# Flex学习

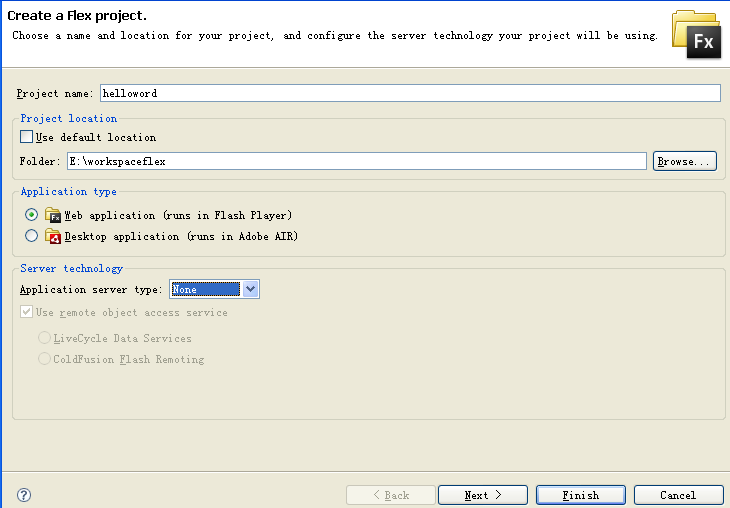
## 第一章 Flex简介和开发环境

1. 简介：
2. Flex3是Adobe开发的RIA（Rich Internet Applications ）的工具。
3. Flex 包括Flex3 SDK,这其中包括编译器,类库,和调试工具。
4. 开发工具Flex3 Bulider,整合了上面的开发环境和Eclipse。
5. 和Flex整合的服务器产品，如Lifestyle Data Services,CodeFushion。
6. <http://www.adobe.com/cn/>官方网站。
7. 什么是Flex Application？
8. 用Flash.swf文件格式封装发布在html里面。
9. 需要客户端安装Flash Player 9.
10. 应用程序使用ActionScript3,简称AS3。(兼容FlashCS3和ActionScript3编写的文档)
11. 开发者可以完全访问Flash绘画API.
12. 下载安装和别的软件意一样



这是我下载的破解版,按照说明就可以了。

1. 创建一个helloword工程



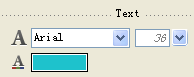
直接finish就可以。

1. 我们目的是调节背景色和添加一个label。

我们调节如下就可以：

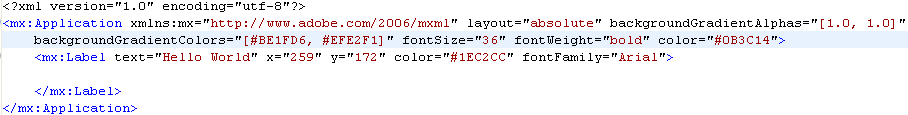


下面是添加Label,并设置lable字体的属性，如下图：

这样就可以设置了,随后我们运行就可以了。



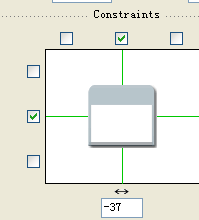
Helloword.mxml内容如下：



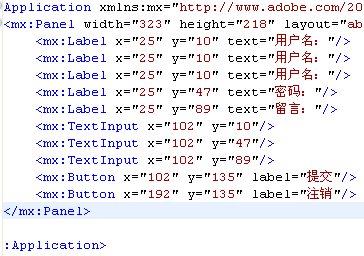
1. 下面我们在创建一个form。
2. 创建工程的过程和上面一样。
3. 我们脱一个panel，在这个面板上面添加几个label和文本框还有按钮。设置什么的和上面基本一样，这里不截图了。字体设为12,一般情况



