## 1、任务指示和通知

两种用于向用户显示消息的API：

* dorado.util.TaskIndicator.showTaskIndicator
* dorado.widget.NotifyTipManager.notify()

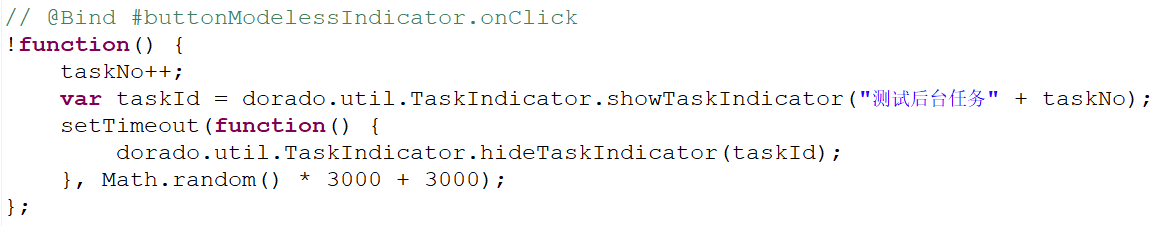
### 非阻塞式的任务指示器



特征：始终显示在页面的右上角。

非阻塞式的指示器在任务执行的过程中还可以继续做页面操作。

用法：



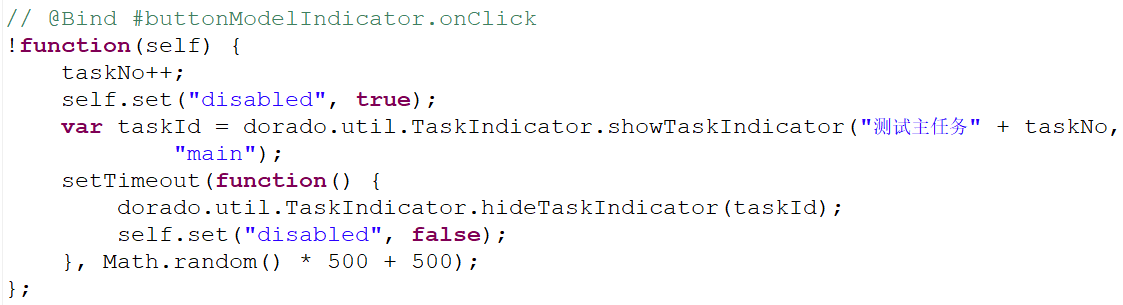
上述代码首先通过showTaskIndicator方法返回的任务指示器的taskId，并调用利用timeout，在3+x秒钟后调用hideTaskIndicator方法隐藏创建的任务指示器。

