columnWidth 参数不同之处在于，columnWidth 是用百分比的小数值设置的，而该值是根据分数值算的，譬如要分 3 列，第 1、2 列占容器宽度的四分之一，则设置为 1，第 3 列占容器宽度的二分之一，则设置为 2。

- pack：该参数控制容器子元素的放置方式。它可设置为以下 3 个值：
  - start：这是默认值，子元素将放置到容器的左边。
  - center：子元素将放置到容器的中间。
  - end：子元素将放置到容器的右边。
- padding：设置容易的 padding 样式，默认值是 0。
- renderHidden：如果设置为 true，则每个容器在渲染时是看不到的，默认值是 false。

现在开始完成这个示例。创建一个名称为“hbox.html”的页面，把需要的 Ext 文件包含在文件里，然后在页面上增加一个 div，用来显示 FormPanel，代码如下：

```
<div id="hbox" style="margin:20px 0 0 20px"><div>
```

在 onReady 函数内创建一个宽度为 600、高度为 400 的 FormPanel，且设置其 layout 为 hbox，代码如下：

```
var frm=new Ext.form.FormPanel({
   title: '在 FormPanel 中使用 HBoxLayout 进行布局',
   renderTo:"hbox",
   width:600,
   height:400,
   labelWidth: 80,
   frame:true,
                        layout: {
                                        type:'hbox',
                           align : 'stretch',
                           pack  : 'start',
                           padding:5
                        },
                        defaults:{
                                 layout:'form',
                                 frame:true,
                                 border:false
                        },
      items:[]
});
```

在 layout 参数中，定义了 layout 类型为 hbox，且对齐方式为 stretch，子元素放置位置为 start，而且 padding 样式值为 5。

参数 defaults 里定义子元素的 layout 为 FormLayout，如果不使用 FormLayout，那么 TextField 控件的标题将看不到，需要通过代码添加，而且上下两个控件之间将没有间隔。一般情况下不设置 frame 为 true，但因本示例用于测试，所以设置 frame 为 true 以便于查看子元素的变化。

然后在 items 中定义两列，每列都包含两个 TextField 控件，代码如下：

```
{
     height:100,
     flex:1,
     items:[
        {xtype:'textfield',fieldLabel:'用户名称',name:"username",anchor:  '90%',
value:'1',tabIndex:1},
```

```
            {xtype:'textfield',fieldLabel :'电子邮件',name:"email",anchor: '90%',value: '3',
tabIndex:3}
          ]
    },
    {
          height:200,
          flex:1,
          items:[
            {xtype:'textfield',fieldLabel :'电话',name:"phone",anchor:'90%', value:'2',
tabIndex:2},

          {xtype:'textfield',fieldLabel  :'QQ',name:"qq",anchor:'90%',value:'4',tabInd
ex:4}
          ]
    }
```

代码中，两列都设置了 flex 的值为 1，表示它们各占容器宽度的一半。代码中还分别为每列设置了不同的高度，这将在后面的测试中用到。要注意的是，在代码里为每个 TextField 控件都设置了 tabIndex 值，这是为什么呢？其主要原因是，如果想使用 Tab 键，以从左到右、从上到下的方式在 TextField 控件中移动光标，不设置 tabIndex 值是无法实现的。因为页面的光标移动方式是在遍历完容器内的控件才会移动到下一个容器，所以如果不设置 tabIndex 值，光标会先完成第 1 列内控件的移动，然后才转移到第 2 列。

