# 去除实验版Ext Js Trial水印

*一、*

*/\*font-family: ext-watermark;\*/  
/\*content: 'd';\*/*

*在 项目名-all\_1.css和 项目名-all\_2.css中*

查找直接注释掉即可。

二、

找到对应的主题下的sass文件

打开项目根目录下的ext/classic/theme-base/sass/src/window/Window.scss注释以下两行代码

*font-family: ext-watermark;*

*content: 'd';*

如果是panel的水印，是在panel/Panel.scss

更绝的，就是把ext/classic/theme-base/sass/src/Component.scss开头3-8行包含ext-watermark的样式全部注释掉，这个应该是全局的样式。

三、

查找ext-6.x.x\classic\theme-base\sass\etc\all.scss

将$ext-trial: true => $ext-trial: false

查找ext-6.x.x\build\ext.js

将"license": "trial" => "license": "commercial"

# Extjs

# Extjs 6.2手动搭建MVC项目

1. 在根目录新建app.js文件，内容如下：

***Ext***.application({  
 **name**: **'ExtjsTest7'**,  
 **extend**: **'ExtjsTest7.Application'**,  
 **requires**: [  
 **'ExtjsTest7.view.main.Main'** ],  
 **mainView**: **'ExtjsTest7.view.main.Main'**});

1. 新建app.json文件，内空可如下。

{  
 **"name"**: **"ExtjsTest7" //指定命名空间，为项目名。**

}

目的是为了sencha-jetbrains-plugin新建Extjs类时，指定正确的命名空间。

1. 建app目录。在此目录下，新建Application.js文件，内容如下：

***Ext***.define(**'ExtjsTest7.Application'**, {  
 **extend**: **'Ext.app.Application'**,  
 **name**: **'ExtjsTest7'**,  
 **stores**: [  
 *//* ***TODO: add global / shared stores here*** ],  
 launch: **function** () {  
 *//* ***TODO - Launch the application*** }  
});

1. 在app目录下建model,store,view目录，
2. 拷贝主题目录到项目中。classic主题目录在ext-6.x.x\build\classic\主题包\resources下。
3. 拷贝中文包locale-zh\_CN.js,此文件在ext-6.x.x\build\classic\locale下。
4. 拷贝ext-all.js文件。此文件在：在ext-6.x.x\build下。
5. 新建ext目录，拷贝ext-6.x.x\packages\core\src到此目录。同时将src目录复制两份到ext\classic\classic\src，ext\src目录。

非必需，主要是为了extjs的代码提示功能在Idea中显示正常。

1. index.html中引用例子如下：

<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>测试页</**title**>  
 <**link rel="stylesheet" href="theme/theme-triton/resources/theme-triton-all.css"** />  
 <**script src="js/ext-all.js"**></**script**>  
 <**script src="js/locale-zh\_CN.js"**></**script**>  
 <**script src="app.js"**></**script**>  
</**head**>

# ExtJS 的选择器

选择器的作用就是通过元素的标签名，属性名，CSS属性值对页面元素快速定位与选择。

选择页面元素，Ext.DomQuery

1. Ext.query(path,[root,type])

Ext.query = Ext.DomQuery.select;

path为选择符或xpath，选择符太多，用时查下。

root为查询开始的节点id，默认为document，为了提高查询速度，建议不要从document开始查询。

type为simple或select,当为simple时用id或标签名属性值等做简单查询。

当root已经通过Ext.get方法获得，可使用以下语法:

var el = Ext.get(Id); //id为节点id;

el.query(path);

1. 获取单一元素Ext.Dom.Element
2. Ext.get(id);

Ext.get = Ext.Dom.Element.get;

id可为元素id属性值，HTMLelement,Element,CompositeElementLite对象，数组，或document对象。

返回的是Element对象，可通过Ext.get(id).dom获取HTMLElement对象。

1. Ext.fly(id);

同上，区别：get能缓存，fly无缓存，fly对获取的元素只能进行一次操作。get会增加内存开销。

1. Ext.getDom(id);

同上，返回的是HTMLElement对象。

1. 其它方法：太多，用时查API。

# 获取元素集合:

Ext.CompositeElementLite与Ext.CompositeElement

1. Ext.select(selector,uniqe,root)

selector不能包括id选择符，

uniqe是否为每个元素创建Element，默认为false表示不创建，返回Ext.CompositeElementLite对象。集合元素类型为HtmlElement。