### 2）阻塞式的任务指示器

在执行阻塞式的任务指示器的时候，指示信息始终在页面中居中显示，另外在其执行时，页面无法操作，直到阻塞式的任务指示器任务执行完毕才可以。



用法：

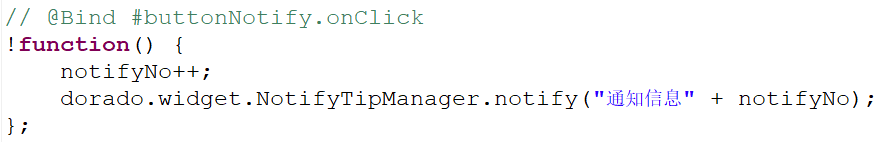


在showTaskIndicator方法中通过main这个参数判断是否需要阻塞效果。其他与非阻塞式任务指示器相似。

### 3）普通的通知消息

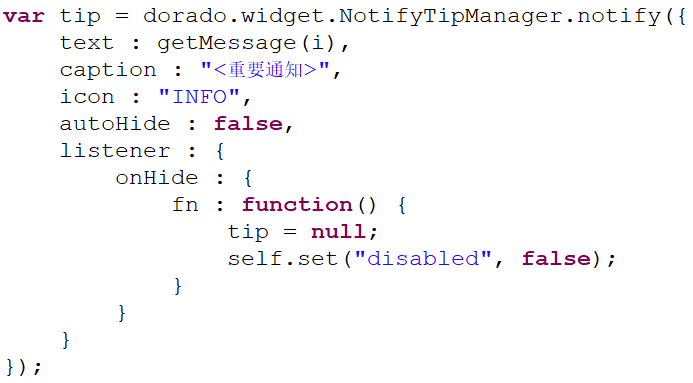


实现：



调用notify方法即可，通知消息会自动显示在屏幕右下角。notify消息默认会自动关闭。

### 4）自定义的通知消息



通过notify方法对通知消息提示框做更多的特性设定，如通过一个JSON对象定制NotifyTip的特性。

* 提示信息通过text设定
* caption修改为“重要通知”（默认为“系统通知”）
* 通过icon属性自定义提示信息的图标
* autoHide设置为false，禁止通知消息自动关闭
* 通知消息提示框关闭按钮才变得可操作

## 2、卡片簿（CardBook）

卡片组件，该组件可以添加多个子组件，但是每次只能显示一个子组件，显示的组件称为活动组件。目前该组件被TabControl使用。

卡片簿就是具有这种功能的一种控件。例如我们在CardBook下添加了多个控件，则可以通过如下代码显示内部的某一个控件：

cardbook.set(“currentIndex”,index);

//currentIndex为当前活动的组件的序号（从0开始计算），index为CardBook下索引号，修改currentIndex的值使不同的控件显示出来。

也可以通过currentControl方法设定当前的控件对象：

cardbook.set(“currentControl”,control1);

//currentControl为当前活动组件

示例：



给按钮绑定onClick事件：

var index = cardbook.get("currentIndex"),

count = cardbook.get("controls").size;

index++;

if (index >= count)

index = 0;

cardbook.set("currentIndex", index);

代码按序列号递增动态设定当前的控件对象。

## 3、多页标签（TabControl）

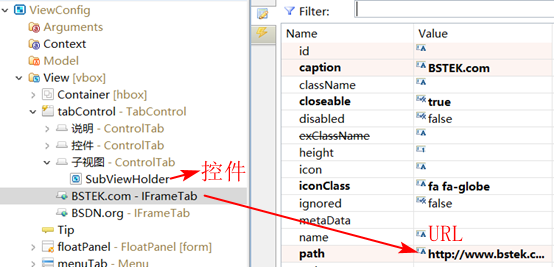
### 1）TabControl：标签页



### 2）标签的分类（ControlTab、IFrameTab）

在一个TabControl中有多个ControlTab（组件标签页）或IframeTab（iframe标签页），他们的区别如下：

ControlTab中可以添加子控件，而IFrameTab与IFrame的用法类似，可以定义一个path属性，内部嵌套一个其他的页面。



### 3）closeable（快速关闭）和disabled（禁止操作）

ControlTab和IframeTab都有closeable和disabled属性。

disabled用于禁止用户的单击操作，closeable用于提供一个X快捷按钮，便于直接关闭这个标签页。



### 4）tabPlacement

通过TabControl的tabPlacement属性，可以设定标签页页头位置调整，如上标签页的页头在顶部。



### 5）NavButtons

当标签页的页头较多时，从而导致TabControl控件无法容纳所有的标签页时，系统会自动出现标签页导航按钮。



### 6）MenuButtons

TabControl提供MenuButtons功能，单击红线内的MenuButtons，系统会自动将当前所有的标签页以菜单的形式展现。



### 7）切换当前标签

除了可以通过鼠标操作切换当前标签页外，也可以通过代码实现当前标签的设定。（尤其是在页面初始化时，希望动态设定当前标签页位置的时候使用；或者当我们对数据做校验后，希望用户定位到一个标签页继续补充内容时很有用。）

tabControl.set("currentIndex", index);

