本文全面介绍了使用ExtJS6开发网页应用，涉及知识面广，是很好ExtJS6入门材料。

作者：[dawnzhou@126.com](mailto:dawnzhou@126.com)

# 开发准备

**ExtJS 6.0 GPL正式版下载地址**

<https://www.sencha.com/legal/gpl/>

**ExtJS 6.0离线文档下载地址**

经典版

<http://docs.sencha.com/downloads/extjs-docs-6.0.0-classic.zip>

现代版

<http://docs.sencha.com/downloads/extjs-docs-6.0.0-modern.zip>

**Sencha Tool下载地址**

<http://cdn.sencha.com/cmd/6.0.0.202/no-jre/SenchaCmd-6.0.0.202-windows-no_jre.zip>

（假设你的系统里已经安装了JRE，否则请下载带JRE版本的Sencha Tool）

**安装Sencha Tool**

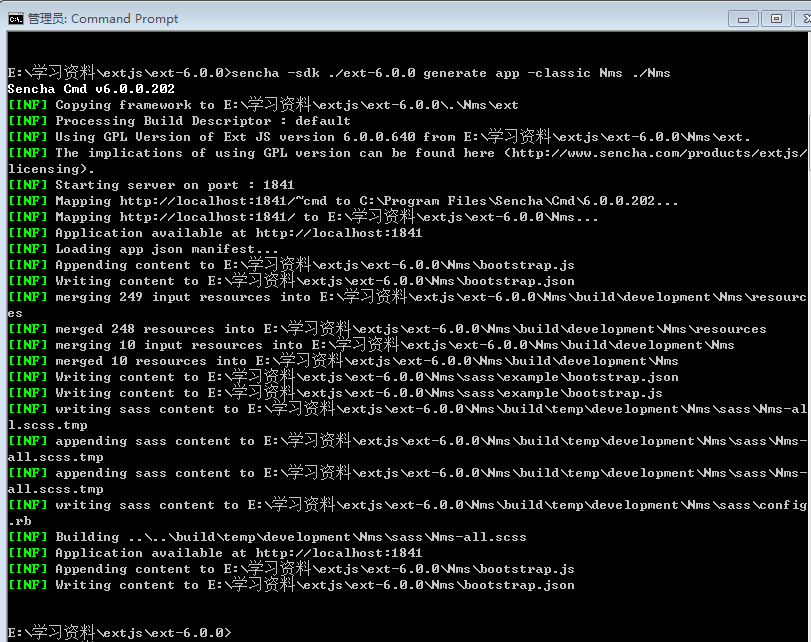
默认安装即可。

解压离线帮助文档extjs-docs-6.0.0-classic.zip

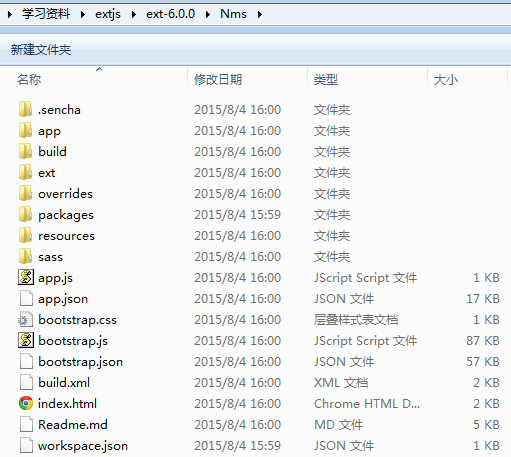
解压ext-6.0.0-gpl.zip

# 生成程序框架

>sencha -sdk ./ext-6.0.0 generate app -classic Nms ./Nms



进入到Nms目录，看到生成的工程

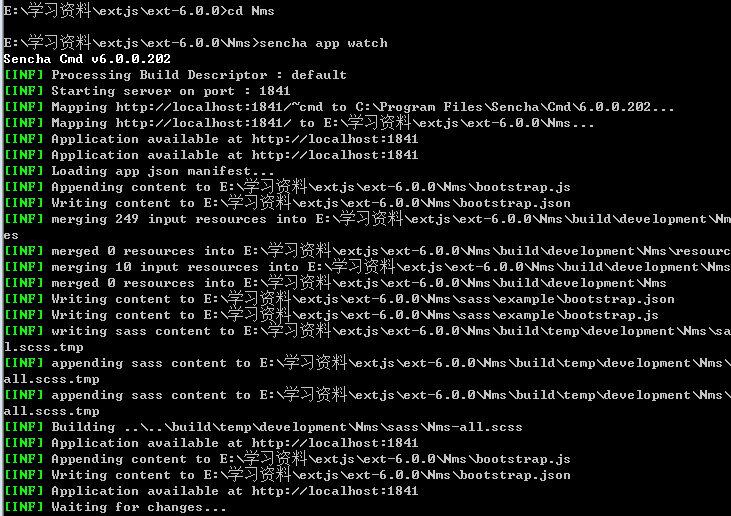


cd Nms

进入到该目录

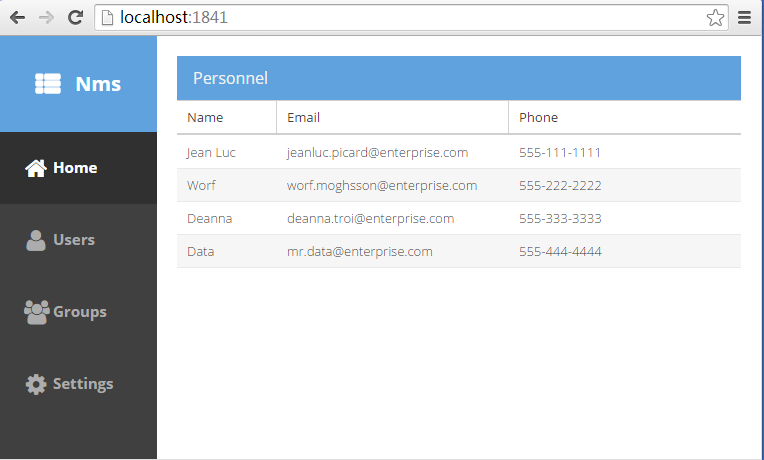
执行

sencha app watch



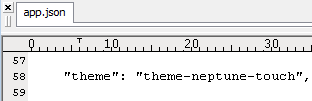
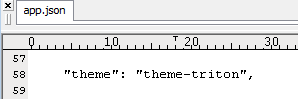
浏览器输入

