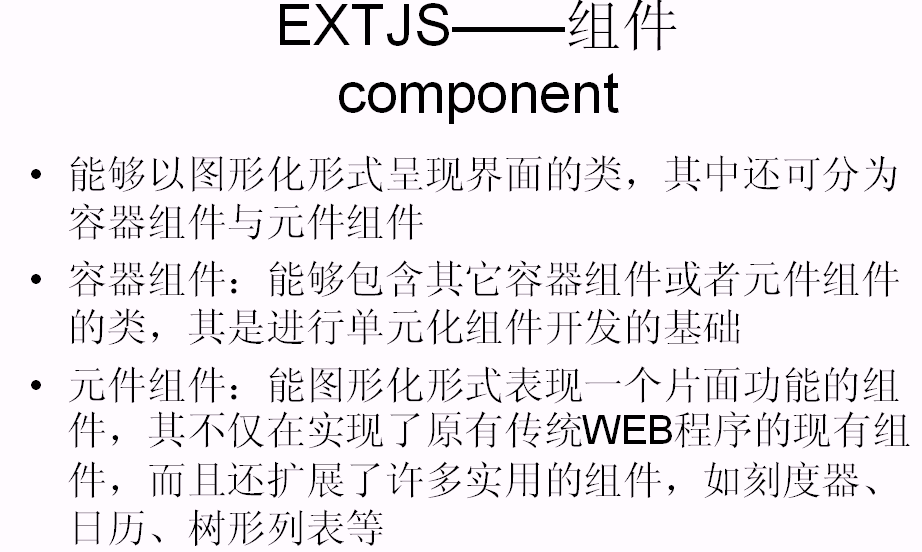
# API使用说明

## API图例

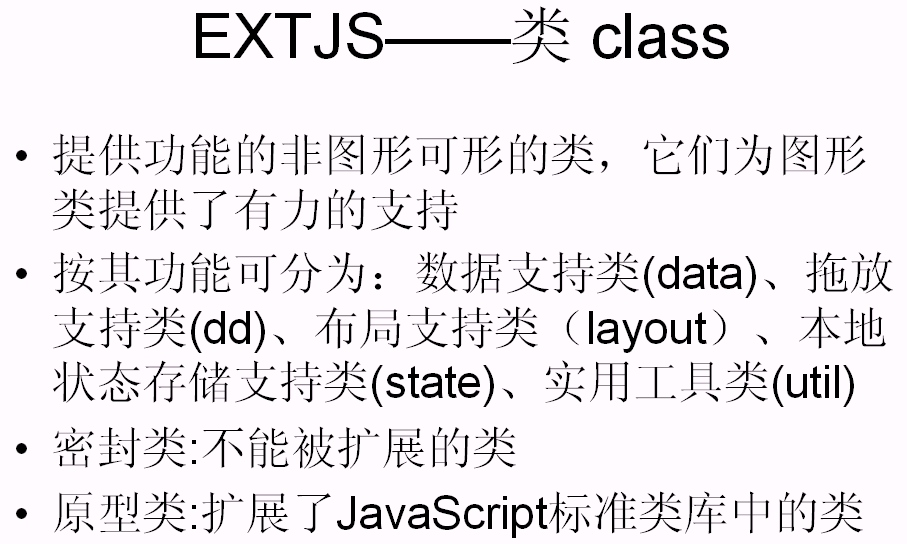
|  |  |
| --- | --- |
|  | **组件** |
|  | **类** |
|  | **方法** |
|  | **事件** |
|  | **配置选项** |
|  | **属性** |
|  | **包** |