Ext.query和Ext.select的作用是一致的，同是根据CSS选择符查找出一个或多个元素。区别在于返回类型上。分别是：query方法返回的是JavaScript标准的数组类型；select方法返回的是CompositeElement类型。

select方法可以用来获得指定标签的DOM对象，其返回值是一个Ext.CompositeElement对象。是一个Element的数组，返回的结果可直接如同Element般地操作，一般比query方法常用。

# query、select、get三者的区别

query返回的是dom数组

select返回的是Ext.CompositeElement或Ext.CompositeElementLite对象。

Select(‘selector’,unique,’root’是一个element的数组。

unique是否为每一个元素创建为element对象，缺省为false,不创建。select返回CompositeElementLite对象，是一个HtmlElement对象数组。否则返回一个CompositeElement对象，是一个element对象数组。

get返回的是element对象

getDom返回的是dom对象（HtmlElement对象）。

element对象类是Ext对dom的封装。

Ext.get(‘id’).dom=Ext.getDom(‘id’);

# 元素生成器 Ext.dom.Helper

1. Ext.dom.Helper.markup(spec);

Ext.dom.Helper.markup = Ext.dom.Helper.CreateHtml；

spec可为字符串，数组，配置对象

配置对象主要有以下4个属性:

1. tag:html标签,如 div,p,a如果没有设置默认为div，
2. cls:元素样式名称，
3. html:元素内html代码(innerHtml的值)，
4. children或cn:要创建的元素下的子元素，根据需要嵌套。

可以直接使用Html元素的属性作为配置对象的属性。

1. 其它方法

insertHtml,append,insertAfter,insertBefore,insertFirst,overview,applyStyles等，

# 事件

## 为元素绑定事件

语法：Ext.EventManager.on(el,eventName,fn[,scope,args])

一次只绑定一个事件的方式：为id为input1的文本框绑定两个事件：

Ext.EventManager.on(“input1”,”foucus”,focus);

Ext.EventManager.on(“input1”,”blur”,blur);

“foucus”为事件名称，focus为事件处理函数。

一次绑定多个事件，但使用相同的配置项

Ext.EventManager.on(“input1”,{

focus: focus,

blur:blur

});

一次绑定多个事件，使用不同的配置项

Ext.EventManager.on(“input1”,{

focus:{

fn:foucs

},

Blur:{

Fn:blur

}

});

## 为组件绑定事件

一种方式是在创建组件时在配置对象中配置listeners属性，另一种方式是在创建组件后使用on方法绑定。

触发事件用fireEvent方法。

# 管理数据模型：

Ext.AbstractManager(基类)与Ext.ModelManager

1. 具有相同管理功能的还有Ext.PluginManager，ComponetManager，都继承自Ext.AbstractManager。

ModelManager对象的作用是注册数据模型以及记录实例化的数据模型，以实现对数据模型及其实例的管理。

AbstractManager包含有两个对象:HashMap，types。Hashmap记录实例化的对象，types记录由Ext.define定义的对象，定义的数据模型存储在types对象中，通过数据模型名称可以检索到该数据模型。

常用方法：(基数中)

get:通过id获取实例化的对象，（从HashMap中）

create:在types找到对象，并实例化该对象。

each:枚举实例化的对象，依次接收三个参数key，value，length。key为HashMap对象中的关键字，value为实例化的对象，length为HashMap中条目总数。

getCount：HaspMap中的条目总数。

(ModelManager中)

getModel：根据模型名称返回模型，

create：实例化模型（在HashMap中创建一条记录）

1. 定义数据模型：Ext.data.Model

常用配置对象项：

extend:

fields:必需项，为该数据模型的字段，由Field对象组成的数组，

validations:可选项，数组，定义数据模型的验证项。

创建一个数据模型就是创建一条记录，

创建（实例化）一个数据模型的三种方法：

1. Ext.create(ModelName,config)
2. new ModelName(config)
3. Ext.ModelManager.create(config,ModelName)
4. 常用数据模型的配置项，属性和方法
5. 配置项

idProperty:定义模型的唯一id，其值为fields中定义的字段名称，默认值为id，

fields: 本模型的字段。它是Field数组。

1. 属性

fields:只读属性，MixedCollection对象，从中可获取字段信息。

stores:只读属性，从中可获取模型所属的store。

modified:只坊属性，保存了修改过的字段及修改值，

raw:如果是reader读取的数据，这里保存原始数据。

1. 方法

get:获取字段的值，

set:设置字段的值，使用对象模式可设置多个字段，如果关联了ui会自动刷新ui的显示。

getProxy,setProxy获取设置proxy

save使用proxy对象提交数据到服务端保存。

beginEdit:进入编辑状态，

cancelEdit:取消所有改变，

endEdit:结束编辑状态。

getChanges:获取被个性的值，

setDirty:标记记录被修改，

commit:确认修改，不可恢复。

destroy:使用proxy通知服务器删除当前模型实例对象对应的数据。

join/unjoin:设置/删除模型实例的store，

# Store

Ext使用的是三层架构，数据与UI组件是分离的，要访问的数据在Store里，不在UI组件里，因而，要操作数据，请使用store 对象的属性和方法

model是一条记录，store是这些记录的集合。store使用MixedCollection对象存储这些模型实例的，因而可以通过MixedCollection对象的属性和方法访问store的数据。

