一直想写一些 extjs4 MVC 的东西，但由于自己的英文水平足够媲美小学 5 年纪的学生，所以在找了一些比我强一点的网友+机器翻译，总结出了以下这篇文章。但个人强烈建议去看英文原版（点击进入）。看完本文后，如有任何错误，欢迎来信或者留言指正(QQ:301109552)。

那么，我们开始吧！

对于 extjs 来说，大客户端程序一直很难写，当你为大客户端程序添加更多的功能和项目的时候，项目的体积往往迅速增长。这样的大客户端程序很难组织和维持 ，所以，extjs4 配备了一个新的应用程序体系结构，它能结构化你的代码，那就是 extjs4 MVC。

extjs4 MVC 有别于其他 MVC 架构，extjs 有他自己定义：

1、Model 是一个 Field 以及他的 Data 的集合，Modes 知道如何通过 Stores 来表示数据，以能用于网格和其他组件。模型的工作很像 extjs3 的记录集（Record class），通常通过数据加载器（Stores）渲染至网格（grid）和其他组件上边。

2、View：用以装载任何类型的组件—grid、tree 和 panel 等等。

3、Controller—用来放使得 app 工作的代码，例如 render views , instantiating Models 或者其他应用逻辑。

本篇文章，我们将创建一个非常简单的应用程序，即用户数据管理，最后，你就会知道如何利用 extjs4 MVC 去创建简单应用程序。extjs4 MVC 应用程序架构提供应用程序的结构性和一致性。这样的模式带来了一些重要的好处：