## API分类说明

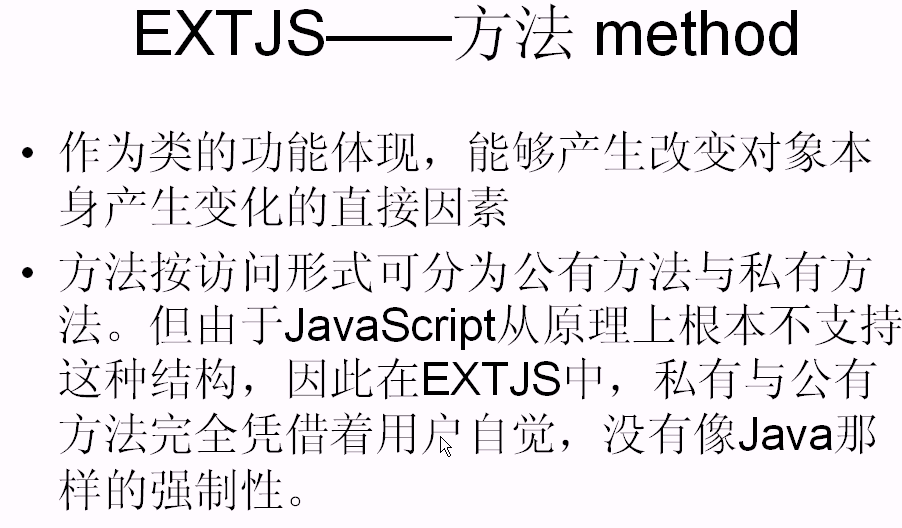
### 组件



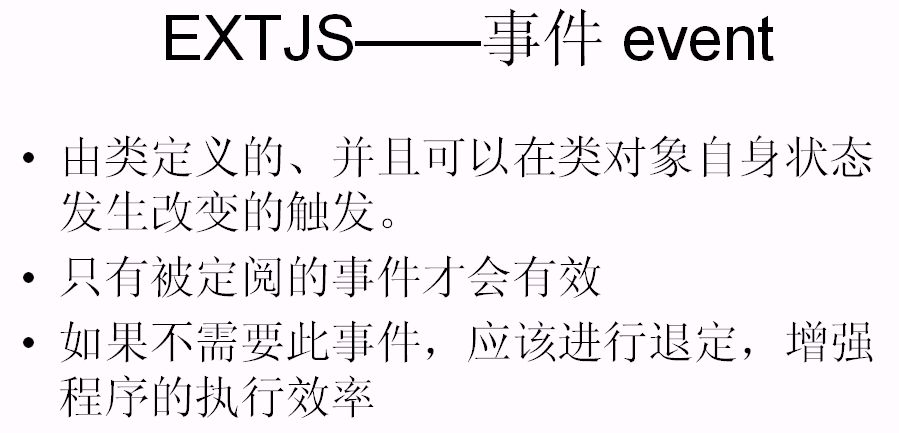
### 类



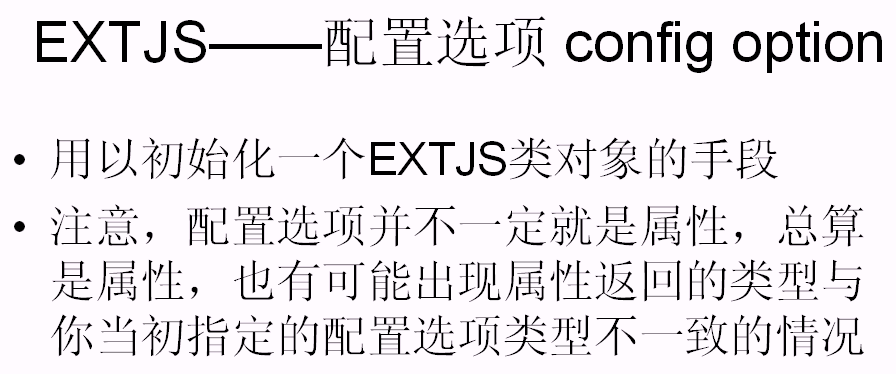
### 方法



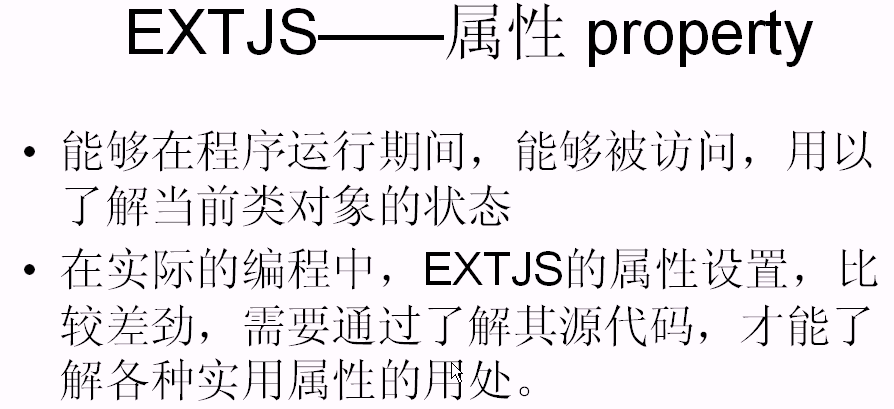
### 事件



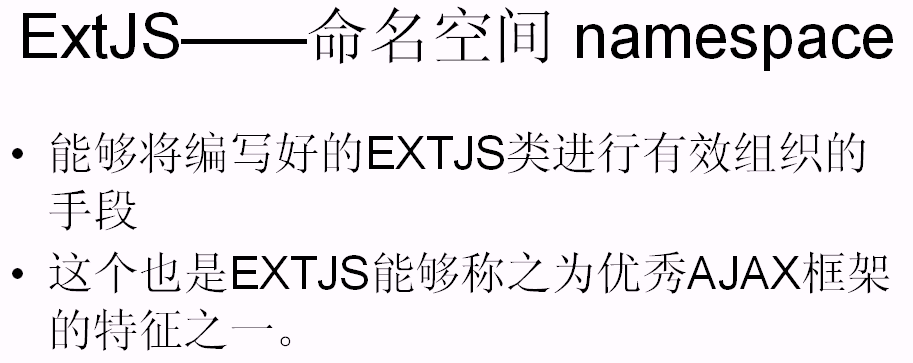
### 配置选项



### 属性



### 命名空间



# 开发环境

|  |
| --- |
| 必须引用的文件  <link rel=*"stylesheet"* type=*"text/css"* href=*"../resources/css/ext-all.css"* />  <script type=*"text/javascript"* src=*"../adapter/ext/ext-base.js"*></script>  <script type=*"text/javascript"* src=*"../ext-all.js"*></script> |

# 语法

## //包

Ext.namespace("Ext.dojochina") ;

//包别名

Dc = Ext.dojochina ;

**var** \_person = **new** Dc.Person() ;

## //类

Ext.dojochina.HelloWorld = Ext.emptyFn ;

//类别名

PN = Ext.dojochina.Person ;

**var** \_person = **new** PN() ;

## //属性

Ext.apply(Ext.dojochina.Person.prototype , {

name:"陈治文"

});

## //创建对象

**new** Ext.dojochina.HelloWorld();

## //实例方法

Ext.apply(Ext.dojochina.Person.prototype , {

name:"",

sex:"",

print:**function**(){

alert(String.format("姓名:{0},性别:{1}" , **this**.name , **this**.sex)) ;

}

}) ;

//创建实例, 调用实例方法

**var** \_person = **new** Ext.dojochina.Person() ;

\_person.name = "陈治文" ;

\_person.sex = "男" ;

\_person.print() ;

## //静态方法

Ext.dojochina.Person.print = **function**(\_name , \_sex){

**var** \_person = **new** Ext.dojochina.Person() ;

\_person.name = \_name ;

\_person.sex = \_sex ;

\_person.print() ;

}

//调用定义静态方法

Ext.dojochina.Person.print("陈治文" , "男") ;

## //构造方法

Ext.dojochina.Person = **function**(\_cfg){

Ext.apply(**this** , \_cfg) ;

} ;

//使用构造方法创建对象

**var** \_person = **new** Ext.dojochina.Person("陈治文" , "男") ;

## //继承

Ext.extend(Ext.dojochina.Student, Ext.dojochina.Person);

//覆盖属性

|  |
| --- |
| Ext.apply(Ext.dojochina.Person.prototype , {  job:"学生",  print:**function**(){  alert(String.format("姓名:{0},性别:{1}" , **this**.name , **this**.sex)) ;  }  }) ; |
| Ext.extend(Ext.dojochina.Student, Ext.dojochina.Person , {  job:"学生"  }) ; |
| Ext.extend(Ext.dojochina.Teacher , Ext.dojochina.Person , {  job:"教师"  }) ; |

//覆盖方法

|  |
| --- |
| Ext.apply(Ext.dojochina.Person.prototype , {  print:**function**(){  alert(String.format("姓名:{0},性别:{1}" , **this**.name , **this**.sex)) ;  }  }) ; |
| Ext.extend(Ext.dojochina.Student , Ext.dojochina.Person , {  print:**function**(){  alert(String.format("{0}是一位{1}学生" , **this**.name , **this**.sex)) ;  }  }) ; |
| Ext.extend(Ext.dojochina.Teacher , Ext.dojochina.Person , {  print:**function**(){  alert(String.format("{0}是一位{1}老师" , **this**.name , **this**.sex)) ;  }  }) ; |

## //事件

|  |
| --- |
| 姓名:<input type=*"text"* id=*"txt\_name"* maxlength=*"10"*/><br/>  性别:<input type=*"text"* id=*"txt\_sex"* maxlength=*"10"*/><br/>  <input type=*"button"* value=*"输入"* onclick="button\_click()"/> |
| //定义Person对象, 添加两个事件  Person = **function**(){  **this**.addEvents(  "namechange",  "sexchange"  ) ;  } ;  //定义Person的方法  Ext.extend(Person , Ext.util.Observable , {  setName:**function**(value){  //触发namechange事件, 参数是value  **this**.fireEvent("namechange", value) ;  },  setSex:**function**(value){  //触发sexchange事件, 参数是value  **this**.fireEvent("sexchange", value) ;  }  }) ;  **var** \_person = **null** ;  Ext.onReady(**function**(){  \_person = **new** Person() ;    //当\_person触发namechange事件, 执行后面的方法  \_person.on("namechange" , **function**(value){  Ext.get("txt\_name").dom.value = value ;  document.title = value ;  });  //当\_person触发sexchange事件, 执行后面的方法  \_person.on("sexchange" , **function**(value){  Ext.get("txt\_sex").dom.value = value ;  });  }) ;  **function** **button\_click**(){  //调用\_person实例的setName方法  \_person.setName(prompt("请输入姓名:" , "")) ;  //调用\_person实例的setSex方法  \_person.setSex(prompt("请输入性别:" , "")) ;  } |

# 组件

## Ext 2.0组件继承结构图

|  |
| --- |
|  |

## Ext.button(按钮)

|  |
| --- |
| **new** Ext.Button({  renderTo:document.body,  text:"确 定",  minWidth:100  }); |
| **new** Ext.Button({  renderTo:Ext.getBody(),  text:"确 定",  handler:**function**(){  alert("欢迎你学习EXTJS") ;  }  }) ; |
| **new** Ext.Button({  renderTo:Ext.getBody(),  text:"确 定",  listeners:{  "click":**function**(){  alert("欢迎你学习EXTJS") ;  }  }  }); |
| **var** \_btn = **new** Ext.Button({  renderTo:Ext.getBody(),  text:"确 定"  }) ;    \_btn.on("click" , **function**(){  alert("欢迎你学习EXTJS") ;  }) ; |

## Ext.form.TextField(单行文本输入框)

|  |
| --- |
| **new** Ext.form.TextField({  renderTo:Ext.getBody(),  fieldLabel:"姓名"  }) ; |
| //创建一个文本输入框  **new** Ext.Panel({  renderTo:Ext.getBody(),  layout:"form",  labelWidth:30,  listeners:{  "render":**function**(\_panel){  \_panel.add(**new** Ext.form.TextField({  id:"txt\_name",  fieldLabel:"姓名"  })) ;  }  }  }) ;    //获取文本输入框的值  **new** Ext.Button({  text:"确 定",  renderTo:Ext.getBody(),  handler:**function**(){  alert(Ext.getCmp("txt\_name").getValue()) ;  }  }) ; |

