. At the bottom are '确定' (OK) and '取消' (Cancel) buttons.

The image shows a 'FormPanel' dialog box with a blue header. It contains two sections: '选填信息' (Optional Information) with an unchecked checkbox, which is currently empty; and '个人信息' (Personal Information) with an unchecked checkbox, containing '爱好:' (Hobby) with the value 'www.cnblogs.com' and '性别:' (Gender) with a dropdown menu showing '请选择性别....'. At the bottom are '确定' (OK) and '取消' (Cancel) buttons.

#### 4. 表单验证实例（空验证，密码确认证，email 验证）

我们可以用单独的 js 写表单验证，但是 extjs 已经为我们想到了（自己单独写反而不方便）。

在验证之前，我不得不提两个小知识点：

//大家在很多的 extjs 代码中都看到了这两个，他们都起提示作用的

Ext.QuickTips.init();//支持 tips 提示

```
Ext.form.Field.prototype.msgTarget='side';//提示的方式,枚举值为"qtip","title","under","side",id(元素id)
```

//side 方式用的较多,右边出现红色感叹号,鼠标上去出现错误提示,其他的我就不介绍了,可自行验证

//大家可以分别去掉这两行代码,看效果就会明白他们的作用,(放在 onReady 的 function() {} 中)

### 1.我们看一个最简单的例子:空验证(其实这不算是个专门的验证例子)

//空验证的两个参数

1.allowBlank:false//false 则不能为空,默认为 true

2.blankText:string//当为空时的错误提示信息

js 代码为:

```
var form1 = new Ext.form.FormPanel({
    width:350,
    frame:true,
    renderTo:"form1",
    labelWidth:80,
    title:"FormPanel",
    bodyStyle:"padding:5px 5px 0",
    defaults:{width:150,xtype:"textfield",inputType:"password"},
    items:[
        {fieldLabel:"不能为空",
        allowBlank:false,//不允许为空
        blankText:"不能为空,请填写",//错误提示信息,默认为 This field is required!
        id:"blanktest",
        anchor:"90%"
        }
    ]
});
```

## 2.用 vtype 格式进行简单的验证。

在此举邮件验证的例子，重写上面代码的 items 配置：

```
items:[
    {fieldLabel:"不能为空",
      vtype:"email",//email 格式验证
      vtypeText:"不是有效的邮箱地址",//错误提示信息,默认值我就不说了
      id:"blanktest",
      anchor:"90%"
    }
]
```

你可以修改上面的 vtype 为以下的几种 extjs 的 vtype 默认支持的验证：

//form 验证中 vtype 的默认支持类型

- 1.alpha //只能输入字母，无法输入其他（如数字，特殊符号等）
- 2.alphanum//只能输入字母和数字，无法输入其他
- 3.email//email 验证，要求的格式是"langsin@gmail.com"
- 4.url//url 格式验证，要求的格式是 http://www.\*\*\*

## 3.确认密码验证（高级自定义验证）

前面的验证都是 extjs 已经提供的验证，我们也可以自定义验证函数，比上面要复杂点了。我们一起做一个密码确认的例子。

我们修改前面的代码：

```
//先用 Ext.apply 方法添加自定义的 password 验证函数（也可以取其他的名字）
Ext.apply(Ext.form.VTypes,{
    password:function(val,field){//val 指这里的文本框值，field 指这个文本框
```

组件，大家要明白这个意思

```
if(field.confirmTo){//confirmTo 是我们自定义的配置参数，一般用来保存另外的组件的 id 值
```

```
var pwd=Ext.get(field.confirmTo);//取得 confirmTo 的那个 id 的值
```

```
return (val==pwd.getValue());
```

```
}
```

```
return true;
```

```
}
```

```
});
```

```
//配置 items 参数
```

```
items:[{fieldLabel:"密码",
```

```
id:"pass1",
```

```
anchor:"90%"
```

```
},{
```

```
fieldLabel:"确认密码",
```

```
id:"pass2",
```

```
vtype:"password",//自定义的验证类型
```

```
vtypeText:"两次密码不一致!",
```

```
confirmTo:"pass1",//要比较的另外一个的组件的 id
```

```
anchor:"90%"
```

```
}
```

关于 vtype 的内容还有很多内容要挖掘，但现在我们就点到这里为止，以后有机会再讨论它的其他高级验证。

不知不觉中写了这么多，大家都要歇息了，我们下次再接着讨论，

（因为本人近期考试和其他锁杂事情，近期可能更新较慢，还请大家海量，耐心，支持！）



前言：说句实话，此 extjs 系列的文章在博客园中的热度不高，可能是学这玩意的人不多吧，但是我觉得有这么个系列的文章对于中国朋友非常有帮助！请大家支持！

上篇 ExtJs2.0 学习系列(5)--Ext.FormPanel 之第二式中我们讨论了下 fieldset 和表单验证的知识，今天我们接着深入解析表元素中 ComboBox 组件的使用。会涉及到 .net 简单服务器数据交互，但暂不做深入讨论，以后会详细分析服务器交互相关，不过可能要等较长一段时间，呵呵！

## 5. 服务器数据作为 ComboBox 的数据源实例

首先从服务器获取 json 数据：

//cs 后台代码，简单起见，示例而已，要主要字符串格式(新手注意，下面的代码放在类里面，不是放在方法里)

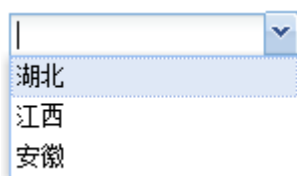
```
public string ServerData="['湖北','江西','安徽']";
```

//aspx 前台 js 介绍代码

```
Ext.onReady(function(){  
    var combo=new Ext.form.ComboBox({  
        store:<%=ServerData%>, //获取 ServerData 的 string 值，不要用"  
引起来，否则就不是 object 数据，而是字符串，这是一个很巧妙的关键点：把服务器的字符串转化为 js 的 object 数据，是不是超级方便。  
        emptyText: '请选择一个省份....',  
        applyTo: 'combo'  
    });  
});
```

//aspx 前台 html 代码

```
<input type="text" id="combo" size="20"/>
```



我们就通过<%=ServerData%>这样的方式获取到了服务器最简单的属性数据。问题来了，js 和 html 怎么调用 c#后台的变量和方法？（变量的调用上面刚刚介绍）

## 6.js 和 html 怎么调用 c#后台的变量和方法

关于这个话题，我不多说，网上很多讲解，在此仅简单说明

1.js 调用 c#后台变量，参考上面，注意，如果想获取 string 类型，请在 js 里用引号 var str="<%=ServerData%>"(返回"['湖北','江西','安徽']")

2.js 调用 c#后台方法：

<!--后台有一个方法：

```
public string ServerData()  
{  
    return "fdfdfdfdsf";  
}
```

前台代码：-->

```
<input id="Text2" type="text" value="<%=ServerData()%>" />
```

3.js 调用 c#后台带参数的方法

```
<!--public string ServerData(string pram)  
{  
    return pram+", 我是参数传进来的";  
}
```

主要是处理好 js 的引号问题，多尝试就会正确-->

```
<script>alert('<%=ServerData("谦虚的天下") %>');</script>
```

好了，现在我们有了解 js 获取后台数据的方法手段，不怕不怕啦，不过，这只是一小步。

## 7.ComboBox 的数据源 store 格式详解

在前面的例子里面，我们一直给 ComboBox 的数据源 store 赋值一维数组，其实 store 支持多维和 Store.data.Store 类型。

//下面就几种数据以代码举例说明

1.一维数组：["江西","湖北"]，值同时赋给 ComboBox 的 value 和 text

2.二维和多维数组：[["one","bbar","111"],["two","tbar","222"]],第一维和第二维分别赋值给 value 和 text，其他维忽略

3.store 类型：包括 GroupingStore, JsonStore, SimpleStore.

//我们分三步走：

//第一步：提供数据：

```
var data=[['湖北','hubei'],['江西','jiangxi'],['安徽','anhui'  
']];
```

```

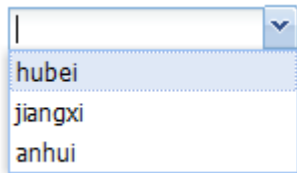
//第二步：导入到 store 中：

var store = new Ext.data.SimpleStore({
    fields: ['chinese', 'english'],
    data : data
});

//第三步：把 store 托付给 comboBox 的 store

var combo = new Ext.form.ComboBox({
    store: store,
    displayField: 'english', //store 字段中你要显示的字段, 多字段必选参数,
    //默认当 mode 为 remote 时 displayField 为 undefined, 当 select 列表时 displayField 为 "text"
    mode: 'local', //因为 data 已经取数据到本地了, 所以 'local', 默认为 "remote", 枚举完
    emptyText: '请选择一个省份...',
    applyTo: 'combo'
});

```



这里我们介绍了两个新的参数 displayField 和 mode，请记住，后面不再专门说明。

## 8. ComboBox 的 value 获取

添加 listeners 事件：

```

//ComboBox 的事件很多(api)，我们无法一一讲解，但是我们可以举一反三，select 事件
就是其中典型的一个

listeners: {
    "select": function() {
        alert(Ext.get("combo").dom.value); //获取 id 为 combo 的值
    }
}

//这里我们提供了一种不是很好的方法，在此不做过多停留

```



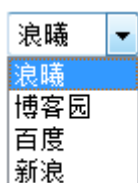
## 9.把 Extjs 的 ComboBox 样式应用到 select 的下拉框中去

核心参数介绍

transform:id//用于转换样式的，TimeField 作为 ComboBox 的子类也有此属性

核心代码：

```
//js 代码
var ExtSelect=new Ext.form.ComboBox({
    transform:"select",//html 中的 id
    width:80//宽度
});
//html 代码
<select id="select">
    <option value="1">***</option>
    <option value="2">博客园</option>
    <option value="3">百度</option>
    <option value="4">新浪</option>
</select>
//是不是超级简单？
```



从中不是也可以看出 extjs 的不同之处的，不过不明显！

## 10.ComboBox 的其他重要参数

```
1.valueField:"valuefield"//value 值字段
2.displayField: "field" //显示文本字段
3.editable: false//false 则不可编辑，默认为 true
4.triggerAction:"all"//请设置为"all",否则默认为"query"的情况下，你选择某个值
后，再此下拉时，只出现匹配选项，如果设为"all"的话，每次下拉均显示全部选项
5.hiddenName:string //真正提交时此 combo 的 name，请一定要注意
6.typeAhead:true, //延时查询，与下面的参数配合
7.typeAheadDelay:3000, //默认 250
//其他参数，请参考 api，或自行尝试
```

关于 combobox 的其他花俏功能在此不多做介绍。

最后一点，如何实现在 aspx 页面更灵活的分离 cs 数据和 js 数据的交互？因为我们都喜欢把 js 放在一个单独的文件，然后在 aspx 页面引用

这样就有个问题，我在 js 里直接获取 cs 数据就有点不方便。我认为可以这样，在 aspx 页面里获取数据，并作为 js，你就 js 变量，你就可

以在 js 里引用了，或者直接通过 url 地址获取。

之所以这么啰嗦的讲 combobox，是因为这个东西有时候真的让人又爱又恨！

下篇中我们继续讲解 form 中其他的表单元素！

N 久没有写 extjs 的，作为一个新手，我为我的这种懒惰行为感到惭愧！

鉴于有朋友反应前面的文章过于简单，我决定以后的文章如果没有闪光点就放在新手区(如果不适合，请跟帖)，不放在首页！

## 11.checkbox 简单示例

效果图：

checkbox简单示例

爱好: ☐ 足球 ☐ 篮球

js 代码：

```
Ext.onReady(function(){
    Ext_quickTips.init();
    var myform=new Ext.FormPanel({
        frame:true,
        width:330,
        layout:"form",
```

```

        labelWidth:30,
        title:"checkbox 简单示例",
        labelAlign:"left",
        renderTo:Ext.getBody(),
        items:[{
            xtype:"panel",
            layout:"column",//也可以是 table,实现多列布局
            fieldLabel:'爱好',
            isFormField:true,//非常重要,否则 panel 默认不显示 fieldLabel
            items:[{
                columnWidth:.5,//宽度为 50%
                xtype:"checkbox",
                boxLabel:"足球",//显示在复选框右边的文字
                name:" "
            },{
                columnWidth:.5,
                xtype:"checkbox",
                boxLabel:"篮球",
                name:" "
            }]
        }]
    });
});

```

关于多列布局，我们可以使用 column 或者 table 布局解决！

//其他几个参数

1.checked: true//true 则选中，默认为 false

2.name:"\*"//name 值

3.value:""//初始化值，默认为 undefine

## 12.radio 简单示例

基本上和 checkbox 一样，不过注意一组单选框必须 name 值相同，才能单选。

效果图：

### radio简单示例

性别: ☐ 男 ☒ 女

代码:

//基本同上, 不做过多解释

```
Ext.onReady(function(){
    Ext.QuickTips.init();
    var myform=new Ext.FormPanel({
        frame:true,
        width:330,
        layout:"form",
        labelWidth:30,
        title:"radio 简单示例",
        labelAlign:"left",
        renderTo:Ext.getBody(),
        items:[{
            xtype:"panel",
            layout:"column",
            fieldLabel:'性别',
            isFormField:true,
            items:[{
                columnWidth:.5,
                xtype:"radio",
                boxLabel:"男",
                name:"sex"
                //inputValue
            },{
                columnWidth:.5,
                checked:true,
                xtype:"radio",
                boxLabel:"女",
                name:"sex"
```