# 组件的管理

1. 组件管理及查询：Ext.ComponentManager与Ext.ComponentQuery

ComponentQuery对象提供了一种类似DomQuery对象的使用选择符查询组件的方法，DomQuery对象可使用DOM元素的标记作为选择。而ComponentQuery对象使用组件的xtype值作为选择符，可与DomQuery对象一样，使用id,在选择符内加入属性或使用伪类选择符来进行搜索。

ComponentQuery只用来搜索ComponentManager对象里的实例，而不是Dom树。

方法:

1. is:检查组件是否匹配选择，如果匹配返回true,否则返回false。
2. query:使用选择符查询组件。

# 焦点管理：

Ext.Focusmanager对象可为组件提供键盘导航功能，还可根据设置选择是否使用可视化手段提示当前焦点所在组件。

要开启组件焦点管理可执行:

Ext.FocusManager.enable(option)

如果option为真，则使用可视化手段显示当前焦点所在组件，否则不显示。

如果要关闭管理，使用disable。

# 格式化输出数据

Ext.String、Ext.Number、Ext.Date、Ext.util.Format

# 模板与超级模板

Ext.Template与Ext.XTemplate

Ext.Template是基础模板，Ext.XTemplate是从Ext.Template扩展出来的，具有强大功能的模板类。

创建模板可用new或Ext.create()。

例子：

var tpl = new Templated([‘<div>’,’<a href=”{src}”>{text}</a>‘,’</div’]);

var tpl = new Templated(‘<div><a href=”{src}”>{text}</a></div>’);

var tpl = new Templated(‘<div>’,’<a href=”{src}”>{text}</a>’,’</div>’);

以上代码结果是一样的。

如果要加入配置项如下：

var tpl = new Templated(‘<div>’,’<a href=”{src}”>{text}</a>’,’</div>’,{compiled:true,disableFormat:true});

Templated只有两个配置项，compiled为布尔值，决定模板是否进行编译，默认值为undefined，不编译，建议设置为true，可以提高运行速度。disableFormat也为布尔值，决定是否使用格式化函数，默认为false，使用格式化函数。如果模板没有格式化要求，设置为true，可以提高运行速度。

XTemplate:模板常用方法：

1. append:将数据应用到模板，并追加到指定元素内。
2. applyTemplate或apply:将数据应用到模板并返回生成的代码。
3. compile:编译模板。
4. overwrite: 将数据应用到模板，覆盖指定元素内容。
5. 例子：

**var** star= 4;  
**var** data = {**starnum**:3};  
**var** starArr = [star>=1,star>=2,star>=3,star>=4,star>=5];  
**var** tpl = **new *Ext***.XTemplate(  
 **'<tpl for=".">'**,  
 **'<span>'**,  
 **'<img src="images/{[values?"Smile.png":"Fury.png"]}"/>'**,  
 **'</span>'**,  
 **'</tpl>'**);  
**var** tplStar = **new *Ext***.XTemplate(  
 **'<tpl for=".">'**,  
 **'<span>'**,  
 **'<img src="images/{[values?"Smile.png":"Fury.png"]}"/>'**,  
 **'</span>'**,  
 **'</tpl>'**);  
**var** tpl2 = **new *Ext***.XTemplate(**'<div>'**,  
 **'{starnum:this.getStarImgs}'**,  
 **'</div>'**,  
 {  
 getStarImgs: **function** (v) {  
 **var** vv = [v>=1,v>=2,v>=3,v>=4,v>=5];  
 **return** tplStar.apply(vv);  
 }  
 }  
);  
**var** el = ***Ext***.get(**'stars1'**);  
**var** e2 = ***Ext***.get(**'stars2'**);  
tpl.append(el,starArr);  
tpl2.append(e2,data);

# 状态管理：

Ext.state.Manager、Ext.state.Provider、Ext.state.LocalStoreProvider、Ext.state.CookieProvider