<http://localhost:1841>



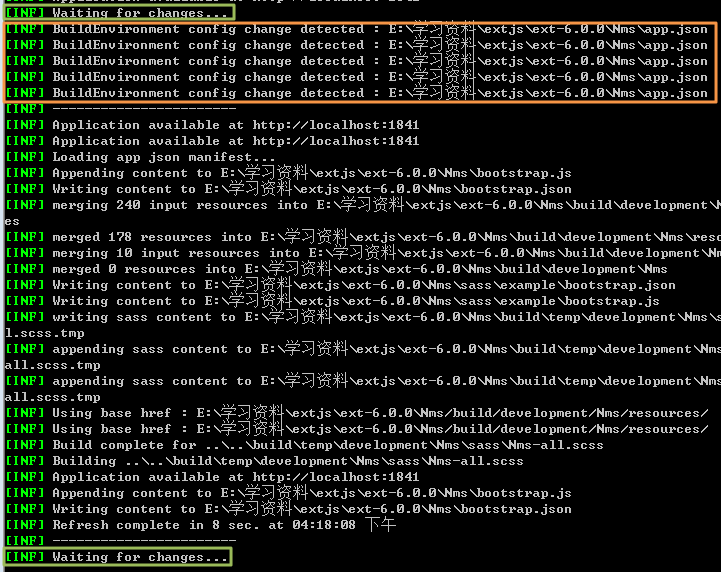
# 定制样式

## 修改主题



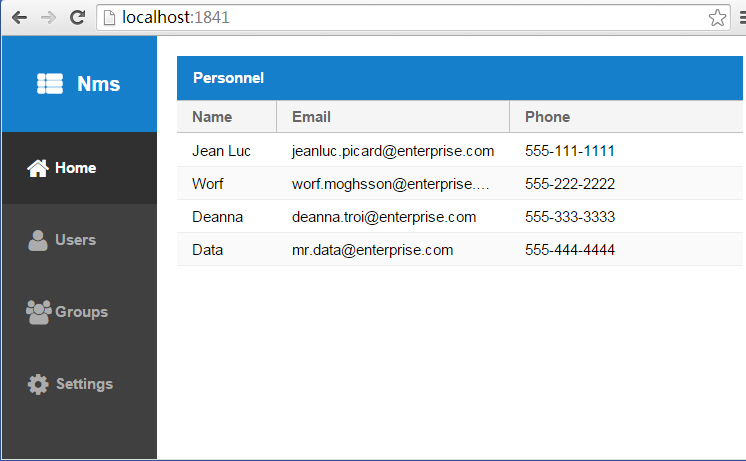
打开app.json，定位到"theme": "theme-triton"改为"theme": "**theme-neptune-touch**"并保存。

观察DOS窗口的变化。



可以看到，橙色标识的部分表示sencha cmd 工具监测到了文件被修改app.json被修改了，工具实时编译，完成后再次输出**Waiting for changes…**，继续监听文件是否有更改。

刷新浏览器，看到主题变化已经生效。

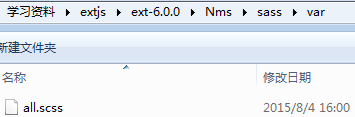


## 修改全局主题变量

为了能够实时看到修改后的效果，我们把浏览器地址增加一个参数

http://localhost:1841/**?platformTags=fashion:true**

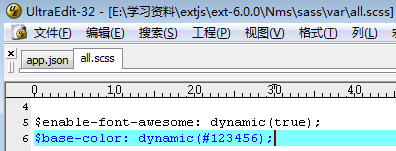
打开all.scss文件



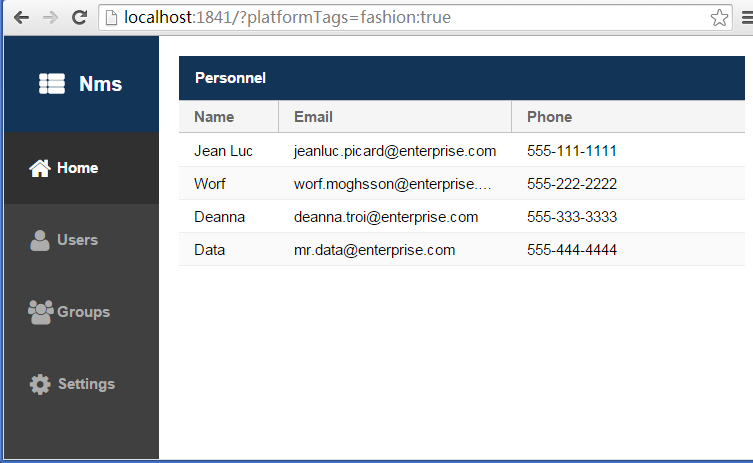
增加一行

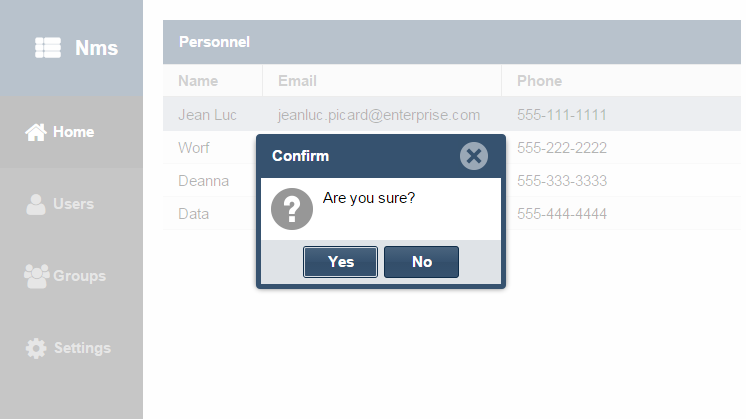
$base-color: dynamic (#123456);