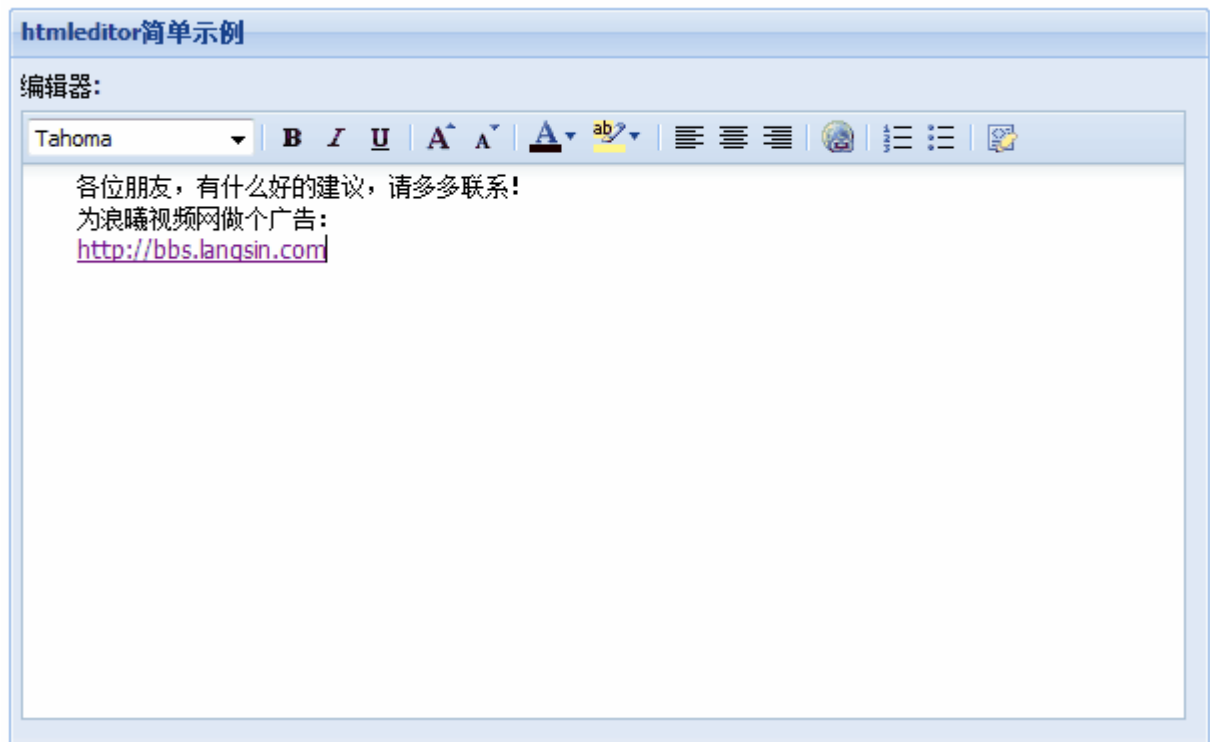
```

    }
}
});
});

```

### 13.htmleditor 简单示例

效果图:



js 代码:

```

//基本上同上
Ext.onReady(function(){
    Ext.QuickTips.init();
    var myform=new Ext.FormPanel({
        frame:true,
        width:600,
        layout:"form",
        labelWidth:50,
        title:"htmleditor 简单示例",
        labelAlign:"top",//items 中的标签的位置
    });
});

```



```

        renderTo:Ext.getBody(),
        items:[{
            xtype:"htmleditor",
            id:"he",
            fieldLabel:"编辑器",
            anchor:"99%"
        }]
    });
});

```

在这里我啰嗦个参数:

**//labelAlign 参数**

labelAlign:此参数是指 form 表单中 items 各项的 label 位置, 默认值为 left, 枚举值有 left,right,top

**//我见过有朋友认为此参数指 title 的位置, 是错误的!**

几个其他的参数:

**//补充几个参数**

1.hideLabel:true//默认为 false, 还适用于有标签的所有表单组件

**//下面的一组参数控制编辑器的工具栏选项, 都是默认值为 true**

2.enableColors: true//默认为 true, 显示字体颜色, 字体背景颜色

3.enableAlignments:true//左, 中, 右对齐

4.enableFont: true//字体

5.enableFontSize:false//字体大小, 就是 A 旁边有个小箭头的

6.enableFormat: false//粗体, 斜体, 下划线

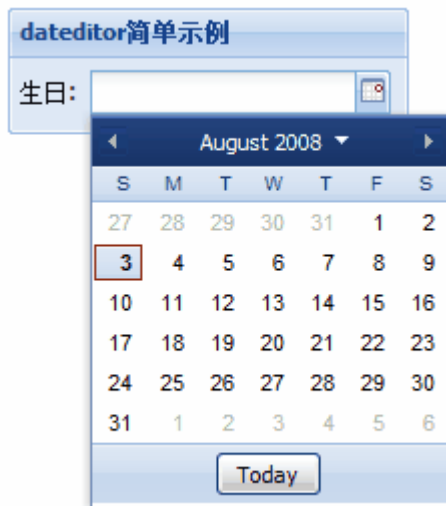
7.enableLinks:true//链接

8.enableLists: true//列表

9.enableSourceEdit: true//源代码编辑

#### 14.datefield 简单示例

效果图:

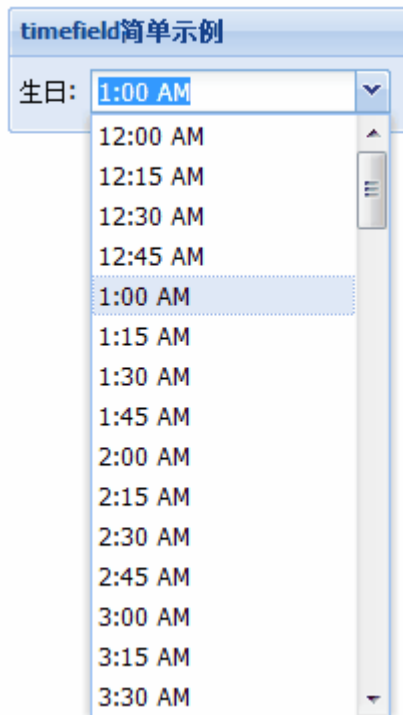


js 代码:

```
Ext.onReady(function(){
    Ext.QuickTips.init();
    var myform=new Ext.FormPanel({
        frame:true,
        width:200,
        layout:"form",
        labelWidth:30,
        title:"dateditor 简单示例",
        labelAlign:"left",
        renderTo:Ext.getBody(),
        items:[{
            xtype:"datefield",
            fieldLabel:"生日",
            anchor:"99%"
        }]
    });
});
```

### 15.timefield 简单示例

把上面的例子中 datefield 改为 timefield,效果图:



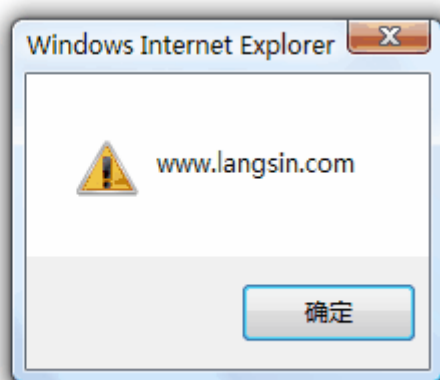
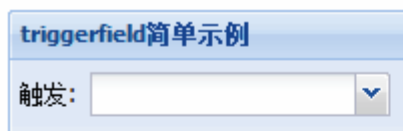
#### 16.numberfield 简单示例:

把上面的 datefield 改为 numberfield, 就只能输入数字了

#### 17.triggerfield 简单示例

说明: 它提供了一个触发的事件 onTriggerClick, datefield 和 combox 都是继承它

效果图: (点击右边下拉按钮)



js 代码:

```
Ext.onReady(function(){
    Ext.QuickTips.init();
    var myform=new Ext.FormPanel({
        frame:true,
```

```

width:200,

layout:"form",

labelWidth:30,

title:"triggerfield 简单示例",

labelAlign:"left",

renderTo:Ext.getBody(),

items:[{

    xtype:"trigger",

    fieldLabel:"触发",

    anchor:"99%",

    onTriggerClick:function(e){

        //在这里写你要实现的事件，很容易扩展

        alert("www.***");

    }

}]

});

});

```

好了，关于 form 的几个基本组件我们都蜻蜓点水的看了一遍，相信大家感性上知道是怎么回事啦！（总算快写完了 formpanel）

前面有朋友说要做个一行多个控件，中间有文字的那种 form 布局，谢谢支持！

下篇我们就做一个复杂点的 form 组件，还能提交服务器的综合示例！敬请期待！

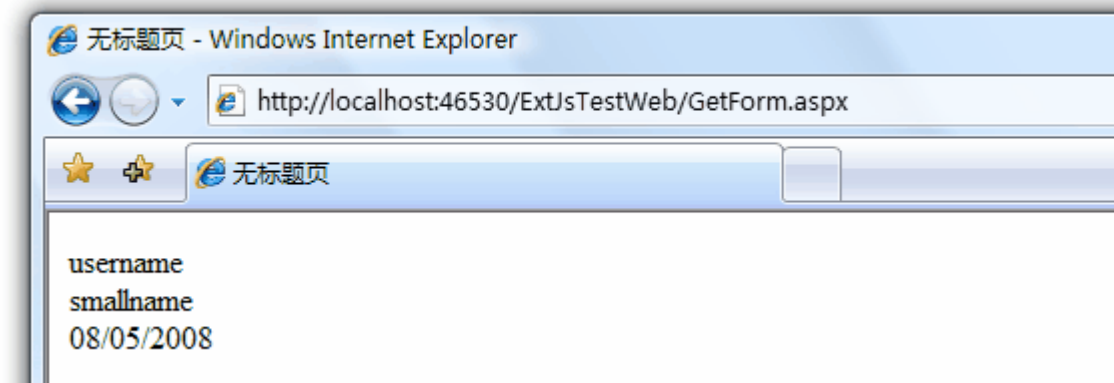
在上篇和前面的介绍中，我们基本上对 form 表单中常见组件有了简单的认识，今天我们做个综合点的例子，向服务器提交下！

其实这篇文章很简单，没有什么发光点，暂放首页半天,忘各位理解！

先来个简单的例子，以说明 formpanel 如何把数据传给其他页面。

效果图：

现在我们要实现的效果是：**点击确定，把值传到另一页面！，如下：**



原页面 js 代码为:

```
Ext.onReady(function(){
    Ext.QuickTips.init();
    var form=new Ext.FormPanel({
        frame:true,
        width:300,
        //monitorValid:true,//绑定验证
        layout:"form",
        labelWidth:70,
        title:"添加个人信息",
        labelAlign:"left",
        renderTo:Ext.getBody(),
        submit: function(){
            this.getEl().dom.action = 'GetForm.aspx',
            this.getEl().dom.method='POST',
            this.getEl().dom.submit();
        },
        items:[{
            xtype:"textfield",
            fieldLabel:"用户名",
            //id:"UserName",
            allowBlank:false,
            blankText:"不能为空，请正确填写",
```

```

        name: "UserName",
        anchor: "90%"
    }, {
        xtype: "textfield",
        fieldLabel: "昵称",
        //id: "SmallName",
        name: "SmallName",
        anchor: "90%"
    }, {
        xtype: "datefield",
        fieldLabel: "注册日期",
        //id: "RegDate",
        name: "RegDate",
        anchor: "90%"
    }],
    });

```

接受页面 GetForm.aspx 的 cs 代码为:

```

protected void Page_Load(object sender, EventArgs e)
{
    string UserName = Request.Form["UserName"];
    string SmallName = Request.Form["SmallName"];
    string RegDate = Request.Form["RegDate"];

    Response.Write(UserName + "<br/>" + SmallName + "<br/>" + Reg
Date);
}

```

因为很简单，我做个简要说明：

//几点说明

1. 首先定义 submit 参数的执行函数，即：

```

submit: function(){
    this.getEl().dom.action = 'GetForm.aspx', //转向页

```

面地址

```

        this.getEl().dom.method='POST', //方式
        this.getEl().dom.submit(); //提交!
    },

```

2. 为按钮添加触发相应的提交（取消）事件（这样就不是默认的 ajax 提交）：

```

        buttons:[{text:"确定",handler:login,formBind:true},{text:"取消",han
dler:reset}]
    });

    function login(){
        form.form.submit(); //提交
    }

    function reset(){
        form.form.reset(); //取消
    }

```

3. 如果你想绑定验证，在 form 表单添加参数 monitorValid:true, 然后在按钮配置参数中  
添加 formBind: true, 如

```

        buttons:[{text:"确定",handler:login,formBind:true},{text:"取消",
handler:reset}]

```

则只有所有的填写字段都满足条件时，"确定"方可提交！如下图，

好了，一个简单的 formpanel 的提交的原理弄清楚啦！

有关 form 提交数据的方法有多种，大家可以参考

<http://www.17ext.com/showtopic-55.aspx>（三种 ext 提交数据的方式），

以后有机会我们再讨论！

下面我们来做个复杂点（只是样子）的 form，示例一下（还是上面的原理）！

效果图：

添加个人信息

用户名: 谦虚的天下

性别: ☒ 男 ☐ 女

出生日期: 08/06/2008



学位: 请选择适合你的学历...

使用框架: ☒ Spring.net ☒ Nhibernate ☒ Linq

Email: 673592063@qq.com

个性签名: 理论是指导，学好理论至关重要！

想说的话: Tahoma

**B** *I* U <sup>A^</sup> <sub>A^</sub> A ab  

其实我没有什么话要说的，就是每天上上[浪曦视频网](#)和[博客园](#)，感觉学到了很多东西！偶尔上上其他的网站！谢谢支持！  

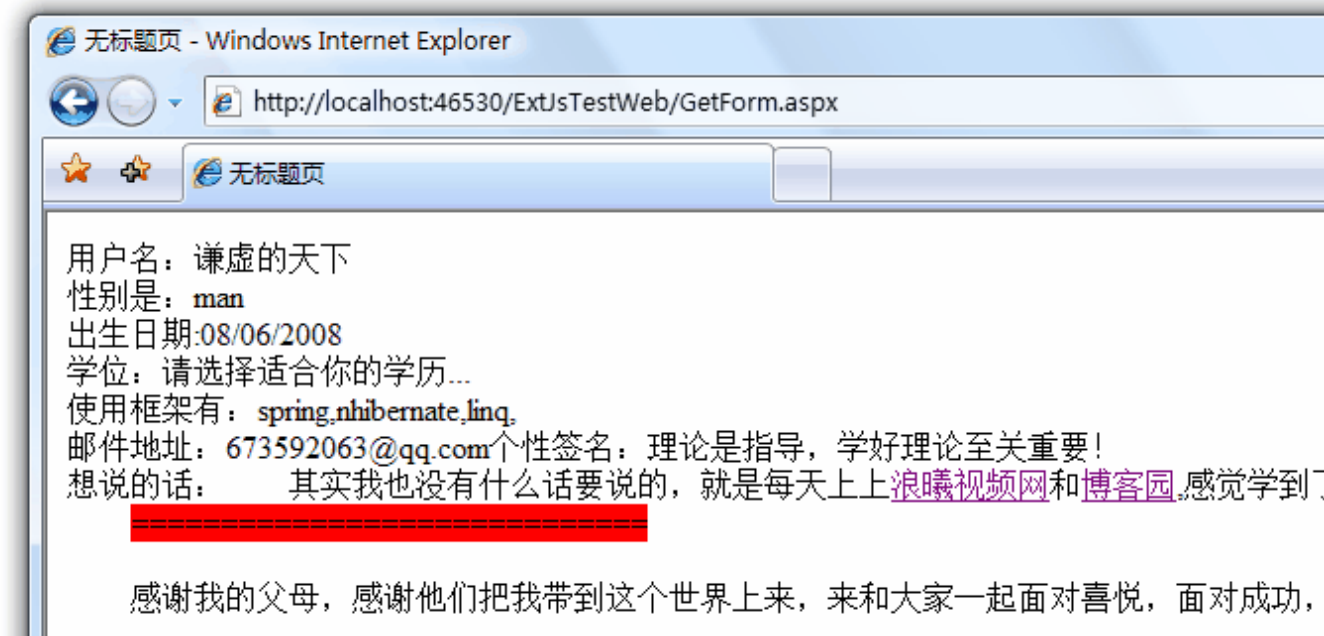
感谢我的父母，感谢他们把我带到这个世界上来，来和大家一起面对喜悦，面对成功，也面对挫折，面对失败，来和大家一起分享生活，分享学习，来和大家一起讨论extjs!