Ext.state.Manager对象使用状态提供者启示组件的状态，默认使用StateProvider对象，也是使用.LocalStoreProvider与CookieProvider。使用set方法可在状态提供内添加一个状态信息，使用get方法，可返回指定的状态id的状态信息，使用clear可清除状态提供者的全部状态信息，使用setProvider可改变状态状态管理器的状态提供者，getProvider返回当前的状态提供者。

StateProvider使用JavaScript对象作为存储状态信息的数据结构，使用set方法可在对象内部添加一个状态信息，其属性名称为状态id,使用get方法返回状态信息，

StateProvider使用的是内存对象，所以在浏览器关闭后所有状态信息都会丢失。

如可保存状态信息可使用Cookie或 html5支持的LocalStorage，

从StateProvider派出了CookieProvider与LocalStorageProvider对象，这两个提供者对象都会在初始化时从Cookie或LocalStorage中将状态信息恢复到内存的JavaScript对象里，然后在initState方法中根据组件的状态id就可以将组件的状态恢复到最后一次使用的情况。

为了将状态保存到Cookie或LocalStorage需要重写set，而且因为数据格式不同，还需要decodeValue和encodeValue将状态信息处理后再保存。

如果要使用CookieProvider与LocalStorage对象，必须在创建组件之前就设置好提供者，

Ext.state.Manager.setProvider(Ext.create(‘Ext.state.CookieProvider’));

或

Ext.state.Manager.setProvider(Ext.create(‘Ext.state.LocalStorageProvider’))

一劳永逸的方法是在本地化文件里，先判断Ext.state.Manager是否存在，如果存在就设置状态提供者，这样，无论在任何页面，只要加载了本地化文件，就已经定义好了状态提供者。

# 可以在Component子类中实现的模板方法

* initComponent 这个方法在constructor中被调用。它可以用来初始化数据，调协配置，添加事件监听器
* beforeShow 这个方法会在显示前调用
* onShow 允许在show操作中添加额外的形为。在调用了 supperclass的onShow后，组件才会被显示
* afterShow 这个方法在组件显示后调用
* onShowComplete 这个方法在afterShow方法完成后调用
* onHide 对组件在隐藏操作时，添加额外形为
* afterHide 在组件隐藏后调用
* onRender 在渲染阶段调用
* afterRender 当渲染完成时，此阶段的组件已经依据配置，应用了样式，我们可以添加样式，配置组件的可见性。
* onEnable 在enable操作时，添加额外的操作，并且调用父类的onEnable, 然后组件成为可用状态
* onDisable, Allows addition of behavior to the disable operation. After calling the superclass’s onDisable, the Component will be disabled.
* onAdded 组件被添加到容器时调用， 在当前阶段，组件已经在父容器的子组件items中。调用了superclass的onAdded后，然后ownerCt引用这个元素，如果有设置ref 配置，refOwner将被设置
* onRemoved 从父容器移除时调用，当前阶段，组件从父容器的子组件items中移除，但还没有被destroyed(如果parent 容器的autoDestroy设置为true时，它将会被destroy， 或者在调用时传递的第二个参数为truthy.) . 在调用supperclass的onRemoved后， ownerCt和refOwner将不在有效
* onResize 在resize操作时，添加额外的形为
* onPosition 在position 操作时，添加额外的形为
* onDestroy 在destroy操作时，添加额外的形为。在调用到superclass后，这个组件被摧毁
* beforeDestroy 在组件被destroy之前调用
* afterSetPosition 在组件的位置已经设置完成之后调用
* afterComponentLayout 在组件被布局后调用
* beforeComponentLayout 在组件被布局前调用

# 如何选择合适的类扩展

选择最合适的类去扩展是非常重要的， 这个基础类必须提供符合我们要求的功能。通常我们选择Ext.panel.Panel, 它可以被渲染，也可以管理其它组件。如果创建的组件，必须有header, footer, or toolbars， 则Ext.panel.Panel非常合适

Panel class有以下的功能

* Border
* Header
* Header tools
* Footer
* Footer buttons
* Top toolbar
* Bottom toolbar
* Containing and managing child Components

如果你定义的组件不需要上面的功能，则使用Panel就浪费了资源，如果一个UI组件不需要包含其它组件，换言之，如果只是简单的封装一些HTML的表单，则继承Ext.Component非常合适。

如果创建的组件只是用来包含其它的组件，而不需要我们在上面提及的Panel的功能。则可以使用 Ext.container.Container. 在Container级别，需要记住的是使用Ext.layout.container.Container 管渲染和管理子组件