## Ext.Panel(嵌入的面板)

|  |
| --- |
| **new** Ext.Panel({  title:"人员信息",  renderTo:Ext.getBody()  }) ; |
| **new** Ext.Panel({  renderTo:Ext.getBody(),  title:"人员信息",  width:200,  height:300  }) ; |
| **new** Ext.Panel({  renderTo:Ext.getBody(),  title:"人员信息",  frame:**true**,  width:200,  height:300  }) ; |
| **var** \_panel = **new** Ext.Panel({  title:"人员信息",  frame:**true**,  width:200,  height:300  }) ;    \_panel.addButton({text:"确 定"}) ;  \_panel.addButton(**new** Ext.Button({text:"取 消"})) ;  \_panel.render(Ext.getBody()) ; |
| **var** \_panel = **new** Ext.Panel({  title:"人员信息",  frame:**true**,  width:300,  height:200,  layout:"form",  listeners:{  "render":**function**(\_panel){  \_panel.add({xtype:"textfield" , fieldLabel:"姓名"}) ;  \_panel.add(**new** Ext.form.TextField({fieldLabel:"地址"})) ;  }  }  }) ;    \_panel.addButton({text:"确 定"}) ;  \_panel.addButton(**new** Ext.Button({text:"取 消" , minWidth:100})) ;  \_panel.render(Ext.getBody()) ; |
| 绝对定位的面板    <style type=*"text/css"*>  .center {  position:*absolute*;  top:*30%*;  left:*43%*;  text-align:*left*;  }  </style>  <script type=*"text/javascript"*>    Ext.onReady(**function**(){    **var** \_panel = **new** Ext.Panel({  title:"登 陆",  frame:**true**,  width:260,  height:130,  layout:"form",  labelWidth:35,  listeners:{  "render":**function**(\_panel){  \_panel.add({xtype:"textfield" , fieldLabel:"姓名"}) ;  \_panel.add(**new** Ext.form.TextField({fieldLabel:"地址"})) ;  }  }  }) ;    \_panel.addButton({text:"确 定", minWidth:80}) ;  \_panel.addButton({text:"取 消", minWidth:80}) ;    \_panel.applyToMarkup(Ext.DomHelper.append(  Ext.getBody()  ,  {  tag:"div",  cls:"contain",  cn:[{  tag:"div",  cls:"center"  }]  }  ,  **true**  ).child("div")  ) ;    }) ;    </script> |
| **与上一效果相同**    <style type=*"text/css"*>  .center {  position:*absolute*;  top:*30%*;  left:*43%*;  text-align:*left*;  }  </style>  <script type=*"text/javascript"*>    Ext.onReady(**function**(){    **var** \_panel = **new** Ext.Panel({  title:"登 陆",  frame:**true**,  width:260,  height:130,  layout:"form",  labelWidth:45,  defaults:{xtype:"textfield" , width:180},  items:[{  fieldLabel:"账号"  },{  fieldLabel:"密码"  }],  buttons:[{  text:"确 定"  },{  text:"取 消"  }]  }) ;    \_panel.applyToMarkup(Ext.DomHelper.append(Ext.getBody(), {  tag:"div",  cls:"contain",  cn:[{  tag:"div",  cls:"center"  }]  } , **true**).child("div")) ;    }) ;    </script> |

## Ext.Window(窗口)

|  |
| --- |
| <script type=*"text/javascript"*>    Ext.onReady(**function**(){    **var** \_window = **new** Ext.Window({  title:"登 陆",  frame:**true**,  width:260,  height:130,  layout:"form",  labelWidth:45,  plain:**true**,  resizable:**false**,  defaults:{xtype:"textfield" , width:180},  bodyStyle:"padding:3px",  buttonAlign :"center",  closeAction:"hide",  listeners:{  "show":**function**(){  alert("窗体显现") ;  },  "hide":**function**(){  alert("窗体隐藏") ;  },  "close":**function**(){  alert("窗体关闭") ;  }  },  items:[{  fieldLabel:"账号"  },{  fieldLabel:"密码"  }],  buttons:[{  text:"确 定"  },{  text:"取 消",  handler:**function**(){  //\_window.hide();  \_window.close();  //this.hide() ;  }  }]  }) ;    \_window.show() ;    }) ;    </script> |
| <script type=*"text/javascript"*>    Ext.onReady(**function**(){    **var** \_window = **new** Ext.Window({  title:"测试窗体",  layout:"form",  width:300,  plain:**true**,  items:[{  xtype:"textfield",  fieldLabel:"姓名"  },{  xtype:"textfield",  fieldLabel:"密码"  }],    buttons:[{  text:"确定",  handler:**function**(){  **var** \_coll = **this**.ownerCt.items ;  **var** \_temp = "" ;  \_temp += "姓名是:" + \_coll.first().getValue() ;  \_temp += "密码是:" + \_coll.itemAt(1).getValue() ;  alert(\_temp) ;  }  }]  }) ;    \_window.show() ;    }) ;  </script> |
| <script type=*"text/javascript"*>    Ext.onReady(**function**(){    **new** Ext.Window({  title:"添加人员",  width:500,  height:350,  plain:**true**,  layout:"form",  labelWidth:59,  defaultType:"textfield",  items:[{  xtype:"panel",  baseCls:"x-plain",  style:"padding:5px",  layout:"column",  items:[{  columnWidth:.5,  layout:"form",  labelWidth:55,  defaultType:"textfield",  defaults:{width:160},  baseCls:"x-plain",  items:[{  fieldLabel:"姓名"  },{  fieldLabel:"年龄"  },{  fieldLabel:"出生日期"  },{  fieldLabel:"联系电话"  },{  fieldLabel:"手机号码"  },{  fieldLabel:"电子邮件"  },{  fieldLabel:"性别"  }]  },{  columnWidth:.5,  layout:"form",  labelWidth:55,  baseCls:"x-plain",  items:{  xtype:"textfield",  fieldLabel:"个人照片",  width:170,  height:177,  inputType:"image"  }  }]  },{  fieldLabel:"身份证号",  width:400  },{  fieldLabel:"具体地址",  width:400  },{  fieldLabel:"职位"  }],  showLock:**false**,  listeners:{  "show":**function**(\_window){  **if**(!\_window["showLock"]){  \_window.findByType("textfield")[7].getEl().dom.src = "default\_pic.gif" ;  \_window["showLock"] = **true**;  }  }  },  buttons:[{  text:"确定",  handler:**function**(){  alert(**this**.ownerCt.buttons[1].text) ;  }  },{  text:"取消"  }]  }).show() ;  }) ;    </script> |
| <script type=*"text/javascript"*>    Ext.onReady(**function**(){    **new** Ext.Window({  title:"添加人员",  width:510,  height:360,  plain:**true**,  layout:"form",  labelWidth:59,  defaultType:"textfield",  defaults:{anchor:"95%"},  items:[{  xtype:"panel",  baseCls:"x-plain",  style:"padding:5px",  layout:"column",  items:[{  columnWidth:.5,  layout:"form",  labelWidth:55,  defaultType:"textfield",  defaults:{width:160},  baseCls:"x-plain",  items:[{  fieldLabel:"姓名"  },{  fieldLabel:"年龄",  value:28,  readOnly:**true**  },{  xtype:"datefield",  format:"Y-m-d",  value:"1981-10-18",  readOnly:**true**,  fieldLabel:"出生日期",  listeners:{  "blur":**function**(\_df){  **var** \_age = \_df.ownerCt.findByType("textfield")[1] ;  \_age.setValue(**new** Date().getFullYear() - \_df.getValue().getFullYear() + 1) ;  }  }  },{  fieldLabel:"联系电话"  },{  fieldLabel:"手机号码"  },{  fieldLabel:"电子邮件"  },{  xtype:"combo",  fieldLabel:"性别",  mode :"local",  displayField :"sex",  readOnly:**true**,  triggerAction : "all",  value:"男",  store:**new** Ext.data.SimpleStore({  fields:["sex"],  data:[["男"] , ["女"]]  })  }]  },{  columnWidth:.5,  layout:"form",  labelWidth:55,  baseCls:"x-plain",  items:{  xtype:"textfield",  fieldLabel:"个人照片",  width:170,  height:177,  inputType:"image"  }  }]  },{  fieldLabel:"身份证号",  width:400  },{  fieldLabel:"具体地址",  width:400  },{  fieldLabel:"职位"  }],  showLock:**false**,  listeners:{  "show":**function**(\_window){  **if**(!\_window["showLock"]){  \_window.findByType("textfield")[5].getEl().dom.src = "default\_pic.gif" ;  \_window["showLock"] = **true**;  }  }  },  buttons:[{  text:"确定",  handler:**function**(){  alert(**this**.ownerCt.buttons[1].text) ;  }  },{  text:"取消"  }]  }).show() ;  }) ;    </script> |
| <script type=*"text/javascript"*>    Ext.onReady(**function**(){    **var** JOB = Ext.data.Record.create([{  name:"job"  }]) ;    **new** Ext.Window({  title:"添加人员",  width:510,  height:360,  plain:**true**,  layout:"form",  labelWidth:59,  defaultType:"textfield",  defaults:{anchor:"95%"},  items:[{  xtype:"panel",  baseCls:"x-plain",  style:"padding:5px",  layout:"column",  items:[{  columnWidth:.5,  layout:"form",  labelWidth:55,  defaultType:"textfield",  defaults:{width:160},  baseCls:"x-plain",  items:[{  fieldLabel:"姓名"  },{  fieldLabel:"年龄",  value:28,  readOnly:**true**  },{  xtype:"datefield",  format:"Y-m-d",  value:"1981-10-18",  readOnly:**true**,  fieldLabel:"出生日期",  listeners:{  "blur":**function**(\_df){  **var** \_age = \_df.ownerCt.findByType("textfield")[1] ;  \_age.setValue(**new** Date().getFullYear() - \_df.getValue().getFullYear() + 1) ;  }  }  },{  fieldLabel:"联系电话"  },{  fieldLabel:"手机号码"  },{  fieldLabel:"电子邮件"  },{  xtype:"combo",  fieldLabel:"性别",  mode :"local",  displayField :"sex",  readOnly:**true**,  triggerAction : "all",  value:"男",  store:**new** Ext.data.SimpleStore({  fields:["sex"],  data:[["男"] , ["女"]]  })  }]  },{  columnWidth:.5,  layout:"form",  labelWidth:55,  baseCls:"x-plain",  items:{  xtype:"textfield",  fieldLabel:"个人照片",  width:170,  height:177,  inputType:"image"  }  }]  },{  fieldLabel:"身份证号",  width:400  },{  fieldLabel:"具体地址",  width:400  },{  bodyStyle:"padding:5px",  xtype:"panel",  layout:"column",  baseCls:"x-plain",  defaults:{baseCls:"x-plain"},  items:[{  columnWidth:.4,  layout:"form",  labelWidth:57,  items:{  xtype:"combo",  fieldLabel:"职位",  anchor:"100%",  mode :"local",  displayField :"job",  readOnly:**true**,  triggerAction : "all",  listeners:{  select:**function**(\_combo , \_record , \_index){  \_combo["selectItem"] = \_record ;  }  },  store:**new** Ext.data.SimpleStore({  fields:["job"]  })  }  },{  columnWidth:.2,  bodyStyle:"padding:0 0 0 5px",  items:{  xtype:"button",  text:"添加职位",  handler:**function**(){  **var** \_window = **this**.ownerCt.ownerCt.ownerCt ;  **var** \_job = \_window.findByType("combo")[1] ;  Ext.MessageBox.prompt("请输入职位的名称" , "职位名称" , **function**(\_btn , \_text){  **if**(\_btn == "ok"){  **var** \_store = **this**.store ;  \_store.insert(0 , **new** JOB({  job:\_text  })) ;  **this**.setValue(\_text) ;  **this**["selectItem"] = \_store.getAt(0) ;  }  } , \_job) ;  }  }  },{  columnWidth:.2,  items:{  xtype:"button",  text:"修改职位",  handler:**function**(){  **var** \_window = **this**.ownerCt.ownerCt.ownerCt ;  **var** \_job = \_window.findByType("combo")[1] ;  **if**(\_job["selectItem"] != **null**)  Ext.MessageBox.prompt("请输入修改后职位的名称" , "职位名称" , **function**(\_btn , \_text){  **if**(\_btn == "ok"){  **this**["selectItem"].set("job" , \_text) ;  **this**.setValue(\_text) ;  }  } , \_job , **false** , \_job.getValue()) ;  }  }  },{  columnWidth:.2,  items:{  xtype:"button",  text:"删除职位",  handler:**function**(){  **var** \_window = **this**.ownerCt.ownerCt.ownerCt ;  **var** \_job = \_window.findByType("combo")[1] ;  **if**(\_job["selectItem"] != **null**)  Ext.MessageBox.confirm("系统提示" , "你确认删除当前职位吗?" , **function**(\_btn){    **if**(\_btn == "yes"){    **try**{    **this**.store.remove(**this**["selectItem"]) ;  }**catch**(\_err){  }    **if**(**this**.store.getCount() != 0){  **this**.setValue(**this**.store.getAt(0).get("job")) ;  **this**["selectItem"] = **this**.store.getAt(0) ;  }  **else**{    **this**.setValue("");    **this**["selectItem"] = **null** ;  }        }    } , \_job) ;    }  }  }]  }],  showLock:**false**,  listeners:{  "show":**function**(\_window){    **if**(!\_window["showLock"]){    \_window.findByType("textfield")[5].getEl().dom.src = "default\_pic.gif" ;    \_window["showLock"] = **true**;    }    /\*var \_job = \_window.findByType("combo")[1] ;    var \_store = \_job.store ;    \_job.setValue(\_store.getAt(0).get("job")) ;    \_job["selectItem"] = \_store.getAt(0) ;\*/  }  },  buttons:[{  text:"确定",  handler:**function**(){    alert(**this**.ownerCt.buttons[1].text) ;    }  },{  text:"取消"  }]  }).show() ;    }) ;    </script> |

