Extjs4 开发笔记（一）——准备工作

2011-06-28　来源：mhzg　作者：mhzg　（共 2 条评论）

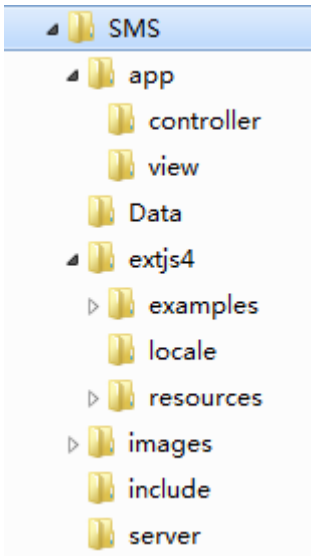
使用 extjs4 mvc+asp 来实现员工管理系统，系统会使用大量的 extjs4 组件及插件，期待着这个系统完成。

本文内容已经重新更新，旧版请查看 www.mhzg.net/a/20116/20116281100279-old.html。

重写原因：由于开始准备的时候，就是按照传统开发去做的，写了一部分之后，有网友和同事提出：“为什么不用 MVC 模式”呢？这样的问题让我对目前传统开发的心发生了一些细微变法，对啊，为什么不用 MVC 模式呢？我征求了一下同事及热心网友的意见，都同意使用 MVC 模式开发。从而，我删掉了原来所有目录，重新按照 MVC 模式去组织目录结构，很快的，目录结构准备好了。

那。。我们重新来过，使用 EXTJS4.0 的 MVC 模式，开发这套员工管理系统。给个简称吧。SMS（你懂得。呵呵！）。

一、建立环境：



Data：数据库文件夹，里面放着管理系统用的数据库文件。数据库目前只有三张表。分别是：

Menu：菜单项

user：员工注册信息

userinfo：员工个人资料信息

Images：图片目录，一些自定义的图片文件

Include：服务端文件目录，里面包含 ASP 所用到的 Conn.asp、Function.asp 等文件

App：整个 SMS 所用到的自定义 JS 文件，里面有一个 controller 文件夹，一个 view 文件夹。controller 文件夹放置主代码，view 文件夹放置各组件。这几个文件夹中的内容会在第二章进行介绍。

Extjs4：此目录放置 Extjs4 的库文件。

Server：服务端目录，里面包含 ASP 服务端获取数据的各种.ASP 文件。目前里面建立了一个叫 MenuLoader.asp 的文件，从名字上来看，这个文件是加载菜单使用。

OK，今天就介绍到这里，明天，我们会从头开始发开基于 Extjs4 MVC 模式的应用。

Extjs4 开发笔记(二)——框架的搭建

2011-06-29　来源：mhzg　作者：mhzg　(共 11 条评论)

头部、菜单、内容区及底部则完全分离成 4 个 JS 文件，我们将先实现这几个文件的基础功能，由于使用的是 Extjs4，所以我们一定要使用 Extjs4 动态加载功能。来动态加载这些文件。

本文内容已经重新更新，旧版请查看：

www.mhzg.net/a/20116/201162913210280-old.html

由于最近老出差，所以代码无法及时更新。导致近一周都没有更新。

废话不多说了，上篇文章建立了比较基础的文件。今天开始搭建大体的框架，由于 Extjs4 在组件建立方面有了很大的改变，所以第一次建立的框架页面还是费了比较长的时间。本章内容增加了一些图片及 CSS 文件，目的是为了美化整个界面。增加的 CSS 文件：

注意事项：layout、region 的使用，如果没有看 API 及相关文档的话，那么面对错误对话框的时候，还不知道是怎么回事。

本文将 main.js 放到了/app/controller 文件夹下，这将是整个项目的主体文件。

而头部、菜单、内容区及底部则完全分离成 4 个 JS 文件，我们将先实现这几个文件的基础功能，之后我们会慢慢完善这些组件。而整个页面的填充，也使用一个 JS 文件来完成。也许有人会问，这么多文件，是不是要都在 index.html 中引入啊。这样想的话，就错了哦。因为我们使用的是 Extjs4，所以我们一定要使用 **Extjs4 动态加载**功能。

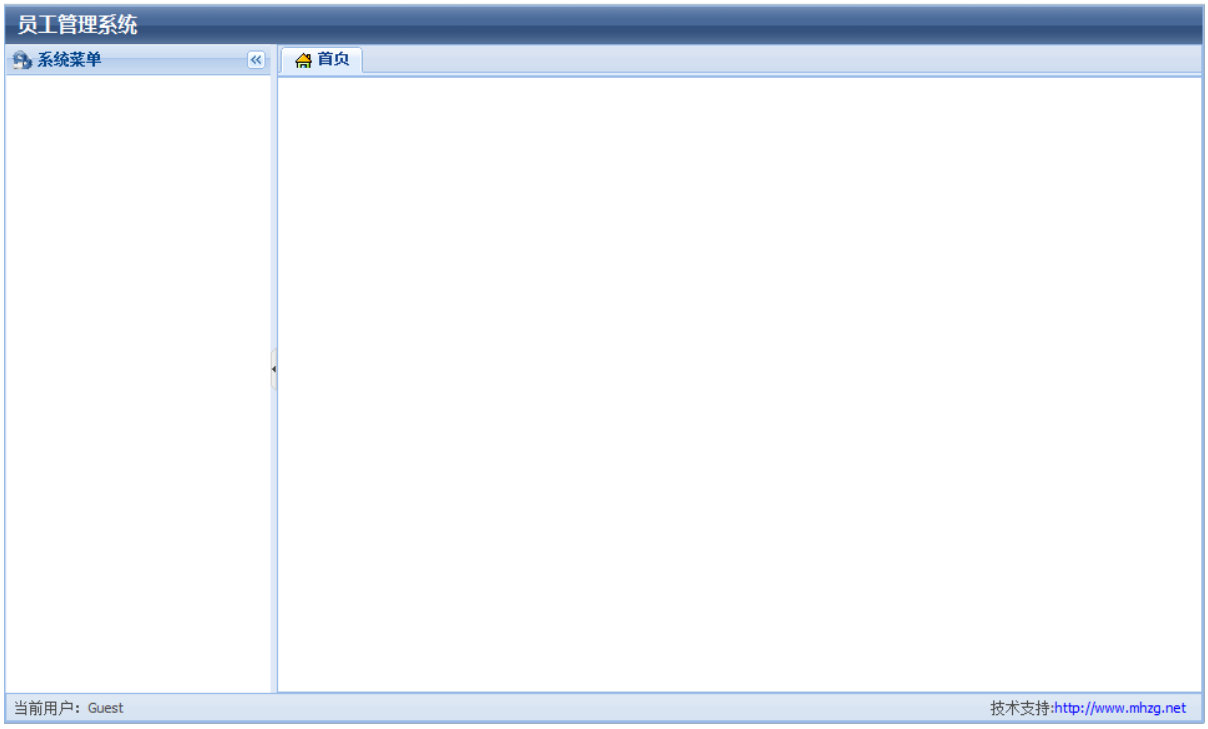
先来看下自定义 CSS（sytle.css）：

```
#header { background: #7F99BE url(/images/layout-browser-hd-bg.gif) repeat-x center; }
#header h1 {font-size: 16px;color: #fff;padding: 3px 10px; font-family:"微软雅黑","黑体"}

.tabs{}
.tabs{background-image: url(../images/menuPanel/bulletin_manager.gif) !important;}
.manage{background-image: url(../images/menuPanel/admin.gif) !important;}
.home{background-image: url(../images/home.gif) !important; line-height:30px;}
.icon-menu{background-image: url(../images/menuPanel/sys.gif) !important;}
```