1、 每个应用程序的工作方式相同，我们只需要学习一次。

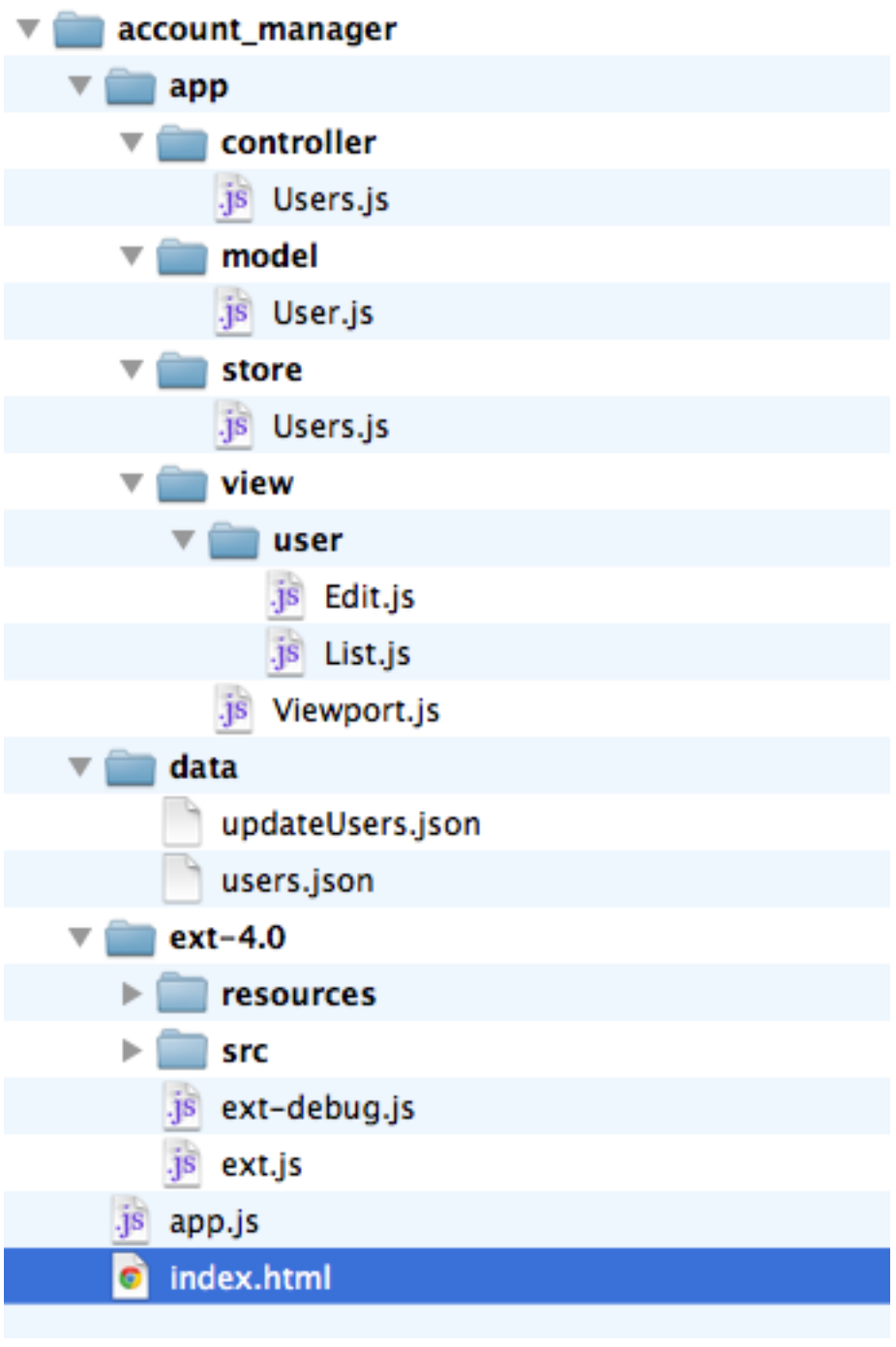
2、 应用程序之间的代码共享很容易，应为他们所有的工作方式都相同

3、 你可以使用 extjs 提供的构建工具创建你应用程序的优化版本。

既然是介绍 Extjs4 MVC，那么，我们开始创建这个应用。

一、文件结构（File Structure）：

Extjs4 MVC 的应用程序和别的应用程序一样都遵循一个统一的目录结构。在 MVC 布局中，所有的类放在应用程序文件夹，它的子文件夹中包含您的命名空间，模型，视图，控制器和存储器。下面来通过简单的例子来看下怎样应用。



在这个例子中，我们将整个应用程序放到一个名为”account_manager”的文件夹下，”account_manager”文件夹中的结构如上图。现在编辑 index.html，内容如下：

```
<html>
<head>
    <title>Account Manager</title>

    <link rel="stylesheet" type="text/css" href="ext-4.0/resources/css/ext-all.css">

    <script type="text/javascript" src="ext-4.0/ext-debug.js"></script>

    <script type="text/javascript" src="app.js"></script>
</head>
<body></body>
</html>
```

二、创建 app.js 文件。

所有 Extjs4 MVC 应用从 Ext.application 的一个实例开始，应用程序中应包含全局设置、以及应用程序所使用的模型(model)，视图（view）和控 制器（controllers），一个应用程序还应该包含一个发射功能(launch function)。