## Ext.grid.GridPanel(嵌入面板)

|  |
| --- |
| **var** grid = **new** Ext.grid.GridPanel({  id:'grid',  title:'用户头像审核',  store: store,  trackMouseOver:**true**,  disableSelection:**false**,  loadMask: **true**,  autoExpandColumn:'contenttypeName',  viewConfig: {  forceFit:**true**  },  tbar:[  "审核状态：",  {  id:'addDicButton',  text:'通过',  tooltip:'选中头像通过审核'  },  **new** Ext.Button(mulPass),  **new** Ext.Button(mulNoPass)  ],  bbar: **new** Ext.PagingToolbar({//下面的翻页  pageSize: 20,  store: store,  displayInfo: **true**  }),  }); |

## Ext.Action(可用来创建button)

|  |
| --- |
| //批量审核通过  **var** mulPass = **new** Ext.Action({  text:'通过',  tooltip:'选中头像通过审核',  //iconCls : 'page\_delete',  handler : **function**() {  // var rows = Ext.getCmp("MenuGridPanel").getSelections();  **var** rows=grid.getSelectionModel().getSelections();  **if**(rows ==0){  Ext.Msg.alert("提示","请至少选择一条记录");  }**else**{  //Ext.MessageBox.confirm("确认", "这些记录删除后将不可恢复,您确认删除吗?", function(button, text){    //if (button == 'yes') {  **var** ids = "";  **for**(**var** i=0;i<rows.length;i++){  ids = ids + "&id=" + rows[i].json.id  //ids += rows[i].json.id + ",";//获取选中的ID  }  Ext.Ajax.request({  //url : '${contextPath}/app/icvServing?delete&ids='+ids,  url:"${contextPath}/app/user/auditUserHeadPortrait?pass=yes&"+ids,  method : 'get',  success : **function**(result, request) {  **var** resp = Ext.util.JSON.decode(result.responseText);  **if** (resp) {  //Ext.MessageBox.alert("提示", '删除成功。');  Ext.getCmp('grid').store.reload();//刷新列表  }**else**{  Ext.MessageBox.alert("提示", '审核失败.');  }  },  failure : **function**(result, request) {  Ext.MessageBox.alert("提示", '审核失败,可能是系统原因,请联系技术人员。');  }  });  //}  //});  }    }  }); |

## Ext.grid.ColumnModel(表头)

|  |
| --- |
| //设置复选框  **var** sm = **new** Ext.grid.CheckboxSelectionModel();  //表头  **var** cm = **new** Ext.grid.ColumnModel([  **new** Ext.grid.RowNumberer(),//自动行号  sm,//添加的地方  {header: "用户名", width: 80, dataIndex: 'loginName', sortable: **true**},  {header: "头像", width: 80, dataIndex: 'avatar',renderer:opt}  ]);  cm.defaultSortable = **true**; |
| 第二步: 在**new** Ext.grid.GridPanel中加入属性sm:sm, |

## Ext.data.ComplexJsonStore() 创建数据存储

|  |
| --- |
| **var** store = **new** Ext.data.ComplexJsonStore({  root: 'items',  //autoLoad:true,  totalProperty: 'total',  idProperty: 'id',  remoteSort: **true**,  fields: [  {name: 'id', type: 'int'},  {name: 'loginName', type: 'string'},  {name:'avatar', type:'string', mapping:'detail.avatar'}  ],  proxy: **new** Ext.data.HttpProxy({  url: 'app/user/listUserHeadPortrait'  })  });  store.load({params:{start:0, limit:20, avatarStatus:0}}); |

## Ext.onReady(dom加载完成要执行的函数)

|  |
| --- |
| Ext.onReady(**function**(){  **var** viewport = **new** Ext.Viewport({  layout : 'fit',  align:'center',  items : grid  });  }); |

# 容器

## 面板(Panel是一种面板)

|  |
| --- |
| 面板:(顶部工具栏、底部工具栏、面板头部、面板尾部、面板主区域  面板可以放入其它任何容器中，  面板本身是一个容器，他里面又可以包含各种其它组件。  组件  容器(面板是一种容器) |
| **new** Ext.Panel( {  renderTo : document.body,  title : "面板头部header",  width : 300,  height : 200,  html : '<h1>面板主区域</h1>',  tools : [ {//窗口右边的操作按钮  id : "save"  }, {  id : "help",  handler : **function**() {  Ext.Msg.alert('help', 'please help me!');  }  }, {  id : "close"  } ],  tbar : [ {  text : '顶部工具栏topToolbar'  } ],  bbar : [ {  text : '底部工具栏bottomToolbar'  } ],  buttons : [ { //与html, tbar, bbar平行(面板上很少包含按钮, 按钮一般放在工具栏上)  text : "按钮位于footer"  } ]  }); |