运行后效果如上。如何使面板居中,如下设置即可：

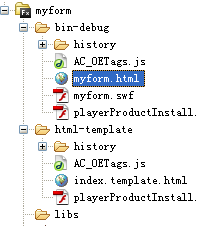


Myform.mxml源代码如下：



标签为<mx>因为第一行xmlns命名空间的定义。

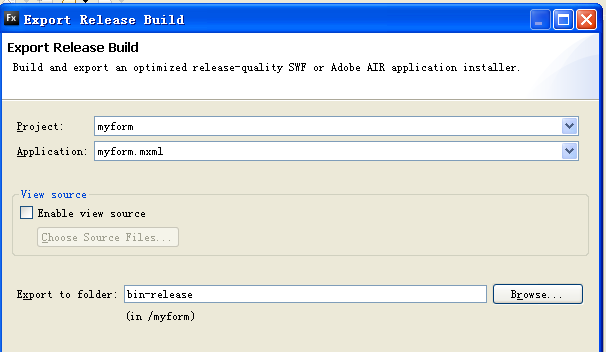
1. 介绍工程的结构。



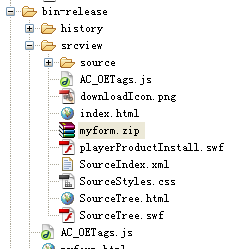
结构如上面,在bin-debug下面的myform.html里面定义了检查flash版本的代码和如果浏览器不支持javascript的代码，这个文件的生成是根据下面的index.template.html生成的。我们如果修改模板的配置就是修改上面的配置了。

## 第二章 AS3语法

1. 我们首先介绍一下怎样发布一个项目和导入一个项目,我们用菜单下的export和import两种来完成.



这是导出的,导入的话,就是我们上面导出的时候生成的bin-release包下的zip就可以。如下：



1. 下面我们做一个例子，左右都有list，并且可以相互拖动，看完代码我们在看语法，AS3是基础，那些mx标签最终也要编译成AS3的语言。



1. 里面用到了函数的定义和数组的定义，还有for循环，还有定义list的属性。这其中还有Number类型和Array类型。在flashplay创建完我们的应用之后就会调用initApp的方法(applicationComplete事件)。
2. 组件上是画布上(canvas)放了两个list。
3. CDATA是只元要素,我们在这个里面写什么”<,!....”都行了,要是写在外面的话,会背解析成别的意思,因为我们的文本是mxml的形式的。
4. **下面介绍基础的语法**

这其中包括if,switch,for,for in,for each,while,do while,break continue.

这里先介绍一下怎样debug，这里我们调用trace(“hahahah”);就可以在控制台中输出内容。要想用trace函数必须安装flashplay的bebug的版本。这样我们的信息就会在控制台进行输出。  
for循环有三种方式：

第一种：

for(var i:Number = 0;i<7;i++){

arr.push("元素"+i);

}

第二种：

var student:Object = new Object();

student.name = "邢健";

student.age = 25;

student.type = "本科";

for(var temp:String in student){

textarea.text = "\n"+textarea.text+"\n"+student[temp.toString()];

}

第三种：

for each(var temp:\* in student){

textarea.text = textarea.text+"\n"+temp.toString();

}

Continue和break的用法和java的一样

If else 和以前一样

Label的用法：跳出两层循环，相当goto的概念

outerLoop: for (var i:int = 0; i < 10; i++) {

for (var j:int = 0; j < 10; j++) {

if ( (i == 8) && (j == 0)) {

break outerLoop;

}

trace(10 \* i + j);

}

}

With的用法：

with (someOther\_mc) {

\_x = 50;

\_y = 100;

gotoAndStop(3);

}等效于下面

someOther\_mc.\_x = 50;

someOther\_mc.\_y = 100;

someOther\_mc.gotoAndStop(3);

特殊类型

有三种： \* void null

构造函数不能重载,必须的参数必须放在可选的参数之前

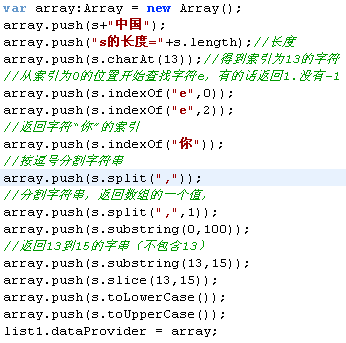
常量的使用const

Internal子类是可以看得见得。

数组的用法Array:



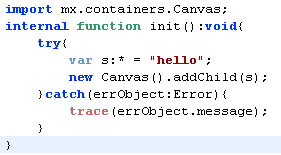
字符串String的用法:



各种数据类型做运算基本都可以,注意的是boolean和int型做运算(boolean的取值是1,0)

类型转换的时候,也可以用as

运行时类型检查:也就是异常处理



注意的是\*为所有类型.