确定

取消



传到 GetForm.aspx 页面后显示:



不过在传过来的页面要记得 `ValidateRequest="false"`, 安全编码我就暂且不讨论了!

js 代码:

```
Ext.onReady(function(){
    Ext.QuickTips.init();
    var form=new Ext.FormPanel({
        frame:true,
        width:500,
        monitorValid:true,//把有 formBind:true 的按钮和验证绑定
        layout:"form",
        labelWidth:55,
        title:"添加个人信息",
        labelAlign:"right",
        renderTo:Ext.getBody(),
        submit: function(){
            this.getEl().dom.action = 'GetForm.aspx',
            this.getEl().dom.method='POST',
            this.getEl().dom.submit();
        },
    },
```

```

items:[{
    xtype:"panel",
    layout:"column",
    fieldLabel:"用户名",
    isFormField:true,
    items:[{
        columnWidth:.5,
        xtype:"textfield",
        allowBlank:false,
        blankText:"不能为空, 请填写",
        name:"UserName",
        anchor:"90%"
    },{
        columnWidth:.20,
        layout:"form",
        labelWidth:40,
        labelAlign:"right",
        items:[{
            xtype:"radio",
            fieldLabel:"性别",
            boxLabel:"男",
            name:"Sex",
            checked:true,
            inputValue:"man",//这里如果用 value,
            值是 on, 所以用 inputValue (出现这种情况的是 radio,checkbox)
            anchor:"95%"
        }]
    },{
        columnWidth:.30,
        layout:"form",
        labelWidth:1,//让标签宽度为很小的值 (奇怪的是为 0
        时反而不行)
    }
]

```

```

        items:[{
            xtype:"radio",
            boxLabel:"女",
            labelSeparator:"",//去除分隔符: "
            name:"Sex",
            inputValue:"woman",
            anchor:"95%"

        }]

    }],{ //上面是第一行

        xtype:"panel",
        layout:"column",
        fieldLabel:"出生日期",
        isFormField:true,
        items:[{

            columnWidth:.5,
            xtype:"datefield",
            name:"BirthDate",
            anchor:"90%"

        },{

            columnWidth:.5,
            layout:"form",
            labelWidth:40,//注意, 这个参数在这里可以调整简单 f
            fieldLabel 的布局位置

            items:[{

                xtype:"combo",
                name:"Degree",
                fieldLabel:"学位",
                store:["小学","初中","高中","专科","本科","硕士","博士"],

                emptyText:"请选择适合你的学历 ... ",

```

```

                                anchor:"90%"
                                ]]
                                ]]
                                },{//上面是第二行
                                xtype:"panel",
                                layout:"column",
                                isFormField:true,
                                fieldLabel:"使用框架",
                                items:[{
                                    columnWidth:.2,
                                    xtype:"checkbox",
                                    boxLabel:"Spring.net",
                                    name:"SpringNet",
                                    inputValue:"spring"//这里如果用 value, 值是 on,

```

所以用 inputValue

```

                                },{
                                    columnWidth:.2,
                                    layout:"form",
                                    labelWidth:1,
                                    items:[{
                                        xtype:"checkbox",
                                        boxLabel:"Nhibernate",
                                        labelSeparator:"",
                                        name:"NHibernate",
                                        inputValue:"nhibernate",
                                        anchor:"95%"
                                    ]]
                                },{
                                    columnWidth:.6,
                                    layout:"form",
                                    labelWidth:1,
                                    items:[{

```

```

        xtype: "checkbox",
        boxLabel: "Linq",
        labelSeparator: "",
        name: "Linq",
        inputValue: "linq",
        anchor: "95%"
    }
}
}

```

}, { // 上面是第三行

```

        xtype: "textfield",
        fieldLabel: "Email",
        name: "Email",
        vtype: "email", // email 验证, 如果想自定义验证的话, 请参见前面的

```

文章

```

        vtypeText: "email 格式错误!",
        anchor: "56%" // 控制文本框的长度

```

}, { // 上面是第四行

```

        xtype: "textarea",
        fieldLabel: "个性签名",
        name: "OneWord",
        height: 50,
        anchor: "95%"

```

}, { // 上面是第五行

```

        xtype: "htmleditor",
        fieldLabel: "想说的话",
        name: "WantToSay",
        anchor: "95%",
        enableAlignments: false, // 去除左右对齐工具栏
        enableLists: false // 去除列表工具栏

```

}],

```

        buttons:[{text:"确定",handler:login,formBind:true},{text:"取消",
handler:reset}]
    });
    function login(){
        form.form.submit();
    }
    function reset(){
        form.form.reset();
    }
});

```

接受 cs 页面的代码:

```

protected void Page_Load(object sender, EventArgs e)
{
    string UserName = Request.Form["UserName"];
    string Sex = Request.Form["Sex"];
    string BirthDate = Request.Form["BirthDate"];
    string Degree = Request.Form["Degree"];
    string SpringNet = Request.Form["SpringNet"];
    string NHibernate = Request.Form["NHibernate"];
    string Linq = Request.Form["Linq"];
    string Email=Request.Form["Email"];
    string OneWord = Request.Form["OneWord"];
    string WantToSay = Request.Form["WantToSay"];

    Response.Write("用户名: " + UserName + "<br/>");
    Response.Write("性别是: " + Sex + "<br/>");
    Response.Write("出生日期:" + BirthDate + "<br/>");
    Response.Write("学位: " + Degree + "<br/>");
    Response.Write("使用框架有: ");
    if (SpringNet != null)
    {

```

```

        Response.Write(SpringNet + ",");
    }
    if (NHibernate != null)
    {
        Response.Write(NHibernate + ",");
    }
    if (Linq != null)
    {
        Response.Write(Linq + ",");
    }

    Response.Write("<br/>");
    Response.Write("邮件地址: " + Email);
    Response.Write("个性签名: " + OneWord + "<br/>");
    Response.Write("想说的话: " + WantToSay);
}

```

经过一个简单的传值原理传值后，一个表单就可以把数据存储到数据库中去了！

//注意几点

- 1.绑定验证的两个参数 monitorValid:true,formBind:true
- 2.精确布局要注意的参数为和 width 有关的: width:500,labelWidth:55,columnWidth:5,anchor:"90%",isFormField:true 等
- 3.radio 和 checkbox 通过 inputValue 获取值，页面传值
- 4.多列多组件布局为 form 和 column 和 form 布局组合使用，请参考源码分析！

至此，formpanel 的简单使用就告一段落，但是 formpanel 的应用还远远没有讲到，有机会我们再到高级篇里讨论！

谢谢各位朋友的支持！

在下篇中我们接着诉说另外一个组件 tabpanel，希望各位支持，拍砖，给我动力！

最后，推荐一个网站：\*\*\*视频网

大家好,接着介绍 extjs 的基础吧,Tabpanel 组件大家喜欢吧!

(暂放首页 2 个小时.)

照旧,看个最简单的先,后面再详细说复杂的!

效果图片:



js 代码:

```
Ext.onReady(function(){
    var tabsDemo=new Ext.TabPanel({
        renderTo:Ext.getBody(),
        width:300,
        activeTab:0,//当前激活标签
        frame:true,
        items:[{
            contentEl:"tabOne",//标签页的页面元素 id(加入你想显示的话)
            title:"***",
            closable:true//是否现实关闭按钮,默认为 false
        },{
            contentEl:"tabTwo",
            title:"博客园兄弟们可好"
        }]
    });
});
```

html 代码:

```
<body style="margin:10px;">
    <div>
        <div id="tabOne" class="x-hide-display">i am tabOne!</div>
        <div id="tabTwo" class="x-hide-display">i am tabTwo!</div>
    </div>
</body>
<!--注意 class 类型,设为 x-hide-display,以正常显示-->
```

在这里例举几个参数:

```
//几个相关参数
1.tabPosition:"bottom"//选项卡的位置,枚举值 bottom,top.默认为 top(只有 top 的
```



时候才能选项卡的滚动!)

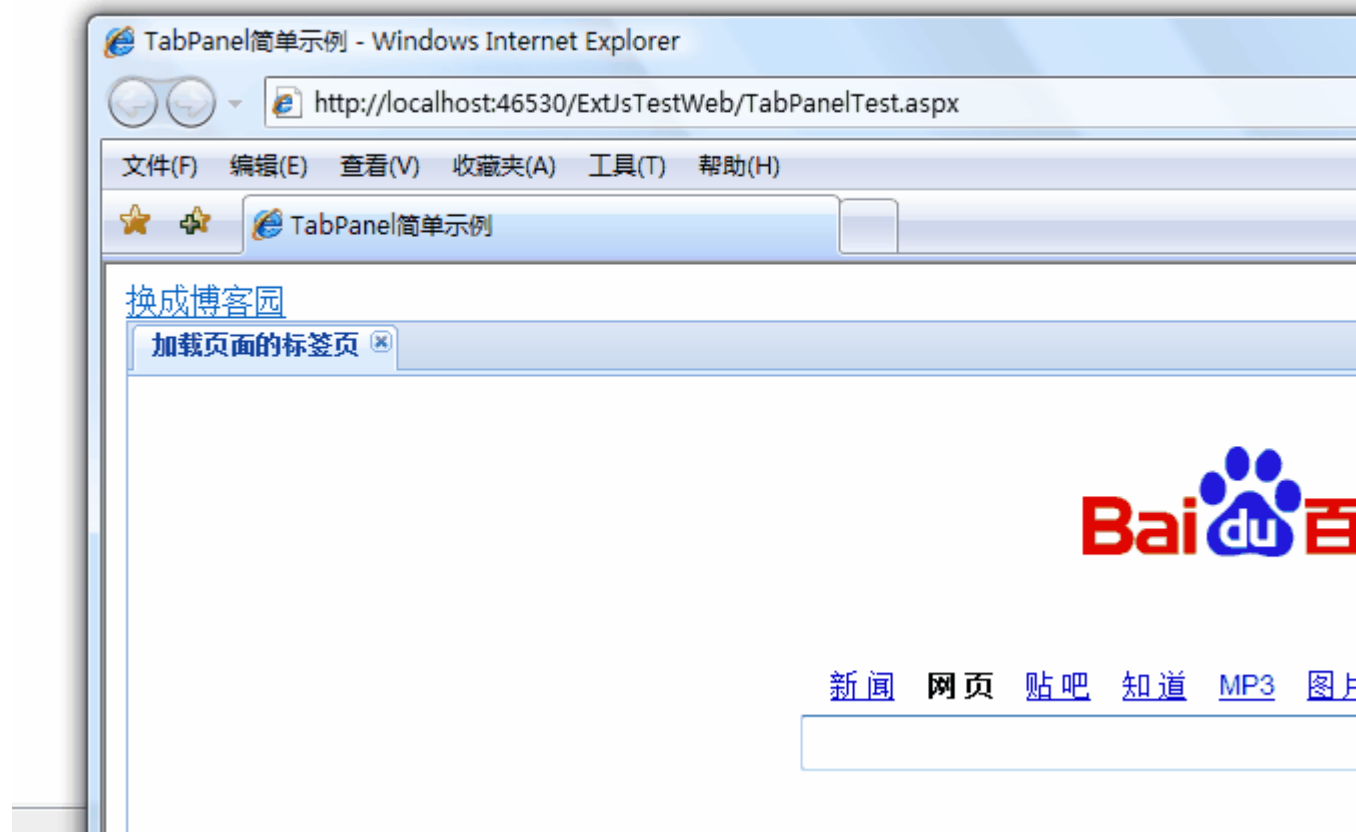
2.tabTip:"提示"//必须先调用 `Ext.QuickTips.init();`才有效果

经常我们有这么个情况,一个选项卡加载一个页面,这里我提供一种不是很好但是很稳定的简单方法(已经在项目中验证没有出现问题).