//通过currentIndex属性实现当前标签页的动态设定。

### 8）删除当前标签

var tab = tabControl.get("currentTab"); //currentTab属性为当前Tab

if (tab) {

tab.close();

}

### 9）删除指定标签

var tab = tabControl.getTab(tabName);//getTab方法取得TabGroup中的Tab

if (tab) {

tab.close();

}

### 10）添加标签页（IFrameTab）

var newTab = tabControl.addTab({ //addTab方法为插入标签页

$type : "IFrame",

caption : options.caption,

closeable : options.closeable,

path : options.path

});

tabControl.set("currentTab", newTab); //设置为当前页

### 11）添加标签页（ControlTab）

var newTab = tabControl.addTab({

$type : "Control",

caption : "测试标题",

closeable : true

});

如果还希望在ControlTab内部添加子控件对象，则方法如下：

var newTab = tabControl.addTab({

$type : "Control",

caption : "测试标题",

closeable : true,

control : {

$type : "Control",

style : {

background : ("#cccccc")

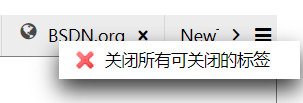
}

}

});

tabControl.set("currentTab", newTab);

### 12）标签页右键菜单



在TabControl的onTabContextMenu事件中添加代码：

view.get("#menuTab").show({ //其中menuTab为一个Menu对象

event: arg.event

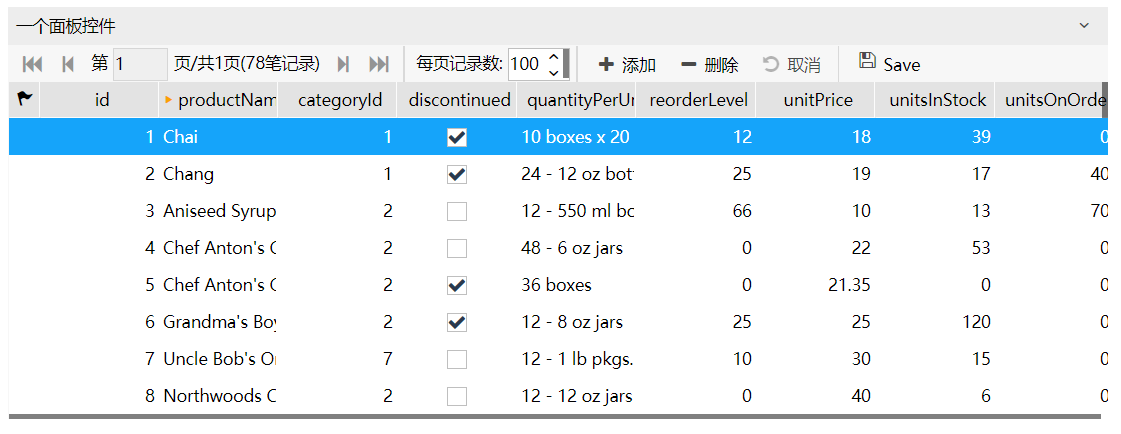
});

## 4、子视图（SubViewHolder）

在View中可以直接通过SubViewHolder控件使用子视图技术。

例如已经开发好一个单表数据维护界面，希望在一个新的页面中嵌入原来已经开发定义好的单表数据维护界面，则可以添加一个SubViewHolder控件，并设置其subView属性指定希望引用的页面的URL：

com.bstek.dorado.sample.data.SimpleCRUD。



## 5、小贴士（Tip）

小贴士就是系统提示信息。

### 1）控件的tip属性

只要设置一个控件的tip属性就可以了。



### 2）dorado.TipManager.initTip

TipManager是一个用于动态定义Tip的工具类，基本方法如：

dorado.TipManager.initTip(self.getDom(), {

text : self.get("tip"), //Tip显示的文本

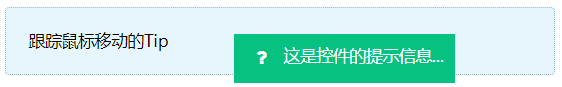
icon: "QUESTION", //图标所在路径

trackMouse: true //跟踪鼠标移动

});