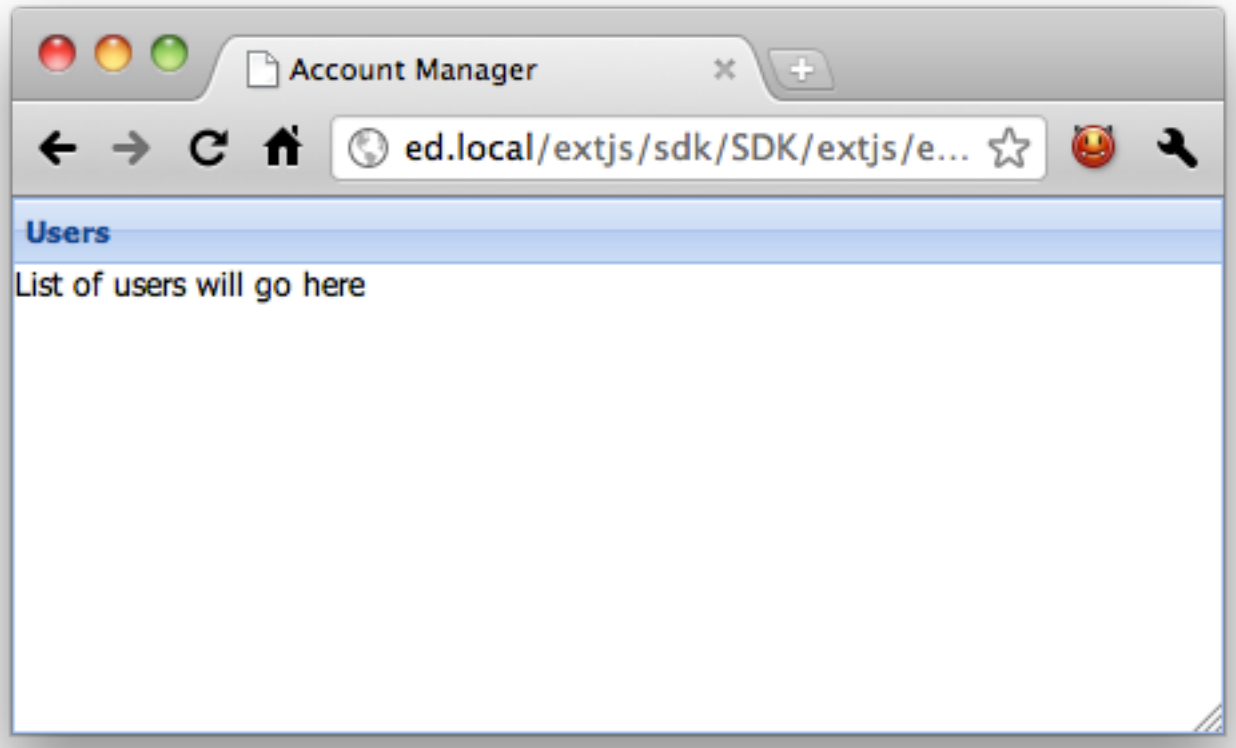
现在来创建一个简单的账户管理应用。首先，需要选择一个命名空间（所有 extjs4 应用应该使用一个单一的全局变来来作为命名空间）。暂时，使用”AM”来作为命名空间。

```
Ext.application({
    name: 'AM',

    appFolder: 'app',

    launch: function() {
        Ext.create('Ext.container.Viewport', {
            layout: 'fit',
            items: [
                {
                    xtype: 'panel',
                    title: 'Users',
                    html : 'List of users will go here'
                }
            ]
        });
    }
});
```

以上代码，做了如下几件事。首先，调用 Ext.application 创建一个应用程序类的实 例，设置了一个”AM”的命名空间，他将作为整个应用的全局变量，也将作为 Ext.Loader 的命名空间，然后通过 appFolder 来指定配置选项设 置相应的路径。最后，创建了一个简单的发射功能，这里仅仅创建了一个 Viewport，其中包含一个 panel，使其充满整个窗口。



三、定义一个控制器

控制器是整个应用程序的关键，他负责监听事件，并对某些时间做出相应的动作。现在我们创建一个控制器，将其命名为 Users.js，其路径是 app/controller/Users.js。然后，我们为 Users.js 添加如下代码：

```
1.  Ext.define('AM.controller.Users', {
2.      extend: 'Ext.app.Controller',
3.
4.      init: function() {
5.          console.log('Initialized Users! This happens before the Application launch function is called');
6.      }
7.  });
```

完成之后，我们将创建好的控制器添加到程序配置文件：app.js 中：

```
1.  Ext.application({
2.      ...
3.
4.      controllers: [
5.          'Users'
6.      ],
7.
8.      ...
9.  });
```