完成后，在浏览器中打开页面，将看到如图 8-1 所示的结果。

从图 8-1 中可以看到，现在每列的高度都是 FormPabel 的 Body 高度。

将 HBoxLayout 的 align 方式修改为“stretchmax”，刷新一下页面，将看到如图 8-2 所示的结果。

图 8-1 在 FormPanel 中使用 HboxLayout 进行布局的显示结果

图 8-2 align 为“stretchmax”的显示结果

从图 8-2 中可以看到，现在两列的高度都变成了第 2 列的高度。

再将 align 的值修改为“top”，将看到如图 8-3 所示的结果。

从图 8-3 可以看到，现在高度都是各自的高度了。

根据以上的测试，可以了解到不同的 align 值可获得不同的列高度，因而可以在设计中通过该设置灵活布局。

图 8-3 align 为“top”的显示结果

8.1.2 在 FormPanel 中使用 HBoxLayout 和 VBoxLayout 进行布局

本示例将要在 FormPanel 中实现如图 8-4 所示的布局。

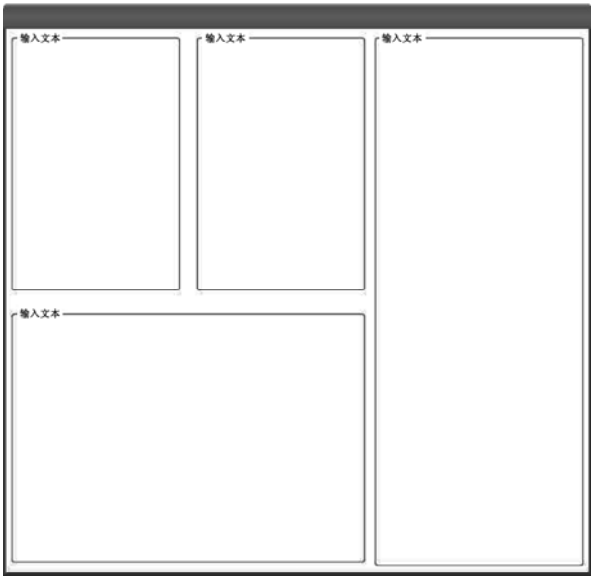

图 8-4 将要实现的布局图

从图 8-4 可以看到，首先要将 FormPanel 分成 2 列，然后左边再分 2 行，最后还要将左边第 1 行再分成 2 列。

首先创建一个名称为“hbox2.html”的页面文件，将需要的 Ext 文件包含在该文件里，然后创建一个 FormPanel，并使用 HBoxLayout 将 FormPanel 分成两部分，其中第 1 列的 flex 值为 2，第 2 列的 flex 值为 1，具体代码如下所示。

```
    var frm=new Ext.form.FormPanel({
    title: '在 FormPanel 中使用 HBoxLayout 和 VBoxLayout 进行布局',
```

```
        renderTo:"hbox",
        width:600,
        height:400,
        labelWidth: 80,
        frame:true,
        layout: {
                    type:'hbox',
            align : 'stretch',
            pack  : 'start',
            padding:5
        },
        defaults:{
            frame:true,
            border:false
        },
        items:[
            {
                    flex:2,
                    items:[
                    ]
            },
            {
                    flex:1,
                    items:[
                    ]
            }
        ]
    });
```

现在已经分好 2 列了，可以在浏览器中打开页面，将看到如图 8-5 所示的结果。

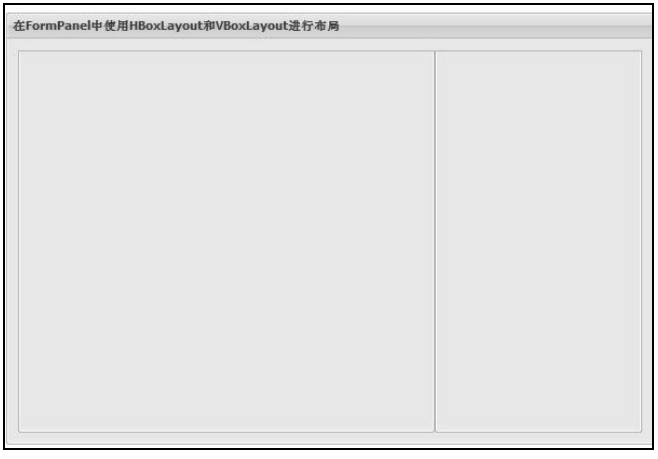

图 8-5　已分成两列的 FormPanel

下面要将第 1 列分成两行，并在所有区域加入 FieldSet 控件。经过修改后的 FormPanel 的 items 参数的代码如下所示：

```
    {
        layout: {
                            type:'vbox',
```

```
            align : 'stretch',
            pack  : 'start',
            padding:5
        },
        defaults:{
            frame:true,
            border:false
        },
        flex:2,
        items:[
            {flex:1,items:[]},
            {flex:1,items:[]}
        ]
    },
    {
        layout:'fit',
        flex:1,
        items:[
            {xtype:'fieldset',title:'分组 4', items:[]}
        ]
    }
```