## 第三章 事件驱动

很多开发人员都知道如何使用事件机制，但却不理解事件机制的基本工作原理。在这里我们探讨一下ActionScript 3的事件机制(无论是新手还是老手，我相信看到最后肯定对你有所帮助 )。   
1. 首先来看一下事件机制中的几个基本概念：   
event objects(这里我们姑且叫它事件对象)：所有的事件对象都是flash.events.Event或者其子类。比如说鼠标事件MouseEvent，计时器事件TimerEvent。每个事件对象中都包括事件类型(type)和事件源(target)，便于event listener知道该事件的类型以及是谁抛出这样的事件。   
event dispatchers(姑且叫它事件源):是事件发起的主体。所有的dispatcher都是EventDispatcher或其子类。事件触发源必须提供API让listener进行对该事件的注册(register)。   
event listeners(姑且叫它事件监听器):在AS 3.0中它往往表现为一个函数。我们必须为每个特殊的事件和事件源注册一个监听器，当事件发生的时候，事件源(dispatcher)会通知该监听器去处理制定的事件。   
2. 如何注册监听器(registering Listener)   
在AS 3.0中可以通过下面的语法格式进行注册。   
dispatcher.addEventListener(eventName, listener);   
当然在Flex中还有另外一种格式，在mxml文件中对一个控件的事件直接制定一个处理函数，如   
<mx:Button label="TestClick" click="clickHandler()"/>   
3. 如何删除监听器(removing Listener)   
在AS 3.0中以下面的格式进行删除   
circle.removeEventListener(MouseEvent.CLICK, onClick);   
应该很容易理解。   
但这里确有一点很重要：尽量删除你已经不再使用的事件监听器，这个涉及到Flex 内存泄漏的问题，以后会有详细介绍。   
4. 理解事件阶段(Event Phases)   
我们把事件触发源分为两类：一类是那些显示在界面中的元素抛出的事件，一类是那些不在界面中显示的元素抛出的事件。很简单，举个例子：第一类如URLLoader对象去load图片或其他资源所抛出的如complete事件，就属于第一类。界面上的一个按钮所抛出的click事件就是第二类，它会触发一个事件流，引发一系列的事件，它又分三个阶段：捕获(capture)，执行(target) 和冒泡(bubble)，姑且这样表达这三个阶段。   
捕获阶段(capture)   
在这个阶段，Flash Player会去查找事件的触发源，它是通过根显示元素(root display object)逐层向下寻踪，直到找到事件的发起源头，如你在button上click后，Flash Player找到的事该button对象。在这个阶段监视器(listener)默认是不会收到任何消息的(默认只在target和bubble阶段收到消息)。   
如果你想在捕获阶段让listener收到消息并有所动作，那么你可以在注册的时候改变第三个参数为true，如下面的方法：object.addEventListener(MouseEvent.CLICK, onClick, true);   
该阶段听起来很陌生，事实上它很少使用。但它却在事件流中扮演很重要的角色。它可以在事件到达子元素(child elements)之前被application捕获，当然也就可以阻 止事件到达子元素。   
目标执行阶段(target)   
在这个阶段是事件的执行阶段，也就是我们常用的用来处理事件流的代码。   
冒泡阶段(bubble)   
最后的这个阶段其实就是第一个阶段的逆向过程。它就是在事件代码执行完毕后通过子元素逐级向父元素发出该事件，一直到根元素。在这个阶段你不需要特别的为监听器指定什么，你只要正常的注册事件就可以了。监听器会在该阶段收到通知。但前提是该事件是能够bubble的。   
5. 事件的优先级(Event Prioriy)   
事件优先级我们用的很少，但理解它却很重要，但你添加一个事件的时候你可以设置监听器的优先级，优先级越高，你的监听器函数会越早被调用到。当然两个优先级相同的监听器是根据他们被添加的顺序来决定优先级的。这个优先级指的是在同一个事件阶段(如capture,target,or bubble).一次，事件处理的优先级是先看阶段再看优先级的。   
6. 弱引用(Weak Reference)   
弱引用时事件处理中非常重要的概念。要理解它为什么这么重要，你首先要理解Flash Player是如何进行垃圾回收的(garbage collection).   
  