如下图所示：



回到浏览器页面，效果如下：





可以看到，标题和按钮的颜色为我们设置的颜色值。

查看哪些全局的主题变量可以修改，需要参考Ext Api Document

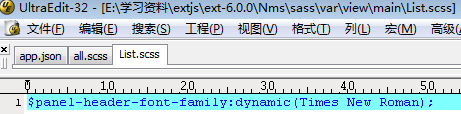


## 配置组件的变量

新建一个文件

Nms\sass\var\view\main\**List.scss**，该文件对应于代码

Nms\app\view\main\List.js，意味着我们将要修改视图Nms.view.main.List 的样式。



设置panel的背景

$panel-body-background-color : dynamic(#4682B4);

设置表头背景

$grid-header-background-color : dynamic(#F0FFF0);

设置行间隔颜色为透明

$grid-row-cell-border-color : dynamic(transparent);

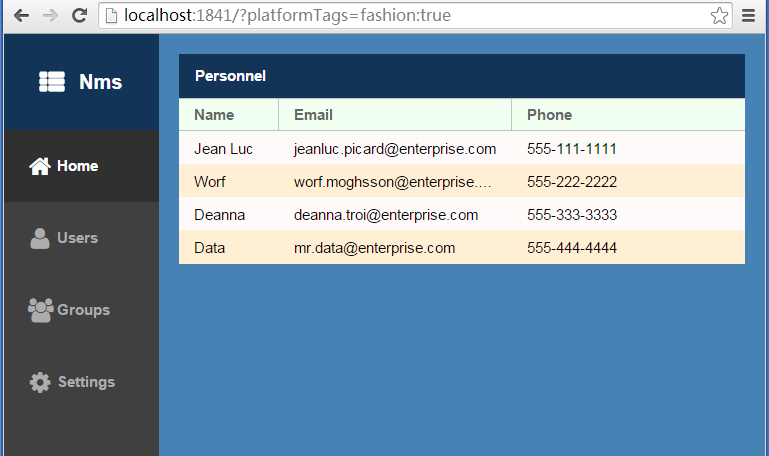
设置列表行背景

$grid-row-cell-background-color : dynamic(#FFFAFA);

设置列表行交替背景

$grid-row-cell-alt-background-color : dynamic(#FFEFD5);

最终效果如下



## 通过JS覆盖(override)主题

创建overrides\panel\Panel.js

加入如下代码

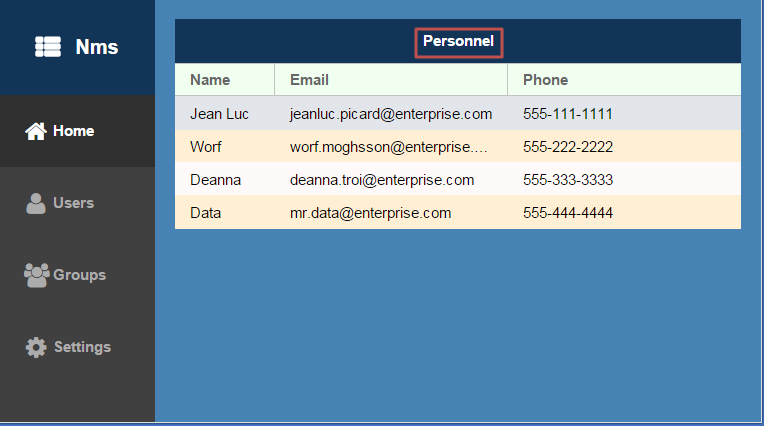
Ext.define('Nms.panel.Panel', {

override : 'Ext.panel.Panel',

titleAlign : 'center'

});

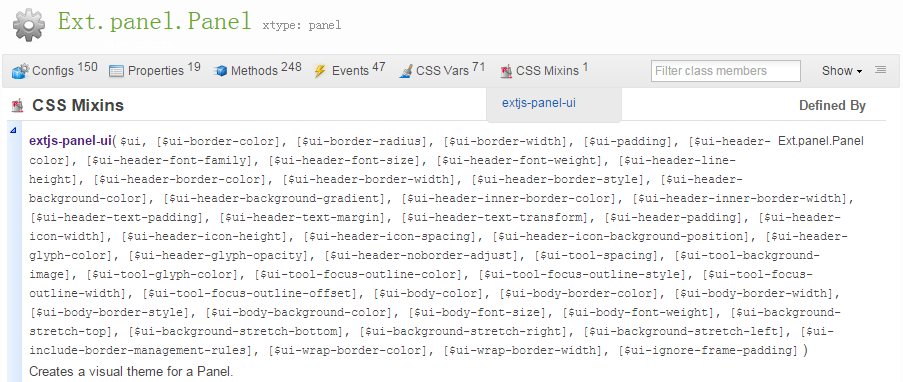
刷新页面，观察页面变化。



## 创建用户组件UI

每一个Ext Framework下的组件都拥有自己的UI设置参数，缺省下都是default，这个属性可以给予单独实体组件拥有自己的UI配参，可以让你在同一类型组件下给予不同的样式。