第 1 列的代码里增加了定义 layout 的代码，现在使用 VBoxlayout 分行，使用 flex 值设置两行平均分配容器的高度。

在第 2 列的代码里，使用了一个 FitLayout 作为容器，其目的是让 FieldSet 控件能自动填满整个容器。

刷新一下页面，将看到如图 8-6 所示的结果。

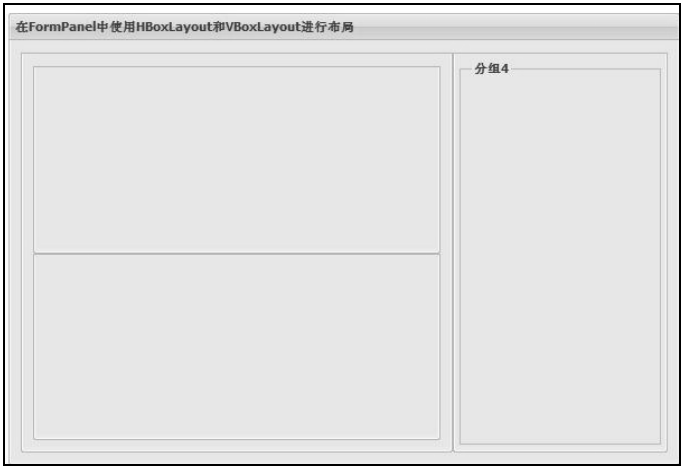

图 8-6 第 1 列已分成两行的 FormPanel

现在要完成布局的最后一步，将左边第 1 行分成两列，并在所有区域显示 FieldSet 控件。经过修改后的第 1 列的 items 参数的代码如下所示：

```
    {
        layout: {
                    type:'hbox',
            align : 'stretch',
            pack  : 'start',
            padding:5
```

```
        },
        defaults:{
            layout:'fit',
            frame:true,
            border:false
        },
        flex:1,
        items:[
            {
                    flex:1,
                    items:[
                            {xtype:'fieldset',title:'分组 1', items:[]}
                    ]
            },
            {
                    flex:1,
                    items:[
                            {xtype:'fieldset',title:'分组 2',items:[]}
                    ]
            }
        ]
    },
    {
        layout:'fit',
        flex:1,
        items:[
            {xtype:'fieldset',title:'分组 3', items:[]}
        ]
    }
```

和第 2 列一样，都要使用 Fitlayout 作为容器，让 FieldSet 自适应容器。

刷新浏览器，将看到如图 8-7 所示的结果。

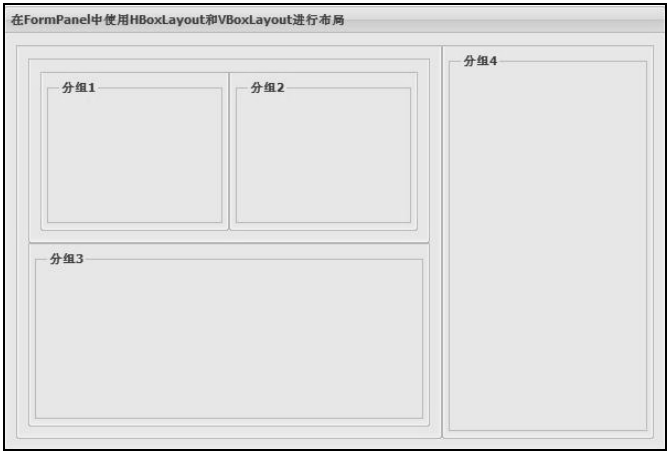

图 8-7 区域划分完成的 FormPanel

现在看上去有点怪，需要做一下调整。将目前代码中除了 FormPanel 的 frame 之外的所有 frame 和 padding 设置全部注释掉，将看到如图 8-8 所示的结果。

以下要解决的问题就是关于 FieldSet 的边界粘连问题了，只要设置容器的 bodystyle 参数就可

以了。在“分组 1”、“分组 2”、“分组 3”的父容器里增加以下代码：

```
    bodyStyle:'padding-right:5px',
```

图 8-8 注释掉 frame 和 padding 设置后的 FormPanel

完成后刷新一下浏览器，将看到如图 8-9 所示的结果。

图 8-9 FormPanel 最终显示的结果

至此，一个比较复杂的 FormPanel 布局就完成了。

注意 本示例在 IE 中会出现看不到 FieldSet 底线的问题，可以在 bodyStyle 的定义中增加“padding-bottom:10px”来解决。也可以不使用 FitLayout，而采取在 FieldSet 中设置 height 属性的方法解决。不过笔者测试过，两种方法都不能让 IE 与 Firefox 的显示一样。

通过上述两个使用 HBoxLayout 和 VBoxLayout 进行布局的示例，可以看到，使用 HBoxLayout 和 VBoxLayout 进行布局比在 Ext 2.0 中使用 ColumnLayout 进行布局简单多了，因而希望大家能花

点功夫研究一下，这样会使你未来的开发事半功倍。

8.1.3 Panel 的 body 的样式范围

在 Ext 2.0 中，如果要在 Panel 的 body 中显示 HTML 文本，会碰到不少麻烦，譬如要使用“<li>”显示一个列表或者使用“”加粗显示文本，都会得不到你想要的结果。主要原因是这些样式定义都会受到 Ext 的样式定义影响。为了解决这个问题，在 Ext 3.0 中，Panel 增加了参数 preventBodyReset。其作用就是，如果设置该参数 true，则 body 内的文本将不受 Ext 样式的影响，而回归 W3C 的标准样式。相反，如果不设置（默认值为 false），或者设置为 false，则会受 Ext 样式影响。