图片文件夹就不放上来了。从以前的项目中拷贝了一些比较靠谱的图片，大家完全可以自己去下载一些 ICON 图标文件而为己所用。

搭建的框架是经典的 EXTJS 布局模式，如图所示：



首先，我们建立 index.html 和 app.js，index.html 代码为：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>员工管理系统</title>
<link rel="stylesheet" type="text/css" href="extjs4/resources/css/ext-all.css" />
<!--引入自定义 CSS-->
<link rel="stylesheet" type="text/css" href="extjs4/resources/style.css" />
<script type="text/javascript" src="extjs4/ext-all-debug.js"></script>
<script type="text/javascript" src="app.js"></script>
</head>

<body>
</body>
</html>
```

App.js：

```
Ext.Loader.setConfig({enabled: true});
Ext.application({
    name: 'SMS',
    appFolder: 'app',
    controllers: [
        'Main'
    ]
});
```

稍做解释：

Ext.Loader.setConfig({enabled: true});//意思是开启 Ext.Loader。Ext.Loader 是动态加载的核心哦。。

Ext.appliction({...});看字面意思吧，不解释。

配置中的 name，我理解为是 Extjs3.x、Extjs2.x 中的命名空间。

appFolder，应用文件夹名字。

controllers，控制单元的名字，这里我们定义为 Main。那么根据 Extjs4 动态加载的要求，我们需要在/app/controllers 文件夹下建立 Main.js 文件，作为控制单元。有关 Extjs4 动态加载机制，请参考：www.mhzg.net/a/20117/2011721040290.html

Main.js：

```
Ext.define('SMS.controller.Main',{
    extend: 'Ext.app.Controller',
    init : function(){
    }
})
```

这里的 Main.js 中只是定义了 Main 这个类，且继承了 Ext.app.Controller，其余都没有写。看到这里，会有人很奇怪了，index.html 中引入了 app.js，而 app.js 只是创建了 Main 这个类，但 Main.js 什么都没有，那么页面中为什么会显示出框架页面呢？这也是最多人所疑惑的。下面我来解释下这个问题。所有的原因就在于 app.js 这个文件中，app.js 文件定义了 Ext.application。而 Ext.application 中有个属性是 autoCreateViewport ，这个属性是 Boolean 类型，如果值为 true，那么 Extjs4 会自动加载 view/Viewport.js 文件，如果值为 flase，那么必须要自己去创建一个 View，这就是为什么 app.js 和 Main.js 文件都没有写相关代码，也会有界面出现。

整理下思路，由于 Extjs4 自动加载了 view/Viewport.js，而 Viewport.js 文件包含了头部、菜单、内容区及底部这 4 个组件，那么我们必须先完成这 4 个文件的编写，同样，由于这 4 个文件是界面型的，我们将这 4 个文件都放到 view 文件夹下。

view 文件夹下共 5 个 JS 文件，分别为：

Header.js、Menu.js、South.js、TabPanel.js 及 Viewport.js

这 5 个 js 文件的作用，我们一一介绍。

5 个 js 文件都包含了 Ext.applyIf、callParent。由于篇幅问题，Ext.applyIf、callParent 等方法，请参考 Extjs4 相关 API。

Header.js：这个是头部，也就是深蓝色底子。白色字体，那块，上面写着员工管理系统。

代码为：

```
Ext.define('SMS.view.Header', {
    extend: 'Ext.Component',
    initComponent: function() {
        Ext.applyIf(this, {
            xtype: 'box',
            cls: 'header',
            region: 'north',
            html: '<h1>员工管理系统</h1>',
            height: 30
        });
        this.callParent(arguments);
    }
});
```

Menu.js：

```
Ext.define('SMS.view.Menu',{
    extend: 'Ext.tree.Panel',
    initComponent : function(){
        Ext.apply(this,{
            id: 'menu-panel',
            title: '系统菜单',
            iconCls:'icon-menu',
            margins : '0 0 -1 1',
            region:'west',
            border : false,
            enableDD : false,
            split: true,
            width : 212,
            minSize : 130,
            maxSize : 300,
            rootVisible: false,
            containerScroll : true,
            collapsible : true,
```

```
            autoScroll: false
        });
        this.callParent(arguments);
    }
})
```

TreePanel 并没有加载菜单项，关于 Extjs4 Tree，我们后面会介绍。

TabPanel.js：

