



EfsFrame Web 表现层 API 文档

版本号：V2.0 版		EfsFrame 框架团队	
联系方式：			
网址： http://www.efsframe.cn/			
联系人：	郭志军	Email：	enjsky@163.com
Q Q：	68098375	联系电话：	13297990437

目 录

第一章 概述	4
第二章 文件引入	4
第三章 页面布局	5
简单布局	5
再布局	7
第四章 Efs 命名空间	10
getDom(el) :	10
getEfs(el):	10
getExt(el):	11
第五章 核心 panel 面板容器.....	11
简单的 panel 容器.....	11
复杂的 panel 容器.....	12
获取 panel 容器对象.....	13
第六章 tabpanel 页签容器.....	13
简单的 tabpanel 容器.....	13
复杂的 tabpanel 容器.....	14
第七章 Grid 网格容器.....	15
GridPanel 构造.....	15
数据源 store	16
列模式 colmodel.....	17
样例及显示效果.....	17
第八章 Treepanel 树面板.....	18
Treepanel 构造.....	18
xtype="root" 节点.....	19
xtype="loader" 节点.....	20
样例及显示效果.....	21
第九章 Window 窗口面板.....	21
Window 窗口构造.....	21
样例及显示效果.....	23
第十章 Form 表单	23
Form 对象构造.....	24
样例及效果	24
form 常用方法.....	25
第十一章 Toolbar 工具条.....	26
Toolbar 构造	26
样例及显示效果.....	27
第十二章 Button 按钮和 Menu.....	27
Button 构造	28
Menu 构造	28
第十三章 Efs 事件	28
onEfs + 大写开头的 Ext 事件名.....	28
第十四章 Efs.Common 常用方法	29

createDocument(strData, isUrl).....	29
getNode(objXML, nodePath, index).....	29
getNodeValue(objXML, nodePath, index).....	29
setAttribValue (objXML,nodePath,attribName,attribValue)	30
selectSingleNode(objXML, nodePath).....	30
selectNodes(objXML, nodePath).....	30
setText(node, value).....	30
getText(node).....	30
getXML(objXML).....	31
getOpXml(mix, isDom).....	31
getQryXml(form, emptyRet).....	31
getEmptyQryXml().....	31
setEditValue(xml doc, form, rootEl).....	31
getRootXml(isDom).....	32
addRootXml: function(xml doc, isDom).....	32
setSpanValue: function(sXML,sRoot,where).....	32
ajax: function(url,param,callback,scope).....	32
load: function(cfg,callback).....	33

第一章概述

Efs 是企业快速开发的 UI 层。这个 UI 层封装 **extjs** 框架，**Ext** 提供了好的页面布局方式、功能强大的组件、优质的页面风格，但是如果直接使用 **Ext** 又是较为复杂的，不仅要从头学习 **EXT** 框架，而且 **Ext** 是通过 **js** 函数创建页面的，这就不得不为每个页面都编写一个相对应初始化函数，这会使得开发人员要将大量的时间放在页面的制作上，而无法集中精力专注在业务组件的开发上，所以我们针对 **Ext** 框架进行再封装，将配置项写在 **div** 标签上，根据对应的 **xtyp** 自动构造 **ext** 对象。开发人员可以不用深入掌握 **Ext** 框架，只需要开发者编写少量的 **html** 代码，也可以做出很炫的页面。当然你也可以使用 **new Ext** 的控件渲染到页面，两者可以混合使用。

Efs 中的页面配置大多数和 **Ext** 的配置项相同，少量不同的再下面介绍组件的篇章会提到。Efs 的常用方法都在 **Efs.Common** 中，其他方法都是 **Ext** 的方法，大家可以参考 **ExtAPI**。Efs 支持 **Ext** 的所有事件，调用方式请参考“Efs 事件”篇章。开发者通常通过 **Efs.getExternal("id")** 就可以得到 **Ext** 组件的原始对象，然后用这个对象再调用 **Ext** 的方法。

第二章 文件引入

要在页面上使用该框架需要引入相关的 **js** 文件：

```
<link rel="stylesheet" type="text/css" href="css/ext-all.css" />
<link rel="stylesheet" type="text/css" href="css/efs-all.css" />
<script type="text/javascript" src="js/loadmask.js"></script>
<script type="text/javascript" src="js/efs-all.js"></script>
```

ext-all.css ext 框架的样式文件

efs-all.css Efs 框架的样式文件

loadmask.js 该文件是页面加载提示层，应该放在样式文件之后，

efs-all.js 之前引入。

efs-all.js 该文件已经包含了 ext 框架的 js 文件，它应该在执行 js 之前引入，而且在样式文件之后。

第三章 页面布局

由于 ext 的布局方式很多，真正常用的不多，所以作者挑选了 ext 的 border 布局进行封装。如果您还有其他布局要求可以通过 new Ext 的控件方式渲染到页面，或者联系请联系我们，我们将根据您的需求进行定制。

简单布局

border 布局运用 east（右）、west（左）、north（上）、south（下）、center（其余部分）最多 5 个方位来满足布局需要，其中 north 和 south 的优先级最高，然后是 east 和 west，最低的是 center：

North 部分		
West 部分	Center 部分	East 部分
South 部分		

通过上图可以理解这里优先级的意思：north 和 south 占用整个页面的宽，高度由我们定义；west 和 east 占用 north 和 south 高度以外剩下的全部高度，最后剩余的部分为 center，下面是定义 5 个 div 标签来完成布局的实例：

```
<HEAD>
<TITLE> layout 布局 </TITLE>
<META http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" type="text/css" href="../css/ext-all.css" />
<script type="text/javascript" src="../js/efs-all.js"></script>
```

```
</HEAD>
<BODY>

<div region="north" height="100" title="north部分">North部分</div>
<div region="south" height="100" title="South部分">South部分</div>
<div region="west" width="150" title="West部分">West部分</div>
<div region="east" width="150" title="East部分">East部分</div>
<div region="center">中间部分</div>
</BODY>
</HTML>
```

north部分		
North部分		
West部分	中间部分	East部分
West部分		East部分
South部分		
South部分		

Ø 整体布局（后面会将谈到再次布局）时这些 **div** 节点一定要都是 **body** 标签的直接子节点，而且 **body** 标签再不应该含有其它需要显示出来的直接子节点，否则将不作为布局方式，直接以 **html** 内容引入。

Ø 5 个 **div** 并不是都需要的，其中 **center** 对应的 **div** 节点是**不可或缺的**，其它 **div** 根据对页面布局的需要来定义，而且每个 **div** 一定要定义 **id** 属性。

Ø 定义布局的 **div** 标签定义了一些扩展标签：

u **region** 属性：每个定义布局的 **div** 一定需要定义 **region** 属性，而且该属性的值只可以是 '**east**'、'**south**'、'**west**'、'**north**'、'**center**' 其中之一，并不能重复，这就定义了该 **div** 在页面上所在的布局位置。

⌋ height/width 属性: region 为 north 或 south 的 div 需要定义 height 属性, 指定占用页面上下的高度是多少, 这里定义 width 属性没有作用, 宽度始终是 100%; region 为 east 或 west 的 div 需要定义 width 属性, 指定占用页面左右的宽度是多少, 这里定义 height 属性没有作用, 高度始终是除去 north 和 south 高以后的 100%; region 为 center 的 div 不需要定义 height 和 width, 它的部分始终是其它部分占用后剩余的部分。