## window是一种特殊的面板(panel)

|  |
| --- |
| 窗口包含了浮动、可拖动、可关闭、最大化、最小化等特性 |
| **function** **newWin**() {  **var** win = **new** Ext.Window( {  title : "窗口",  width : 400,  height : 300,  maximizable : **true**  });  win.show();  } |
| **var** i = 0;  //定义组  **var** mygroup;  //新建窗口  **function** **newWin**() {  **var** win = **new** Ext.Window( {  title : "窗口" + i++,  width : 400,  height : 300,  maximizable : **true**,  manager : mygroup  });  win.show();  }    //把组中的某个窗口移到组的底层  **function** **toBack**() {  mygroup.sendToBack(mygroup.getActive());  }  //隐藏所有  **function** **hideAll**() {  mygroup.hideAll();  }  Ext.oReay(function(){  mygroup = **new** Ext.WindowGroup();  Ext.get("btn").on("click", newWin);  Ext.get("btnToBack").on("click", toBack);  Ext.get("btnHide").on("click", hideAll);  }); |

## Ext.VeiwPort

|  |
| --- |
| VeiwPort 代表整个浏览器显示区域，该对象渲染到页面的body区域 ，不需要再指定 renderTo, 并会随着浏览器显示区域的大小自动改变，一个页面中只能有一个ViewPort 实例。 |
| **new** Ext.Viewport({  enableTabScroll:**true**,  layout:"fit",  items:[  {  title:"面板",  html:"",  bbar:[  {text:"按钮1"},  {text:"按钮2"}  ]  }  ]  }); |
| **使用Viewport来设计页面布局**  **new** Ext.Viewport( {  enableTabScroll : **true**,  layout : "border",  items : [ {  title : "面板",  region : "north",  height : 50,  html : "<h1>网站后台管理系统!</h1>"  }, {  title : "功能菜单",  region : "west",  width : 200,  collapsible : **true**,  html : "菜单栏"  }, {  xtype : "tabpanel",  region : "center",  items : [ {  title : "面板1"  }, {  title : "面板2"  } ]  } ]  }); |

## 对话框

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ext 的对话框都封装在 Ext.MessageBox 类，该类还有一个简写形式即 Ext.Msg，可以直  接通过 Ext.MessageBox 或 Ext.Msg 来直接调用相应的对话框方法来显示 Ext 对话框  普通对话框一般包括四个参数  比如 confirm 的方法签, 名为 confirm ( String title, String msg, [Function fn], [Object scope] )   |  |  | | --- | --- | | title | 对话框的标题 | | msg | 对话框中的提示信息 | | fn | 当关闭对话框后执行的回调函数 | | scope | 回调函数的执行作用域  回调函数可以包含两个参数，即 button 与 text,button 表示点击的按钮  text 表示对话框中有活动输入选项时输入的文本内容  我们可以在回调函数中通过 button 参数来判断用户作了什么什么选择，可  以通过 text 来读取在对话框中输入的内容 | |
| **alert**  Ext.get("btnAlert").on("click",function(){  Ext.MessageBox.alert("请注意","这是ExtJS的提示框");  }); |
| **confirm**  Ext.get("btn").on(  "click", **function**() {  Ext.MessageBox.confirm("请确认", "是否真的要删除指定的内容", **function**(button, text) {  alert(button); //button的值为yes或no  alert(text);  });  }); |
| **prompt**  Ext.get("btn").on(  "click",  **function**() {  Ext.MessageBox.prompt("输入提示框", "请输入你的新年愿望:", **function**(button, text) {  **if** (button == "ok") {  alert("你的新年愿望是:" + text);  } **else**  alert("你放弃了录入!");  });  }); |
| **progress** |
| **wait** |
| **Ext.Msg.show()**  **function** **save**(button) {  **if** (button == "yes") {  alert("yes");  //执行数据保存操作  } **else** **if** (button == "no") {  alert("no");  //不保存数据  } **else** {  alert("cancel");  //取消当前操作  }  }  Ext.onReady(**function**() {  Ext.get("btn").on("click", **function**() {  Ext.Msg.show( {  title : '保存数据',  msg : '你已经作了一些数据操作，是否要保存当前内容的修改？',  buttons : Ext.Msg.YESNOCANCEL,  fn : save,  icon : Ext.MessageBox.QUESTION  });  });  }); |

# 组件

|  |
| --- |
|  |

# 分类

## 面板

## 窗口

## 对话框

## 布局

布局就是指容器组件中子元素的分布、排列组合方式

ExtJS 的布局基类为 Ext.layout.ContainerLayout，其它布局都是继承该类。

ExtJS 的容器组件包含一个 layout 及 layoutConfig 配置属性，这两个属性用来指定容器使用的布局及布局的详细配置信息，如果没有指定容器组件的 layout 则默认会使用 ContainerLayout 作为布局，该布局只是简单的把元素放到容器中，有的布局需要 layoutConfig 配置，有的则不需要 layoutConfig 配置。

## 布局layout属性的值

|  |  |
| --- | --- |
| containerLayout | 该布局只是简单的把元素放到容器中 |
| column | 列布局(分成几列) |
| border | 区域布局(上下左右) |
| fit | 层叠式布局(如果容器组件中有多个子元素，则只会显示一个元素) |
| form | 表单布局 |
| card | 卡片布局(选项卡) |
| tabel | 表格布局(可合并单元格) |
| accordion | 可折叠 |

|  |
| --- |
| Ext中的一些容器组件都已经指定所使用的布局，比如  TabPanel使用card布局、  FormPanel使用form布局，  GridPanel中的表格使用column布局  等，我们在使用这些组件的时候，不能给这些容器组件再指定另外的布局。 |
| Ext2.0 中还包含其它的 Ext.layout.AbsoluteLayout、Ext.layout.AnchorLayout 等布局类，这些布局主要作为其它布局的基类使用，一般情况下我们不会在应用中直接使用。另外，我们也可以继承 10 种布局类的一种，来实现自定义的布局。 |

### border区域布局

|  |
| --- |
| Border 布局由类Ext.layout.BorderLayout定义，布局名称为border。该布局把容器分成东南西北中五个区域 ，  分别由east，south, west，north, cente来表示，在往容器中添加子元素的时候，我们只需要指定这些子元素  所在的位置，Border布局会自动把子元素放到布局指定的位置。 |
| **new** Ext.Viewport( {  layout : "border",  items : [ {  region : "north",  height : 50,  title : "顶部面板"  }, {  region : "south",  height : 50,  title : "底部面板"  }, {  region : "center",  title : "中央面板"  }, {  region : "west",  width : 100,  title : "左边面板"  }, {  region : "east",  width : 100,  title : "右边面板"  } ]  }); |

### column列布局

|  |
| --- |
| Column 列布局由 Ext.layout.ColumnLayout 类定义，名称为 column。列布局把整个容器  组件看成一列，然后往里面放入子元素的时候，可以通过在子元素中指定使用 columnWidth  或 width 来指定子元素所占的列宽度。columnWidth 表示使用百分比的形式指定列宽度，而  width 则是使用绝对象素的方式指定列宽度，在实际应用中可以混合使用两种方式。 |
| **new** Ext.Panel({  renderTo:document.body,  title:"容器组件",  layout:"column",  width:500,  height:100,  items:[  {title:"列1",columnWidth:.2},  {title:"列2",columnWidth:.3},  {title:"列3",columnWidth:.5},  {title:"列4",width:200}  ]  }  );  注意: columnWidth的中和等于1的时候, 所有列才会完全填充面板  实际效果: columnWidth为除去width后, 剩下的宽度作百分比分割 |

### fit

|  |
| --- |
| 如果容器组件中有多个子元素，则只会显示一个元素， |
| **new** Ext.Panel( {  renderTo:document.body,  title : "容器组件",  layout : "fit",  width : 500,  height : 100,  items : [ {  title : "子元素1",  html : "这是子元素1中的内容"  }, {  title : "子元素2",  html : "这是子元素2中的内容"  } ]  }); |

### form

|  |
| --- |
| Form 布局由类 Ext.layout.FormLayout 定义，名称为 form，是一种专门用于管理表单中输入字段的布局，这种布局主要用于在程序中创建表单字段或表单元素等使用。  在实际应用中，Ext.form.FormPanel 这个类默认布局使用的是 Form 布局，而且 FormPanel还会创建与 <form> 标签相关的组件，因此一般情况下我们直接使用 FormPanel 即可。 |
| **new** Ext.form.FormPanel( {  renderTo : document.body,  title : "容器组件",  width : 300,  labelAlign : "right",  height : 120,  defaultType : 'textfield',  items : [  {  fieldLabel : "请输入姓名",  name : "name"  emptyText:'请输入...',  }, {  fieldLabel : "请输入地址",  name : "address"  }, {  fieldLabel : "请输入电话",  name : "tel"  }  ]  }); |