```
Ext.define('SMS.view.TabPanel',{
    extend: 'Ext.tab.Panel',
    initComponent : function(){
        Ext.apply(this,{
            id: 'content-panel',
            region: 'center',
            defaults: {
               autoScroll:true,
               bodyPadding: 10
            },
            activeTab: 0,
            border: false,
           //plain: true,
            items: [{
              id: 'HomePage',
              title: '首页',
              iconCls:'home',
              layout: 'fit'
            }]
        });
        this.callParent(arguments);
    }
})
```

South.js：

```
Ext.define('SMS.view.South',{
    extend: 'Ext.Toolbar',
    initComponent : function(){
        Ext.apply(this,{
            id:"bottom",
            //frame:true,
```

```
            region:"south",
            height:23,
            items:["当前用户：Guest",'->',"技术支持:<a href='http://www.mhzg.net' target='_blank' style='text-decoration:none;'><font color='#0000FF'>http://www.mhzg.net</font></a>  "]
        });
        this.callParent(arguments);
    }
})
```

文件都创建好了。我们进行最后一部，布局。

Viewport.js：

```
Ext.define('SMS.view.Viewport',{
    extend: 'Ext.Viewport',
    layout: 'fit',
    hideBorders: true,
    requires : [
        'SMS.view.Header',
        'SMS.view.Menu',
        'SMS.view.TabPanel',
        'SMS.view.South'
    ],
    initComponent : function(){
        var me = this;
        Ext.apply(me, {
            items: [{
                id:'desk',
                layout: 'border',
                items: [
                    Ext.create('SMS.view.Header'),
                    Ext.create('SMS.view.Menu'),
                    Ext.create('SMS.view.TabPanel'),
                    Ext.create('SMS.view.South')
                ]
            }]
        });
        me.callParent(arguments);
    }
})
```

重点：requires 属性，这个我理解为创建引用。稍懂编程语言的人应该都明白。但是光引用还不够，我们还需要去实例化它，也就是 Ext.create。至此，我们的框架已经顺利搭建完毕。

今天的工作也就是这么多，搭建完框架之后，会慢慢丰富整个系统。本来想连菜单的树也完成，最后想了想，这工作还是留到明天吧。因为树涉及到了异步获取，需要有服务端程序，今天弄好框架之后，把服务端代码写好了，明天来完成这棵树的实现吧。

需要注意的一点，在 extjs4 中，只要定义了布局为 border，那么他的 items 中必须要有 **region:'center'** 的组件，否则将会提示错误。貌似在 extjs3.x 甚至是以前的版本都没发现有这样的要求，因为这个，费了老大的劲才调整过来，再一看，代码全部变了，已经跟 extjs3.x 的风格完全不同了。令人欣喜的是，现在的代码完全符合 extjs4 的风格，也完全符合我的预期。

好了，本章内容就结束了。下篇文章，将实现菜单的功能。敬请期待。

关于 Extjs4 开发笔记（二）的补充说明

2011-07-27　来源：mhzg　作者：mhzg　(共 1 条评论)

Extjs4 开发笔记二，主要讲述了框架的搭建。最近有网友跟我反映，说按照文章内容照着完成后，不显示框架，只有空白一片，问了几位网友下，发现他们都是使用的 Exjts4.02a 的版本，最后，在官方最新的 API 中发现，Ext.application 下的属性 autoCreateViewport 默认

Extjs4 开发笔记二，主要讲述了框架的搭建，链接地址是：www.mhzg.net/a/20116/201162913210280.html，最近有网友跟我反映，说按照文章内容照着完成后，不显示框架，只有空白一片，问了几位网友下，发现他们都是使用的 Exjts4.02a 的版本，最后，在官方最新的 API 中发现，Ext.application 下的属性 autoCreateViewport 默认变成了 false，如此一来，就不能自动创建 Viewport 了。这里提供下解决办法。

如果你的 Extjs4 版本为 Extjs4.02a 以下版本，则不需要任何改动。完全可以显示完整框架。

如果你的 Extjs4 版本为 Extjs4.02a 或以上版本，则需要修改 app.js 文件，其内容为：

```
1.  Ext.Loader.setConfig({enabled: true});
2.  Ext.application({
3.      name: 'SMS',
4.      autoCreateViewport: true,
5.      appFolder: 'app',
6.      controllers: [
7.          'Main'
```

```
8.      ]
9.  });
```

注意，此 app.js 加入了 autoCreateViewport: true 设置。

这里要特别感谢网友“西门吹牛”和“火焰楼兰”。

最后，谢谢大家对 mhzg 的关注。

Extjs4 开发笔记(三)——菜单的实现

2011-06-30　来源：mhzg　作者：mhzg　(共 3 条评论)

本文实现以及点击菜单后在右边内容区打开一个新的 Panel。本篇文章的内容主要包括两个方面，Extjs4 Tree 及 Extjs4 TablePanel。

本文内容已经更新，旧文档请看这里：www.mhzg.net/a/20116/20116301260285-old.html

上篇文章介绍了搭建一个空的框架（链接地址：www.mhzg.net/a/20116/201162913210280.html），使得管理系统有了大致的模样，今天工作的主要内容就是菜单的实现以及点击菜单后在右边内容区打开一个新的 Panel。本篇文章的内容主要包括两个方面，Extjs4 Tree 及 Extjs4 tabPanel。

在 Extjs 应用中实现菜单，无疑 Tree 是最好的选择，将菜单项直接写到 Tree 中，也未尝不可，但后期的维护会非常麻烦，那么最好的选择就是异步获取菜单节点，这样既有利于后期维护，也可以节省 JS 代码的编写量。

实现异步加载，那必须要有数据库和服务端程序。管理系统使用的是 ACCESS 数据库。在数据库中建立 Menu 表，表一共有 4 个字段，分别是 ID、MenuName、ParentID 和 cls.

ID：自增长类型

MenuName：代表菜单的名字。ParentID

ParentID：父 ID

cls：样式名（这个需要在样式表中体现出来，才会有效果）

数据库有了，接下来就是服务端的编写，其实 JS 框架的好处在于服务端无论用什么都可以，这里我就使用 ASP 来实现了。由于服务端涉及的方面比较多，较多代码帖出来会比较乱，这里只说下返回的形式。

菜单需要的 JSON 数据格式如下：