⌋ border 属性: 5 个布局 div 都可以设置整个属性, 默认是为 true, 如果不想布局区域有边框就设为 false。

⌋ title 属性: 在布局 div 中定义 title 可以在整个区域上添加一个标题头, 该属性指定标题头的内容。

⌋ split 属性: 该属性默认为 'false', 布局 div 中定义 split 为 'true' 表示该区域可以通过拖拉边框来实现伸缩, region 为 north 和 south 的区域高度可以伸缩; region 为 east 和 west 的区域宽度可以伸缩, 该属性对 center 区域无效。

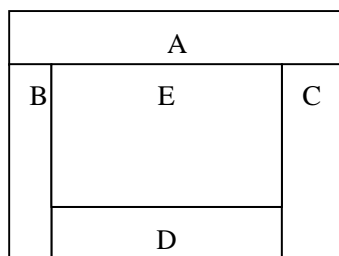
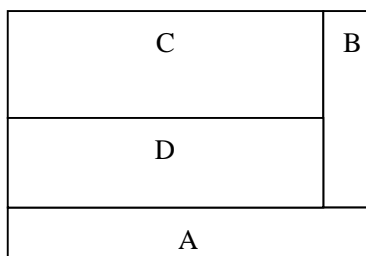
⌋ minSize/ maxSize 属性: 该属性指定了区域拖动范围的最小/大值, 这两个属性只有在 split 属性为 'true' 时, 设置这两个属性才有效。

⌋ collapsible 属性: 该属性默认为 'false', 布局 div 中定义 collapsible 为 'true' 表示该区域缩回, 这里注意, 在页面上对区域的缩回是需要通过点击区域的标题头实现的, 所以如果没有设置 title 属性, 就无法通过鼠标来完成区域的缩回。该属性对 center 区域无效。

⌋ collapseMode 属性: 只有在 collapsible 属性为 'true' 时才有意义, 它指定了区域缩回的方式, 这里要么不设置, 要设置就只能为 'mini'。

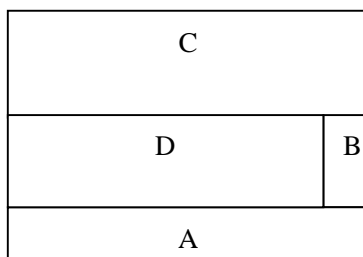
再布局

通过第一节现在就可以对整个页面进行一次布局, 但是有些时候需要的布局仅仅依靠 5 个区域是无法实现的或是 5 个区域的分布不是我们所希望的, 例如:

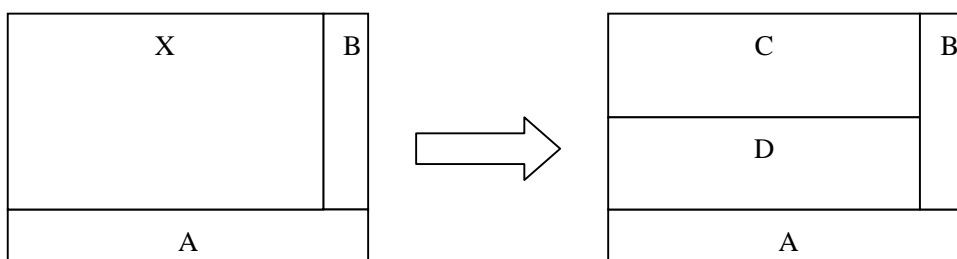


这里图片显示的布局都无法运用前面提到的简单布局方式实现，那么就需要对页面进行再次布局，在介绍布局方式以前，先来看看实现布局的思路。

从第一个图来看，A 区域可以对应到 **south** 部分，B 区域可以对应到 **east** 部分（注意前面已经提到，不是 5 个部分都必不可少），如果这个时候把 C 设为 **north** 部分，那么布局就会形成如下样式：



这个就是因为布局区域优先级的问题，所以现在要改变布局方式，还是将 A 区域可以对应到 **south** 部分，B 区域可以对应到 **east** 部分，而 C、D 两个区域先不管，合并起来对应到 **center** 部分，形如：

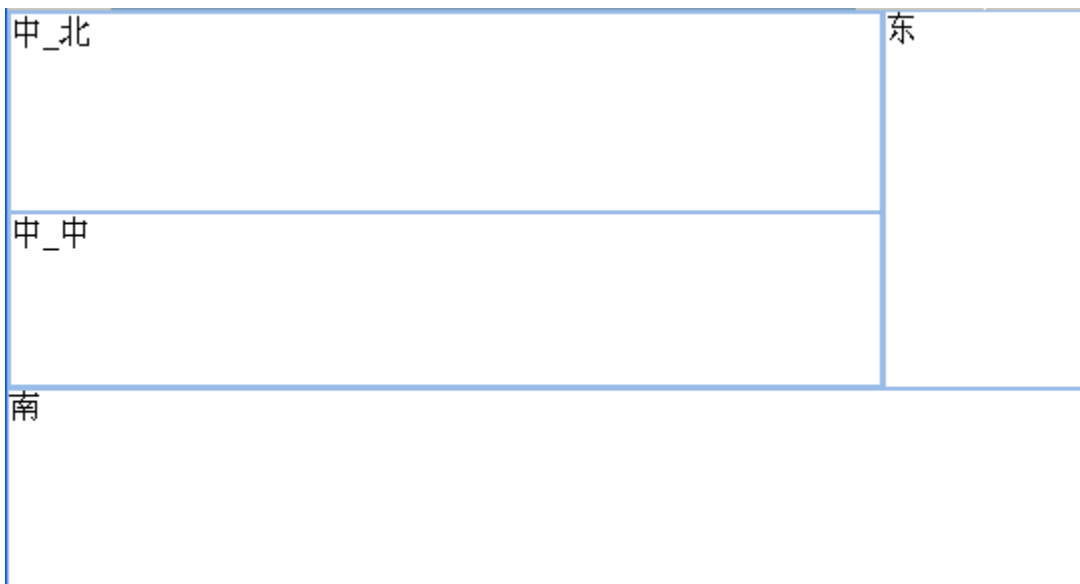


然后我们将 **center** 对应的 X 区域再一次布局，将 C 区域对应到 X 区域的 **north** 部分，D 区域对应到 X 区域的 **center** 部分。

思路已经很清晰了，再来看如何实现。第一步将 A 区域可以对应到 **south** 部分，B 区域可以对应到 **east** 部分的方式和前面一样，关键来看第二步，如何将 C

区域对应到 X 区域的 north 部分，D 区域对应到 X 区域的 center 部分：

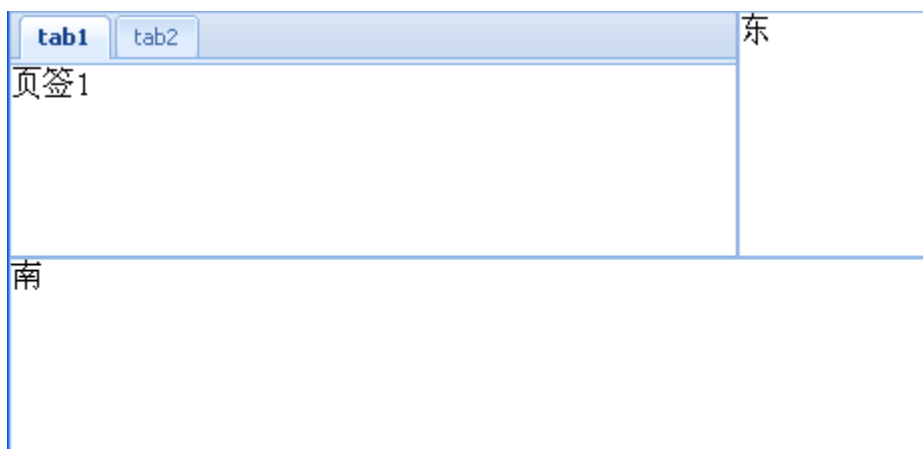
```
<div region="south" height="100">南</div>
<div region="east" width="100">东</div>
<div region="center">
  <div region="north" height="100">中_北</div>
  <div region="center">中_中</div>
</div>
```