Ext为各种组件提供了很多SASS mixins混入，你可以用这些mixins来生成新的UI，这可以从API中找到它们。如



创建sass\src\panel\Panel.scss，增加

@include **extjs-panel-ui** (

$ui:'**highlight-framed**',

$ui-header-background-color:#FF6347,

$ui-border-color:#FF6347,

$ui-header-border-color:#FF6347,

$ui-body-border-color:#FF6347,

$ui-header-color:white

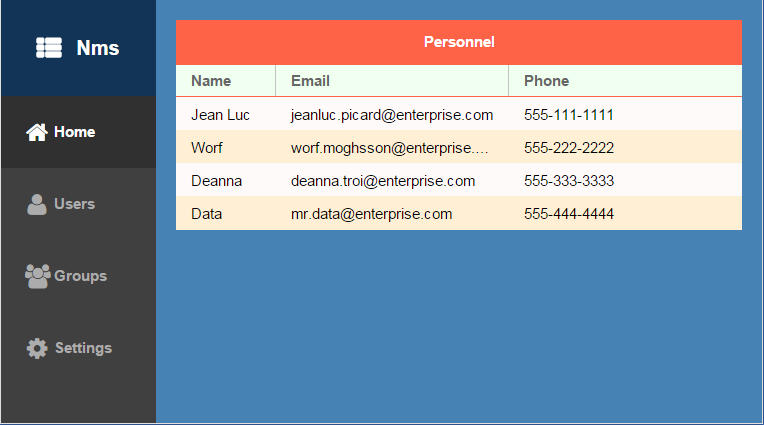
);

修改Nms\app\view\main\List.js代码，增加

**ui : 'highlight',**

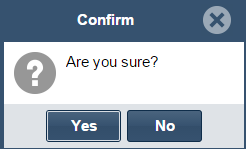
**frame : 'true'**,

两个配置，刷新页面，观察页面变化

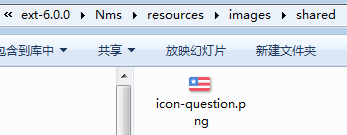


## 定制Image

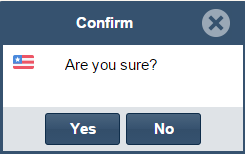
双击列表的某条记录，弹出如下对话框，



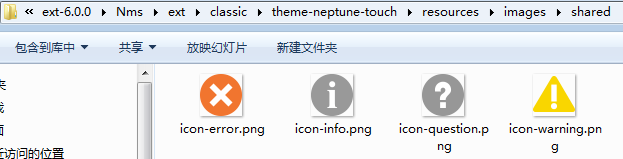
建立一个png文件



当再双击列表记录时，弹出的对话框图标变为了我们刚刚设置的图标。



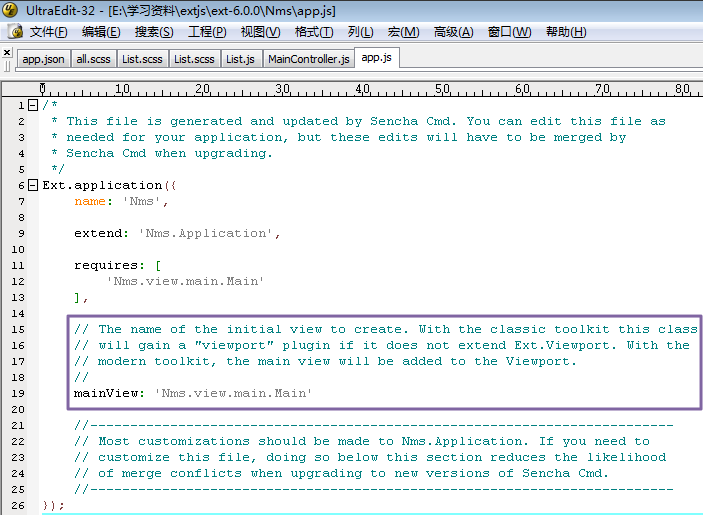
我们使用的主题是theme-neptune-touch，对比一下目录和文件名，就明白我们为什么要在Nms\resources\images\shared\下面创建同名icon-question.png图片了。



# 开发详解

## 理解框架代码

### 程序入口app.js



mainView配置项是用来创建的初始化视图，也就是说程序在初始化时会创建类为“Nms.view.main.Main”的实例。

注意，匿名类继承的是**Nms.Application**类。

### Nms.Application



stores存放的是全局或者共享的store，如果定义的store不在这里配置，则程序不会加载该store。

### Nms.view.main.Main

extend: 'Ext.tab.Panel', 继承自Ext.tab.Panel

controller: 'main', 使用名字为main的控制器，

Nms.view.main.MainController里有配置项alias: 'controller.**main**'

viewModel: 'main', 使用名字为main的视图模型

Nms.view.main.MainModel里有配置项alias: 'viewmodel.**main**'

ui: 'navigation', 见第二章介绍

header: {

layout: {

align: 'stretchmax'

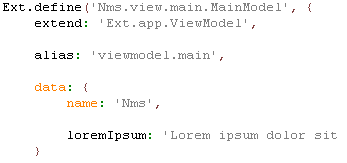
},

title: {

**bind**: {

text: '{**name**}' 绑定的是对应的viewModel里配置项data.name，见

Nms.view.main.MainModel



viewModel的好处是，当里面的数据发生变化时，bind到

viewModel的视图能立即更新

},

flex: 0

},

iconCls: 'fa-th-list'