```
1.  [{"text":"管理员管理","id":"1","iconCls":"manage","leaf":true}]
```

这里注意一点：由于整棵菜单树都是异步获取的，所以节点并不需要递归，而树的 node.id 就是 JSON 数据中的 ID。点击父节点展开子节点的时候，发送的数据也是 node.id，这样正好解决获取子节点的问题。所有根节点的 ParentID 都为 0。那么第一次加载菜单的时候，正好可以获取所有的根节点了。

重点，菜单树的数据源编写，根据 MVC 原则，在 app 目录下建立 store 文件夹，这个文件夹下放置所有的获取数据的 js 文件，在 store 文件夹下建立 Menus.js，编写如下代码：

```
Ext.define('SMS.store.Menus',{
    extend: 'Ext.data.TreeStore',
    root: {
        expanded: true
    },
    proxy: {
        type: 'ajax',
        url: '/server/MenuLoader.asp'
    }

})
```

然后修改 view 文件夹下的 Menu.js 文件：

```
Ext.define('SMS.view.Menu',{
    extend: 'Ext.tree.Panel',
    alias: 'widget.smsmenu',
    requires:['SMS.store.Menus'],
    initComponent : function(){
        Ext.apply(this,{
            id: 'menu-panel',
            title: '系统菜单',
            iconCls:'icon-menu',
            margins : '0 0 -1 1',
            region:'west',
            border : false,
            enableDD : false,
            split: true,
            width : 212,
            minSize : 130,
            maxSize : 300,
            rootVisible: false,
            containerScroll : true,
            collapsible : true,
```

```
            autoScroll: false,
            store:Ext.create('SMS.store.Menus'),
        });
        this.callParent(arguments);
    }
})
```

如此，当页面打开时，就会自动加载菜单项了。但是目前来说，点击菜单的任何节点都没有任何作用，那么由于整个项目都要使用 Extjs 来完成，那么必须要实现点击节点在右边内容区显示相应的 Grid 或 Panel 等等。下面，我们编写代码实现这个功能。

原理：当点击菜单树的节点时，先要判断要打开的组件是否存在，如果存在，则在右边内容区激活当前组件，如果不存在，则创建一个组件，然后在右边内容区增加一个组件。当然。这里为了通用性更高，创建出的组件一律按 panel 为准。为了实现在右边内容区的 tabPanel 上增加或删除对应的 table，根据分离原则，我们需要在控制器上完成该操作，接下来，我们在 controller 文件夹下建立 Menu.js 文件，Menu.js 会完成这一系列的工作，具体代码如下：

```
Ext.define('SMS.controller.Menu',{
    extend: 'Ext.app.Controller',
    refs:[
        {ref: 'smsmenu',selector: 'smstablepanel'},
        {ref: 'tabPanel',selector:'smstablepanel'}
    ],
    init:function(){
        this.control({
            'smsmenu': {
                itemmousedown: this.loadMenu
            }
        })
    },
    loadMenu:function(selModel, record){
        if (record.get('leaf')) {
            var panel = Ext.getCmp(record.get('id'));
            if(!panel){
                panel ={
                    title: 'New Tab ' + record.get('id'),
                    iconCls: 'tabs',
                    html: 'Tab Body ' + record.get('id') + '<br/><br/>',
                    closable: true
                }
                this.openTab(panel,record.get('id'));
            }else{
```

```
                var main = Ext.getCmp("content-panel");
                main.setActiveTab(panel);
            }
        }

    },
    openTab : function (panel,id){
        var o = (typeof panel == "string" ? panel : id || panel.id);
        var main = Ext.getCmp("content-panel");
        var tab = main.getComponent(o);
        if (tab) {
            main.setActiveTab(tab);
        } else if(typeof panel!="string"){
            panel.id = o;
            var p = main.add(panel);
            main.setActiveTab(p);
        }
    }

})
```

关键点：refs 和 this.control，refs 在官方 API 中没有找到其解释，网上查了下，对该属性的解释是：凡是 component 都可以使用该属性在它的归属容器及归属容器的父节点中注入一个对该属性的引用名称。有了该引用名，和该组件有共同父节点的组件就可以比较方便的引用该组件。

而 this.control 则很容易理解了，即通过 Ext.ComponentQuery 为选定的组件添加监听。上面代码为我们的菜单节点添加了一个事件 loadMenu，loadMenu 中，先获取对应的 panel，如果该 panel 不存在，则使用 openTab 方法在内容区的 tabPanel 上增加一个 panel，如果存在，则激活该 panel。而 panel 不存在的话，这里只简单的创建了一个 panel，并没有任何意义，下篇文章，我们将详细讨论这个问题。

最后，我们需要修改 app.js，将我们创建的控制器加载进来。app.js：

```
Ext.Loader.setConfig({enabled: true});
Ext.application({
    name: 'SMS',
    appFolder: 'app',
autoCreateViewport:true,
    controllers: [
        'Menu'
    ]
});
```

这里唯一修改的地方就是将 Main 改为了 Menu。后面，我们会逐步完善。

今天的工作比较多，整理下今日的工作内容：

1、在 app 目录下建立了 store 文件夹，并建立了 Menus.js 文件，这个文件负责菜单数据的加载。

2、修改了 view 文件夹下的 Menu.js,使其可以加载菜单数据。

3、在 controller 文件夹下建立了 Menu.js，使其可以在内容区的 tabPanel 上增加新的 panel 组件。

4、修改 app.js，使其可以使菜单项正常使用。

Extjs4 开发笔记(四)——实现登录功能

2011-11-01　　来源：mhzg　　作者：mhzg　　(共 2 条评论)

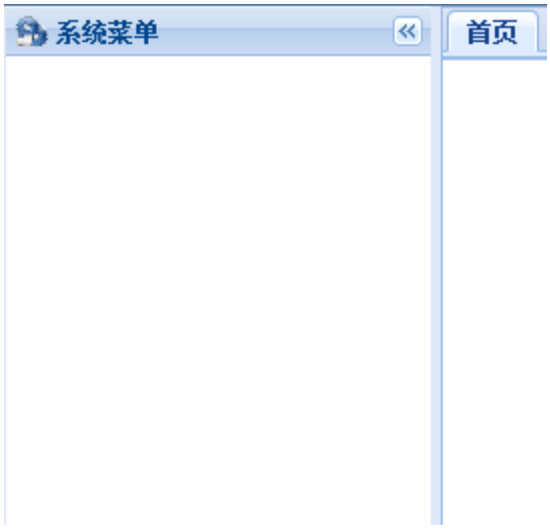
上篇文章介绍了如何实现菜单功能（点击查看），但是有个问题，就是管理系统必须是登录后才会显示菜单，而且菜单还要实现不同权限有不同的菜单项，本文将实现这个功能。

上篇文章介绍了如何实现菜单功能（点击查看），但是有个问题，就是管理系统必须是登录后才会显示菜单,而且菜单还要实现不同权限有不同的菜单项,本文将实现这个功能。

首先，将 server/MenuLoader.asp 修改，增加管理员验证功能。即