下面我们通过一个示例来测试一下这两种情况。新建一个名称为“preventBodyReset.html”的页面文件，然后将其代码替换为以下代码。

```
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/
strict.dtd">
    <html>
         <head>
         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
         <title>第 8 章 Ext 用户界面控件</title>
        <link rel="stylesheet" type="text/css" href="../resources/css/ext-all.css"/>
         <script src="../ext-base.js"></script>
         <script src="../ext-all-debug.js"></script>
      </head>
      <body>
      </body>
      <div id="panel1" style="margin:20px 0 0 20px;"></div>
      <div id="panel2" style="margin:20px 0 0 20px;"></div>

         <script>
       Ext.onReady(function(){
                              var html=[
                                    '<h1>标题一</h1>',
                                    '<h2>标题 2</h2>',
                                    '<p><strong>STRONG</strong>, <em>EMPHASIS</em> and
a <a href="#">Link</a></p>',
                                    '<ul>',
                                          '<li>行 1</li>',
                                          '<li>行 2</li>',
                                    '</ul>',
                                    '<ol>',
                                          '<li>行 1</li>',
                                          '<li>行 2</li>',
                                    '</ol>'
                  ];


         var panel1=new Ext.Panel({
            title:"preventBodyReset:true",
            width:500,
            height:350,
            bodyStyle:"padding:5px;",
            renderTo:"panel1",
            preventBodyReset:true,
            html:html.join('')
```

```
        });

        var panel2=new Ext.Panel({
            title:"preventBodyReset:false",
            width:500,
            height:200,
            bodyStyle:"padding:5px;",
            renderTo:"panel2",
            html:html.join('')
        });

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

代码中定义了两个 Panel，第 1 个设置了 preventBodyReset 为 true，第 2 个则没有设置。在浏览器中打开这个页面，将看到如图 8-10 所示的结果。

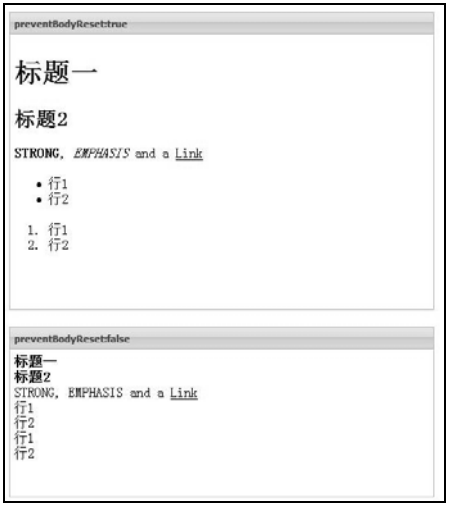

图 8-10 参数 preventBodyReset 测试页面的显示结果

从图 8-10 可以看到，设置 preventBodyReset 为 true 的 Panel 里的文本使用的是 W3C 的标准样式，而没有设置的则显示为 Ext 的样式。

我们可通过 firebug 检查一下这到底发生了什么。

在 Firebug 单击“单击查看页面中的元素”按钮，然后在页面的第 1 个 Panel 中单击“STRONG”。也可以直接在 HTML 标签页中使用查找功能找到“STRONG”。

将 HTML 标签页中右边的小窗口切换到“样式”标签页，我们可以看到如图 8-11 所示的结果。

从图 8-11 中可以看到，“STRONG”能恢复到 W3C 的标准样式是因为重定义了 Body 里的样式，使之与 W3C 里的样式是一样的。

重复刚才的动作，不过目标是没有设置 preventBodyReset 的 Panel 里的“STRONG”，单击后将看到如图 8-12 所示的结果。

从图 8-12 中可以看到，这里“STRONG”使用的是默认的 EXT 里的样式，而其显示是用正常

字体显示的。

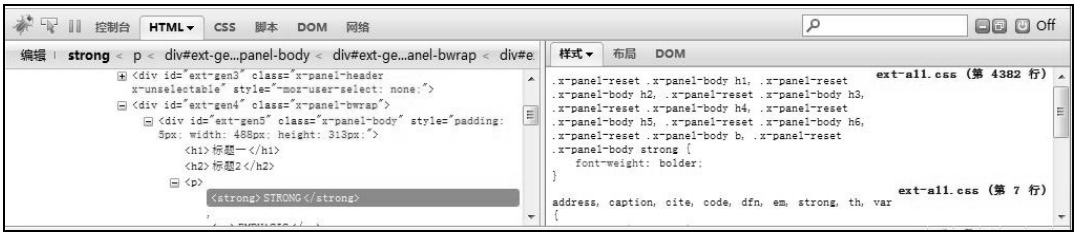

图 8-11 设置了 preventBodyReset 为 true 的“STRONG”的样式

图 8-12 没有设置 preventBodyReset 的“STRONG”的样式

从以上测试可以知道，参数 preventBodyReset 是一个不错的改进，增加了 Panel 输出结果的灵活性。因为参数 preventBodyReset 是在 Panel 里定义的，所以继承自 Panel 的对象都具有该属性，譬如 FormPanel。