顾名思义，垃圾回收就是回收那些不再使用的资源，主要是内存资源，否则会致系统崩溃。   
  
Flash Player是通过引用计数器(reference counting)以及标记与清除(mark and sweep)来实现垃圾回收的。   
引用计数器简单的讲就是给每个对象分配的技术器，一旦该对象的计数器为0时，就表示该对象可能不再被使用，可以安全的清除了。它的弊端就是不能解决循环引用问题，一旦出现循环引用，就不能通过该方法进行垃圾回收。   
而标记与清除(mark and sweep)就是为了解决循环引用问题的。它是通过在程序中遍历每个活动的、可到达的节点，一旦发现一些节点是处在活动节点之外的，就被认为不再有用，就可以被回收了。   
这两种机制是联合运行在垃圾回收过程中来保证正确回收资源。但他们并不能回收程序错误导致的垃圾。而监听器的注册就最容易导致这个问题的发生。

## 第四章 自定义事件

1. 这样的方式也可以:

添加自定义的事件

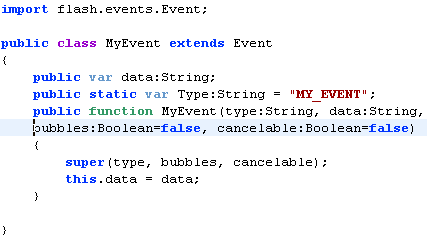
button2.addEventListener(**"testevent"**,onShow);

出发时间

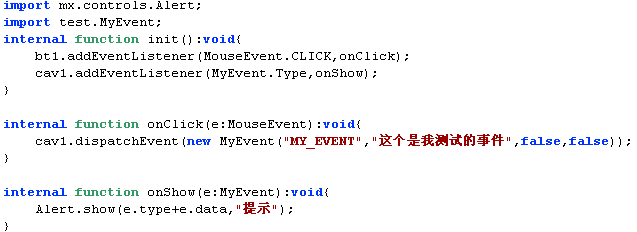
button2.dispatchEvent(**new** Event(**"testevent"**,**true**,**false**));

1. 下面的是手动写的类

说一个注意的地方,如果两个组件注册了相同的事件,而且两个组件有叠加,那么默认的情况他会按照事件流的方式从里到外的触发该事件,要想分开对这两个事件的区分可以采用当前的target==目标的target



下面是测试的地方:

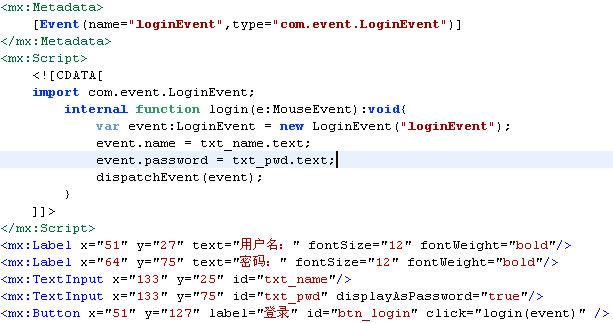


下面做一个登录验证的例子:

首先我们创建一个工程:

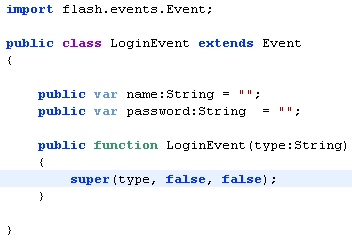
1. 定义一个自定义的组件,mxml components,这里我们继承的panel

这个组件上面放着几个控件.



我们通过元数据的方式对整个form进行添加事件,就是最上面得那段,这个事件也是我们自己定义的,最后点击按钮的时候我们触发这个新增加的事件.(自定义的事件里面包含登录信息),注意button里面的click事件的方法里面必须写event.

下面是自定义的事件:



测试的里面如下:



注意因为上面自定义的form里面是采用元数据的方式添加事件的,所以在form里面有loginEvent的事件.

## 第五章 数据绑定

**数据绑定：**

当数据源对象的数据发生变化时，目标对象的数据会自动更新，而不需要我们再编写代码去强制更新.