在布局默认情况是 `xtype="panel"` (`xtype` 请查看 `Efs.xtype`)，您还可以指定 `xtype` 类型。如：

`tabpanel` 在布局：

```
<div region="south" height="100">南</div>
<div region="east" width="100">东</div>
<div region="center" xtype="tabpanel">
  <div title="tab1">页签1</div>
  <div title="tab2">页签2</div>
</div>
```



类似的还可以是 `xtype="grid"` `xtype="treepanel"` 等，`xtype` 的直接子节点的配置请参照对应的章节。

第四章 Efs 命名空间

Efs 拷贝了 Ext 的属性和方法，而且扩展了一些方法。常用的方法有：

getDom(el) :

描 述：获取页面（document）节点对象

参数 el： 节点 ID 或者节点对象

返 回： 页面（document）节点对象

getEfs(el):

描 述：获取 Efs 对象

参数 el： 节点 ID 或者节点对象（必须是 `xtype` 节点）

返 回： Efs 对象

getExt(el):

描 述: 获取 Ext 对象

参数 el: 节点 ID 或者节点对象 (必须是 xtype 节点)

返 回: Ext 对象

第五章 核心 panel 面板容器

Efs 的核心就是 panel 面板容器, 很多面板都是继承它。Panel 容器有的插件, 其他继承类容器也拥有。

若要 border 布局外, 单独创建一个 panel 容器, 只需要在节点上加上 `xtype="panel"` 即可, Efs 框架将会帮您创建 Ext.Panel 并渲染到 xtype 节点上。

简单的 panel 容器

在 document 上的任意 div 节点上加入 `xtype="panel"`, 即能创建出一个 panel 容器。如:

```
<div id="mypanel" xtype="panel" height="100" title="简单的panel容器">  
    这是最简单的panel容器!  
</div>
```



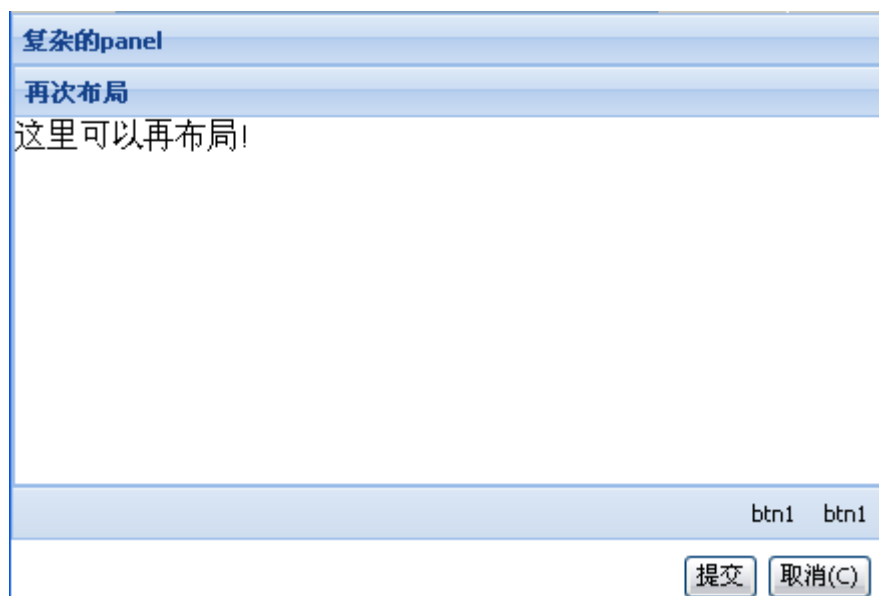
简单的panel容器
这是最简单的panel容器!

复杂的 panel 容器

Panel 容器有 tbar、bbar 等，Efs 配置对一些常用的进行封装：tbar、bbar、buttons。我们把这样的对象成为“插件”

样例如下：

```
<div id="mypanel" xtype="panel" height="300" title="复杂的panel">
  <div xtype="bbar">
    <div text="->"></div>
    <div text="btn1"></div>
    <div text=" "></div>
    <div text="btn1" onEfsClick="alert(2);"></div>
  </div>
  <div xtype="buttons">
    <div text="提交"></div>
    <div text="取消" onEfsClick="alert(3);"></div>
  </div>
  <div region="center" title="再次布局">
    这里可以再布局!
  </div>
  <!-- 或者放普通html代码-->
  <!--
  <form action="" method="post">
    .....
  </form>
  -->
</div>
```



在 `xtype="panel"` 的直接子节点下加入 `xtype="tbar"`、`xtype="bbar"`（子节点写法请参照 `toolbar`）或 `xtype="buttons"`（子节点写法请参照 `button`）

获取 panel 容器对象

如果想要获取 `panel` 容器 `ext` 对象，则需要在 `xtype` 节点上定义一个 `id`，然后通过调用 `Efs.getExt("mypanel")` 获取。

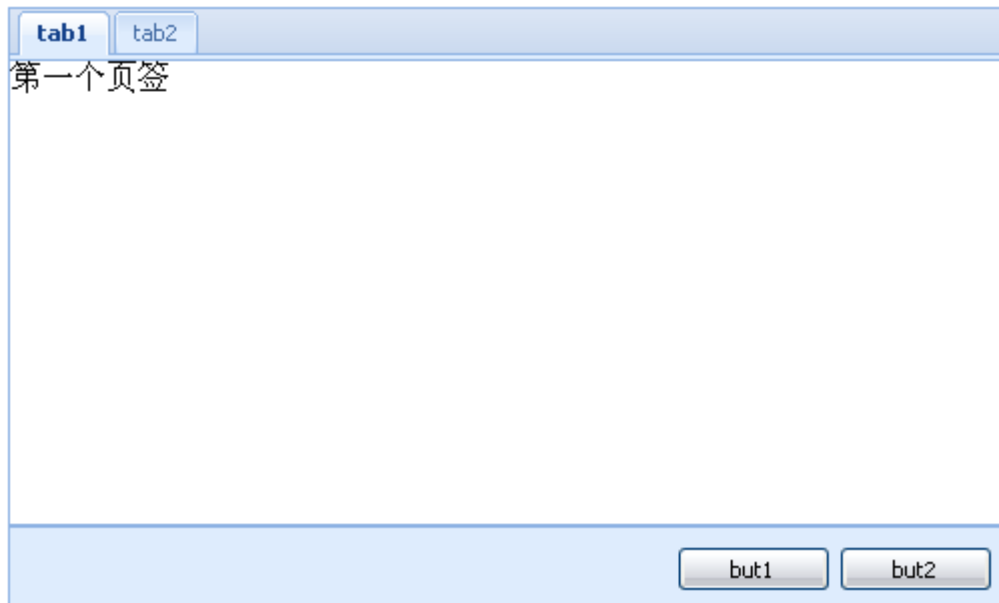
第六章 tabpanel 页签容器

`Efs.Tabpanel` 继承了 `Efs.Panel`，因此 `panel` 封装过的插件，在 `tabpanel` 上也适用。要创建一个 `tabpanel`，只需要在 `div` 节点上加入 `xtype="tabpanel"`