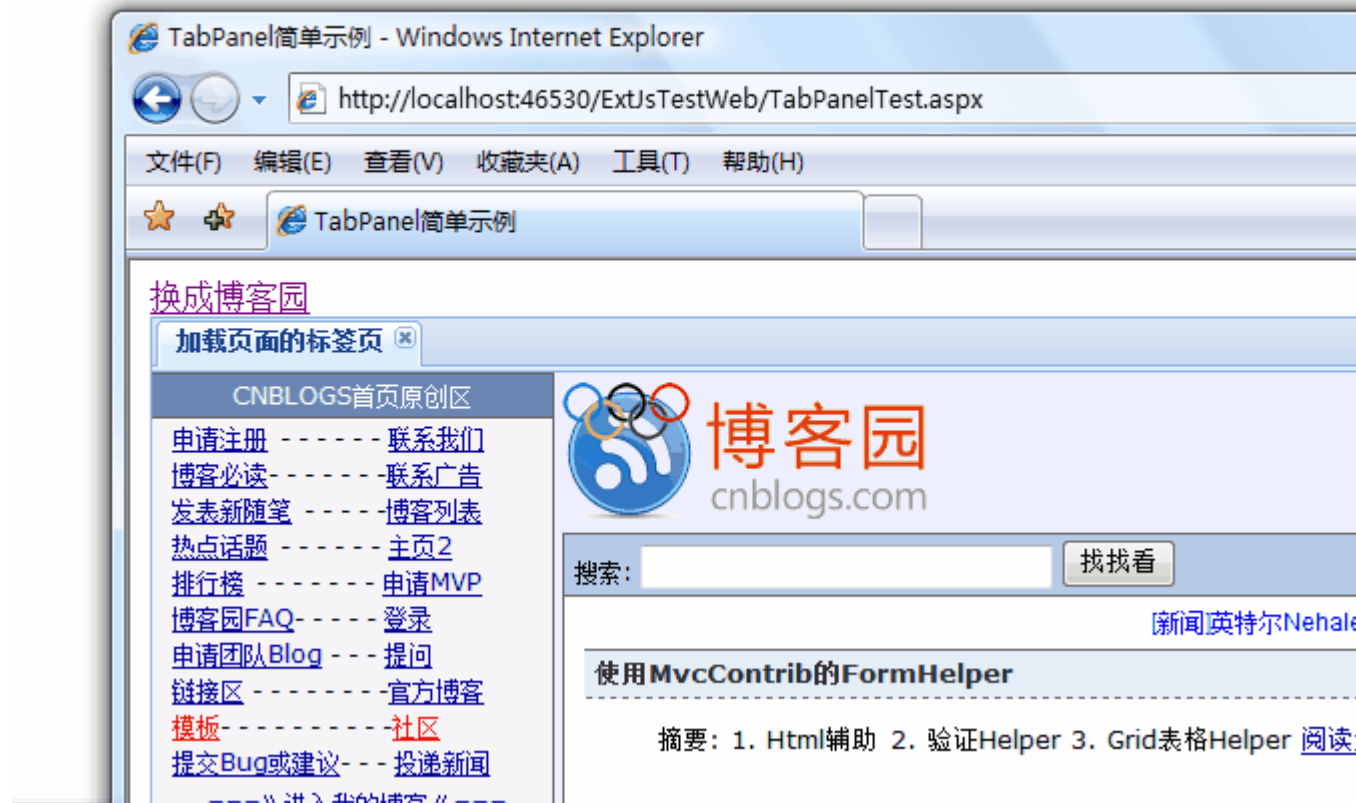
就是:使用 **iframe** 作为 **tab** 的标签页内容.

## 1.用 **iframe** 加载其他完整页面的 **tabpanel**.

效果图:



点击链接"换成博客园",



html 代码:

```
<body style="margin:10px;">
    <div>
        <a href="javascript:void(0)" onclick="parent.frames['mainFrame'].location='http://www.cnblogs.com'">换成博客园</a>
        <iframe id="mainFrame" name="mainFrame" src="http://www.baidu.com" frameborder="0" height="100%" width="100%" style="overflow:hidden;"></iframe>
    </div>
</body>
```

js 代码:

```
Ext.onReady(function(){
    var tabsDemo=new Ext.TabPanel({
        renderTo:Ext.getBody(),
        activeTab:0,
```

```

        height:700,//当用 viewport 布局时,这个 height 可以省去

        frame:true,

        items:[{

            contentEl:"mainFrame",

            tabTip:"fengjian",

            title:"加载页面的标签页",

            closable:true

        }]

    });
});

```

今天的内容简单.就暂且说做到了这里,下篇中我们说说 tabpanel 的滚动标签和动态添加标签的 tabpanel!

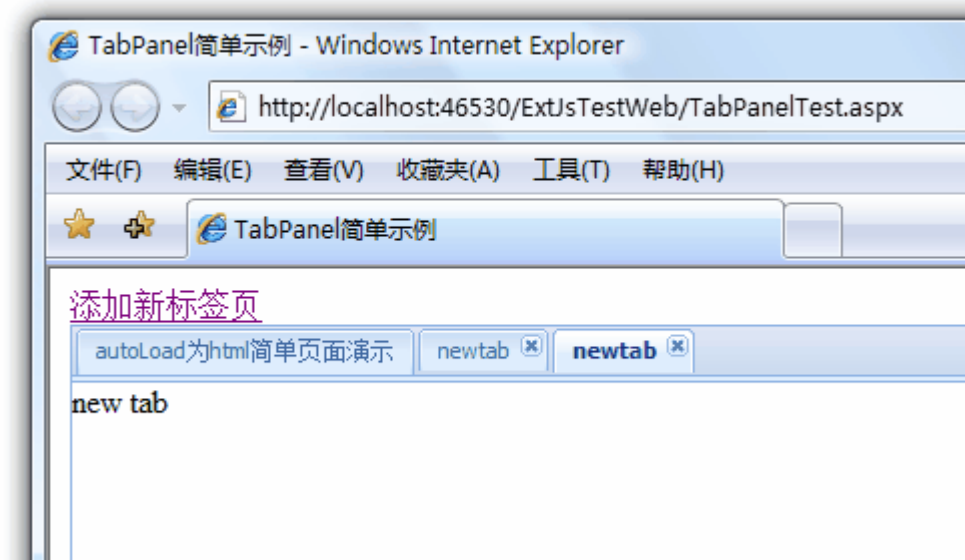
敬请期待!

上一篇种我们简单的了解了下 tabpanel

下面我们要介绍的是,如何动态的添加标签页!

## 2.动态添加 tabpanel 的标签页

效果图:



点击"添加新标签页",会添加一个标签页,而且激活这个新的标签页.

html 代码:

```
<body style="margin:10px;">

    <div>

        <a id="AddNewTab" href="javascript:void(0)">添加新标签页</a>

    </div>

</body>
```

js 代码:

```
Ext.onReady(function(){

    Ext_quickTips.init();

    var tabsDemo=new Ext.TabPanel({

        renderTo:Ext.getBody(),

        activeTab:0,

        height:700,

        frame:true,

        items:[{

            title:"autoLoad 为 html 简单页面演示",

            autoLoad:{url:"tab1.htm",scripts:true}

        }]

    });

    //下面是添加新标签页的关键代码,很简单方便

    var index=0;

    Ext.get("AddNewTab").on("click",function(){

        tabsDemo.add({

            title:"newtab",

            id:"newtab"+index,

            html:"new tab",

            closable:true

        });

        tabsDemo.setActiveTab("newtab"+index);

        index++;

    })

});
```

简单说明:

其实添加的话,只要 add() 方法就可以了,但是我们还要激活这个新的标签页,就必须 set ActiveTab(newtab 的索引或 id),关键就是我们不好判断这个索引,所以只好设置个递增的全局变量 index 来给 newtab 取名,这样我们也能准确的获取新的不重复的 newtab 了,也就容易激活了。而且我们可以通过下图看出来。

```
<div id="ext-comp-1003" class="x-panel x-panel-noborder x-hide-  
display" style="width: 956px;">  
<div id="newtab0" class="x-panel x-panel-noborder x-hide-display" style="width:  
956px;">  
<div id="newtab1" class="x-panel x-panel-noborder x-hide-display" style="width:  
956px;">  
<div id="newtab3" class="x-panel x-panel-noborder" style="width: 939px;">
```

### 3.稍微修改上面的例子 tabpanel(官方的例子)

效果图:



我就不多说了, 关键的几个参数注释了下

```
<body style="margin:10px;">  
  <div>  
    <div id="AddBtn"></div>  
  </div>  
</body>
```

js 代码:

```
Ext.onReady(function(){  
  Ext.QuickTips.init();  
  var tabsDemo=new Ext.TabPanel({  
    renderTo:Ext.getBody(),
```

```
        //resizeTabs:true,宽度能自动变化,但是影响标题的显示
        activeTab:0,
        height:200,
        enableTabScroll:true,//挤的时候能够滚动收缩
        width:200,
        frame:true,
        items:[{
            title:"tab advantage",
            html:"sample1"
        }]
    });
```

```
var index=0;
```

//就是下面这个函数,关键的地方,非常简单也非常实用

```
function addTab()
{
    tabsDemo.add({
        title:"newtab",
        id:"newtab"+index,
        html:"new tab"+index,
        closable:true
    });
    tabsDemo.setActiveTab("newtab"+index);
    index++;
}
```

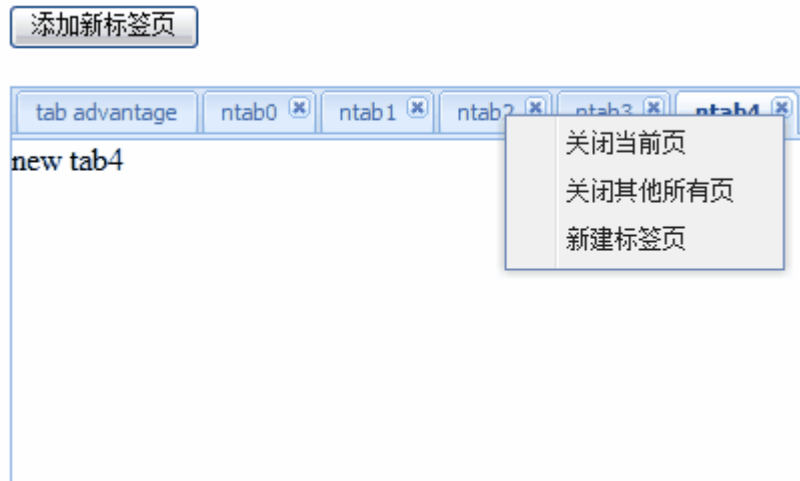
//设置一个按钮(上面的是一个链接,应用有点不同哦)

```
new Ext.Button({
    text:"添加新标签页",
    handler:addTab
```

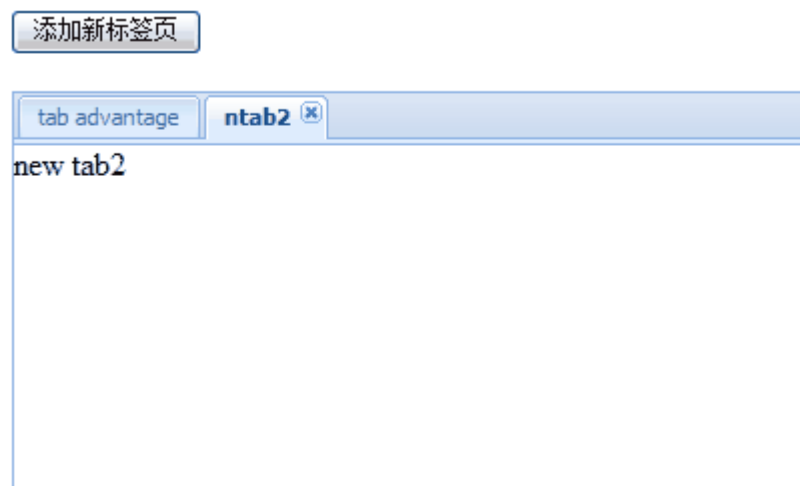
```
}).render(document.body, "AddBtn");  
});
```

#### 4.为 tabpanel 标签页添加右键菜单

效果图:



点击"关闭其他所有页"后,



其他两个右键菜单还是道理相同.

//几个参数说明

1.enableTabScroll:true//前面已经说过了

2. listeners:{"contextmenu":function(参数1,参数2,参数3){...}}

//右键菜单事件, 三个参数分别为当前 tabpanel, 当前标签页 panle, 时间对象 e

3.//扩充 2, 每个标签页都有激活和去激活事件

activate 和 deactivate, 他们的执行函数有个参数, 就是当前标签页。

例如: items:[{

```

        title:"tab advantage",
        listeners:{
            deactivate:function(a){alert("删除,a表示当前标签页");},
            activate:function(){alert("激活");}
        },
        html:"sample1"
    }]

```

4.menu=new Ext.menu.Menu();//menu 组件，就不多说了，后面会专门分析下，不过不要忘记 menu.showAt(e.getPoint());了

html 代码和上面的例子的 html 代码一样.

js 代码:

```

Ext.onReady(function(){
    Ext_quickTips.init();
    var tabsDemo=new Ext.TabPanel({
        renderTo:Ext.getBody(),
        //resizeTabs:true,宽度能自动变化,但是影响标题的显示
        activeTab:0,
        height:200,
        enableTabScroll:true,//挤的时候能够滚动收缩
        width:400,
        frame:true,

        //下面是比上面例子新增的关键右键菜单代码
        listeners:{
            //传进去的三个参数分别为:这个 tabpanel(tabsDemo),当前
            //标签页,事件对象 e
            "contextmenu":function(tdemo,myitem,e){
                menu=new Ext.menu.Menu([
                    {
                        text:"关闭当前页",
                        handler:function(){

```



```

        tdemo.remove(myitem);
    }
}, {
    text: "关闭其他所有页",
    handler: function () {
        //循环遍历
        tdemo.items.each(function
(item){
            if(item.closable&&item!=myitem)
            {
                //可以关闭的其他所有
                标签页全部关掉
                tdemo.remove(item);
            }
        });
    }
}, {
    text: "新建标签页",
    handler: addTab
});
//显示在当前位置
menu.showAt(e.getPoint());
}
},

items:[{
    title: "tab advantage",
    html: "sample1"
}]
});

```

```

var index=0;

function addTab()
{
    tabsDemo.add({
        title:"ntab"+index,
        id:"newtab"+index,
        html:"new tab"+index,
        closable:true
    });

    tabsDemo.setActiveTab("newtab"+index);

    index++;
}

new Ext.Button({
    text:"添加新标签页",
    handler:addTab
}).render(document.body,"AddBtn");
});

```

关于 tabpanel 的简单使用就说到了这里.

XTemplate 是 Extjs 里面的模板组件.

下面我们看个最简单的例子.

效果图:

XtemplateData简单示例
名称:博客园 编号:1,书:<<道不远人>>,日期:2007-7-7 编号:2,书:<<大话设计模式>>,日期:2006-6-6

js 代码:

```

Ext.onReady(function(){
    //数据源
    var data={
        name:"博客园",

```

```

        read:[{
            book:'<<道不远人>>',
            date:'2007-7-7'
        },{
            book:"<<大话设计模式>>",
            date:"2006-6-6"
        }]
    }
}

//呈现组件
var mypanel=new Ext.Panel({
    width:400,
    id:"mypanel",
    title:"XtemplateData 简单示例",
    renderTo:Ext.getBody()
});

//创建模板
var tpl=new Ext.XTemplate(
    '<table><tr><th>名称:{name}</th></tr>',
    '<tr><td>',
    '<tpl for="read">',
        '<p>编号:{#},书:{book},日期:{date}</p>',
    '</tpl></td></tr></table>'
);

//重写绑定模板
tpl.overwrite(mypanel.body,data);
})

```

简要说明:

```

/*
var tpl=new Ext.XTemplate(
    '<table><tr><th>名称:{name}</th></tr>',
    '<tr><td>',
    '<tpl for="read">',

```

```

        '<p>编号:{#},书:{book},日期:{date}</p>',
        '</tpl></td></tr></table>'
    );

```

```
tpl.compile();
```

```
    tpl.overwrite(mypanel.body,data);
```

```
*/
```

1.tpl.compile();//可以在创建模板后,添加 tpl.compile();编译代码,速度快点.