```
1.  If Session("Manage") <> "" Then
2.  '显示菜单项
3.  End If
```

这时，重新打开页面，由于有了基本的管理员验证，菜单不显示了。



接下来，开始制作登录，在 view 文件夹下建立 Login.js，checkcode.js，其中 Login.js 实现登录功能，有用户名、密码和验证码，验证码的实现，就是 checkcode.js，由于篇幅问题，checkcode.js 请查看本站另一篇文章， ExtJS4 学习笔记(十)---ExtJS4 图片验证码的实现。

主要是 Login.js：

```
Ext.define(SMS.view.Login',{
    extend:'Ext.window.Window',
    alias: 'widget.loginForm',
    requires: ['Ext.form.*','SMS.view.CheckCode'],
    initComponent:function(){
        var checkcode = Ext.create('SMS.view.CheckCode',{
            cls : 'key',
            fieldLabel : '验证码',
            name : 'CheckCode',
            id : 'CheckCode',
            allowBlank : false,
            isLoader:true,
            blankText : '验证码不能为空',
            codeUrl: '/include/checkCode.asp',
            width : 160
        })
        var form = Ext.widget('form',{
            border: false,
            bodyPadding: 10,
            fieldDefaults: {
                labelAlign: 'left',
                labelWidth: 55,
                labelStyle: 'font-weight:bold'
            },
            defaults: {
                margins: '0 0 10 0'
            },
            items:[{
                xtype: 'textfield',
                fieldLabel: '用户名',
                blankText : '用户名不能为空',
                name:'UserName',
                id:'UserName',
                allowBlank: false,
                width:240
```

```
36.             },{
37.                 xtype: 'textfield',
38.                 fieldLabel: '密　码',
39.                 allowBlank: false,
40.                 blankText : '密码不能为空',
41.                 name:'PassWord',
42.                 id:'PassWord',
43.                 width:240,
44.                 inputType : 'password'
45.             },checkcode],
46.             buttons:[{
47.                 text:'登录',
48.                 handler:function(){
49.                     var form = this.up('form').getForm();
50.                     var win = this.up('window');
51.                     if(form.isValid()){
52.                         form.submit({
53.                             clientValidation: true,
54.                             waitMsg:'请稍后',
55.                             waitTitle:'正在验证登录',
56.                             url:'/server/checklogin.asp',
57.                             success: function(form, action) {
58.                                 //登录成功后。
59.                                 //隐藏登录窗口，并重新加载菜单
```

??

??

??

??

```
60.                                 win.hide();
61.                                 Ext.getCmp('SystemMenus').store.load();
62.
63.                             },
64.                             failure: function(form, action) {
65.                                 Ext.MessageBox.show({
66.                                     width:150,
67.                                     title:"登录失败",
68.                                     buttons: Ext.MessageBox.OK,
69.                                     msg:action.result.msg
70.                                 })
71.                             }
72.                         });
73.                     }
74.                 }
```

```
            }]
        })
        Ext.apply(this,{
            height: 160,
            width: 280,
            title:'用户登陆',
            closeAction:'hide',
            closable : false,
            iconCls:'win',
            layout:'fit',
            modal : true,
            plain : true,
            resizable: false,
            items:form
        });
        this.callParent(arguments);
    }
});
```

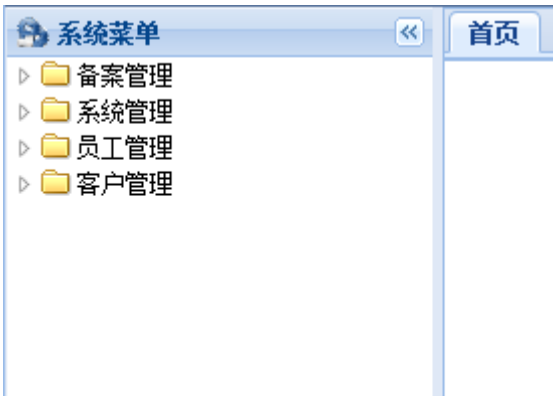
最终效果：

修改 controller 目录下的 Main.js:

```
Ext.define('SMS.controller.Main',{
    extend:'Ext.app.Controller',
    requires:['SMS.view.Login'],

    onLaunch : function(){
        var win;
        if(!win){
            win = Ext.create('SMS.view.Login').show();
        }
    }
})
```

这时，当页面加载的时候，会显示登录窗口，而登录成功后， 会隐藏登录窗口并加载菜单。最后附上登录成功后页面效果。



Extjs4 开发笔记(五)——动态 grid

2011-11-10　来源：mhzg　作者：mhzg　(共 4 条评论)

现在为止，框架有了，菜单有了。下一步工作就是实现每个菜单项目的数据显示，一般来讲，我们都是固定思维，通过编写大量的 grid 来实现各种数据的显示，在网络上有一些案例，都是每一个菜单项目，写一个 grid，而且点击树节点，都是通过判断或其他方法来创建相应

上一节我们说了如何实现登录，其地址是：Extjs4 开发笔记(四)——实现登录功能。现在为止，框架有了，菜单有了。下一步工作就是实现每个菜单项目的数据显示，一般来讲，我们都是固定思维，通过编写大量的 grid 来实现各种数据的显示，在网络上有一些案例，都是每一个菜单项目，写一个 grid，而且点击树节点，都是通过判断或其他方法来创建相应 grid 的实例，例如：