当我们访问 index.html 时，用户控制器(Users.js)自动加载，因为我们在上面的 app.js 中的定义中指定了。Init 函数一个最 棒的地方是控制器与视图的交互，这里的交互是指控制功能，因为他很容易就可以监听到视图类的事件处理函数，并采取相应的措施，并及时渲染相关信息到面板上 来。

编写 Users.js：

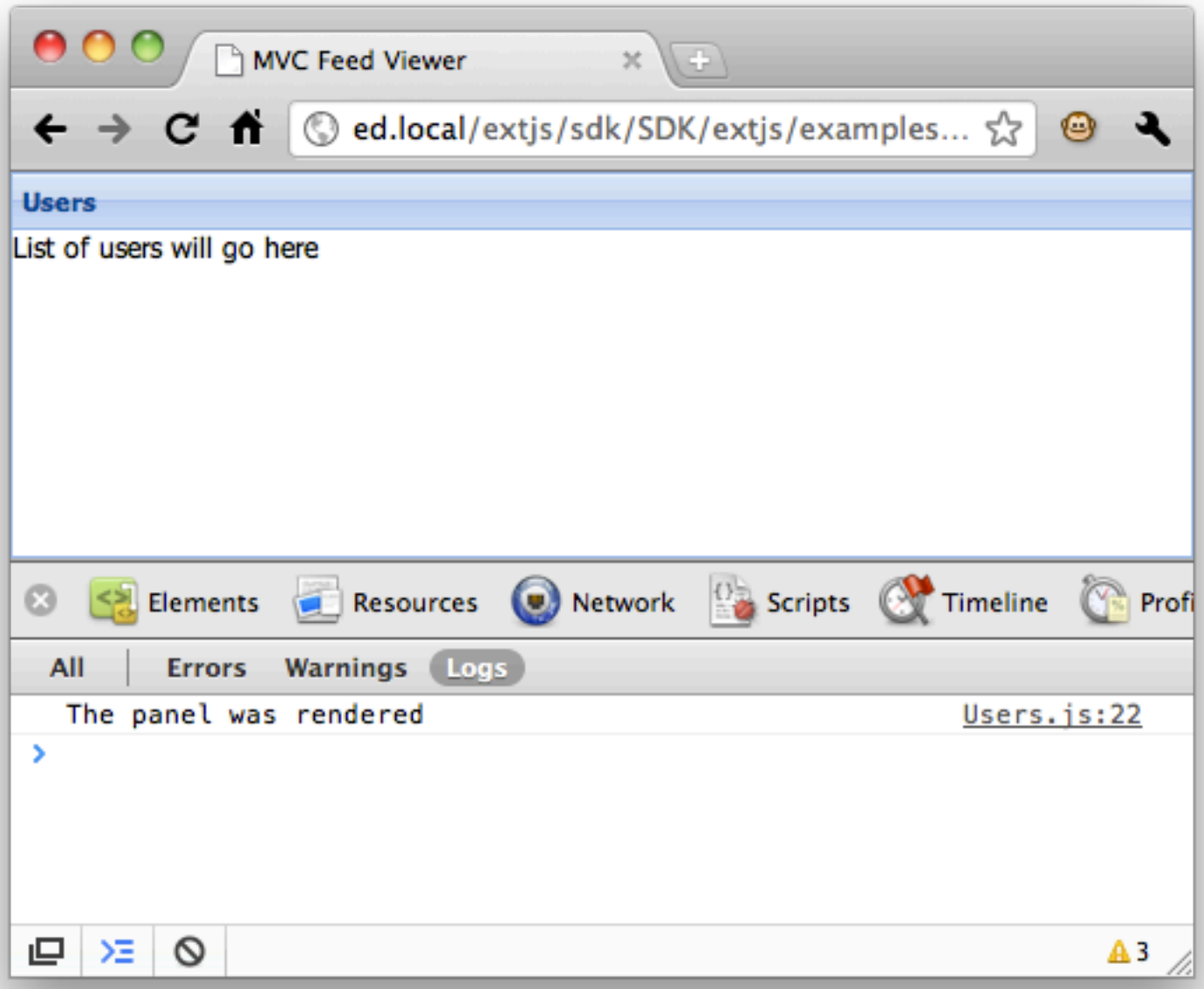
```
1.  Ext.define('AM.controller.Users', {
2.      extend: 'Ext.app.Controller',
3.
4.      init: function() {
5.          this.control({
```

```
            'viewport > panel': {
                render: this.onPanelRendered
            }
        });
    },

    onPanelRendered: function() {
        console.log('The panel was rendered');
    }
});
```