2. tpl.overwrite(mypanel.body,data);//把数据填充到模板中去,并呈现到目标组件

3.名称:{name} //对于一维单数据对象,直接用{名称}输出,

4.,<tpl for="read"> //对于多维对象(如拥有多条数据的表),使用 tpl 和 for 配合使用,会使用 tpl 的格式把数据一条一条输出,read 为上级节点

5.{.} //对于一维对数据的对象,如 color: ['Red', 'Blue', 'Black'],可以用{.}按照 tpl 模板逐一输出,如:

```

    '<tpl for="color">',
        '<div> {.}</div>',
    '</tpl>'

```

6.{#} //表示循环的索引

7.parent.\*\*\* //在子对象中访问父对象元素,使用 parent,如:{parent.name}

8.if // '<tpl if="age > 1">',

```
'<p>{name}</p>',
```

```
'</tpl>',
```

//if 实现有条件的逻辑判断,很容易使用

9.其他几个常用的参数:

xindex //循环模板的当前索引 index (从 1 开始),用[]。

xcount //循环模板循环的次数。 用[]

举例:

```
'<tpl for="read">',
```

```
'<p>编号:{#},书:{book},日期:{date},奇偶:{[xindex%2==0?"偶数
```

```
":"奇数"}},次数:[xcount]}</p>',
```

```
'</tpl>
```

10.模板成员函数(借用 api 下):

```
var tpl = new Ext.XTemplate(
```

```

    '<tpl for="kids">',
    '<tpl if="this.isGir1(name)">',
    '    <p>Gir1: {name} - {age}</p>',
    '</tpl>',
    '<tpl if="this.isGir1(name) == false">',
    '    <p>Boy: {name} - {age}</p>',
    '</tpl>',
    '</tpl></p>', {
    isGir1: function(name){
        return name == 'Sara Grace';
    },
    isBaby: function(age){
        return age < 1;
    }
});

```

接下来,我们做个服务器的例子(好像很多朋友对这个要求很强烈啊)

实例演示:用模板呈现服务器数据

效果图:

编号	名称	地址	位置
1	博客园	<a href="http://www.cnblogs.com">http://www.cnblogs.com</a>	1
2	浪曦视频网	<a href="http://bbs.langsin.com">http://bbs.langsin.com</a>	3
3	实验室	<a href="http://www.shiyanshi.com">http://www.shiyanshi.com</a>	4
4	大活二楼	<a href="http://www.langsin.com">http://www.langsin.com</a>	5
5	湖北日报	<a href="http://www.baidu.com">http://www.baidu.com</a>	6
6	学生工作处	<a href="http://www.cnblogs.com">http://www.cnblogs.com</a>	2

html 代码:

```

<div id="container">
    </div>

```

css 代码:

```

<style type="text/css">
    body
    {
        font-size:12px
    }

```

```

    }
    #container
    {
        border:1px solid black;
        width:330px;
    }
    td,th
    {
        border-bottom:1px dashed black;
    }
    th
    {
        text-align:center;
    }
    .namewidth
    {
        width:120px;
    }
    .urlwidth
    {
        width:150px;
    }
</style>

```

js 代码:

```

Ext.onReady(function(){

    var mydata;

    Ext.Ajax.request({
        url:"getXtemplateData.ashx",//服务器端地址
        success:function(request){
            mydata=request.responseText;//服务器端文本数据
            mydata=eval('(' +mydata+')');//使用 eval 把文本数据转换为 json

```

对象

```
//或者用 extjs 自带的方法:mydata=Ext.util.JSON.decode(mydata),效果相同

var tpl2=new Ext.XTemplate(
    '<table><thead><tr><th>编号</th><th class="namewidth">名称</th><th class="urlwidth">地址</th><th>位置</th></tr></thead><tbody>' ,
    '<tpl for="results">' ,
    '<tr><td>{#}</td><td>{linkname}</td><td>{linkurl}</td><td>{linkpos}</td><tr>' ,
    '</tpl></tbody></table>'
);
tpl2.compile();
tpl2.overwrite(Ext.get("container"),mydata);

},
failure:function()
{
    alert("failure!");
}
});
})

/**简单说明**
1.Ext.Ajax.request(),这里暂且对 ajax 不多谈论,后面会详细叙述
2.eval 用"()"可以把规范文本转换为 json 对象,很重要!mydata=eval('(' +mydata+' ')');
3.如果我们把模板创建和绑定放到 ajax 外面,会出错,因为 ajax 为异步调用,记住哦~
4.关于 success 函数的 request 参数,我截个图看看,就明白了
*/
```

request	Object tId=0 status=200 statusText=OK
argument	undefined
getAllResponseHeaders	"Server: ASP.NET Development Server/9.0.0.0\r\nDate: Mon, 18 Aug 2008 18:30:00\r\nVersion: 2.0.50727\r\nCache-Control: private\r\nContent-Type: text/html; charset=utf-8\r\nLength: 467\r\nConnection: Close\r\n"
getResponseHeader	Object Server=ASP.NET Development Server/9.0.0.0
responseText	"{results:[{id:'1',linkname:'博客园',linkurl:'http://www.cnblogs.com',linkpos:1},{id:'2',linkname:'浪曦视频网',linkurl:'http://bbs.langsin.com',linkpos:3},{id:'3',linkname:'实验室',linkurl:'http://www.shiyanshi.com',linkpos:4},{id:'5',linkname:'大活二楼',linkurl:'http://www.langsin.com',linkpos:5},{id:'6',linkname:'湖北日报',linkurl:'http://www.baibao.com',linkpos:6},{id:'8',linkname:'学生工作处',linkurl:'http://www.cnblogs.com',linkpos:2}]}"
responseXML	null
status	200
statusText	"OK"
tId	0

先看看数据库数据:

	id	linkname	linkurl	linkpos
▶	1	博客园	http://www.cn...	1
	2	浪曦视频网	http://bbs.lan...	3
	3	实验室	http://www.shi...	4
	5	大活二楼	http://www.la...	5
	6	湖北日报	http://www.ba...	6
	8	学生工作处	http://www.cn...	2
*	NULL	NULL	NULL	NULL

字段,数据都很清楚.下面是服务器 getXTemplateData.ashx 的代码:

```
<%@ WebHandler Language="C#" Class="getXTemplateData" %>

using System;
using System.Web;
using System.Data;
using System.Data.SqlClient;
using System.Text;

public class getXTemplateData : IHttpHandler {

    public void ProcessRequest (HttpContext context) {

        StringBuilder tempResult = new StringBuilder("{results:[" );
```



```

        string connstr = @"Data Source=WIN-7JW7UWR0M27\MSSQL2005;Initial Catalog=xgc;Persist Security Info=True;User ID=sa;Password=*****
*****";

        SqlConnection sqlconn = new SqlConnection(connstr);

        SqlCommand comm = new SqlCommand("select * from xLink", sqlconn);

        sqlconn.Open();

        SqlDataReader sdr = comm.ExecuteReader();

        while (sdr.Read())
        {
            tempResult.Append("{id:");

            tempResult.Append((int)sdr["id"]);

            tempResult.Append(",linkname:");

            tempResult.Append((string)sdr["linkname"]);

            tempResult.Append(",linkurl:");

            tempResult.Append((string)sdr["linkurl"]);

            tempResult.Append(",linkpos:");

            tempResult.Append((int)sdr["linkpos"]);

            tempResult.Append("},");
        }

        tempResult.Remove(tempResult.Length - 1, 1); //去掉多余的那个逗号
        tempResult.Append("]}");

        context.Response.Write(tempResult); //输出
    }

    public bool IsReusable {
        get {
            return false;
        }
    }
}

```

```
}
```

今天对 XTemplate 做了个简单介绍,下篇我们开始 TreePanel 的学习讨论~!

最后推荐个网站:\*\*\*视频网 ,谢谢支持!

=====补充=====

我们在上面对基础上作个查询:

html 代码为:

```
<div id="head"><input id="linkid" type="text" /></div>

<div id="container">

</div>
```

js 代码在上面 js 的基础上添加下面部分:

```
new Ext.Button( {
    text: "查询大于指定 id 值的记录",
    handler: function() {
        Ext.Ajax.request( {
            url: "getXtemplateData.ashx?id="+Ext.get("linkid").dom.v
            alue, //获取文本框的值

            success: function(request) {
                mydata=request.responseText;
                mydata=eval('('+mydata+')');
                tpl2.overwrite(Ext.get("container"),mydata);
            },
            failure: function()
            {
                alert("failure!");
            }
        });
    }
}).render(document.body, "head");
```

服务器 GetTemplateData.ashx 代码也要稍微修改下:

//主要代码

```
public void ProcessRequest (HttpContext context) {
    string sql="";
    if (context.Request.QueryString["id"] != null)
    {
        sql = "select * from xLink where id>"+Convert.ToString(context.Request.QueryString["id"]);
    }
    else
    {
        sql="select * from xLink";
    }
    StringBuilder tempResult = new StringBuilder("{results:["");

    string connstr = @"Data Source=WIN-7JW7UWR0M27\MSSQL2005;Initial Catalog=xgc;Persist Security Info=True;User ID=sa;Password=673592063@qq.com";

    SqlConnection sqlconn = new SqlConnection(connstr);
    SqlCommand comm = new SqlCommand(sql, sqlconn);
    sqlconn.Open();
    SqlDataReader sdr = comm.ExecuteReader();
    while (sdr.Read())
    {
        tempResult.Append("{id:");
        tempResult.Append((int)sdr["id"]);
        tempResult.Append(",linkname:");
        tempResult.Append((string)sdr["linkname"]);
        tempResult.Append(",linkurl:");
        tempResult.Append((string)sdr["linkurl"]);
        tempResult.Append(",linkpos:");
        tempResult.Append((int)sdr["linkpos"]);
        tempResult.Append("},");
    }
}
```

```

    }

    tempResult.Remove(tempResult.Length - 1, 1);

    tempResult.Append("] }");

    context.Response.Write(tempResult);

}

```

效果图:

查询大于指定id值的记录			
6			
编号	名称	地址	位置
8	学生工作处	http://www.cnblogs.com	2

查询大于指定id值的记录			
4			
编号	名称	地址	位置
5	大活二楼	http://www.langsin.com	5
6	湖北日报	http://www.baidu.com	6
8	学生工作处	http://www.cnblogs.com	2

代码有点乱!

今天开始,我们就开始一起学习 TreePanel 了, 道个歉, 上篇的代码很乱阿.

我总是喜欢用最简单的例子开始,去理解最基本的使用方法,减少对 i 后面高级使用的干扰!

TreePanel 是继承自 Panel,所以很多在 Panel 中谈到的属性这里可能会一笔带过,如有问题,请参考 ExtJs2.0 学习系列(2)--Ext.Panel

### 1.第一个静态树--最简单的树

效果图:



html 代码:

```

<div id="container">

</div>

```

js 代码:

```

Ext.onReady(function() {

    var mytree=new Ext.tree.TreePanel( {

```

```

    el:"container",//应用到的html 元素 id
    animate:true,//以动画形式伸展,收缩子节点
    title:"Extjs 静态树",
    collapsible:true,
    rootVisible:true,//是否显示根节点
    autoScroll:true,
    autoHeight:true,
    width:150,
    lines:true,//节点之间连接的横竖线
    loader:new Ext.tree.TreeLoader(),//
    root:new Ext.tree.AsyncTreeNode({
        id:"root",
        text:"根节点",//节点名称
        expanded:true,//展开
        leaf:false,//是否为叶子节点
        children:[{text:'子节点一',leaf:true},{id:'child2',text:'子
节点二',children:[{text:"111"}]}]}
    })
    });
    mytree.render();//不要忘记 render() 下,不然不显示哦
})

```

在这里,我谈一个问题:

*/\*只有 loader 和 AsyncTreeNode 才能使 children 显示出来,为什么?\*/*

我在 api 中没有找到答案,甚至连 children 都没有看到,但是在原代码中,我们可以确定这个事实,只有 loader 实例后,AsyncTreeNode 的 children 才会被递归的添加(appendChild)到它的父节点下,所以象示例中的代码,一定要注意条件.

其他的子节点问题不受此限制!

TreePanel 基本配置参数:

*//TreePanel 配置参数*

1.animate:true//展开,收缩动画,false 时,则没有动画效果

2.autoHeight:true//自动高度,默认为 false

3.enableDrag:true//树的节点可以拖动 Drag(效果上是),注意不是 Draggable

```
4.enableDD:true//不仅可以拖动,还可以通过 Drag 改变节点的层次结构(drap 和 drop)
5.enableDrop:true//仅仅 drop
6.lines:true//节点间的虚线条
7.loader:Ext.tree.TreeLoader//加载节点数据
8.root:Ext.tree.TreeNode//根节点
9.rootVisible:false//false 不显示根节点,默认为 true
10.trackMouseOver:false//false 则 mouseover 无效果
11.useArrows:true//小箭头
```

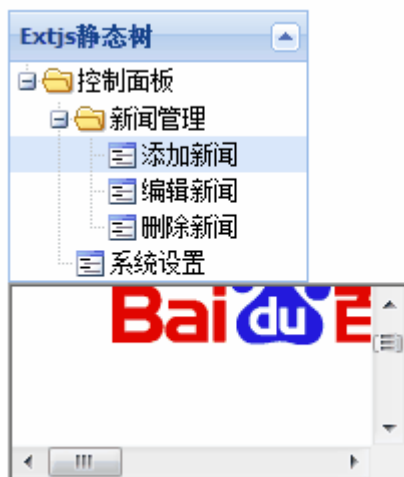
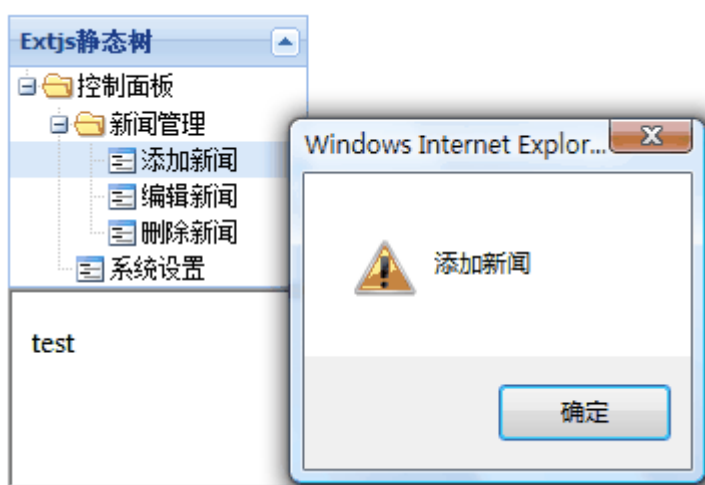
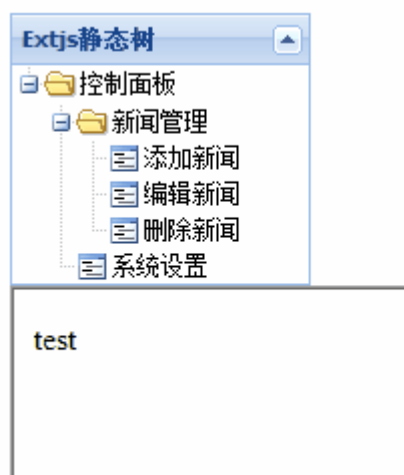
## 2.通过 TreeNode 自定义静态树