},

responsiveConfig: {

tall: {

headerPosition: 'top' 当浏览器视图高度大于宽度时，tab显示在视图顶部

},

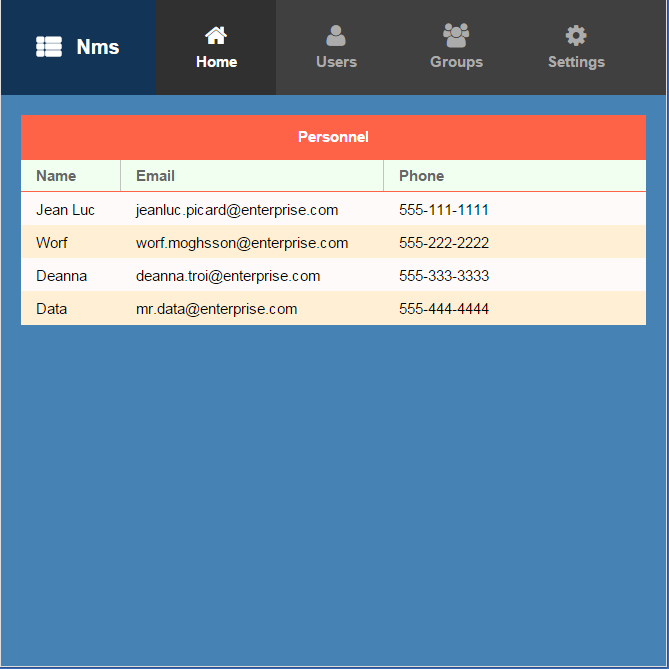
wide: {

headerPosition: 'left' 当宽度大于高度时，tab显示在视图左边

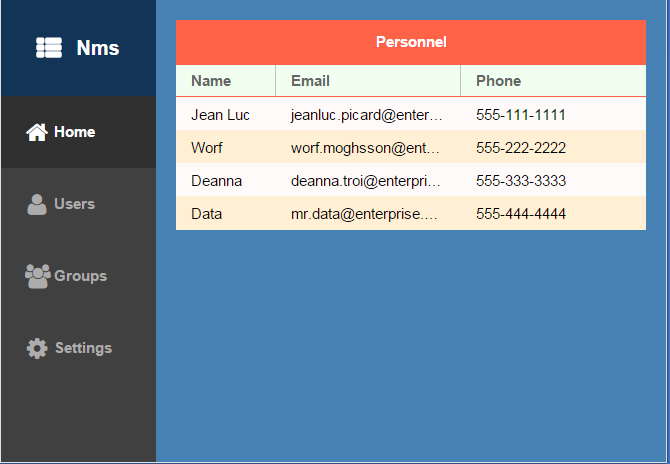
}

},

高度大于宽度效果



宽度大于高度效果



items: [{

title: 'Home',

iconCls: 'fa-home',

// The following grid shares a store with the classic version's grid as well!

items: [{

xtype: 'mainlist' Nms.view.main.List类的 **xtype: 'mainlist'**，是对类名的简写

}]

}, {

title: 'Users',

iconCls: 'fa-user',

bind: {

html: '{loremIpsum}' 见dataModel的data. **loremIpsum**

}

}, {

title: 'Groups',

iconCls: 'fa-users',

bind: {

html: '{loremIpsum}'

}

}, {

title: 'Settings',

iconCls: 'fa-cog',

bind: {

html: '{loremIpsum}'

}

}]

可以看出，配有4个Tab项。

### Nms.view.main.List

Ext.define('Nms.view.main.List', {

extend: 'Ext.grid.Panel',

xtype: 'mainlist',

ui : 'highlight',

frame : 'true',

requires: [

'Nms.store.Personnel'

],

title: 'Personnel',

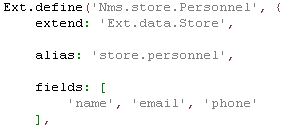
store: {

type: 'personnel' 使用的Nms.store.Personnel，其配置有alias: 'store.**personnel**',

},

columns: [ 有三列，每列要显示的值通过dataIndex指示

{ text: 'Name', dataIndex: 'name' }, dataIndex的配置值对应于

Nms.store.Personnel的fields里的项

{ text: 'Email', dataIndex: 'email', flex: 1 },

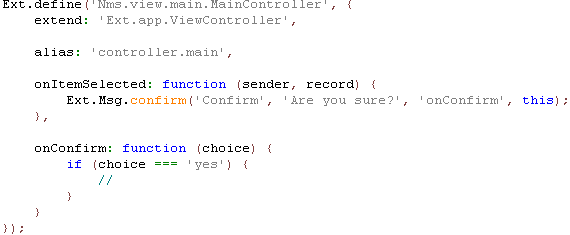
{ text: 'Phone', dataIndex: 'phone', flex: 1 }