### accordion

|  |
| --- |
| Accordion 布局由类 Ext.layout.Accordion 定义，名称为 accordion，表示可折叠的布局，也就是说使用该布局的容器组件中的子元素是可折叠的形式。 |
| **new** Ext.Panel( {  renderTo : document.body,  title : "容器组件",  width : 500,  height : 200,  layout : "accordion",  layoutConfig : {  animate : **false** //展开和折叠时候应用动画效果  },  items : [  {title : "子元素1", html : "这是子元素1中的内容"},  {title : "子元素2", html : "这是子元素2中的内容"},  {title : "子元素3", html : "这是子元素3中的内容"}  ]  }); |
| **提交form表单**  formPanel.form.submit({  //waitTitle:'提示',  //waitMsg : '正在验证数据,请稍侯 ......',  url : 'app/systemUser/modifyPassword',  timeout:100,  method:'post',  success:**function**(result, request) {  //var resp = Ext.util.JSON.decode(result.responseText);  //alert(resp);  //if(resp){  window.location.href="app/systemUser/initList";  //}else{  // Ext.MessageBox.alert('提示', '数据保存失败！');  //}  },failure:**function**(form,action){  Ext.MessageBox.alert("提示", '数据保存失败, 可能是系统原因, 请联系技术人员。');  }  }); |

### table

|  |
| --- |
| Table 布局由类 Ext.layout.TableLayout 定义，名称为 table，该布局负责把容器中的子元素按照类似普通 html 标签 |
| **var** panel = **new** Ext.Panel( {  renderTo : document.body,  title : "容器组件",  width : 500,  height : 200,  layout : "table",  layoutConfig : {  columns : 3  },  items : [  {  title : "子元素1",  html : "这是子元素1中的内容",  rowspan : 2, //占两行  height : 100  }, {  title : "子元素2",  html : "这是子元素2中的内容",  colspan : 2 //占两列  }, {  title : "子元素3",  html : "这是子元素3中的内容"  }, {  title : "子元素4",  html : "这是子元素4中的内容"  }  ]  }); |

## 表格

|  |
| --- |
| 表格由类Ext.grid.GridPanel定义，继承自Panel，其xtype为grid。  ExtJS中，表格Grid必须包含列定义信息，并指定表格的数据存储器Store。表格的列信息由类Ext.grid.ColumnModel定义、而表格的数据存储器由Ext.data.Store定义，数据存储器根据解析的数据不同分为JsonStore、SimpleStroe、GroupingStore等。 |
| //定义表格中要显示的数据  **var** data = [  [ 1, 'EasyJWeb', 'EasyJF', 'www.easyjf.com' ],  [ 2, 'jfox', 'huihoo', 'www.huihoo.org' ],  [ 3, 'jdon', 'jdon', 'www.jdon.com' ],  [ 4, 'springside', 'springside', 'www.springside.org.cn' ]  ];  //创建一个数据存储  **var** store = **new** Ext.data.SimpleStore( {  data : data,  fields : [ "id", "name", "organization", "homepage" ]  });    **var** grid = **new** Ext.grid.GridPanel( { //创建一个表格  renderTo : Ext.getBody(),  title : "中国Java开源产品及团队",  height : 150,  width : 600,  columns : [ //表格包含的列由 columns 配置属性来描述, columns是一个数组, 每一行数据元素描述表格的一列信息  {  header : "项目名称", //列头显示文本  dataIndex : "name" //列对应的记录集字段  //sorable 列是否可排序  //renderer 列的渲染函数  //width 宽度  //format 格式化信息  }, {  header : "开发团队",  dataIndex : "organization"  }, {  header : "网址",  dataIndex : "homepage"  }  ],  store : store,  autoExpandColumn : 2  }); |
| **var** data = [  [ 1, 'EasyJWeb', 'EasyJF', 'www.easyjf.com' ],  [ 2, 'jfox', 'huihoo', 'www.huihoo.org' ],  [ 3, 'jdon', 'jdon', 'www.jdon.com' ],  [ 4, 'springside', 'springside', 'www.springside.org.cn' ]  ];      **var** store = **new** Ext.data.SimpleStore( {  data : data,  fields : [ "id", "name", "organization", "homepage" ]  });      **var** cm = **new** Ext.grid.ColumnModel( [  {  header : "项目名称",  dataIndex : "name",  sortable : **true**  }, {  header : "开发团队",  dataIndex : "organization",  sortable : **true**  },  {  header : "网址",  dataIndex : "homepage"  }  ]);    **var** grid = **new** Ext.grid.GridPanel( {  renderTo : document.body,  title : "中国Java开源产品及团队",  height : 200,  width : 600,  cm : cm,  store : store,  autoExpandColumn : 2  }); |
| **var** data = [ [ 1, 'EasyJWeb', 'EasyJF', 'www.easyjf.com' ],  [ 2, 'jfox', 'huihoo', 'www.huihoo.org' ],  [ 3, 'jdon', 'jdon', 'www.jdon.com' ],  [ 4, 'springside', 'springside', 'www.springside.org.cn' ] ];      **var** store = **new** Ext.data.SimpleStore( {  data : data,  fields : [ "id", "name", "organization", "homepage" ]  });      **var** colM = **new** Ext.grid.ColumnModel( [  {  header : "项目名称",  dataIndex : "name",  sortable : **true**  }, {  header : "开发团队",  dataIndex : "organization",  sortable : **true**  }, {  header : "网址",  dataIndex : "homepage",  renderer : showUrl  }  ]);    **function** **showUrl**(value) {  **return** "<a href='http://"+value+"'>" + value + "</a>";  }    **var** grid = **new** Ext.grid.GridPanel( {  renderTo : "hello",  title : "中国Java开源产品及团队",  height : 200,  width : 600,  cm : colM,  store : store,  autoExpandColumn : 2  }); |

## 数据存储store

|  |
| --- |
| 存储器 Store 负责把各种各样的数据（如二维数组、JSon 对象数组、xml 文本）等转换成 ExtJS的数据记录集 Record  表格的数据是存放类型为 Store 的数据存储器中，通过指定表格 Grid 的 store 属性来设置表格中显示的数据，通过调用 store 的 load 或 reload方法可以重新加载表格中的数据。ExtJS 中用来定义控件中使用数据的 API 位于 Ext.dd 命名空间中 |

### Record

|  |
| --- |
| (关系数据表中的一行，也可以称为记录)  ExtJS 中有一个名为 Record 的类，表格等控件中使用的数据是存放在Record 对象中，一个 Record 可以理解为关系数据表中的一行，也可以称为记录。Record 对象中即包含了记录（行中各列）的定义信息（也就是该记录包含哪些字段，每一个字段的数据类型等），同时又包含了记录具体的数据信息（也就是各个字段的值） |
| **var** MyRecord = Ext.data.Record.create( [ //使用 Record 的 create 方法创建一个记录集 MyRecordd，MyRecord 其实是一个类，该类包含了记录集的定义信息，可以通过 MyRecord 来创建包含字段值的 Record 对象  {  name : 'title'  }, {  name : 'username',  mapping : 'author'  }, {  name : 'loginTimes',  type : 'int'  }, {  name : 'lastLoginTime',  mapping : 'loginTime',  type : 'date'  }  ]);    **var** r = **new** MyRecord( {  title : "日志标题",  username : "easyjf",  loginTimes : 100,  loginTime : **new** Date()  });    alert(MyRecord.getField("username").mapping); //获取映射, 结果: author  alert(MyRecord.getField("lastLoginTime").type); //获取类型, 结果: data  alert(r.get("loginTimes")); //获取值, 结果: 100  alert(r.data.username); //获取值, 结果: easyjf  r.set("username", "张三");  alert(r.data.username); //获取值, 结果: easyjf  alert(r.dirty); //当前记录的值时候被改动过, 改动过为true  }); |

### store

|  |
| --- |
| (数据存储器, 可以理解为客户端的小型数据表，提供缓存等功能) |
| **var** store = **new** Ext.data.JSonStore( {  url : "link.ejf?cmd=list",  id : "id",  totalProperty : "total",  root : "items",    fields : [ 'title', {  name : 'username',  mapping : 'author'  }, {  name : 'loginTimes',  type : 'int'  }, {  name : 'lastLoginTime',  mapping : 'loginTime',  type : 'date'  } ]  });  store.load(); |

### dataReader

|  |
| --- |
| 数据读取器，也就是数据解析器，其负责把从服务器或者内存数组、xml文档中获得的杂乱信息转换成 ExtJS 中的记录集 Record 数据对象，并存储到 Store 里面的记录集数组中 |

#### ArrayReader

|  |
| --- |
| Ext.data.ArrayReader－数组解析器，用于读取二维数组中的信息，并转换成记录集 Record对象。 |
| //[ [1, '测试', '小王',3], [2, '新年好', 'williamraym',13] ]  **var** MyRecord = Ext.data.Record.create([  {name: 'title', mapping:1},  {name: 'username', mapping:2},  {name: 'loginTimes', type:3}  ]);  **var** myReader = **new** Ext.data.ArrayReader({  id: 0  }, MyRecord); |