```
1.  if(node.id=="depmanage"){
2.          panel = new depManagePanel();
3.          main.openTab(panel);
4.        }else if(node.id=="telmanage"){
5.          panel = new telManagePanel();
6.          main.openTab(panel);
7.        }else if(node.id=="usermanage"){
8.          panel = new userManagePanel();
9.          main.openTab(panel);
10.       }else if(node.id=="userlog"){
```

```
        panel = new operationPanel();
        main.openTab(panel);
      }else{.....}}
```

但是在很多优秀的案例中，也不排除动态生成 gird 的可能，至少目前为止，我没见过此种案例，可能是孤陋寡闻吧。。

在这个项目的开发过程中，我不断思考，是否能动态生成 grid，使得工作一劳永逸。在查阅一些资料和自己动手尝试后，发现这样的办法可行，但前提是，需要服务端支持。单纯从开发角度来讲，我感觉这样的方法更加适合实际的开发。

在本节开始前，我们先来看一个问题。

树节点服务端返回的 json 数据：看了 API 之后，发现树的 sotre 返回 json 数据后，如果不指定其 fields，那么再整个树的节点集合中，只有固定的几种属性值，即 text，id、iconCls 和 leaf（注意：这里只是例举了这几种常见的属性值，至于其他的属性值，还是留给大家去探究吧）。所以大多数网友问我，为什么自定义的属性值，在 extjs4.0 中使用 record.get("xxxx")为什么获取不到？这其中的原因就是在 store 中没有指定 fields。也就是说，在本项目中，如果菜单树不指定 fields，那么是无法往下进行开发的。

回到正题，要实现动态创建 grid，我们先要可以让树有很多属性供我们所用，新建 Menu.js，将这个 js 文件放置到 app/model 文件夹下。其代码为：

```
Ext.define('SMS.model.Menu', {
extend: 'Ext.data.Model',
fields: ['id', 'text','iconCls','stores','columns'],
root: {
 expanded: true
},
proxy: {
 type: 'ajax',
 url: '/server/MenuLoader.asp'
}
})

```

MenuLoader.asp 返回如下 json：

```
[{"text":"备案列表
","id":"4","iconCls":"Folder","stores":"bastore","columns":[{text:'序号
',dataIndex:'ID'},{text:'公司名称',dataIndex:'kehu_name'}],"leaf":true},{"text":"新
增备案","id":"5","iconCls":"Folder","stores":"","columns":[],"leaf":true}]

```

修改 app/store 下的 Menus.js，更改为如下代码：

```
Ext.define('SMS.store.Menus',{
 extend: 'Ext.data.TreeStore',
 requires: 'SMS.model.Menu',
 model: 'SMS.model.Menu'
})
```

新建 bastore.js，将其放到 app/store 目录下，代码为：

```
Ext.define('SMS.store.bastore', {
    extend: 'Ext.data.Store',
    requires: 'SMS.model.beianlistmodel',
    model: 'SMS.model.beianlistmodel'
});


```

新建 beianlistmodel.js，将其放到 app/model 目录下，其代码为：

```
Ext.define('SMS.model.beianlistmodel', {
    extend: 'Ext.data.Model',
    fields: ['ID', 'kehu_name']//,这里只列出了自增值 id 和客户名称。

    //proxy: {
//        type: 'ajax',
//        url: '/server/getbeianlist.asp',
//        reader: {
//            type: 'json',
//            root: 'results'
//        }
//    }
});
```

服务端代码没有编写，所以将获取数据的代码暂时注释掉。

接下来，我们修改 app/controller 目录下的 Menu.js，增加一个 stores 属性，其代码为

stores: ['bastore'],

然后将测试代码

```
if (record.get('leaf')) {
                var panel = Ext.getCmp(record.get('id'));
                if(!panel){
                    panel ={
                        title: 'New Tab ' + record.get('id'),
                        iconCls: 'tabs',
                        html: 'Tab Body ' + record.get('id') + '<br/><br/>',
                        closable: true
                    }
                    this.openTab(panel,record.get('id'));
                }else{
                    var main = Ext.getCmp("content-panel");
                    main.setActiveTab(panel);
                }
            }
```

全部删除，重新写如下代码：

```
if (record.get('leaf')) {
                var panel = Ext.getCmp(record.get('id'));
                if(!panel){
        panel = Ext.create("Ext.grid.Panel",{
         store:record.get('stores'),
         columns:record.get('columns'),
         title: record.get('text'),
         id:record.get('text')+record.get('id'),
          viewConfig: {
          stripeRows: true
         },
         closable: true
        })
        this.openTab(panel,record.get('id'));
                }else{
        var main = Ext.getCmp("content-pane");
        main.setActiveTab(panel);
    }
```

```
19.          }
20.
```

稍做说明：Ext.create("Ext.grid.Panel"..... 首先创建了一个 gridpanel，由于我们增加了菜单树的 fields 属性，那么我们可以获取所有 fields 指定的属性值，通过上面的 json 数据

{"text":"备案列表","id":"4","iconCls":"Folder","stores":"bastore","columns":[{text:'序号',dataIndex:'ID'},{text:'公司名称',dataIndex:'kehu_name'}],"leaf":true}

我们就实现了动态生成 grid。app/controller 目录下的 Menu.js 全部代码为:

```
1.  Ext.define('SMS.controller.Menu',{
2.   extend: 'Ext.app.Controller',
3.   refs:[
4.     {ref: 'smsmenu',selector: 'smstablepanel'},
5.                   {ref: 'tabPanel',selector:'smstablepanel'}
6.
7.   ],
8.   stores: ['bastore'],
9.   init:function(){
10.   this.control({
11.    'smsmenu': {
12.                 itemmousedown: this.loadMenu
13.             }
14.   })
15.  },
16.  loadMenu:function(selModel, record){
17.   if (record.get('leaf')) {
18.             var panel = Ext.getCmp(record.get('id'));
19.             if(!panel){
20.     panel = Ext.create("Ext.grid.Panel",{
21.      store:record.get('stores'),
22.      columns:record.get('columns'),
23.      title: record.get('text'),
24.      id:record.get('text')+record.get('id'),
25.       viewConfig: {
26.       stripeRows: true
27.      },
28.      closable: true
29.     })
30.     this.openTab(panel,record.get('id'));
```