],

listeners: {

select: 'onItemSelected' 注册事件处理函数，当select事件发生时，

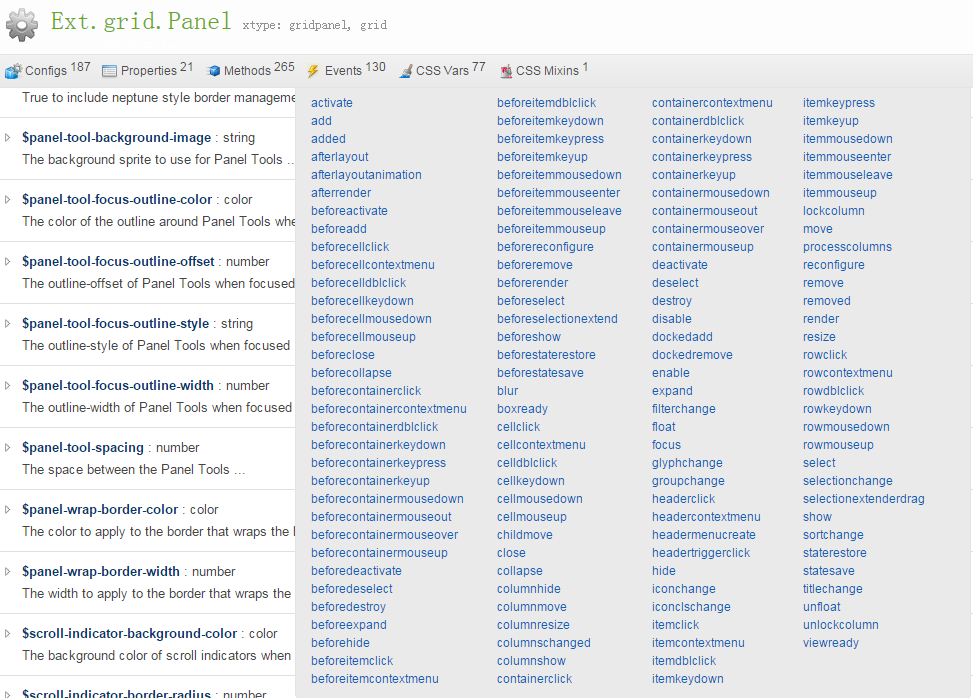
调用controller的'**onItemSelected**'函数



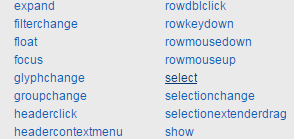
}

});