简单的 tabpanel 容器

`xtype="tabpanel"` 节点的子节点加入 `title` 属性，每个 `title` 节点作为 `tabpanel` 的一个页签。`Title` 节点默认的 `xtype="panel"`，可以指定其他的 `xtype`。

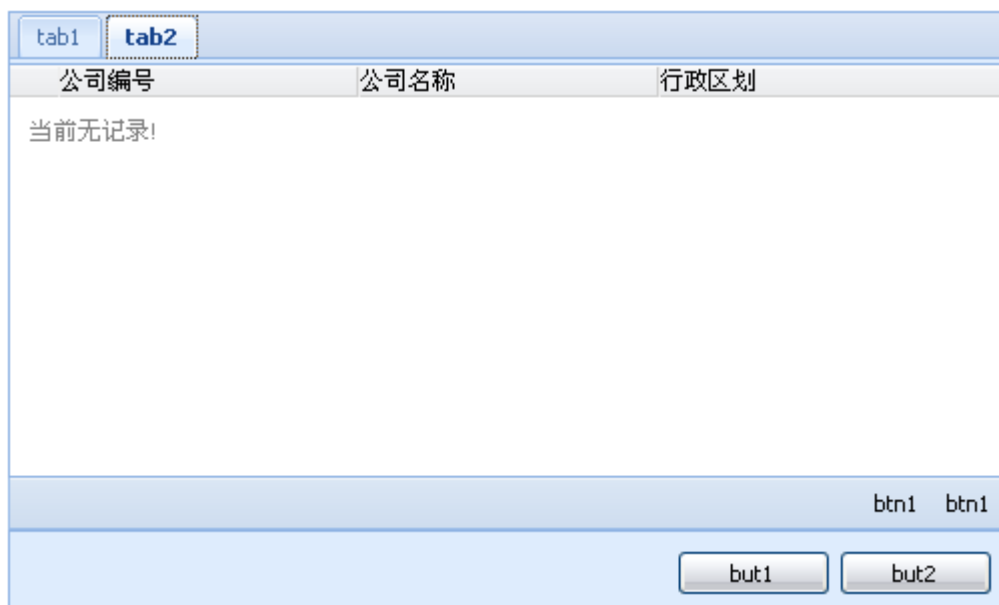
```
<div xtype="tabpanel" width="500" height="300" title="father"
buttons="[{text: 'but1'}, {text: 'but2'}]" >
  <div id="tab1" title="tab1">第一个页签</div>
  <div id="tab2" title="tab2">第二个页签</div>
</div>
```



复杂的 tabpanel 容器

```
<div xtype="tabpanel" width="500" height="300" title="father"
buttons="[{text: 'but1'}, {text: 'but2'}]" >
  <div xtype="bbar">
    <div text="->"></div>
    <div text="btn1"></div>
    <div text=" "></div>
    <div text="btn1" onEfsClick="alert(2);"></div>
  </div>
  <div id="tab1" title="tab1">第一个页签</div>
  <div id="tab2" title="tab2" xtype="grid">
    <div xtype="groupstore" url="your_url">
      <div xtype="xmlreader" recordid="SSCID" record="ROW">
        <div name="SSCID" mapping="SSCID"></div>
        <div name="SSCNAME" mapping="SSCNAME"></div>
        <div name="REGADDRCODEDES" mapping="REGADDRCODEDES"></div>
        <div name="REGADDRESS" mapping="REGADDRESS"></div>
        <div name="OPTIME" mapping="OPTIME"></div>
      </div>
    </div>
    <div xtype="colmodel">
      <div header="公司编号" width="150" sortable="true" dataIndex="SSCID"></div>
      <div header="公司名称" width="150" sortable="true" dataIndex="SSCNAME"></div>
      <div header="行政区划" width="200" sortable="true" dataIndex="REGADDRCODEDES"></div>
      <div header="注册地详址" width="200" sortable="true" dataIndex="REGADDRESS"></div>
      <div header="操作时间" dataIndex="OPTIME" width="150"></div>
    </div>
  </div>
</div>
```

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



第七章 Grid 网格容器

Grid 容器是一个常用的容器，它继承了 Panel 容器。要构造一个 grid 必须要有两个对象：store（数据源）和 colmodel（列模式）。

xtype="grid" 构造 Grid，xtype="grid" 的直接子节点必须要有 xtype="store" 或 xtype="groupstore"，xtype="colmodel"。

GridPanel 构造

xtype="grid" 节点配置：

```
<div xtype="grid" title="Grid网格容器" width="500" height="400" onEfsRowClick = "doRowClick()">
    .....
</div>
```

Ext 常用配置：

autoHeight : Boolean
autoWidth : Boolean
bodyStyle : String
border : Boolean
buttonAlign : String
iconCls : String
id : String
title : String

height : Number

width : Number

扩展配置:

xtype: String 构造类型 (必须)

region: String 'east'、'south'、'west'、'north'、'center'其中之一, 配置此属性, 将该容器用于布局. 具体用法参照"页面布局"

pagingBar: Boolean 当为 true 时, 有分页工具条, 显示 "displayMsg" 信息, 优先级高于 simplePagingBar

simplePagingBar: Boolean 当为 true 时, 有分页工具条, 不显示 "displayMsg" 信息

pageSize: Number Grid 一页显示记录数, 默认为 10

elStyle : String 相当于 Ext 原始配置的 style, 整个 panel 样式

loadMask: String 加载遮罩效果 默认 msg:"加载..."

Ext 常用事件:

activate : (Ext.Panel p)

disable : (Ext.Component this)

enable : (Ext.Component this)

titlechange : (Ext.Panel p, String The)

扩展事件:

cellclick : (Object data, Grid this, Number rowIndex, Number columnIndex, Ext.EventObject e)

Object data: 通过 data[xmlreader 的 name]获取数据, 如: data["SSCID"] 获取公司编号. 以下相同

celldblclick : (Object data, Grid this, Number rowIndex, Number columnIndex, Ext.EventObject e)

rowclick : (Object data, Grid this, Number rowIndex, Ext.EventObject e)

rowdblclick : (Object data, Grid this, Number rowIndex, Ext.EventObject e)

数据源 store

xtype="groupstore/store" 默认为分组 store, 其直接子节点必须要定义 xtype="xmlreader/reader" 读取模式. 读取模式为 xml 结构.

xtype="store" 或 xtype="groupstore" 节点的配置:

```
<div xtype="groupstore" url="your_url" autoLoad="true">
  <div xtype="xmlreader" recordid="SSCID" record="ROW">
    <div name="SSCID" mapping="SSCID"></div>
    <div name="SSCNAME"></div>
  </div>
</div>
```


autoLoad: Boolean 自动请求数据 默认为 false
url:String xml 数据请求地址
recordid: String 数据结构 id, 如果不调用 getDelXml() 方法可以忽略。
record: String 数据节点, 通常为表名
mapping: String record 节点的子节点, 通常为字段名
name: String 列模式 dataIndex 绑定名, 如果 mapping 和 name 相同, 可以只写 name