例子 1 其实很受数据的限制,必须先要准备好数组对象,我们用另外的方式再写个例子,在写例子前,我们来学习下

TreeNode 的基本配置参数:

```
//TreeNode 常用配置参数
1.checked:false//true 则在 text 前有个选中的复选框,false 则 text 前有个未选中的复选框,默认没有任何框框
2.expanded:false//展开,默认不展开
3.href:"http://www.cnblogs.com"//节点的链接地址
4.hrefTarget:"mainFrame"//打开节点链接地址默认为 blank,可以设置为 iframe 名称 id,则在 iframe 中打开
5.leaf:true//叶子节点,看情况设置
6.qtip:"提示"//提示信息,不过要 Ext.QuickTips.init();下
7.text:"节点文本"//节点文本
8.singleClickExpand:true//用单击文本展开,默认为双击
```

效果图:



html 代码:

```
<body style="margin:10px;">
  <div id="container">
    </div>
```

```
<iframe name="mainFrame" id="mainFrame" height="100px" width="200px" src="jstest.htm"></iframe>
</body>
```

js 代码:

```
Ext.onReady(function(){
    Ext.QuickTips.init();
    var mytree=new Ext.tree.TreePanel({
        el:"container",
        animate:true,
        title:"Extjs 静态树",
        collapsible:true,
        enableDD:true,
        enableDrag:true,
        rootVisible:true,
        autoScroll:true,
        autoHeight:true,
        width:150,
        lines:true
    });

    //根节点
    var root=new Ext.tree.TreeNode({
        id:"root",
        text:"控制面板",
        expanded:true
    });

    //第一个子节点及其子节点
    var sub1=new Ext.tree.TreeNode({
        id:"news",
        text:"新闻管理",
```



```
        singleClickExpand:true
    });
    sub1.appendChild(new Ext.tree.TreeNode({
        id:"addNews",
        text:"添加新闻",
        href:"http://www.baidu.com",
        hrefTarget:"mainFrame",
        qtip:"打开百度",
        listeners:{//监听
            "click":function(node,e){
                alert(node.text)
            }
        }
    }));
    sub1.appendChild(new Ext.tree.TreeNode({
        id:"editNews",
        text:"编辑新闻"
    }));
    sub1.appendChild(new Ext.tree.TreeNode({
        id:"delNews",
        text:"删除新闻"
    }));

    root.appendChild(sub1);

    root.appendChild(new Ext.tree.TreeNode({
        id:"sys",
        text:"系统设置"
    }));

    mytree.setRootNode(root);//设置根节点
```

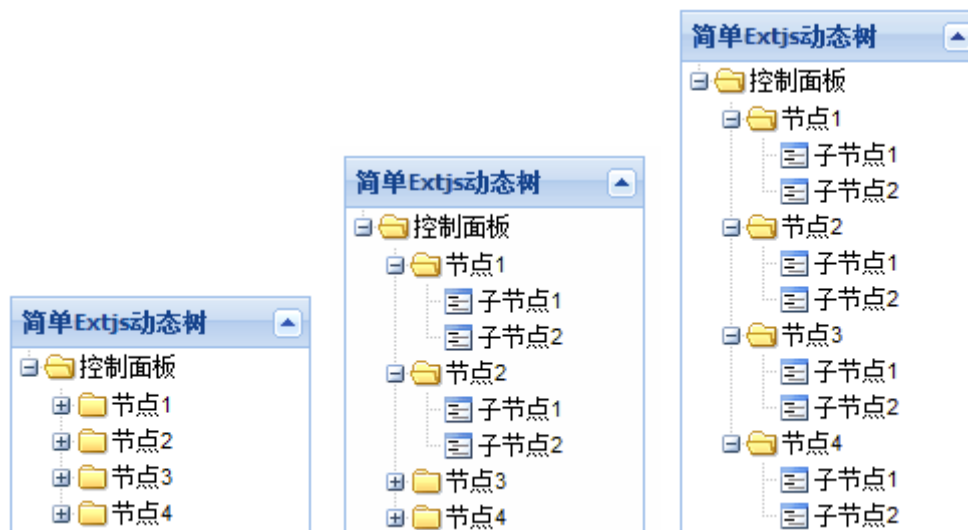
```
mytree.render();//不要忘记 render() 下,不然不显示哦
})
```

### 3.用 TreeLoader 加载数据生成树

//这里只介绍 TreeLoader 的参数一个

```
dataUrl:"*****.***"//地址
url:"*****.***"//url 参数和 dataUrl 参数一样
```

效果图:



html 代码:

```
<div id="container">
</div>
```

js 代码:

```
Ext.onReady(function(){
    Ext_quickTips.init();
    var mytree=new Ext.tree.TreePanel({
        el:"container",
        animate:true,
        title:"简单 Extjs 动态树",
        collapsible:true,
        enableDD:true,
        enableDrag:true,
```

```

        rootVisible:true,
        autoScroll:true,
        autoHeight:true,
        width:150,
        lines:true,
        //这里简简单单的 loader 的几行代码是取数据的,很经典哦
        loader:new Ext.tree.TreeLoader({
            dataUrl:"json.ashx"
        })
    });

    //根节点
    var root=new Ext.tree.AsyncTreeNode({
        id:"root",
        text:"控制面板",
        expanded:true
    });

    mytree.setRootNode(root);
    mytree.render();//不要忘记 render() 下,不然不显示哦
})

```

上面的代码中 dataUrl 地址为 json.ashx 的代码是怎样呢?

让我们先来思考一个问题:

```

/*---dataUrl 的地址返回的内容必须为数组对象形式,但是 .net 页面如何返回这样的格式?
---*/

```

我的解决方案:把 .net 数据转换为 json 对象。

```

        using System.Web.Script.Serialization;

        public string ToJson(object o)
        {
            JavaScriptSerializer j = new JavaScriptSerializer
()

```

```
        return j.Serialize(o);  
    }  
}
```

这里的 o，我们可以定义个类的对象，传进去，转换为 json 对象

json.ashx 代码:

```
using System;  
using System.Web;  
using System.Collections.Generic;  
using System.Web.Script.Serialization;  
  
public class jsongdata  
{  
    //定义 jsongdata 类，存放节点数据  
    public string id;  
    public string text;  
    public bool leaf;  
    public List<jsongdata> children=new List<jsongdata>(); //存放子节点  
}  
  
public class json : IHttpHandler {  
    public List<jsongdata> jsdata=new List<jsongdata>();  
    public void ProcessRequest (HttpContext context) {  
  
        for (int i = 1; i < 5; i++)  
        {  
            jsongdata jd = new jsongdata();  
            jd.id="num"+i;  
            jd.text = "节点"+i;  
            jd.leaf = false;  
            for (int j = 1; j < 3; j++)  
            {  
                jsongdata subjd = new jsongdata();
```

```

        subjd.id = "sub" + j;
        subjd.text = "子节点" + j;
        subjd.leaf = true;
        jd.children.Add(subjd);
    }
    jsdata.Add(jd);
}

context.Response.Write(ToJson(jsdata.ToArray())); //ToArray()

```

在 IE 里面好像缺了不行

```

    }

    public bool IsReusable {
        get {
            return false;
        }
    }

    public string ToJson(object o)
    {
        //序列化对象为 json 数据，很重要！

        JavaScriptSerializer j = new JavaScriptSerializer();
        return j.Serialize(o);
    }
}

```

异步获取它的数据:

POST json.ashx	200 OK	localhost:46530	709 B
<div> <div>Headers</div> <div>Post</div> <div>Response</div> </div> <pre> [[{"id":"num1","text":"节点1","leaf":false,"children":[{"id":"sub1","text":"子节点1","leaf":true,"children":[]}, {"id":"sub2","text":"子节点2","leaf":true,"children":[]}]}, {"id":"num2","text":"节点2","leaf":false,"children":[{"id":"sub1","text":"子节点1","leaf":true,"children":[]}, {"id":"sub2","text":"子节点2","leaf":true,"children":[]}]}, {"id":"num3","text":"节点3","leaf":false,"children":[{"id":"sub1","text":"子节点1","leaf":true,"children":[]}, {"id":"sub2","text":"子节点2","leaf":true,"children":[]}]}, {"id":"num4","text":"节点4","leaf":false,"children":[{"id":"sub1","text":"子节点1","leaf":true,"children":[]}, {"id":"sub2","text":"子节点2","leaf":true,"children":[]}]}]] </pre>			

好了。

这里是模拟出数据，从数据库中取出数据再处理为节点数据，一个道理！

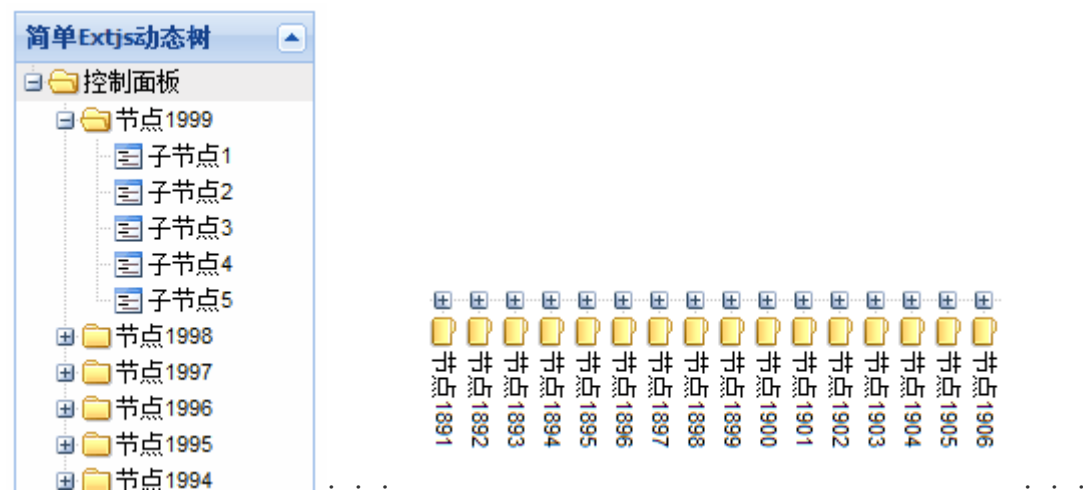
今天我们算是对 TreePanel 的最基本的使用有个基础的认识，谢谢支持！

昨天有朋友说,在 IE 下有的时候 ashx 传过来的节点不能加载,其实我也出现这样的问题,很是烦人!

今天 extjs 上看到了一个解决方案，觉得很好:如果 treeLoader 加载失败，会继续加载，直到成功加载。

#### 4.解决 IE 下非正常加载节点问题

即使从服务器取到大量的数据，也没有问题。



基本代码一样，只有 js 代码的 loader 小小的修改下：

```
loader:new Ext.tree.TreeLoader({
    url:"json.ashx",
    listeners:{
        "loadexception":function(loader,node,response){
            node.loaded = false;
            node.reload.defer(10,node);//不停的加载，直到 true
        }
    }
})
```

#### 5.使用 TreeNodeUI

在 node 中我们可以用专门的类控制 node 的 UI.

```
//TreeNodeUI 的基本配置参数,node.getUI()=>TreeNodeUI
1.addClass("class");//添加 css 类
2.getAnchor()//返回 a 元素(对象),控制 a 链接
3.getIconEl()//返回 img 元素(对象),控制 icon 图标
4.getTextEl()//返回 span 元素(对象),控制节点文本
5.hide()
6.show()
7.removeClass()
```

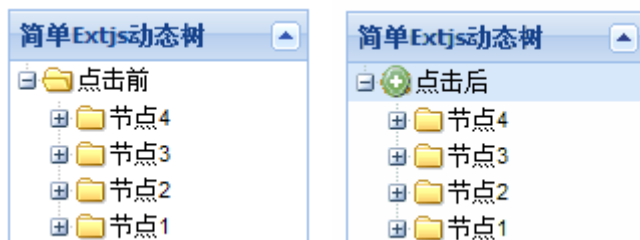
我们来做个例子，不过我先推荐个网站 [extjs 中文社区](#)，感觉就是内容太少了

例子：动态改变节点的 **icon** 图标

关键 js 代码：

```
//以根节点为例
root.on("click",function(node){
    node.getUI().getTextEl().innerHTML="点击后";//a 元素(对象)
    node.getUI().getIconEl().src="Image/add.gif";//img 元素(对象)
});
```

效果图：




再看看节点 `node.getUI().getTextEl()` 到底是什么？firebug 下，

	<b>TextEl</b>	
	<b>clientLeft</b>	span#extdd-3
	<b>clientTop</b>	0
	<b>contentEditable</b>	0
		"inherit"
	<b>getBoundingClientRect</b>	getBoundingClientRect()
	<b>getClientRects</b>	getClientRects()
	<b>getElementsByClassName</b>	getElementsByClassName()
	<b>id</b>	"extdd-3"
	<b>className</b>	"
	<b>nodeType</b>	1
	<b>tagName</b>	"SPAN"
	<b>nodeName</b>	"SPAN"
	<b>localName</b>	"SPAN"
	<b>prefix</b>	null
	<b>namespaceURI</b>	null
	<b>nodeValue</b>	null
	<b>ownerDocument</b>	<b>Document</b> TreePanelTest.aspx
	<b>parentNode</b>	a.x-tree-node-anchor TreePanelTest.aspx
	<b>offsetParent</b>	div#ext-gen12.x-panel-body
	<b>nextSibling</b>	null
	<b>previousSibling</b>	null
	<b>firstChild</b>	"点击前"
	<b>lastChild</b>	"点击前"
	<b>childNodes</b>	<b>NodeList</b> 0=Text length=1
	<b>attributes</b>	<b>NamedNodeMap</b> 0=Attr 1=Attr length=2
	<b>dir</b>	"
	<b>baseURI</b>	"http://localhost:46530/ExtJsTestWeb/TreePanelTest.aspx"
	<b>textContent</b>	"点击前"
	<b>innerHTML</b>	"点击前"
	<b>clientWidth</b>	0
	<b>clientHeight</b>	0
	<b>offsetLeft</b>	32
	<b>offsetTop</b>	1



node.getUI().getIconEl()的内容:

 <b>IconEl</b>	
clientLeft	0
clientTop	0
contentEditable	"inherit"
 <b>getBoundingClientRect</b>	getBoundingClientRect()
 <b>getClientRects</b>	getClientRects()
 <b>getElementsByClassName</b>	getElementsByClassName()
id	"extdd-2"
className	"x-tree-node-icon"
nodeType	1
tagName	"IMG"
nodeName	"IMG"
localName	"IMG"
 <b>style</b>	<b>CSSStyleDeclaration</b> length=0
tabIndex	-1
title	" "
lang	" "
align	" "
spellcheck	false
<b>src</b>	"http://localhost:46530/ExtJsTestWeb/ExtJs/resources/images/default/..."
naturalWidth	1
naturalHeight	1
width	16
height	18
x	16
y	0
name	" "
alt	" "

其他类似.