绑定实际也是借助事件机制来完成的，当目标使用了数据绑定的时候，目标对象就会监听数据源对象的某一固定事件。当数据源发生变化时，数据源会派发改变事件（ChangeEvent），通知目标对象更新数据。这个过程由Flex完成，不用我们手动干预。

绑定的前提条件：源对象的数据和目标对象的数据格式相同。

实现数据绑定的方法

1. 在对象的属性标签中,使用{ }把数据源直接绑定到对象的某个属性上。

例如:

<mx:Label x="136" y="53" text="邢健加油!" fontSize="{hslider.value}"/>

<mx:HSlider x="109" y="211" minimum="10" maximum="50" id="hslider"/>

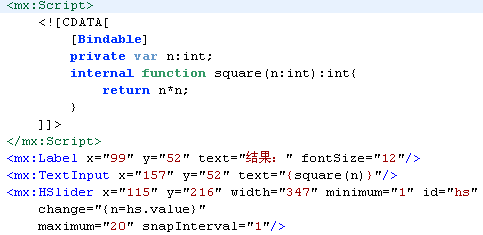
2.在对象的属性标签中,使用{ }把某个函数的返回值作为数据源绑定到对象属性上。

函数的参数要使用[Bindable]绑定符号[Bindable]

**[Bindable(event=“eventname”)]**

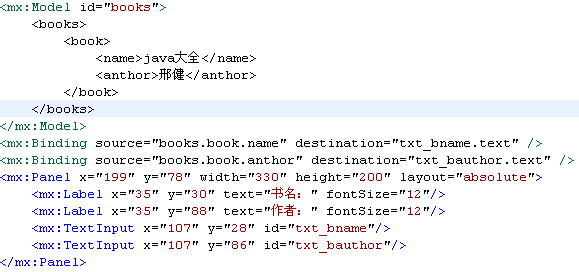
**Event表示当数据源发生变化时，数据源所在对象派发的事件类型，它是可选项，默认的事件名是“propertyChange”，一般情况下只需要使用[Bindable]标签**

**例如:**

****

**3.使用<mx:Binding>标签**

**source=“…” destination=“…” 例如:**

****

**注意要是book是多个话,可以用books.book[0].name**

**4.<mx:Model>标签**

**<mx:Model id=“books”>**

**<books>**

**<book>**

**<name>FLEX教程</name>**

**<author>张三</name>**

**</book>**

**<book>**

**<name>JAVA高级编程</name>**

**<author>李四</name>**

**</book>**

**</books>**

**</mx:Model>**

**5.Object对象是一个动态对象**

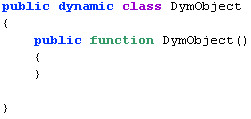
**var o:Object=new Object();**

**o.name=“Jack”;**

**o.age=20;**

**o.address=“北京海淀100号”;**

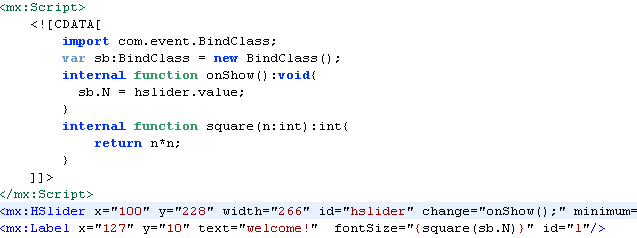
**创建动态类:**

****

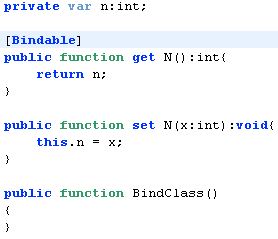
**Model中的节点数据默认是Object类型，作数据源必须转换成ObjectProxy类型。**

**[Bindable]标签用于函数时，只能在setter和getter方式定义的函数前使用。称为函数级别绑定。**

**如下:**

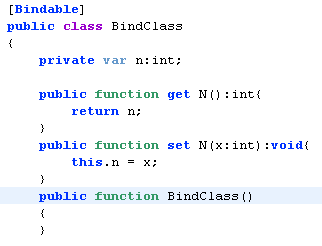
****

**类如下:**

****

**注意get函数的绑定.**

**[Bindable]标签用于公有类时，这个类的所有公有变量、 setter和getter方式定义的函数都可以用于绑定。成为对象级别绑定。例如:**

****

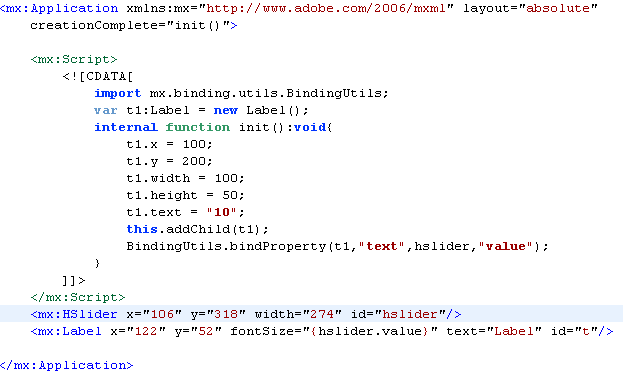
**注意get和set都有绑定的功能了.因为是public的,上面的变量就不可以了.因为是private**