列模式 colmodel

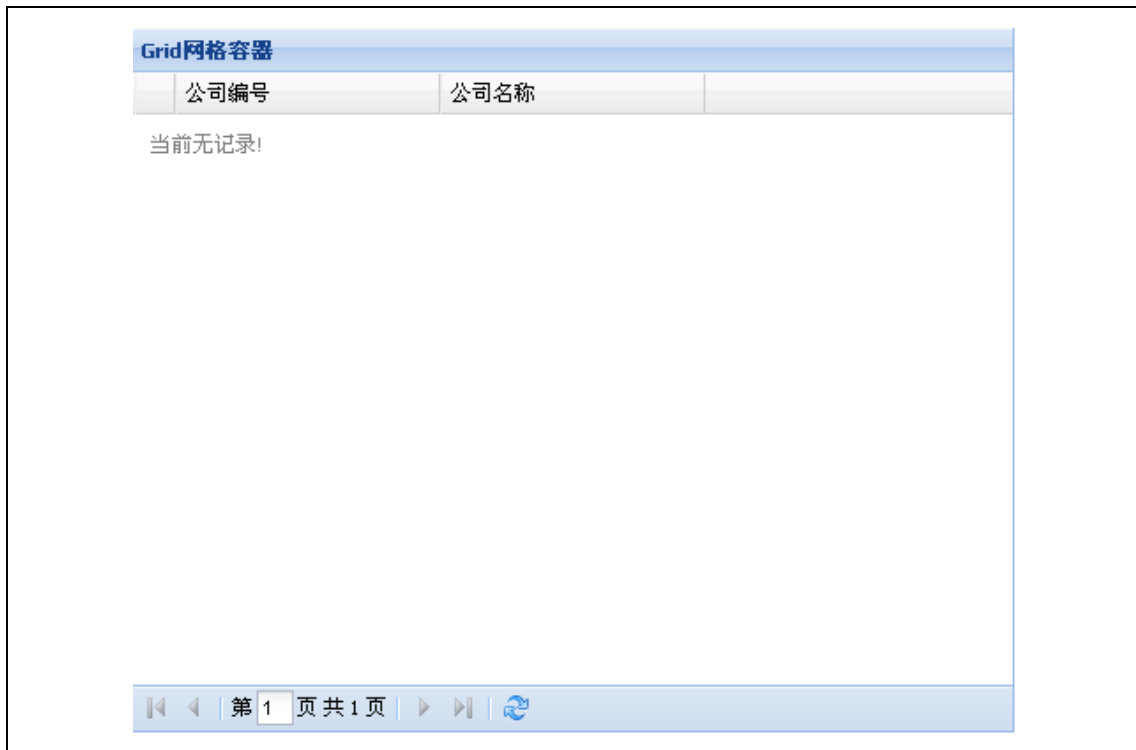
xtype="colmodel" 的直接子节点解析成 grid 列模式配置.xtype="colmodel" 只能在 grid 下使用
xtype="colmodel" 节点的配置:

```
<div xtype="colmodel">  
  <div header="公司编号" width="150" sortable="true" dataIndex="SSCID"></div>  
  <div header="公司名称" width="150" sortable="true" dataIndex="SSCNAME"></div>  
</div>
```

header: String 显示列文本
dataindex:String 绑定数据源中的 name

样例及显示效果

```
<div xtype="grid" title="Grid网格容器" width="500" height="400" pagingBar="true"  
onEfsRowClick="doRowClick()">  
  <div xtype="groupstore" url="your_url" autoLoad="true">  
    <div xtype="xmlreader" recordid="SSCID" record="ROW">  
      <div name="SSCID" mapping="SSCID"></div>  
      <div name="SSCNAME"></div>  
    </div>  
  </div>  
  <div xtype="colmodel">  
    <div header="公司编号" width="150" sortable="true" dataIndex="SSCID"></div>  
    <div header="公司名称" width="150" sortable="true" dataIndex="SSCNAME"></div>  
  </div>  
</div>
```



第八章 Treepanel 树面板

treepanel 继承了 panel。Treepanel 有两种模式，一种是静态的节点 root 模式，一种是加载动态 XML 模式。（只能选其中一种模式）

Treepanel 构造

要构造一个 treepanel，首先配置 xtype="treepanel"，其次配置其 node 节点或者 loader 节点。

xtype="treepanel"节点配置：

```
<div xtype="treepanel" title="treepanel 面板" width="500" height="400"
onEfsClick="doTreeClick()">
    .....
</div>
```

Ext 常用配置：

autoScroll: Boolean
autoHeight : Boolean
autoWidth : Boolean
bodyStyle : String
border : Boolean

```

buttonAlign : String
collapsedCls : String
collapsible : Boolean
collapseMode: String 只有在 collapsible 属性为'true'时才有意义，它指定了
区域缩回的方式，这里要么不设置，要设置就只能为'mini'
iconCls : String
id : String
title : String
height : Number
width : Number

```

扩展配置:

```

xtype: String 构造类型（必须）
region: String 'east'、'south'、'west'、'north'、'center'其中之一，配置
此属性，将该容器用于布局.具体用法参照"页面布局"
elStyle : String 相当于 Ext 原始配置的 style，整个 panel 样式

```

Ext 常用事件:

```

activate : ( Ext.Panel p )
disable : ( Ext.Component this )
enable : ( Ext.Component this )
titlechange : ( Ext.Panel p, String The )
beforeclick : ( Node node, Ext.EventObject e )
beforecollapse : ( Ext.Panel p, Boolean animate )
beforecollapsenode : ( Node node, Boolean deep, Boolean anim )
beforeexpand : ( Ext.Panel p, Boolean animate )
beforeexpandnode : ( Node node, Boolean deep, Boolean anim )
beforeload : ( Node node )
click : ( Node node, Ext.EventObject e )
collapse : ( Ext.Panel p )
dblclick : ( Node node, Ext.EventObject e )
expand : ( Ext.Panel p )
expandnode : ( Node node )
load : ( Node node )
textchange : ( Node node, String text, String oldText )

```

xtype="root" 节点

只需要定义根节点 `xtype="root"`，如果其有子节点，直接当成功能树的子节点。`xtype="root"` 只能在 `treepanel` 下使用。

`xtype="root"` 节点配置:

```
<div id="root" text="rootNode" xtype="root" expanded="true">
```

```
<div text="node1" expanded="true" onEfsClick="doNodeClick()">
  <div text="node11">
    <div text="node111" opurl="a.html"></div>
  </div>
  <div text="node12"></div>
</div>
<div text="node2"></div>
</div>
```

Ext 常用配置:

id : String

text:String 节点显示文本

icon : String

iconCls : String

isTarget : Boolean

expanded: Boolean 是否展开, 默认 false

leaf : Boolean 是否叶节点, 默认 false

singleClickExpand : Boolean 是否单击展开, 默认 false

常用事件:

beforeclick : (Node this, Ext.EventObject e)

beforecollapse : (Node this, Boolean deep, Boolean anim)

beforeexpand : (Node this, Boolean deep, Boolean anim)

click : (Node this, Ext.EventObject e)

collapse : (Node this)

dblclick : (Node this, Ext.EventObject e)

expand : (Node this)

textchange : (Node this, String text, String oldText)

xtype="loader" 节点

只需要定义根节点 `xtype="root"`, 如果其有子节点, 直接当成功能树的子节点. `xtype="loader"` 这个 `xtype` 只能在 `treepanel` 下使用。

xtype="loader" 节点配置:

```
<div xtype="loader" url="#" onEfsBeforeload="beforeTreeLoad()"></div>
```