#### JsonReader

|  |
| --- |
| 用于读取 JSON 格式的数据信息，并转换成记  录集 Record 对象。 |
| **var** data = {'results': 2, 'rows': [  { id: 1, title: '测试', author: '小王', loginTimes: 3 },  { id: 2, title: 'Ben', author: 'williamraym', loginTimes:13}  ]  };    **var** MyRecord = Ext.data.Record.create([  {name: 'title'},  {name: 'username', mapping: 'author'},  {name: 'loginTimes', type: 'int'}  ]);    **var** myReader = **new** Ext.data.JsonReader({  totalProperty: "results",  root: "rows",  id: "id"  }, MyRecord); |
| JSonReader 还有比较特殊的用法，就是可以把 Store 中记录集的配置信息存放直接保存在从服务器端返回的 JSON 数据中，  var myReader = new Ext.data.JsonReader();  这一个不带任何参数的 myReader，可以处理从服务器端返回的下面 JSON 数据： |

#### XmlReader

## 树TreePanel

# ExtJs xtype一览

基本组件：

|  |  |  |
| --- | --- | --- |
| xtype | Class | 描述 |
| button | Ext.Button | 按钮 |
| splitbutton | Ext.SplitButton | 带下拉菜单的按钮 |
| cycle | Ext.CycleButton | 带下拉选项菜单的按钮 |
| buttongroup | Ext.ButtonGroup | 编组按钮(Since3.0) |
| slider | Ext.Slider | 滑动条 |
| progress | Ext.ProgressBar | 进度条 |
| statusbar | Ext.StatusBar | 状态条，2.2加进来，3.0又去了 |
| colorpalette | Ext.ColorPalette | 调色板 |
| datepicker | Ext.DatePicker | 日期选择面板 |

容器及数据类组件

|  |  |  |
| --- | --- | --- |
| xtype | Class | 描述 |
| window | Ext.Window | 窗口 |
| viewport | Ext.ViewPort | 视口，即浏览器的视口，能随之伸缩 |
| box | Ext.BoxComponent | 盒子组件，相当于一个<div> |
| component | Ext.Component | 组件 |
| container | Ext.Container | 容器 |
| panel | Ext.Panel | 面板 |
| tabpanel | Ext.TabPanel | 选项面板 |
| treepanel | Ext.tree.TreePanel | 树型面板 |
| flash | Ext.FlashComponent | 显示Flash的组件(Since3.0) |
| grid | Ext.grid.GridPanel | 表格 |
| editorgrid | Ext.grid.EditorGridPanel | 可编辑的表格 |
| propertygrid | Ext.grid.PropertyGrid | 属性表格 |
| editor | Ext.Editor | 编辑器 |
| dataview | Ext.DataView | 数据显示视图 |
| listview | Ext.ListView | 列表视图 |

工具栏组件：

|  |  |  |
| --- | --- | --- |
| xtype | Class | 描述 |
| paging | Ext.PagingToolbar | 分页工具条 |
| toolbar | Ext.Toolbar | 工具栏 |
| tbbutton | Ext.Toolbar.Button | 工具栏按钮 |
| tbfill | Ext.Toolbar.Fill | 工具栏填充区 |
| tbitem | Ext.Toolbar.Item | 工具条项目 |
| tbseparator | Ext.Toolbar.Separator | 工具栏分隔符 //等价于 "-", |
| tbspacer | Ext.Toolbar.Spacer | 工具栏空白 |
| tbsplit | Ext.Toolbar.SplitButton | 工具栏分隔按钮 |
| tbtext | Ext.Toolbar.TextItem | 工具栏文本项 |

菜单组件：

|  |  |  |
| --- | --- | --- |
| xtype | Class | 描述 |
| menu | Ext.menu.Menu | 菜单 |
| colormenu | Ext.menu.ColorMenu | 颜色选择菜单 |
| datemenu | Ext.menu.DateMenu | 日期选择菜单 |
| menubaseitem | BaseItem |  |
| menucheckitem | Ext.menu.CheckItem | 选项菜单项 |
| menuitem | Ext.menu.Item |  |
| menuseparator | Ext.menu.Separator | 菜单分隔线 |
| menutextitem | Ext.menu.TextItem | 文本菜单项 |

表单及表单域组件：

|  |  |  |
| --- | --- | --- |
| xtype | Class | 描述 |
| form | Ext.FormPanel/Ext.form.FormPanel | 表单面板 |
| checkbox | Ext.form.Checkbox | 多选框 |
| combo | Ext.form.ComboBox | 下拉框 |
| datefield | Ext.form.DateField | 日期选择项 |
| timefield | Ext.form.TimeField | 时间录入项 |
| field | Ext.form.Field | 表单字段 |
| fieldset | Ext.form.FieldSet | 表单字段组 |
| hidden | Ext.form.Hidden | 表单隐藏域 |
| htmleditor | Ext.form.HtmlEditor | HTML编辑器 |
| label | Ext.form.Label | 标签 |
| numberfield | Ext.form.NumberField | 数字编辑器 |
| radio | Ext.form.Radio | 单选按钮 |
| textarea | Ext.form.TextArea | 多行文本框 |
| textfield | Ext.form.TextField | 单行文本框 |
| trigger | Ext.form.TriggerField | 触发录入项 |
| checkboxgroup | Ext.form.CheckboxGroup | 编组的多选框(Since2.2) |
| displayfield | Ext.form.DisplayField | 仅显示，不校验/不被提交的文本框 |
| radiogroup | Ext.form.RadioGroup | 编组的单选按钮(Since2.2) |

图表组件：

|  |  |  |
| --- | --- | --- |
| xtype | Class | 描述 |
| chart | Ext.chart.Chart | 图表组件 |
| barchart | Ext.chart.BarChart | 柱状图 |
| cartsianchart | Ext.chart.CartesianChart |  |
| columnchart | Ext.chart.ColumnChart |  |
| linechart | Ext.chart.LineChart | 连线图 |
| piechart | Ext.chart.PieChart | 扇形图 |

数据集Store：

|  |  |  |
| --- | --- | --- |
| xtype | Class | 描述 |
| arraystore | Ext.data.ArrayStore |  |
| directstore | Ext.data.DirectStore |  |
| groupingstore | Ext.data.GroupingStore |  |
| jsonstore | Ext.data.JsonStore |  |
| simplestore | Ext.data.SimpleStore |  |
| store | Ext.data.Store |  |
| xmlstore | Ext.data.XmlStore |  |

# 代码摘抄

|  |
| --- |
| json方法应用:decode:将json字符串转换为对象，encode：将对象转换为json字符串  var Mystrjson="{Name:'陈建强',Sex:'男',Age:26,Married:false}";//第一json字符串  var Myobjson=Ext.decode(Mystrjson);//将json字符串转换为对象  Myobjson.Name;//使用json对象属性  var Mystrjson2=Ext.encode(Myobjson);//将json对象转换为json字符串 其中Mystrjson=Mystrjson2 |
| **弹出提示框**  Ext.onReady(function(){      Ext.MessageBox.alert('Message', 'Hello World ! ');  }); |
| **事件延迟处理**  Ext.get("btnAlert").addListener("click",a, **this**, {delay:1000});  Ext.get("btnAlert").on("click", a, **this**, {delay:2000});//单位毫秒 |
| **禁止销毁**  所有组件(Component)都有beforedestroy这么一个事件  win.on("beforedestroy", **function**() {  alert("想关闭我，这是不可能的!");  **this**.show();  **return** **false**;  }); |
| **写入body**  renderTo : document.body,  renderTo : Ext.getBody(), |
| **获取某ID的值**  Ext.get('loginName').getValue(); |
| **设置图片的src**  Ext.get('ext-gen159').dom.src='2222222222'; |
| **设置某文本输入框的值**  Ext.getCmp("newPassword").setValue(null); |