## 6.带有 checkbox 的树

类似 vista 下 IIS7.0 的配置(就是带有 checkbox 的哦)

先看几个关键字:

//关键代码

1.node.getUI().checkbox.checked//返回节点选择,true 和 fasle

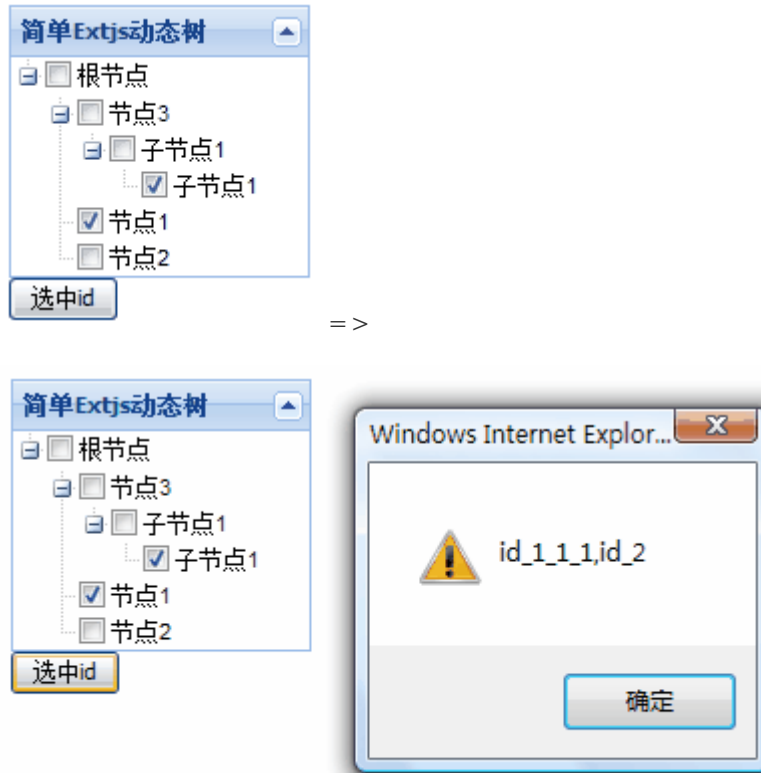
2.用 .net 书写 json 文本,引用两个 dl:Jayrock.dll 和 Jayrock.Json.dll,我会在后面提供下载

```
using Jayrock;
using Jayrock.Json;
using Jayrock.JsonRpc;
```

//使用 Jayrock 的 JsonTextWriter 可以书写 json 文本

3. checkchange 事件, 选择变化时激发

看个简单的效果图:



两个问题:

1. 如何隐藏复选框前的文件夹等小图标?

//答: 图标 img 元素的 css 的 class 名为 x-tree-node-icon, 把 display 设为 none 就可以了

2. 如何使用 Jayrock.dll 和 Jayrock.json.dll 把字符串转换为 json 对象?

```
using (JsonTextWriter writer = new JsonTextWriter(context.Response.OutputStream))
{
    writer.WriteStartArray(); //[
    writer.WriteStartObject(); //{
    writer.WriteEndObject(); //}
    writer.WriteEndArray(); //]

    writer.WriteMember("text"); //"text":
```

```

        writer.WriteString("节点 3");//节点三
        writer.WriteMember("checked");
        writer.WriteBoolean(false);//bool 值

        ...

    }
}

```

详细请参见

[http://msdn.microsoft.com/zh-cn/library/bb299886.aspx#intro\\_to\\_json\\_topic5](http://msdn.microsoft.com/zh-cn/library/bb299886.aspx#intro_to_json_topic5)

/\*

很罗嗦,我为什么要这么做?

其实我也不想这么罗嗦,但是我用下面两种方法

1.用 ashx 字符串拼接返回的文本,没有成功!

2.类对象序列化 json,因为 checked 为 C#的关键字,定义一个含 checked 字段的类,我没有成功!

只好用这么繁琐的办法,请高手有什么好的建议,多多指点!

\*/

css 代码:

```

//隐藏节点前的图标

<style type="text/css">

    .x-tree-node-icon
    {

        display:none;

    }

</style>

```

html 代码:

```

<body style="margin:10px;">

    <div id="container">

        </div>

```

```
<div id="btn"></div>
</body>
```

js 代码:

```
Ext.onReady(function(){
    Ext.QuickTips.init();
    Ext.BLANK_IMAGE_URL="ExtJs/resources/images/default/s.gif";

    function mytoggleChecked(node)
    {
        //迭代复选=>父节点影响子节点选择,子节点不影响父节点
        if(node.hasChildNodes())
        {
            //eachChild(fn),遍历函数
            node.eachChild(function(child){
                child.getUI().toggleCheck(node.attributes.checked);

                child.attributes.checked = node.attributes.checked;

                //child.getUI().checkbox.checked=node.getUI().checkbox.checked;
                //有其父必有其子

                child.on("checkchange",function(sub){
                    mytoggleChecked(sub); //传递子节点
                });

                mytoggleChecked(child); //处理子节点(第三级,有点晕阿)
            })
        }
    }

    var mytree=new Ext.tree.TreePanel({
        el:"container",
```

```

        animate:true,
        title:"简单 Extjs 动态树",
        collapsible:true,
        enableDD:true,
        enableDrag:true,
        rootVisible:true,
        autoScroll:true,
        autoHeight:true,
        width:150,
        lines:true,
        loader:new Ext.tree.TreeLoader({
            url:"checkjson.ashx",
            listeners:{
                "loadexception":function(loader,node,response){
                    //加载服务器数据,直到成功

                    node.loaded = false;

                    node.reload.defer(10,node);
                }
            }
        })
    });

//根节点
var root=new Ext.tree.AsyncTreeNode({
    id:"root",
    text:"根节点",
    checked:false,
    listeners:{
        "checkchange":function(node){ //选中是否切换激发事件
            mytoggleChecked(node); //自定义级联选择函数
        }
    },

```

```

        expanded:true
    });

    mytree.setRootNode(root);
    mytree.render();
    root.expand(true);

    new Ext.Button({
        text:"选中 id",
        handler:function(){
            var b=mytree.getChecked();
            var checkid=new Array;//存放选中 id 的数组
            for(var i=0;i<b.length;i++)
            {
                if(b[i].leaf)
                {
                    checkid.push(b[i].id);//添加 id 到数组
                }
            }
            alert(checkid.toString());//checkid.toString()这个结果,我们
            可以传到服务器,想怎么处理就怎么处理
        }
    }).render(document.body,"btn");
})

```

checkjson.ashx 代码:

```

//先添加引用两个 Jayrock.Json.dll 和 Jayrock.dll
<%@ WebHandler Language="C#" Class="checkjson" %>

using System;

using System.Web;

using System.Text;

```

```

//添加 Jayrock 命名空间
using Jayrock;
using Jayrock.Json;
using Jayrock.JsonRpc;

public class checkjson : IHttpHandler {

    public void ProcessRequest(HttpContext context)
    {
        using (JsonTextWriter writer = new JsonTextWriter(context.Res
ponse.Output))
        {
            //下面的代码好顶啊,算了,朋友,不想看了,就别看这个例子,我自己都有点看
            不下去了

            writer.WriteStartArray();
            //111
            writer.WriteStartObject();
            writer.WriteMember("text");
            writer.WriteString("节点 3");
            writer.WriteMember("id");
            writer.WriteString("id_1");
            writer.WriteMember("checked");
            writer.WriteBoolean(false);
            //111--111

            writer.WriteMember("children");
            writer.WriteStartArray();
            writer.WriteStartObject();
            writer.WriteMember("id");
            writer.WriteString("id_1_1");
            writer.WriteMember("text");
            writer.WriteString("子节点 1");
            writer.WriteMember("checked");

```

```
        writer.WriteBoolean(false);

//111-111-111

        writer.WriteMember("children");
        writer.WriteStartArray();
        writer.WriteStartObject();
        writer.WriteMember("id");
        writer.WriteString("id_1_1_1");
        writer.WriteMember("text");
        writer.WriteString("子节点 1");
        writer.WriteMember("leaf");
        writer.WriteBoolean(true);
        writer.WriteMember("checked");
        writer.WriteBoolean(true);
        writer.WriteEndObject();
        writer.WriteEndArray();

writer.WriteEndObject();
writer.WriteEndArray();
writer.WriteEndObject();

//2222

writer.WriteStartObject();
writer.WriteMember("id");
writer.WriteString("id_2");
writer.WriteMember("text");
writer.WriteString("节点 1");
writer.WriteMember("leaf");
writer.WriteBoolean(true);
writer.WriteMember("checked");
writer.WriteBoolean(false);
writer.WriteEndObject();

//333

writer.WriteStartObject();
```



```

        writer.WriteMember("id");
        writer.WriteString("id_3");
        writer.WriteMember("text");
        writer.WriteString("节点 2");
        writer.WriteMember("leaf");
        writer.WriteBoolean(true);
        writer.WriteMember("checked");
        writer.WriteBoolean(false);
        writer.WriteEndObject();

        writer.WriteEndArray();
    }
}

public bool IsReusable
{
    get
    {
        return false;
    }
}
}

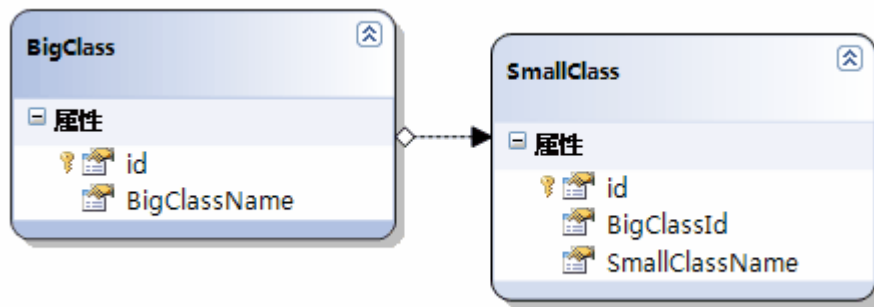
```

哇，好乱阿！感觉不是很好，就推荐一个很好的网站吧：

\*\*\*视频在线

继续 tree 的 learn! 今天就来个可增删改的树吧,操作数据库就使用比较方便的 Linq,无非就是增删改!

LinqData.dbml:



html 代码:

```
<body>

    <div id="container" style="float:left; margin-right:10px;">

    </div>

    <iframe name="mainFrame" id="mainFrame" src="http://www.baidu.com"
width="200px" height="200px"></iframe>

</body>
```

css 代码:

```
<style type="text/css">

    .leaf
    {

        background-image:url(ExtJs/resources/images/default/tree
/leaf.gif) !important;

    }<!--节点右键菜单的叶子图片-->

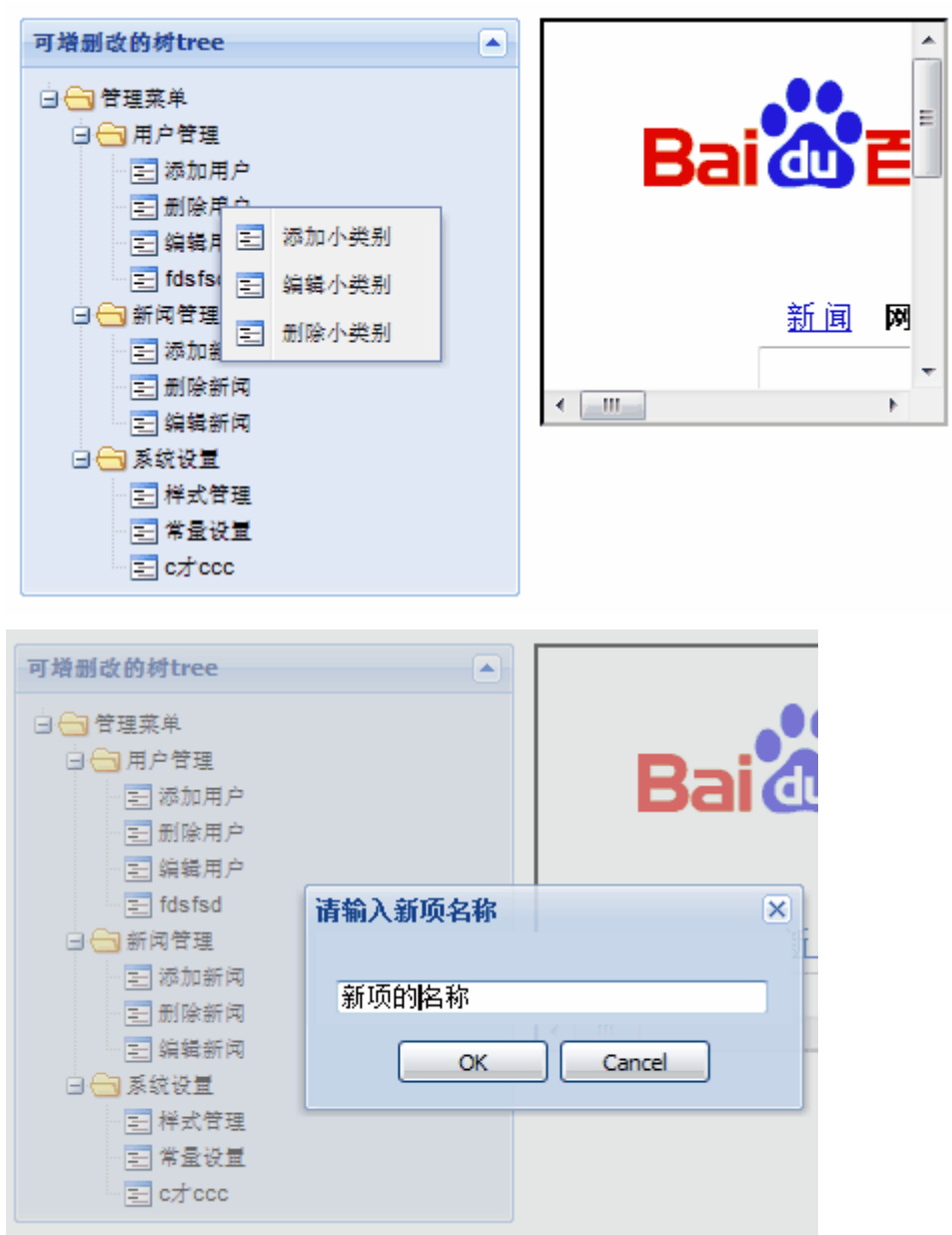
    .folder
    {

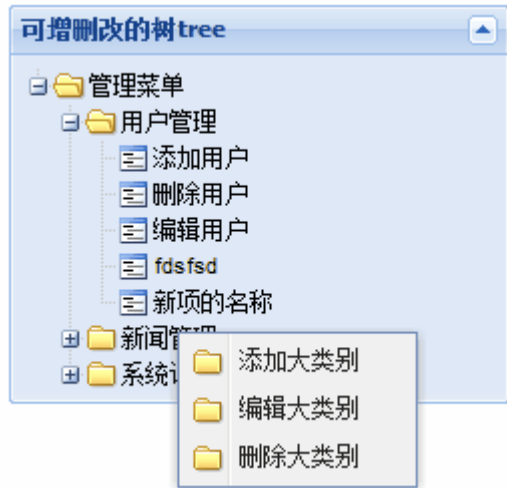
        background-image:url(ExtJs/resources/images/default/tr
ee/folder.gif) !important;

    }<!--节点右键菜单的文件夹图片-->

</style>
```