Ext 常用配置:

url:String xml 数据请求地址

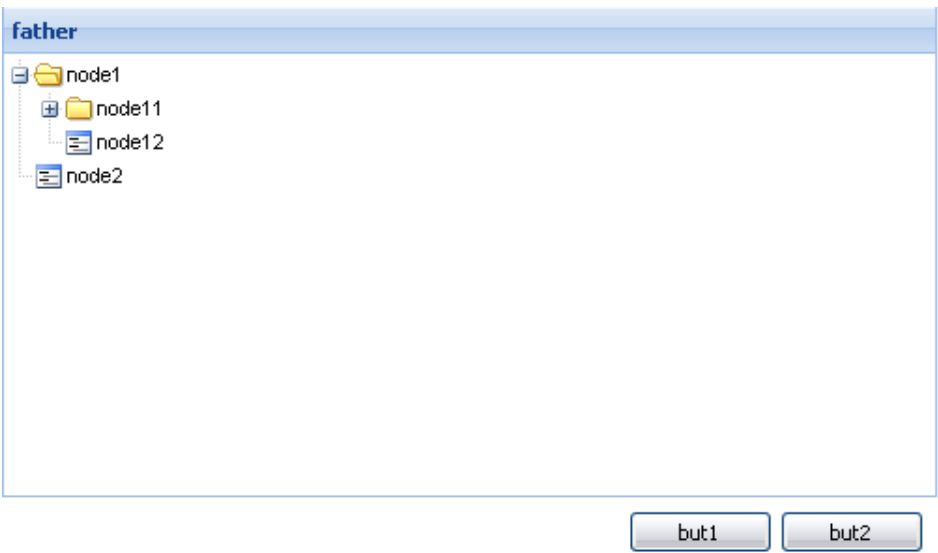
baseParams : Object http 参数, baseParams="{myparam:'发送到后台的参数'}"

常用事件:

load(Ext.tree.TreeNode node, Function callback)

样例及显示效果

```
<div xtype="treepanel" name="name" width="500" height="300" title="father"
buttons="[{text: 'but1'}, {text: 'but2'}]" onEfsClick="doClick()">
  <div id="root" text="rootNode" xtype="root" expanded="true">
    <div text="node1" expanded="true">
      <div text="node11"><div text="node111" opurl="a.html"></div></div>
      <div text="node12"></div>
    </div>
    <div text="node2"></div>
  </div>
</div>
```



第九章 Window 窗口面板

Window 窗口继承了 panel，默认的窗口是浮动并且隐藏的，关闭窗口后默认不销毁 window 对象，而是把它隐藏。窗口也可以作 border 再布局。

Window 窗口构造

要构造一个 window 窗口，需要配置 `xtype="window"`，window 直接子节点配置与 panel 相同。

`xtype="window"` 节点配置

```
<div id="ww" xtype="window" width="500" height="400" title="window" autoShow="true">
    .....
</div>
```

Ext 常用配置:

autoShow: Boolean
autoScroll: Boolean
autoHeight : Boolean
autoWidth : Boolean
bodyStyle : String
border : Boolean
buttonAlign : String
closable : Boolean
closeAction : String
onEsc : Function
iconCls : String
id : String
title : String
titleCollapse : Boolean
height : Number
width : Number

扩展配置:

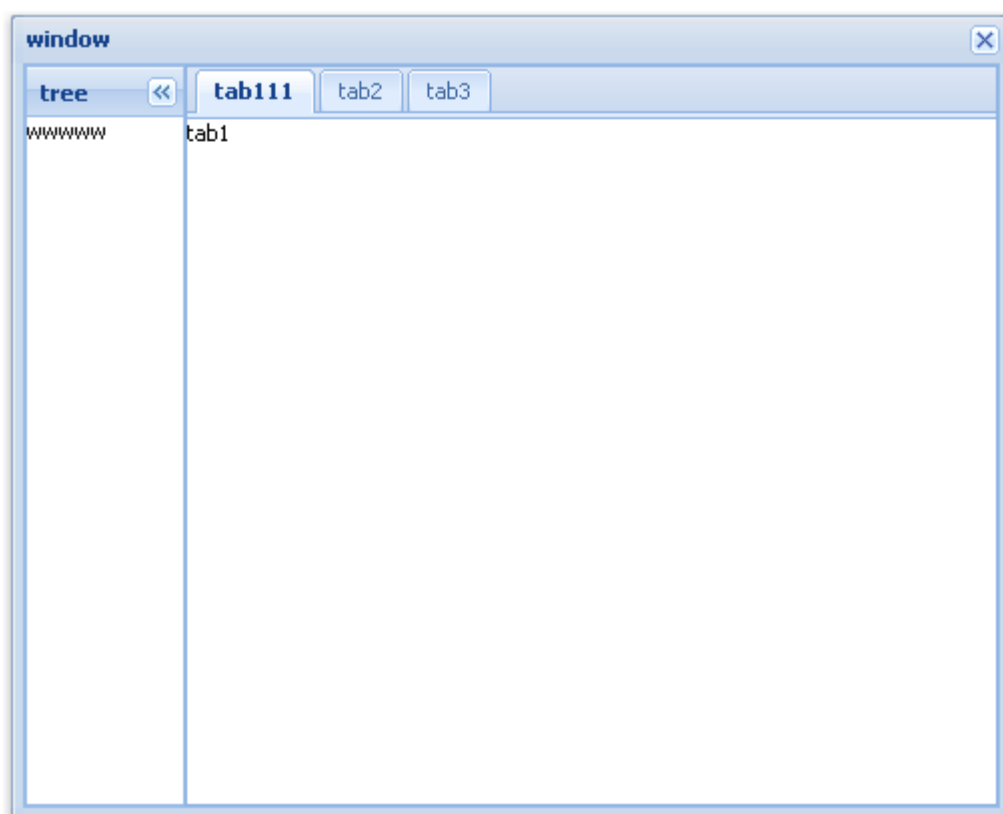
xtype: String 构造类型 (必须)
region: String 'east'、'south'、'west'、'north'、'center' 其中之一, 配置此属性, 将该容器用于布局. 具体用法参照"页面布局"
elStyle : String 相当于 Ext 原始配置的 style, 整个 panel 样式

Ext 常用事件:

activate : (Ext.Panel p)
disable : (Ext.Component this)
enable : (Ext.Component this)
titlechange : (Ext.Panel p, String The)
beforeclose : (Ext.Panel p)
beforecollapse : (Ext.Panel p, Boolean animate)
beforeexpand : (Ext.Panel p, Boolean animate)
beforehide : (Ext.Component this)
close : (Ext.Panel p)
collapse : (Ext.Panel p)
disable : (Ext.Component this)
enable : (Ext.Component this)
expand : (Ext.Panel p)
hide : (Ext.Component this)
move : (Ext.Component this, Number x, Number y)
show : (Ext.Component this)