在 Users.js 中，init 函数使用 this.control 来负责监听，由于使用了新的 ComponentQuery 引擎，所以可以快速方便的找到页面上组件的引用(viewport > panel)，这有些类似 CSS 选择器，通过匹配，快速的找到相匹配的组件。

在上面的 init 函数中，我们使用 viewport > panel，来找到 app.js 中 Viewport 下的 panel 组件，然后，我们提供了一个对象的处理函数(this. onPanelRendered，注意，这里的对象是 this，他的处理函数是 onPanelRendered)。整个效果是，只要符合触发 render 事件的任何组件，那么 onPanelRendered 函数将被调用。当运行我们的应用程序，我们将看到以下内容。



四、定义一个视图

到目前为止，应用程序只有几行，也只有两个文件,app.js 和 app/controller /Users.js。现在我们来增加一个 grid,来显示整个系统中的所有用户。作为视图组件的一个子类，我们创建一个新的文件，并把他放到 app /view/user 目录下。命名为 List.js。整个路径是这样的。app/view/user/List.js，下面，我们写 List.js 的代 码：

```
Ext.define('AM.view.user.List' ,{
    extend: 'Ext.grid.Panel',
    alias : 'widget.userlist',
    title : 'All Users',
    initComponent: function() {
```

```
6.          this.store = {
7.              fields: ['name', 'email'],
8.              data  : [
9.                  {name: 'Ed',    email: 'ed@sencha.com'},
10.                 {name: 'Tommy', email: 'tommy@sencha.com'}
11.             ]
12.         };
13.         this.columns = [
14.             {header: 'Name',  dataIndex: 'name',  flex: 1},
15.             {header: 'Email', dataIndex: 'email', flex: 1}
16.         ];
17.          this.callParent(arguments);
18.     }});
```

我们创建好的这个类，只是一个非常普通的类，并没有任何意义，为了能让我们更好的使用这个定义好的类，所以我们使用 alias 来定义一个别名，这个时候，我们的类可以使用 Ext.create()和 Ext.widget()创建，在其他组件的子组件中，也可以使用 xtype 来创建。