## 工具栏(tbar顶部工具栏, bbar底部工具栏)的写法

|  |
| --- |
| Ext中的工具栏项目主要包含下面的类：  Ext.Toolbar.Button－，xtype为tbbutton //按钮  TextItem－ //文本  Ext.Toolbar.Fill－ //空白填充, 使后面的右对齐  Separator－ //竖线分隔符  Spacer－ //  SplitButton－ // |
| tbar : [  new Ext.Toolbar.TextItem('工具栏:'), //工具栏标题, 位于最左侧  {pressed : **true**, text : '刷新'},  {xtype:"tbfill"}, //填充, 后面的组件靠右对其  {pressed:**true**,text:'添加'},  {xtype:"tbseparator"},  {pressed:**true**,text:'保存'},  "注册时间:", validTimeBegin, " - ",validTimeEnd,  new Ext.Button(search)  ] |
| //高级查询  **var** search = **new** Ext.Action({  text:'高级查询',  iconCls:'page\_find',  handler:**function**(){  advancedSearch.show();  }  }); |
| //录入开始时间  **var** validTimeBegin = **new** Ext.form.DateField({  id:'regTimeBegin',  name:'regTimeBegin',  width : 100,  //allowBlank : false,//不允许为空  readOnly:**true**,//设置只读  //value : new Date(),//设置默认值  ///dateRange:{begin:'validTimeBegin',end:'validTimeEnd'},//用于vtype类型dateRange  //vtype:'dateRange',  format:'Y-m-d'//显示日期的格式  });    //录入结束时间  **var** validTimeEnd = **new** Ext.form.DateField({  id:'regTimeEnd',  name:'regTimeEnd',  width : 100,  //allowBlank : false,//不允许为空  readOnly:**true**,//设置只读  //value : new Date(),//设置默认值  //dateRange:{begin:'validTimeBegin',end:'validTimeEnd'},//用于vtype类型dateRange  //vtype:'dateRange',  format:'Y-m-d'//显示日期的格式  }); |

## form窗口

|  |
| --- |
| Ext.onReady(**function**() {  **var** form = **new** Ext.form.FormPanel({  baseCls: 'x-plain',  labelWidth: 55,  url:'save-form.php',  defaultType: 'textfield',  items: [{  fieldLabel: 'Send To',  name: 'to', //文本输入框的name="to"  emptyText:'请输入...', //文本输入框中的灰色提示文字  anchor:'100%', // anchor width by percentage  allowBlank:**false,** //验证: 不能为空  blankText:'不能为空'  },{  fieldLabel: 'Subject',  name: 'subject',  anchor: '100%' // anchor width by percentage  }, {  xtype: 'textarea',  hideLabel: **true**,  name: 'msg',  anchor: '100% -53' // anchor width by percentage and height by raw adjustment  }]  });  **var** window = **new** Ext.Window({  title: 'Resize Me',  width: 500,  height:300,  minWidth: 300,  minHeight: 200,  layout: 'fit',  plain:**true**,  bodyStyle:'padding:5px;',  buttonAlign:'center',  items: form,  buttons: [{  text: 'Send'  },{  text: 'Cancel'  }]  });  window.show();  }); |

## validate(校验)

|  |
| --- |
| items : [ {  xtype : 'textfield',  name : 'username',  fieldLabel : 'username',  selectOnFocus:true,  //得到焦点时自动选择文本  allowBlank : **false**,  blankText : '请输入用户名',  msgTarget : 'qtip',  regex:/^([\w]+)(.[\w]+)[\*@([\w-]+\.){1,5}([A-Za-z]){2,4}$/](mailto:*@([\w-%5d+\.)%7b1,5%7d(%5bA-Za-z%5d)%7b2,4%7d$/),  regexText:"用户名必须为Email格式"  }, {  xtype : 'textfield',  fieldLabel : 'password',  inputType : 'password',  name : 'pwd',  allowBlank : **false**,  blankText : '密码不能为空',  msgTarget : 'qtip'  } ], |
| 有一个方法, 可以不用在每个表单里写msgTarget, 只需在定义表单前面写:  Ext.QuickTips.init();// 初始化显示提示信息。没有它提示信息出不来。  Ext.form.Field.prototype.msgTarget = "under";//title,qtip,side |

**msgTarget(提示信息显示地方)**

|  |  |
| --- | --- |
| msgTarget:"qtip" |  |
| msgTarget:"title" |  |
| msgTarget:"under"  使用under时要注意表单的高度 |  |
| msgTarget:"side"  使用side是要注意表单的宽度 |  |
| regex:/^([\w]+)(.[\w]+)\*@([\w-]+\.){1,5}([A-Za-z]){2,4}$/,     regexText:"用户名必须为Email格式" |  |

## JS框架中的onDisable方法无效, 需要重写onDisable方法

|  |
| --- |
| Ext.override(Ext.form.HtmlEditor, {  onDisable: **function**(){  **if**(**this**.rendered){  **this**.wrap.mask();  }  Ext.form.HtmlEditor.superclass.onDisable.call(**this**);  },  onEnable: **function**(){  **if**(**this**.rendered){  **this**.wrap.unmask();  }  Ext.form.HtmlEditor.superclass.onEnable.call(**this**);  }  }); |

## RadioGroup重写的getValue和setValue

|  |
| --- |
| Ext.override(Ext.form.RadioGroup, {  getValue: **function**(){  **var** v;  **if** (**this**.rendered) {  **this**.items.each(**function**(item){  **if** (!item.getValue())  **return** **true**;  v = item.getRawValue();  **return** **false**;  });  }  **else** {  **for** (**var** k **in** **this**.items) {  **if** (**this**.items[k].checked) {  v = **this**.items[k].inputValue;  **break**;  }  }  }  **return** v;  },  setValue: **function**(v){  **if** (**this**.rendered)  **this**.items.each(**function**(item){  item.setValue(item.getRawValue() == v);  });  **else** {  **for** (**var** k **in** **this**.items) {  **this**.items[k].checked = **this**.items[k].inputValue == v;  }  }  }  }); |
| 调用  **var** lastNameAuditRadio= **new** Ext.form.RadioGroup({  labelSeparator:'',  height:20,  items : [  {  boxLabel: '通过',  inputValue: **true**,  name: 'lastNameAudit',  checked: **true**  },  {  boxLabel: '不通过',  inputValue: **false**,  name: 'lastNameAudit'  }  ]  });  lastNameAuditRadio.getValue(); |

# ------------------------------------------

# 组件

# 面板

# 窗口 及 对话框

# 布局Layout

## border区域布局

## Form布局回显数据

|  |
| --- |
| **表单数据回显, 手动拼接数据内容**  Ext.get('homeAddress').dom.value = data.result.data.county + data.result.data.homeAddress;  备注:  1. JS对象数据结构, 使用firebug来查看;  2. data.result.data.county, 其中data.result.data为ext中的json对象结构, county为数据所在的key |

# 表格

|  |
| --- |
| 渲染数据  **function** **getUserName**(value, p, r) {  **return** r.data['lastName']+ r.data['firstName'];  };  {header: "用户名", columnWidth:.2, dataIndex: 'userName', sortable: **true**, renderer:**getUserName**}, |
| //设置复选框  **var** sm = **new** Ext.grid.CheckboxSelectionModel();    //表头  **var** cm = **new** Ext.grid.ColumnModel([  **new** Ext.grid.RowNumberer(),//自动行号  sm,//复选框  {header: "国安ID", columnWidth:.2, dataIndex: 'loginName', sortable: **true**},  {header: "用户状态", columnWidth:.2, dataIndex: 'status'},  {header: '操作', width: 130, dataIndex:'', renderer:operate}  ]);  cm.defaultSortable = **true**; |

# 数据存储Stroe

# TreePanel

# 表单验证

|  |  |  |  |
| --- | --- | --- | --- |
| **if**(!formPanel.form.isValid()){  **return**;  }  校验textfield一般来说有这几个常见的方式：  **1.不能为空**  allowBlank:false  **2.最大最小长度**  minLength: 10,  maxLength: 100,  **3.邮件格式**  vtype: "email",  **4.url格式**  vtype: "url",  5. **注册校验器**  例如: 密码输入两次，这个比较复杂，所幸的是extjs里的例子有的   |  | | --- | | 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** val2 = Ext.get(field.confirmTo).getValue(); //取得confirmTo的那个id的值  **return** (val == val2);  } **else** {  **return** **true**;  }  }  },  phonecheckText: "两次密码不一致！"  ); | | {  fieldLabel:"请输入新密码",  inputType:'password',  name:'newPassword',  id:'newPassword',  allowBlank:**false**,  blankText:'不能为空'  },  {  fieldLabel:"请再次输入新密码",  inputType:'password',  name:'newPasswordToo',  id:'newPasswordToo',  allowBlank:**false**,  blankText:'不能为空',    vtype:"password",//自定义的验证类型  confirmTo:"newPassword"//要比较的另外一个的组件的id  },  {  fieldLabel:'电话',  vtype: "phonecheck",  name:'telephone',  id:'telephone'  }, | | //extjs验证手机(11位)或座机电话(7位或8位)  Ext.apply(Ext.form.VTypes, {  phonecheck: **function**(val, field) {  **var** reg = /(^[0-9]{3,4}\-[0-9]{7,8}$)|(^[0-9]{7,8}$)|(^\([0-9]{3,4}\)[0-9]{3,8}$)|(^0{0,1}13[0-9]{9}$)/;  **return** reg.test(val);  },  phonecheckText: "号码不匹配!"  }  ) | |

# ------------------------------------------