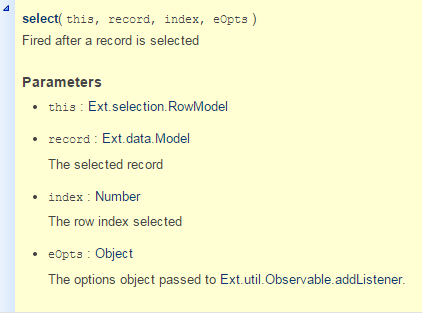
事件及其处理函数的参数，参考帮助文档。如



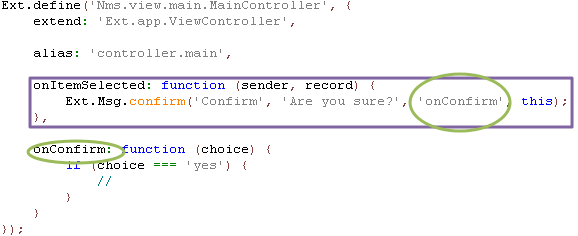
点击select



跳转到select事件处理的函数原型



### Nms.view.main.MainController



onItemSelected对应于Nms.view.main.List注册的回调处理函数。

### Nms.view.main.MainModel

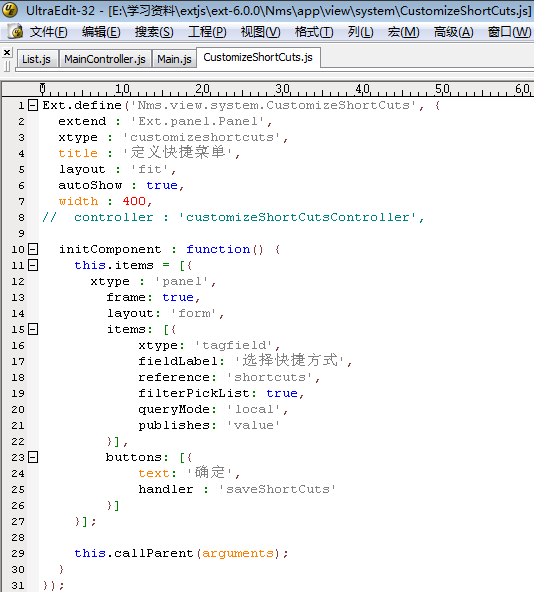
略。

## 开发实践

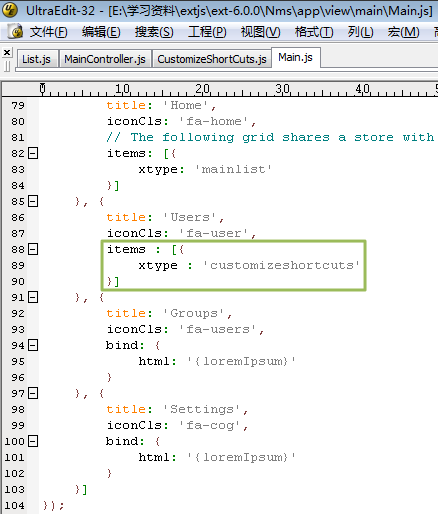
这一节，我们在原有程序上进行扩充，实现一个简单的页面。该页面用来定制快捷菜单。

### 创建页面视图Nms.view.system.CustomizeShortCuts

创建一个js文件，如下图所示：



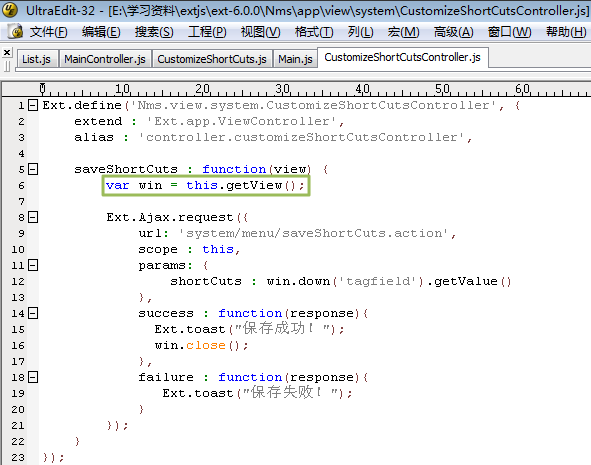
修改主视图，将第二个tab页面改为刚创建的视图。



刷新页面，看一下页面效果。

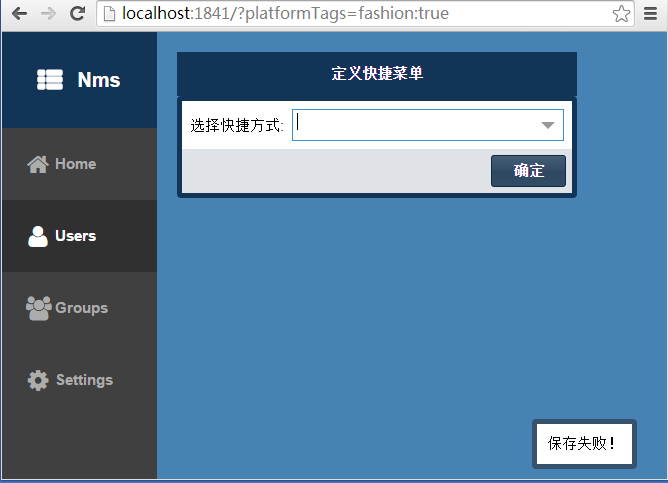


### 创建视图对应的控制器Nms.view.system.CustomizeShortCutsController



该控制器里有视图中【保存】按钮对应的事件处理函数。其中，getVeiw()函数获取的是注册到该控制器的视图，在这里指Nms.view.system.CustomizeShortCuts。我们需要注册该控制器到视图。在Nms.view.system.CustomizeShortCuts里放开注释的那一行。

刷新页面，看一下效果。



当点击确定按钮后，提示保存失败。

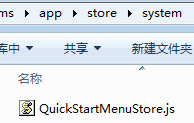
失败的原因在于，程序需要向url: 'system/menu/saveShortCuts.action'提交结果，而url对应的文件不存在，因此，我们在Nms\app\view\system下创建一个文件saveShortCuts.action内容为空。

再次运行，提示保存成功。

### 初始化视图Nms.view.system.CustomizeShortCuts的数据

我们注意到，视图的下拉框内容为空，因为我们没有为tagfield设置store。

### 创建store Nms.store.system.QuickStartMenuStore



内容如下：

Ext.define('Nms.store.system.QuickStartMenuStore', {

extend: 'Ext.data.Store',

model: 'Nms.model.system.MenuModel',

proxy: {

type: 'ajax',

api: {

read: 'system/menu/getMenusOfLeaf.action'

},

reader: {

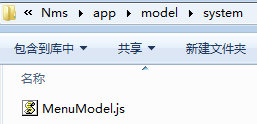
type: 'json'

}

}

});

### 创建model Nms.model.system.MenuModel



内容如下

Ext.define('Nms.model.system.MenuModel', {

extend : 'Ext.data.Model',

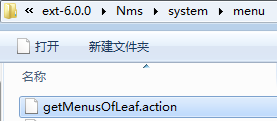
fields : [ 'id', 'text', 'cntType', 'icon', 'root', 'smallIcon', 'pId',

'pName', 'shortCut', 'quickStart', 'size', 'sort', 'mode' ]

});