效果图:





图就不多贴出来,下面看核心代码:

核心代码:

js 代码:

```
///<reference path="Extjs_Intellisense.js" />
Ext.onReady(function() {
    var mytree=new Ext.tree.TreePanel({
        id:"mytree",
        el:"container",
        animate:true,
        title:"可增删改的树 tree",
        collapsible:true,
        frame:true,
        enableDD:true,
        enableDrag:true,
        rootVisible:true,
        autoScroll:true,
        autoHeight:true,
        width:250,
        lines:true,
        loader:new Ext.tree.TreeLoader({
            url:"CURDTree.ashx",//服务器处理数据代码
        })
    })
    mytree.render('container')
```

```

        listeners:{
            "loadexception":function(loader,node,response){
                //加载服务器数据,直到成功
                node.loaded = false;
                node.reload.defer(10,node);
            }
        }
    })),
    listeners:{
        "contextmenu":function(node,e){
            if(node.isLeaf())
            {
                var nodemenu=new Ext.menu.Menu({
                    items:[{
                        text:"添加小类别",
                        iconCls:'leaf',//右键名称前的小图片
                        handler:function(){
                            Ext.MessageBox.prompt("请输入新项名称","",function(e,text){
                                if(e=="ok")
                                {
                                    Ext.Ajax.request({
                                        url: 'CURDTree.ashx?newitemParentid='+node.parentNode.id.substring(2)+"&newitemValue="+text,
                                        success:function(request)
                                        {
                                            mytree.root.reload();//数的重新加载
                                        }
                                    });
                                }
                            });
                        }
                    }
                });
            }
        }
    }
}

```

```

expand(true,false);

},

failure:function

n(){

alert("添加

失败");

}

});

}

else

{

alert("取消了");

}

})

}

},{

text:"编辑小类别",

iconCls:'leaf',

handler:function(){

Ext.MessageBox.prompt("

请输入此项新名称","",function(e,text){

if(e=="ok")

{

Ext.Ajax.request({

url: 'CURDTree.

ashx?editItemid='+node.id+"&editItemvalue="+text,//传递需要的值,服务器会

执行修改

success:function

n(request)

{

mytree.root.

reload();

```

```

mytree.root.

expand(true,false);

    },
    failure:function
n(){
    alert("编辑
失败");

    }
    });
    }
    else
    {
        alert("取消了");
    }
    })
    }
    },{
        text:"删除小类别",
        iconCls:'leaf',
        handler:function(){
            Ext.Ajax.request({
                url: 'CURDTree.
ashx?delItemid='+node.id,//根据 id 删除节点
                success:function
n(request)
                {
                    mytree.root.
reload();
                    mytree.root.
expand(true,false);
                },
                failure:function

```

```

n(){

                                alert("删除

失败");

                                }

                                });

                                }

                                }]]

                                ));

                                nodemenu.showAt(e.getPoint()); //men

u 的 showAt，不要忘记

                                }

                                else if (!node.isLeaf() && node.parentNode!

=null)

                                {

                                var nodemenu=new Ext.menu.Menu({

                                items:[{

                                text:"添加大类别",

                                iconCls:'folder',

                                handler:function(){alert(no

de.id)} //在此略去

                                },{

                                text:"编辑大类别",

                                iconCls:'folder',

                                handler:function(){alert("执

行事件代码")}}

                                },{

                                text:"删除大类别",

                                iconCls:'folder',

                                handler:function(){alert("执

行事件代码")}}

                                ]]

                                ));

```



```

                                nodemenu.showAt(e.getPoint());
                                }
                                }

        }
    });

    var root=new Ext.tree.AsyncTreeNode({
        id:"root",
        text:"管理菜单",
        expanded:true
    });

    mytree.setRootNode(root);
    mytree.render();
})

```

服务器端代码:

```

//CURDTree.ashx 文件
//引用 Newtonsoft.Json.dll
<%@ WebHandler Language="C#" Class="CURDTree" %>

using System;
using System.Web;
using System.IO;
using System.Linq;
using Newtonsoft.Json;

public class CURDTree : IHttpHandler {

    public void ProcessRequest (HttpContext context) {
        LinqDataDataContext lddc = new LinqDataDataContext(@"Data Source=WIN-7JW7UWR0M27\MSSQL2005;Initial Catalog=LinqData;Persist Security Info=True;User ID=sa;Password=*****");
    }
}

```

```

        if (context.Request.QueryString["newitemParentid"] != null &
& context.Request.QueryString["newitemValue"] != null)
        {
            //添加

            SmallClass sc = new SmallClass();

            sc.BigClassId = Convert.ToInt32(context.Request.QueryStri
ng["newitemParentid"]);

            sc.SmallClassName = Convert.ToString(context.Request.Quer
yString["newitemValue"]);

            lddc.SmallClass.InsertOnSubmit(sc);

            lddc.SubmitChanges();
        }
        else if (context.Request.QueryString["editItemid"] != null&&c
ontext.Request.QueryString["editItemvalue"]!=null)
        {
            //编辑

            SmallClass sc = lddc.SmallClass.First(s => s.id == Int32.
Parse(context.Request.QueryString["editItemid"]));

            sc.SmallClassName = Convert.ToString(context.Request.Quer
yString["editItemvalue"]);

            lddc.SubmitChanges();
        }
        else if (context.Request.QueryString["delItemid"] != null)
        {
            //删除

            SmallClass sc = lddc.SmallClass.First(c => c.id == Int32.
Parse(context.Request.QueryString["delitemid"]));

            lddc.SmallClass.DeleteOnSubmit(sc);

            lddc.SubmitChanges();
        }
        else

```

```

{
    StringWriter sw = new StringWriter();
    JsonWriter writer = new JsonWriter(sw);

    var results = from small in lddc.SmallClass
                  join big in lddc.BigClass on small.BigClass
Id equals big.id

                  group small by small.BigClassId;

    //下面开始拼接 json 数据字符串
    writer.WriteStartArray(); //[, 其他类似, 请参见网上 Newtonsoft.
Json.dll 的使用

    foreach (var r in results)
    {
        writer.WriteStartObject();
        writer.WritePropertyName("id");
        writer.WriteValue("b_" + r.First().BigClass.id);
        writer.WritePropertyName("text");
        writer.WriteValue(r.First().BigClass.BigClassName);
        writer.WritePropertyName("children");
        writer.WriteStartArray();
        foreach (var s in r)
        {
            writer.WriteStartObject();
            writer.WritePropertyName("id");
            writer.WriteValue(s.id);
            writer.WritePropertyName("href");
            writer.WriteValue("FormSubmit.aspx?id=" + s.id);
            writer.WritePropertyName("hrefTarget");
            writer.WriteValue("mainFrame");
            writer.WritePropertyName("text");
            writer.WriteValue(s.SmallClassName);

```

```

        writer.WritePropertyName("leaf");

        writer.WriteValue(true);

        writer.WriteEndObject();
    }

    writer.WriteEndArray();

    writer.WriteEndObject();
}

writer.WriteEndArray();//]

writer.Flush();

string myjson = sw.ToString();

context.Response.Write(myjson);//输出 json 数据结果
}

}

public bool IsReusable {
    get {
        return false;
    }
}

}
}

```

今天就做个例子,下回我们再继续讨论 extjs 的其他内容.

thanks!

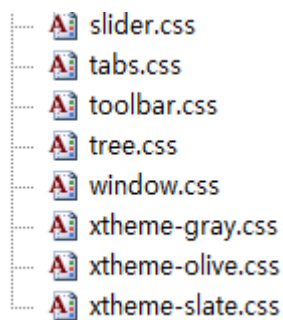
哦,忘记了 Jayrock 的 dll 库文件:Jayrock 的 dll

extjs 的默认皮肤很好看,但是我们还可以变换样式切换其他皮肤.

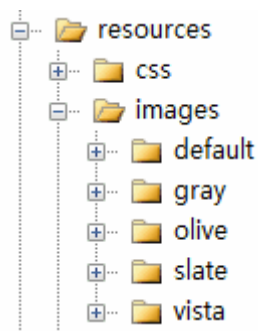
**1.直接添加其他 css 文件换肤.**

皮肤文件:xtheme-olive.zip 下载

把皮肤文件解压,把 css 文件(如 xtheme-olive.css)拷贝到 extjs 的 resources 目录下 css 文件夹里面:



解压皮肤文件,把里面的相应的 image 文件夹下的目录(比如 olive)拷贝到 extjs 的 resources 目录下 images 文件夹下



设置 css 文件如下:

```
<link type="text/css" href="ExtJs/resources/css/ext-all.css" rel="stylesheet" />
<link type="text/css" href="ExtJs/resources/css/xtheme-olive.css" rel="stylesheet" />
<script type="text/javascript" src="ExtJs/adaptor/ext/ext-base.js"></script>
<script type="text/javascript" src="ExtJs/ext-all.js"></script>
```

其实就是在原有的基础上添加了个 xtheme-olive.css 文件。

效果图:

添加个人信息

用户名:

性别:

☒ 男
☐ 女

出生日期:

学位:

请选择适合你的学历...

使用框架:

☐ Spring.net
☐ Nhibernate
☐ Linq

Email:

个性签名:

想说的话:

Tahoma

B

I

U

A<sup>+</sup>

A<sup>-</sup>

A

ab

## 2.配合 cookie 实现网站换肤

我们的目标:

```

/*
    用户可以选择自己的样式，当下次用户打开的网站(在 cookie 的有效期内)的时候，自动
    读取 cookie，显示先前选择的爱好的样式！
*/

```

我们的原理:

```

//关键点
1.document.getElementsByTagName("link")[1].href//获取或者设置第二个css文
件的href值
2.var date=new Date();//今天的日期
    date.setTime(date.getTime()+30*24*3066*1000);//30天后的日期
    document.cookie="css="+name+";expires="+date.toGMTString();//设置30
天后过期的cookies(名称为css)
3.var cookiesArr=document.cookie.split(";");
    var css;
    for(var i=0;i<cookiesArr.length;i++)
    {
        var arr=cookiesArr[i].split("=");
        if(arr[0]=="css")
        {

```

```

        css=arr[1];

        break;
    }
}

```

//这段代码是获取 cookies 中想要的名称为 css 的 cookies，并保存在变量 css 中去

4.如果不设置 expires 的话，会在关闭浏览器后 cookies 失效。

html 代码:

```

<!--切换标签-->

<a href="javascript:void(0)" onclick="changeCSS('')">还原</a>

<a href="javascript:void(0)" onclick="changeCSS('xtheme-olive.css')">
绿色</a>

<a href="javascript:void(0)" onclick="changeCSS('xtheme-gray.css')">
灰色</a>

<a href="javascript:void(0)" onclick="changeCSS('xtheme-purple.css')">
purple</a>

```

关键js 代码:

```

function changeCSS(name)
{
    var date=new Date();

    date.setTime(date.getTime()+30*24*3066*1000);

    document.getElementsByTagName("link")[1].href="ExtJs/resources
/css/"+name;

    document.cookie="css="+name+";expires="+date.toGMTString();//
设置 cookies
}

var cookiesArr=document.cookie.split(";");
var css;

for(var i=0;i<cookiesArr.length;i++)
{
    var arr=cookiesArr[i].split("=");

```

```

        if(arr[0]=="css")
        {
            css=arr[1];
            break;
        }
    }

    document.getElementsByTagName("link")[1].href="ExtJs/resources/css/"
    "+css;//读取并应用css

```

效果图(下次打开浏览器还是这样):

[还原](#) [绿色](#) [灰色](#) [purple](#)

OK, 对于 extjs 的换肤, 就聊到这里, 其实觉得默认的皮肤还可以拉,

下面我提供几套从网上下载到的皮肤, 如果你有什么其他好看的皮肤, 请跟帖答复, 给个下载地址 .

<http://files.cnblogs.com/qianxudetianxia/xtheme-slate.zip>

<http://files.cnblogs.com/qianxudetianxia/xtheme-purple.zip>

<http://files.cnblogs.com/qianxudetianxia/xtheme-galdaka.zip>

<http://files.cnblogs.com/qianxudetianxia/xtheme-darkgray.zip>

<http://files.cnblogs.com/qianxudetianxia/xtheme-black.zip>

thanks!