//self.getDom()是希望用于tip特性的目标dom对象

//Tip属性声明的JSON对象，通过其创建一个Tip控件



Tip控件的trackMouse属性：设置为true时，跟踪鼠标移动。

Tip控件的anchorToTarget属性：如果设置为true，则Tip就会出现指向所对应的Dom对象的一个箭头。

dorado.TipManager.initTip(self.getDom(), {

caption: "提示信息",

text : self.get("tip"),

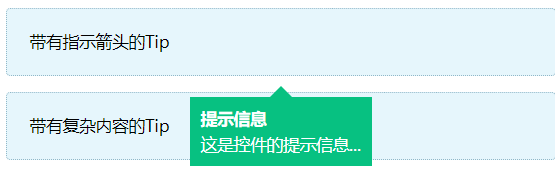
anchorToTarget: true,

align: "center",

vAlign: "bottom",

arrowDirection: "top"

});



Tip控件的Content属性：复杂内容的Tip。

dorado.TipManager.initTip(self.getDom(), {

caption: "提示信息",

content: [{

tagName: "IMG",

src: $url(">images/dorado7.jpg"),

style: "border: 1px lightgray solid; width: 180px; height: 180px"

}, {

tagName: "DIV",

content: "<a href='http://dorado.bstek.com' target='\_blank'>Dorado7</a>",

style: "text-align: center"

}],

anchorToTarget: true,

align: "right",

vAlign: "center",

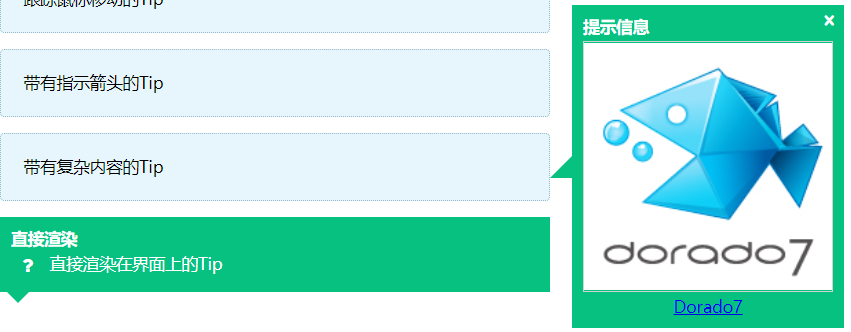
arrowDirection: "left",

autoHide: false,

closeable: true

});

该Tip的内容有图片和Dorado7超链接。



### 3）直接使用Tip控件

直接添加Tip控件，就可以设置相关属性：

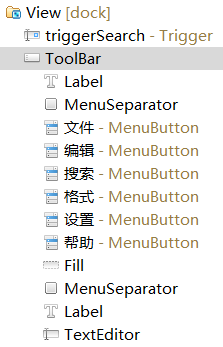
|  |  |
| --- | --- |
| 属性 | 值 |
| arrowAlign | left |
| arrowDirection | bottom |
| caption | 直接渲染 |
| floating | false |
| icon | QUESTION |
| text | 直接渲染在界面上的Tip |
| width | 400 |