```
            }else{
     var main = Ext.getCmp("content-pane");
     main.setActiveTab(panel);
    }
        }

 },
 openTab : function (panel,id){
        var o = (typeof panel == "string" ? panel : id || panel.id);
  var main = Ext.getCmp("content-pane");
        var tab = main.getComponent(o);
        if (tab) {
            main.setActiveTab(tab);
        } else if(typeof panel!="string"){
            panel.id = o;
            var p = main.add(panel);
            main.setActiveTab(p);
        }
    }

})
```

声明，进行一个 Extjs4.0MVC 模式开发，是一个尝试的过程，随着 Extjs4.x 的不断升级、改进和修复，可能会造成当前项目无法运行或者报错，mhzg 会持续关注 extjs 的最新动态，如果有大幅改动，mhzg.net 会将最新的 extjs4 版本下载到本地进行测试，如果本项目无法运行，那么 mhzg.net 会进行修复并在 mhzg.net 上发布补充文档进行说明和修正。而在整个开发过程中，难免会有一些地方不尽人意，欢迎大家相互探讨、研究，共同进步，extjs4 开发笔记将在第六章更名为 extjs4 MVC 开发笔记，请大家持续关注，谢谢！！

Extjs4 开发笔记(六)——数据的增删改

2011-12-15　来源：mhzg　作者：mhzg　(共 0 条评论)

上一章，我们介绍了动态 Grid 的显示，其地址是：Extjs4 开发笔记(五)——动态 grid，在上一章，我们只做了数据的显示，并没有添加、删除和修改功能，本章内容，介绍如何进行添加、删除和修改。

上一章,我们介绍了动态 Grid 的显示,其地址是:Extjs4 开发笔记(五)——动态 grid,在上一章，我们只做了数据的显示，并没有添加、删除和修改功能，本章内容，介绍如何进行添加、删除和修改。

一般的项目中，Gird 功能不是很复杂的话，都会使用 window 来实现数据的添加、修改功能，而本实例中，由于使用了动态 grid 功能，这样就使得再使用动态 window 来实现数据的添加和修改就会变的非常困难。

幸好，Grid 有 RowEditing，下面，我们就用 RowEditing 来实现数据的添加和修改功能。在开始之前，我们先回顾下上一章的一些重点内容：

1、给 Menu 增加了 field 属性，使得我们在数据库中的一些字段可以被当做是 Menu 的节点集合中的对象来调用。

2、给菜单表增加了两个字段，分别是 store 和 columns。在 app/store/文件夹下，我们新建了 bastore.js 文件,那么再数据库中对应的字段中,填写内容为 bastore,在 columns 字段中，我们添加了内容为{text:'序号',dataIndex:'ID'},{text:'公司名称','dataIndex:'kehu_name'}的数据。

最后，我们修改了 Menu.js 文件，使得 Grid 可以显示数据。

现在，我们修改 columns 中的数据为：

```
{text:'序号',dataIndex:'ID',width:50},{text:'公司名称','dataIndex:'kehu_name',width:260,editor:{allowBlank: false}},{text:'备案号','dataIndex:'beianhao',width:140,editor:{allowBlank: false}},{text:'备案密码','dataIndex:'beianpass',width:100,editor:{allowBlank: false}},{text:'备案邮箱','dataIndex:'beianemail',width:160,editor:{allowBlank: false}},{text:'备案邮箱密码','dataIndex:'emailpass',width:120,editor:{allowBlank: false}},{text:'备案账号','dataIndex:'beianzh',width:160,editor:{allowBlank: false}},{text:'账号密码','dataIndex:'beianzhpa',width:120,editor:{allowBlank: false}}
```

在这些数据中，将所有字段都列了出来，并且制定了 editor 中 allowBlank 的数据位 flase,就是说，这些内容必须填写。

接下来，我们需要修改 app/controller 下的 menu.js 文件，增加一些功能，使其可以添加、删除信息。修改内容如下：

```
if (record.get('leaf')) {
                var panel = Ext.getCmp(record.get('id'));
                if(!panel){
                    Ext.require(['Ext.grid.*']);
                    var rowEditing = Ext.create('Ext.grid.plugin.RowEditing',{
                        clicksToMoveEditor: 1,
                        autoCancel: true
                    });
```

```
                    panel = Ext.create("Ext.grid.Panel",{
                        store:record.get('stores'),
                        columns:record.get('columns'),
                        errorSummary:false,
                        title: record.get('text'),
                        id:record.get('text')+record.get('id'),
                        columnLines: true,
                        bodyPadding:0,
                        closable: true,
                        bbar: Ext.create('Ext.PagingToolbar', {
                            store: record.get('stores'),
                            displayInfo: true,
                            displayMsg: '显示 {0} - {1} 条，共计 {2} 条',
                            emptyMsg: "没有数据"
                        }),
                        dockedItems: [{
                            xtype: 'toolbar',
                            items: [{
                                text: '增加信息',
                                iconCls: 'icon-add',
                                handler: function(){
                                    rowEditing.cancelEdit();
                                    var panelStore = this.up("panel").getStore();
                                    var panelModel = this.up("panel").getSelectionModel(
);
                                    panelStore.insert(0,panelModel);
                                    rowEditing.startEdit(0, 0);
                                }
                            }, '-', {
                                itemId: 'delete',
                                text: '删除信息',
                                iconCls: 'icon-delete',
                                disabled: true,
                                handler: function(){
                                    var selection = panel.getView().getSelectionModel().
getSelection()[0];
                                    var panelStore = this.up("panel").getStore();
                                    if (selection) {
                                        panelStore.remove(selection);
                                    }
                                }
                            }, '-' ,{
                                text:'刷新数据',
                                iconCls:'icon-refresh',
```