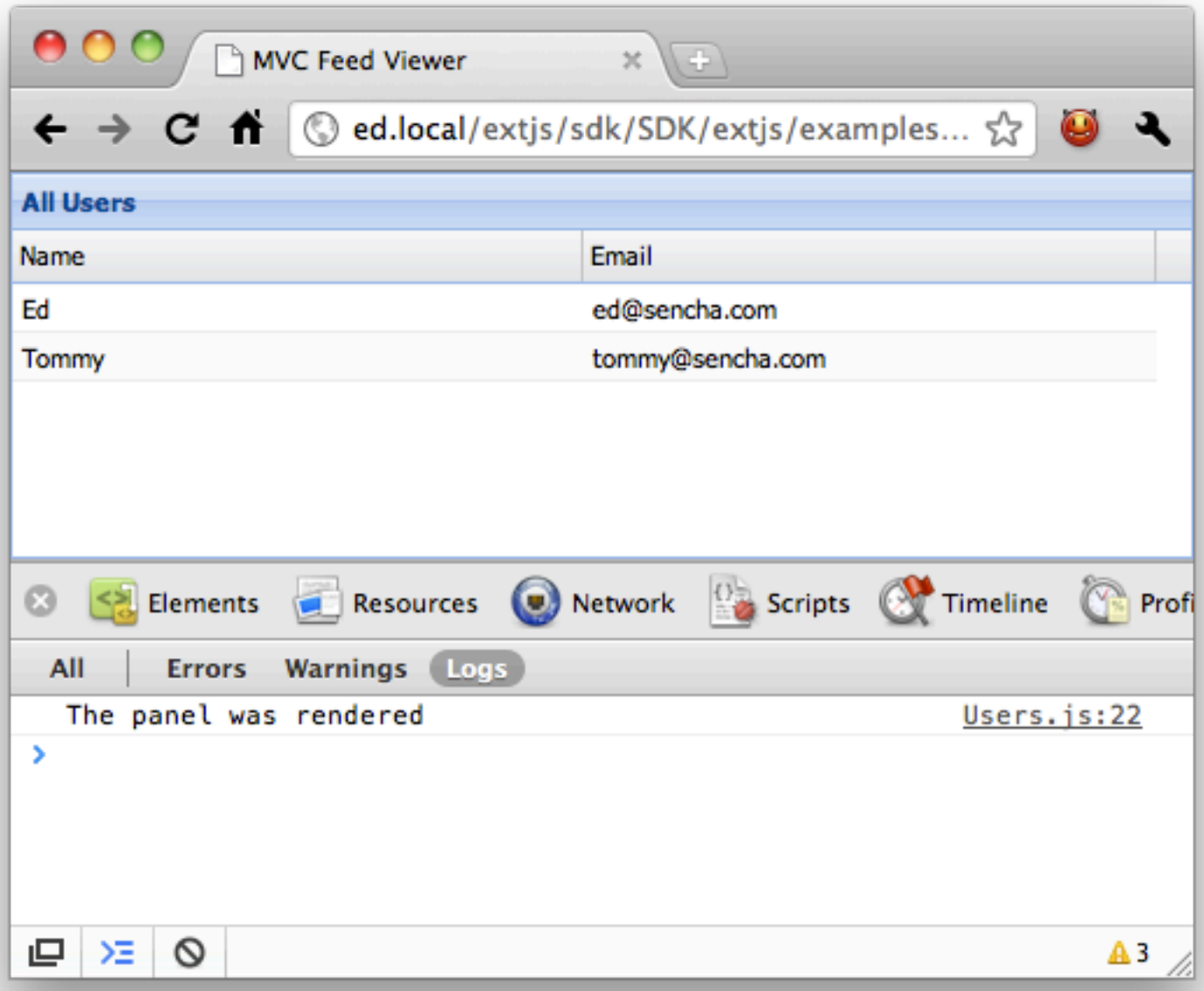
```
1.  Ext.define('AM.controller.Users', {
2.      extend: 'Ext.app.Controller',
3.      views: [
4.          'user.List'
5.      ],
6.      init: ...    onPanelRendered: ...
7.  });
```

修改 Users.js,增加 views 属性，修改 app.js 中的 launch 方法，将 List 渲染到 Viewport。

```
1.  Ext.application({
2.      ...    launch: function() {
3.          Ext.create('Ext.container.Viewport', {
4.              layout: 'fit',
5.              items: {
6.                  xtype: 'userlist'
7.              }
8.          });
9.  }});
```

看到这里，也许会有人开始抓狂了，这个 user.List 到底是怎么来的，为什么要这样写？如果你开始感到疑惑，那么不妨看看 Ext.Loader 是如何工作的（传送门）， 在看过 Ext.Loader 之后，你就会明白了，User.List 就是 app/view/user 下的 List.js 文件。为什么 Ext 要从 view 下 找呢？因为我们在控制器中定

了 views: ['user.List']。这就是 Extjs 动态加载的强大之处，具体 Ext.Loader，请看本站的其他文章，你就会明白了。当我们刷新页面。



未完待续...