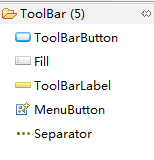
## 6、工具栏

工具栏一般被用在主框架页面当中。

一个工具栏的设计界面如下：



只要添加一个ToolBar控件，并在内部加一些其他的子控件就可以完成一个工具栏的设计，可以添加的子元素有：



### 1）ToolBarButton

是ToolBarButton的一个最基本的按钮控件

### 2）Separator

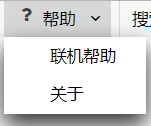
Separator是一个分割栏，效果如下：



### 3）MenuButton

MenuButton控件是一个支持下拉菜单的按钮控件。

在MenuButton下添加1到多个MenuItem元素，最终效果：



### 4）Fill

Fill控件是一个占位控件，在格式MenuButton控件后添加一个Fill控件，则效果如图：



最终格式按钮与搜索标题之间留了很大一块空白，Fill控件会将之后所有的控件挤到ToolBar控件的最右边，实现了控件靠右布局的显示效果。

### 5）icon

ToolBar中可以添加的子控件提供了icon属性，用于定义菜单按钮的图标。

## 7、手风琴

布局组件，分为多个Section，每个Section有一个标题栏和组件，但同时激活的只能有一个Section，类似TabBar或者TabControl。



按不同的手风琴按钮，会打开不同的面板。使用简单，只需添加一个Accordion控件，然后在其中添加不同的Section控件即可。

### 1）currentSection属性

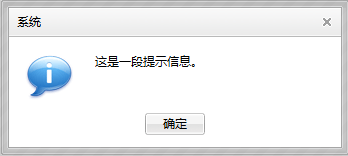
同currentSection属性设定默认展开的Section：

accordion.set("currentSection", section1");

## 8、系统对话框

### 1）alert（默认系统对话框）

dorado.MessageBox.alert(“这是一段提示信息”);



### 2）自定义系统图标

系统对话框中的图标可以自定义：

dorado.MessageBox.alert("这是一段提示信息。", {

icon: "WARNING"

});

//添加一个JSON参数，设置icon属性

### 3）comfirm（确认对话框）

dorado.MessageBox.confirm("请作出您的选择...", {

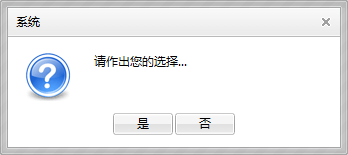
detailCallback: function(button, text) {

dorado.widget.NotifyTipManager.notify("您选择了\"" + button + "\"。");

}

});

通过detailCallback函数的定义。无论用户选择了“是”还是“否”，都会触发自定义的function，并且button参数表示用户选择的是哪个按钮，“是”的button值为“yes”，“否”的button值为“no”。



### 4）prompt（显示单行输入对话框）

用户可以在弹出的系统对话框中输入一段文字。

dorado.MessageBox.prompt("请输入一段文本：", function(text) {

dorado.widget.NotifyTipManager.notify("您输入了：" + text);

});

//function函数相当于用户单击了确定按钮后接受用户输入的文本框的值。

如果希望系统对话框中采用多文本编辑框，则可以使用promptMultiLines方法：

dorado.MessageBox.promptMultiLines("请输入一段文本：", {

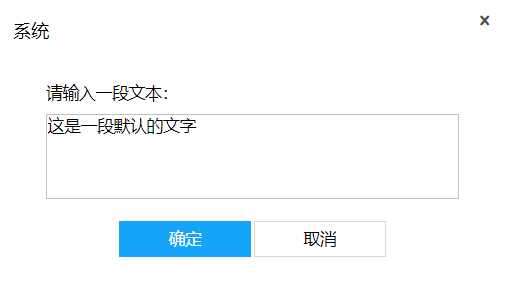
defaultText: "这是一段默认的文字",

callback: function(text){

dorado.widget.NotifyTipManager.notify("您输入了：" + text);

}

});



### 5）show（通过自定义参数显示对话框）

用于灵活控制系统对话框的详细特性。

dorado.MessageBox.show({

icon: "WARNING",

title: "自定义个的标题",

message: "这是一个自定义的对话框",

buttons: ["yes", "no", "cancel"],

closeAction: "cancel",

detailCallback: function(button, text) {

dorado.widget.NotifyTipManager.notify("您选择了\"" + button + "\"。");

}

});