### 创建模拟数据

我们的store设置了从read: '**system/menu/getMenusOfLeaf.action**'下面获取数据，因此，我们需要创建文件。



文件内容如下。

[ {

"id" :"5c919ee03772a0f50137731a772f9923",

"text" :"全网监视",

"root" :false,

"icon" :"resources/images/menustore/area.png",

"smallIcon" :"resources/images/smallmenustore/area.png",

"cntType" :"globaltopo",

"size" :-1,

"shortCut" :1,

"quickStart" :1,

"mode" :0,

"pid" :null,

"pName" :null,

"sort" :0,

"menuModel" :null,

"pname" :null,

"menu" :null

}, {

"id" :"5c919ee03772a0f50137731a772f9922",

"text" :"拓扑监测",

"root" :false,

"icon" :"resources/images/menustore/config.png",

"smallIcon" :"resources/images/smallmenustore/config.png",

"cntType" :"gistopo",

"size" :-1,

"shortCut" :1,

"quickStart" :1,

"mode" :0,

"pid" :null,

"pName" :null,

"sort" :0,

"menuModel" :null,

"pname" :null,

"menu" :null

}, {

"id" :"5c919ee03772a0f50137731a772f9924",

"text" :"物理拓扑",

"root" :false,

"icon" :"resources/images/menustore/topocfg.png",

"smallIcon" :"resources/images/smallmenustore/topocfg.png",

"cntType" :"topoconfiggraph",

"size" :800,

"shortCut" :1,

"quickStart" :1,

"mode" :0,

"pid" :null,

"pName" :null,

"sort" :0,

"menuModel" :null,

"pname" :null,

"menu" :null

}, {

"id" :"5c919ee03772a0f50137731a772f9925",

"text" :"系统监视",

"root" :false,

"icon" :"resources/images/menustore/faultstatistic.png",

"smallIcon" :"resources/images/smallmenustore/faultstatistic.png",

"cntType" :"systemstatus",

"size" :800,

"shortCut" :1,

"quickStart" :1,

"mode" :0,

"pid" :null,

"pName" :null,

"sort" :0,

"menuModel" :null,

"pname" :null,

"menu" :null

}, {

"id" :"5c919ee03772a0f50137731acaf12301",

"text" :"设备配置",

"root" :false,

"icon" :"resources/images/menustore/cmdterminal.png",

"smallIcon" :"resources/images/smallmenustore/cmdterminal.png",

"cntType" :"devframeconfigview",

"size" :800,

"shortCut" :1,

"quickStart" :1,

"mode" :0,

"pid" :null,

"pName" :null,

"sort" :0,

"menuModel" :null,

"pname" :null,

"menu" :null

}, {

"id" :"5c919ee03772a0f50137731acaf2200",

"text" :"故障统计",

"root" :false,

"icon" :"resources/images/menustore/faulthistroy.png",

"smallIcon" :"resources/images/smallmenustore/faulthistroy.png",

"cntType" :"panel",

"size" :800,

"shortCut" :1,

"quickStart" :1,

"mode" :0,

"pid" :null,

"pName" :null,

"sort" :0,

"menuModel" :null,

"pname" :null,

"menu" :null

}, {

"id" :"5c919ee03772a0f50137731acaf33300",

"text" :"任务管理",

"root" :false,

"icon" :"resources/images/menustore/buildstrategy.png",

"smallIcon" :"resources/images/smallmenustore/buildstrategy.png",

"cntType" :"panel",

"size" :800,

"shortCut" :1,

"quickStart" :1,

"mode" :0,

"pid" :null,

"pName" :null,

"sort" :0,

"menuModel" :null,

"pname" :null,

"menu" :null

}, {

"id" :"5c919ee03772a0f50137731acaf32117",

"text" :"用户管理",

"root" :false,

"icon" :"resources/images/menustore/system.png",

"smallIcon" :"resources/images/smallmenustore/system.png",

"cntType" :"userlist",

"size" :800,

"shortCut" :1,

"quickStart" :1,

"mode" :0,

"pid" :null,

"pName" :null,

"sort" :0,

"menuModel" :null,

"pname" :null,

"menu" :null

}, {

"id" :"5c919ee03772a0f50137731acaf32118",

"text" :"权限组管理",

"root" :false,

"icon" :"resources/images/menustore/usergroup.gif",

"smallIcon" :"resources/images/smallmenustore/usergroup.gif",

"cntType" :"usergroup",

"size" :800,

"shortCut" :1,

"quickStart" :1,

"mode" :0,

"pid" :null,

"pName" :null,

"sort" :0,

"menuModel" :null,

"pname" :null,

"menu" :null

}, {

"id" :"5c919ee03772a0f50137731a772f9992",

"text" :"菜单管理",

"root" :false,

"icon" :"resources/images/menustore/menu.png",

"smallIcon" :"resources/images/smallmenustore/menu.png",

"cntType" :"menulist",

"size" :-1,

"shortCut" :1,

"quickStart" :1,

"mode" :0,

"pid" :null,

"pName" :null,

"sort" :0,

"menuModel" :null,

"pname" :null,

"menu" :null

}, {

"id" :"5c919ee03772a0f50137731a772f9991",

"text" :"数据管理",

"root" :false,

"icon" :"resources/images/menustore/log.png",

"smallIcon" :"resources/images/smallmenustore/log.png",

"cntType" :"sysparam",

"size" :800,

"shortCut" :1,

"quickStart" :1,

"mode" :0,

"pid" :null,

"pName" :null,

"sort" :0,

"menuModel" :null,

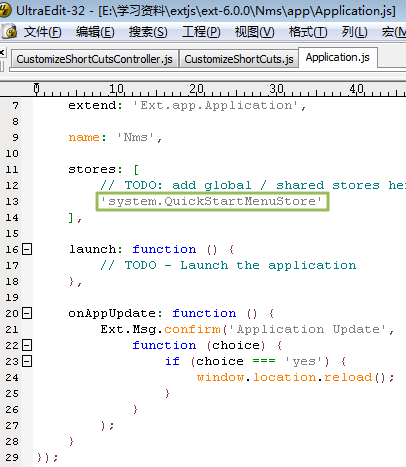
"pname" :null,

"menu" :null

} ]

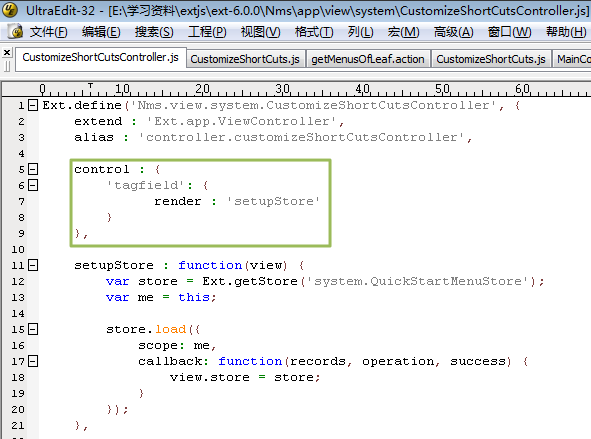
保存文件时，选择utf-8编码

### 配置全局store

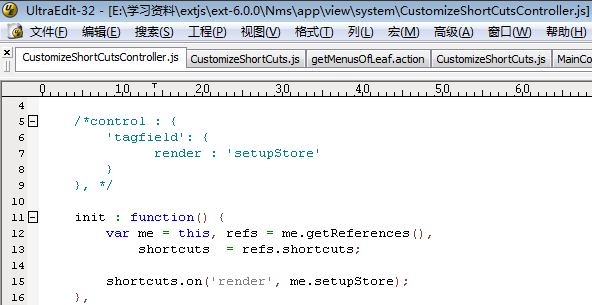


### 初始化视图的store

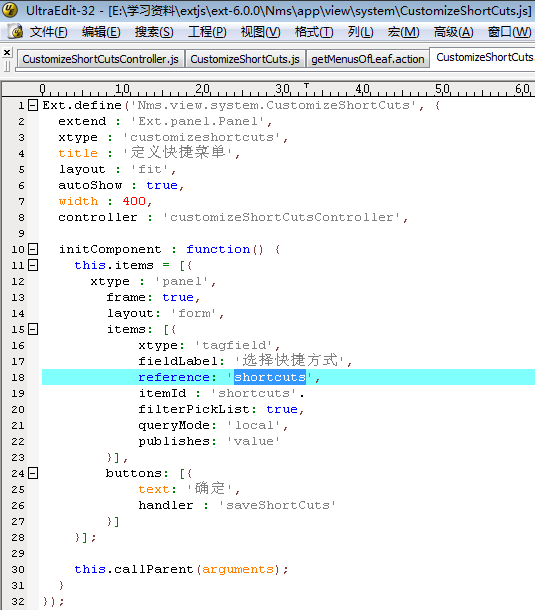
控制器的control配置项是另外一种注册事件响应函数的方式。



在init函数中使用视图的on函数注册事件响应函数



上述例子中，演示了viewController的gerReferences方法的使用方式，可以快速定位指定控件。



组件使用reference配置项进行配置。