样例及显示效果

```
<div id="ww" xtype="window" width="500" height="400" title="window" autoShow="true">
  <div id="w1" region="west" width="80" title="tree" collapsible="true">www</div>
  <div id="c1" region="center" xtype="tabpanel">
    <div title="tab111">tab1</div>
    <div title="tab2">tab2</div>
    <div title="tab3">tab3</div>
  </div>
</div>
```



第十章 Form 表单

form 表单包含了常用的几个元素：text, hidden, textarea

Form 对象构造

构造 form 对象不需要指定 xtype，直接写 form 标签即可。配置如下：

```
<form id="frmPost" url="add.do" method="post" onEfsBeforeaction="doFocus1()"
onEfsFailure="alert('www')">
    .....
</form>
```

Ext 常用配置：

id : String

扩展配置：

action: String 提交地址，非异步提交

url: String 提交地址，主要用于异步提交，action 和 url 两者选其一。

method: String 提交方式 post/get

baseParams: Object http 请求参数,格式如:baseParams="{foo1:'val1',foo2:'val2'}"

Ext 常用事件：

actionfailed : (Form this, Action action)

beforeaction : (Form this, Action action)

扩展事件：

success : (Ext.form.BasicForm form,Ext.form.Action action) 用于监听异步提交成功返回

failure : (Ext.form.BasicForm form,Ext.form.Action action) 用于监听异步提交失败返回

样例及效果

```
<table width="600">
    <tr>
        <td>
            <form id="frmData" class="efs-box" method="post">
                <TABLE class="formArea">
                    <tr>
                        <td width="80">用户名称</td>
                        <td width="160">
                            <input type="text" kind="text" fieldname="USERLIST/USERTITLE"
state="0" datatype="0" maxlength="30" value="" must="true">
                        </td>
                        <td width="20">&nbsp;</td>
                        <td width="80">用户姓名</td>
                        <td width="160">
                            <input type="text" kind="zhuni code" fieldname="USERLIST/USERNAME"
```



```
state="0" datatype="0" maxlength="30" must="true">
    </td>
    <tr>
        <td>出生日期</td>
        <td>
            <input type="text" kind="date" id="birthdate"
fieldname="USERLIST/BIRTHDAY" state="0" datatype="3">
        </td>
        <td width="20">&nbsp;</td>
        <td>民族</td>
        <td>
            <input type="text" kind="dic" src="DIC_NATIVE"
fieldname="USERLIST/NATION" state="0" datatype="0">
        </td>
    </tr>
</tr>
</TABLE>
</form>
</td>
</tr>
</table>
```

用户名称 用户姓名

出生日期 民族

字典面板

混 请输入查询关键字

编码	字典值
01	汉族
02	蒙古族
03	回族
04	藏族
05	维吾尔族
06	苗族
07	彝族

第 1 页 共 6 页

form 常用方法

submit(mix, isQry)

参数mix: 混合型参数

Boolean 为true 时忽略组织xml, 后台将获取不到txtXML值

String 如果为字符串, 忽略组织form表单域xml, 直接将该值赋予txtXML

参数isQry: Boolean 为true 时将组织查询xml, 只有mix参数不为true情况下有效。

用法实例：

1、Efs.getExt(“frmData”).isValid();

验证 id 为 frmData 的 Form 中的所有表单输入是否有效，主要根据表单的 kind 属性类型进行验证；

2、Efs.getExt(“frmData”).reset();

清空 id 为 frmData 的 Form 中的所有表单输入的值；

3、Efs.getExt(“frmData”).submit();

将 id 为 frmData 的 Form 中的所有 input、textarea 表单根据其 fieldname 属性值，组织出标准的业务操作型 XML 接口数据临时保存到 txtXML 表单中提交（txtXML 表单会自动创建），后台即可通过 Request(“txtXML”)获得标准 XML 结构数据。

4、Efs.getExt(“frmData”).submit(sValue);

忽略组织 form 表单域 xml，直接将 sValue 值赋予 txtXML 表单传递到后台组件。

5、Efs.getExt(“frmData”).submit(false,true);

将 form 表单域中的输入值组织成标准的查询型 xml，并将组织好的 xml 赋予 txtXML 表单传递到后台组件,进行查询操作。

第十一章 Toolbar 工具条

前面所用的 tbar、bbar 是构造了 Toolbar 对象。toolbar 工具条中可以放 button、menu 以及原始的 html 片段。

Toolbar 构造

在页面上配置 xtype="toolbar"，可以将工具条渲染到 xtype="toolbar"这个节点上。
xtype="toolbar"节点配置：

```
<div xtype="toolbar">
    .....
</div>
```

Ext 常用配置：

id : String
autoHeight : Boolean
autoWidth : Boolean
height : Number
width : Number

扩展配置：

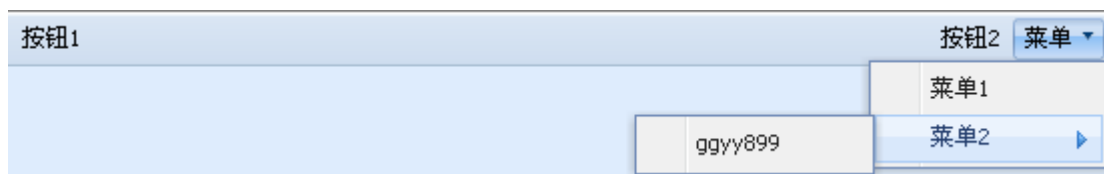
xtype: String 构造类型（必须）
elStyle : String 相当于 Ext 原始配置的 style, 整个 panel 样式

Ext 常用事件:

beforehide : (Ext.Component this)
beforeshow : (Ext.Component this)
disable : (Ext.Component this)
enable : (Ext.Component this)
hide : (Ext.Component this)
show : (Ext.Component this)

样例及显示效果

```
<div xtype="toolbar">
  <div text="按钮1"></div>
  <div text="->"></div>
  <div text="按钮2"></div>
  <div text="菜单">
    <div text="菜单1"></div>
    <div text="菜单2">
      <div text="ggyy899" onEfsClick="alert('ggyy899')"></div>
    </div>
  </div>
</div>
```



第十二章 Button 按钮和 Menu

button 的构造有两种方式，一种是直接写 `<input type="button">`，另一种是 `xtype="button"`。

Button 添加快捷键在文本值后加上 **#快捷键字母**，默认的快捷键：完成、确定（O）；取消、关闭（C）；查询（Q）；返回（B）；转到（G）；保存（S）；打印设置（N）。

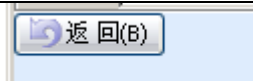
menu 的构造是建立在 button 基础上的，含有子节点的 button 即为 menu。因此要构造 menu 必须用 `xtype="button"` 方式，`input type="button">` 没有子节点。

Button 构造

第一种方式:

直接将 html 原始 button 对象渲染成 Ext.Button 对象。例如:

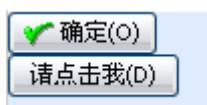
```
<input type="button" iconCls="icon-back" value="返回" onEfsClick="doClick()">
```



第二种方式:

Div 节点上配置 xtype="button"。例如:

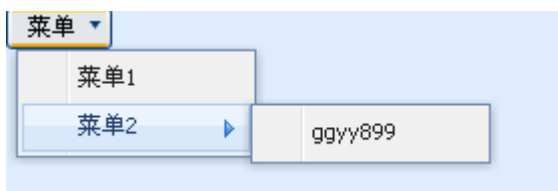
```
<div xtype="button" iconCls="icon-ok" text="确定" onEfsClick="doClick()"></div>
<div xtype="button" text="请点击我#D" onEfsClick="doClick()"></div>
```



Menu 构造

给 button 加上子节点即可得到 menu，子节点上不需要加入 xtype，配置同 Button。例如:

```
<div xtype="button" text="菜单" onEfsClick="doClick()">
  <div text="菜单1"></div>
  <div text="菜单2">
    <div text="ggyy899"></div>
  </div>
</div>
```



第十三章 Efs 事件

在页面上属性以 onEfs 开头的为事件。

onEfs + 大写开头的 Ext 事件名

```
<div xtype="window" height="400" width="300" onEfsShow="doWinShow()">
  .....
</div>
```

```
</div>
<script type="text/javascript">
    function doWinShow(){
        //监听函数
    }
</script>
```

第十四章 Efs.Common 常用方法

createDocument(strData, isUrl)

描述：创建 dom 对象

参数 **strData**: mix 混合型参数

XML 字符串 【可选】 或者 URL（第二个参数为 true）【可选】

参数 **isURL**: Boolean 是否为 url 请求数据 【可选】

返回：dom 对象

getNode(objXML, nodePath, index)

描述：获得指定路径下的节点对象

参数 **objXML** Object dom 对象

参数 **nodePath** String 选择节点路径

参数 **index** Int 存在多个节点时，指定节点的索引，默认为 0

返回：Object 节点对象

getNodeValue(objXML, nodePath, index)

描述：获得指定路径下的节点对象的 text 值，找不到返回""

参数 **objXML** Object dom 对象

参数 **nodePath** String 选择节点路径

参数 **index** Int 存在多个节点时，指定节点的索引，默认为 0

返回：String 节点对象的 text 值,或""

setAttribute(objXML, nodePath, attributeName, attributeValue)

描述：设置指定路径下的节点对象的属性值

参数 objXML Object dom 对象

参数 nodePath String 选择节点路径

参数 attributeName String 设置的属性名称

参数 attributeValue String 设置的属性值

返回：Boolean 是否成功

selectSingleNode(objXML, nodePath)

描述：根据选择路径获取文档对象中的首个节点

参数 objXML Object dom 对象

参数 nodePath String 选择节点路径

返回：node 节点

selectNodes(objXML, nodePath)

描述：根据选择路径获取文档对象中的节点数组

参数 objXML Object dom 对象

参数 nodePath String 选择节点路径

返回：node 数组

setText(node, value)

描述：设置节点对象的值

参数 node Object dom 的节点对象

参数 value String 赋予节点的值

getText(node)

描述：获得节点对象的值

参数 node Object dom 的节点对象

返回：String

getXML(objXML)

描述：获取文档对象的 xml 字符串结构

参数 objXML Object dom 对象

返回：文档对象的 xml 字符串结构

getOpXml(mix, isDom)

描述：组织输入框的值成标准 xml 结构返回

参数：mix 混合型参数 如果是 form 对象(可以是 Efs 的 form, Ext 的 form, document 的 form), 组织该 form 内的输入框值, 否则组织指定 mix 为 ID 的区域输入框的值

参数：isDom 默认返回字符串, 如果是 true 返回 Dom 对象

返回：组织好的 XML 结构

样例：Efs.Common.getOpXml(); //表示获取 document 对象中所有含有 fieldname 属性对象值组织成 xml 结构

getQryXml(form, emptyRet)

描述：组织查询 xml 结构

参数：form mix 混合参数 Ext.form.BasicForm|document 的 form 对象|Efs.BasicForm

参数：emptyRet Boolean 是否返回空串, 默认为 false 返回标准的空查询条件, 标准空查询条件请参照 Efs.Common.getEmptyQryXml()方法

返回：组织好的 XML 结构

getEmptyQryXml()

描述：获取空查询条件 xml 结构

返回：返回标准的空查询条件 "<?xml version='1.0'?><EFSFRAME efsframe='urn=www-efsframe-cn' version='1.0'><QUERYCONDITION/></EFSFRAME>"

setEditValue(xmlDoc, form, rootEl)

描述：设置 form 表单中的输入框值, 通常在做修改操作时候用。

参数：xmlDoc Sting|Object 标准的查询返回 XML 结构字符串, 或者标准查询返回 Dom 对象

参数：form mix 混合参数 Ext.form.BasicForm|document 的 form 对象|Efs.BasicForm

参数：rootEl 指定跟节点名称

返回：返回标准的空查询条件 "<?xml version='1.0'?><EFSFRAME

efsframe='urn=www-efsframe-cn' version='1.0'><QUERYCONDITION/></EFSFRAME>"

getRootXml(isDom)

描述: 获取根节点 xml 结构

参数: isDom Boolean 是否返回 dom 对象, 默认返回 XML 字符串

返回: 根节点 xml 结构

addRootXml: function(xmlDoc, isDom)

描述: 给指定的 XML 结构增加标准根节点

参数: xmlDoc String|Dom XML 字符串或者 XML 结构 DOM 对象

参数: isDom Boolean 是否返回 Dom 对象, 默认返回 XML 字符串

返回: 增加根节点后的 xml 结构

setSpanValue: function(sXML, sRoot, where)

描述: 设置 Span 值, 通常用于信息查看, 传入一个标准的查询返回结构, 将数据设置到指定的 SPAN 中

参数: sXML mix 混合型参数, 可以是 XML 字符串, 也可以是 DOM 对象

参数: sRoot String 指定根节点 xpath, 默认为 "/"

参数: where mix 指定从哪个节点 ID 或者节点开始设置值 默认为 document

样例:

js:

```
Efs.Common.setSpanValue("<EFSDOC><QUERYINFO><TAB><FIELD1>显示在 SPAN 上的文字</FIELD1></TAB></QUERYINFO></EFSDOC>", "EFSDOC/QUERYINFO", Efs.getDom("show"));
```

html:

```
<DIV id = "show"><span id="test1" fieldname="TAB/FIELD1"></span></DIV>
```

ajax(url, param, callback, scope)

描述: 通用异步请求函数

参数: url String 请求地址

参数: param mix 如果是 String 类型, 请求以 txtXML 发送, 否则该类型必须为 Object 类型。

如: {foo:'123'}

参数: callback function 回调函数, 该函数会自动注入 3 个参数, succ: 为是否成功,

response: XMLHttpRequest 请求返回值对象, options: 请求时的方法参数, 较少用

参数: scope Object 回调函数 callback 的 this 对象【可选】

样例: Efs.Common.ajax("test.do?method=add","这里是 txtXML 值",cb);

//回调函数

```
function cb(succ,response,options){  
  
}
```

load(cfg,callback)

描述: 动态载入 URL 的页面内容

参数: cfg mix 混合型参数。如果为 string 类型, 则被认为 URL, 否则是一个配置对象

配置对象如: cfg = {

```
    renderTo: 'myDiv',      // div 节点, 默认为 document.body  
    url: 'test.html',      // url 请求地址, 必须  
    callback: function(){  // 回调函数, 可选  
  
    }  
}
```

参数: callback function 回调函数, 只当 cfg 为 string 类型时有效

例如: load("test.html");

或者 var cfg = {

url: "test.html"

};

load(cfg);