```
                                handler:function(){
                                    var panelStore = this.up("panel").getStore();
                                    var currPage = panelStore.currentPage;
                                    panelStore.removeAll();
                                    panelStore.load(currPage);
                                }
                            }]
                        }],
                        plugins: [rowEditing],
                        listeners: {
                            'selectionchange': function(view, records) {
                                panel.down('#delete').setDisabled(!records.length);
                            }
                        }
                    })
                    this.openTab(panel,record.get('id'));
                }else{
                    var main = Ext.getCmp("content_panel");
                    main.setActiveTab(panel);
                }
            }
```

代码中，启用了插件 rowEditing，增加了一个 toolbar，上面添加了三个按钮，分别是添加信息，删除信息和刷新数据。添加信息的按钮，我们创建了一个 handler，并且在其中获取了 gird 的 Store 和 Model，并将其插入到 grid 的 store 中，删除信息的按钮中，设想这个按钮在没有选中任何行的时候，是不可用的，所以设置其 disabled 为 true。还设置了 handler，并且通过获取选择的行，将其从 grid 的 store 中删除。并且，我们为 grid 增加了一个监听 selectionchange，这样，我们就可以在选择中一条数据后，让删除信息的按钮变的可用。

目前，当点击添加按钮并添如信息后，只是存在于 grid 的 store 中，并没有将数据更新到数据库中，删除信息也是一样。接下来，需要修改 app/store 下的 bastore.js，使其可以更新到数据中。

bastore.js：

```
Ext.define('SMS.store.bastore', {
    extend: 'Ext.data.Store',
    requires: 'SMS.model.beianlistmodel',
    model: 'SMS.model.beianlistmodel',
    pageSize: 20,
    remoteSort: true,
    autoLoad:true,
    proxy: {
```

```
        type: 'ajax',
        url: '/server/getbeian.asp',
        reader: {
            root: 'items',
            totalProperty  : 'total'
        },
        simpleSortMode: true
    },
    listeners:{
        update:function(store,record){
            var currPage = store.currentPage;
            //alert(record.get("ID"))
            Ext.Ajax.request({
                url:'/server/getbeian.asp?action=save',
                params:{
                    id : record.get("ID"),
                    kehu_name:record.get("kehu_name"),
                    beianhao:record.get("beianhao"),
                    beianpass:record.get("beianpass"),
                    beianemail:record.get("beianemail"),
                    emailpass:record.get("emailpass"),
                    beianzh:record.get("beianzh"),
                    beianzhpa:record.get("beianzhpa"),
                },
                success:function(response){
                    store.removeAll();
                    store.load(currPage);
                }
            });
        },
        remove:function(store,record){
            var currPage = store.currentPage;
            //alert(record.get("ID"))
            Ext.Ajax.request({
                url:'/server/getbeian.asp?action=del',
                params:{
                    id : record.get("ID")
                },
                success:function(response){
                    store.removeAll();
                    store.load(currPage);
                }
            });
        }
```

```
53.     },
54.     sorters: [{
55.         property: 'ID',
56.         direction: 'DESC'
57.     }]
58. });
```

代码中，为 store 增加了两个监听，update 和 remove，并且将数据通过 AJAX 发送到服务端，在服务端进行处理，这里，只使用了 update 和 remove。store 中还有个 add 方法，此方法也是增加一条数据，按照常理来说。这个方法才是增加数据，但是我使用了 add 方法之后，点击添加信息，就会添加一条空数据，索性就使用 update 方法，将 id 值也发送到服务端，在服务端进行处理，服务端处理流程是：接收 id 值，判断 id 值是否为空，如果为空，则新增数据，如果不为空，则更新数据。

至此，一个 grid 的功能全部完成了。而且本项目所有的功能，都是如此，这样，只要再数据库中插入相应的行，在 app/store 和 app/model 中建立相关的 js 就可以了。至于其他功能，就不在此一一例举了。

声明一点，这个开发笔记实施到最后，有一些东西已经脱离了 MVC 的范畴，而且也没想到六章内容就结束了这个项目。从文章一到六，只是起一个抛砖引玉的作用。由于 Extjs4 有很大的变动，所以任何基于 Extjs4.x MVC 的项目，都是摸着石头过河，一点一点积累起来的，并不是说我的这些文章起到了指导性的作用，而是实际开发过程中的一些体会和经验。所有项目，有很多种方法可以实现需求。

最后，希望大家能通过阅读这些开发笔记，都有所提高、进步！下面列出所有开发笔记的链接。

Extjs4 开发笔记（一）——准备工作

Extjs4 开发笔记(二)——框架的搭建

关于 Extjs4 开发笔记（二）的补充说明

Extjs4 开发笔记(三)——菜单的实现

Extjs4 开发笔记(四)——实现登录功能

Extjs4 开发笔记(五)——动态 grid

源码下载：http://www.mhzg.net/a/201112/2011121517480473.html

Extjs4 MVC 开发笔记源码

2011-12-15　来源：mhzg　作者：mhzg　（共 0 条评论）

Extjs4 MVC 开发笔记源码，此源码已经将命名空间修改为 Soyi，由于涉及到一些公司信息，将所有数据库中的数据清除，由于时间仓促。只有备案管理的 grid 是可以使用的，其他的功能，按照备案管理的文件照猫画虎即可实现。

Extjs4 MVC 开发笔记源码，此源码已经将命名空间修改为 Soyi,由于涉及到一些公司信息，将所有数据库中的数据清除，由于时间仓促。只有备案管理的 grid 是可以使用的，其他的功能，按照备案管理的文件照猫画虎即可实现。

下载地址：本地下载

CSDN 下载：http://download.csdn.net/detail/sd2208464/3925491

另附所有开发笔记链接：

Extjs4 开发笔记（一）——准备工作

Extjs4 开发笔记(二)——框架的搭建

关于 Extjs4 开发笔记（二）的补充说明

Extjs4 开发笔记(三)——菜单的实现

Extjs4 开发笔记(四)——实现登录功能

Extjs4 开发笔记(五)——动态 grid

Extjs4 开发笔记(六)——数据的增删改