**Mx.binding.utils**

**BindingUtils提供了两个静态方法处理动态绑定**

**bindProperty(site:Object,prop:String,host:Object,chain:Object,commitOnly:Boolean=false)**

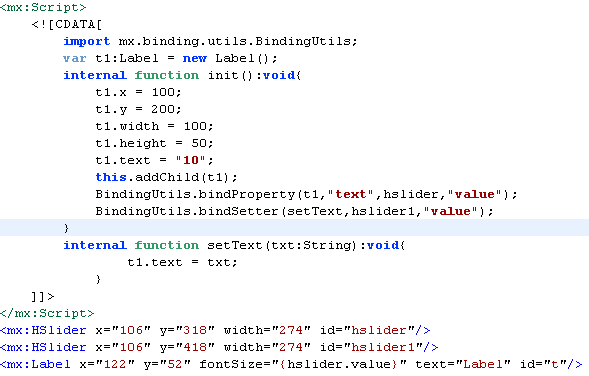
**例如:**

****

**这就是动态的label绑定数据**

**注意bindProperty的用法,第一参数是要改变的对象,第二个是要改变对象的属性,第三个是数据源,第四个是数据源的属性**

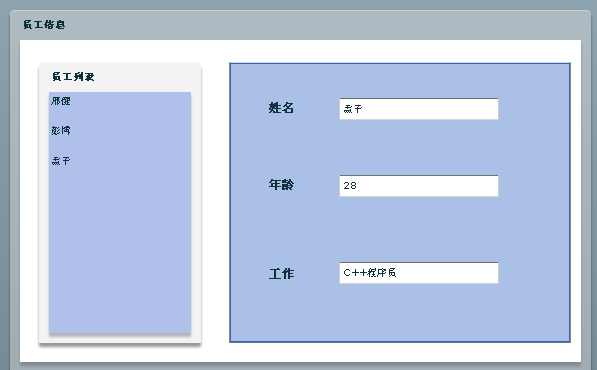
**bindSetter(setter:Function,host:Object,chain:Object,commitOnly:Boolean=false)如下:**

****

**注意bindSetter的用法第一参数是方法,第二个是数据源,第三个是数据源的属性.**

**下面过一个人员列表的例子:**

**如下图:**

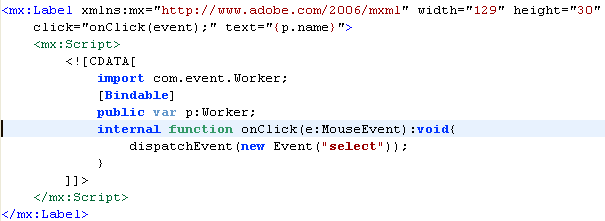
****

**点击左面的时候右面变化:**

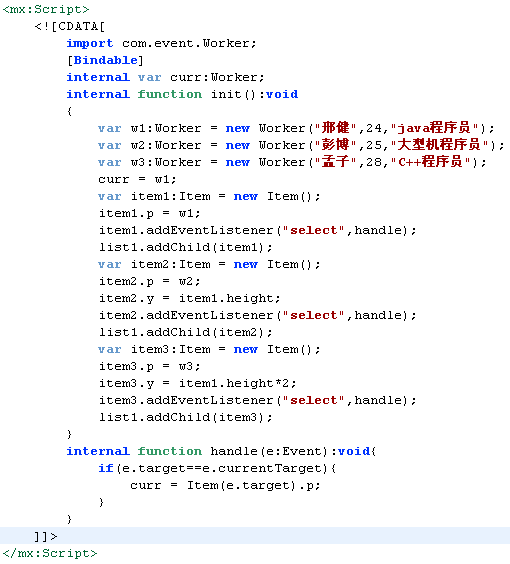
**首先说一下做的思路:**

**左面的是一个面板上面放置多个lable,点击lable的时候发生事件,这个lable是我们自定义的,在它的里面已经和对象对其进行绑定.事件也是我们自己定义的，这里为“select”.点击lable的时候就派发这个事件，所以用到这个组件的时候就要添加这个事件。右面几个属性值，也是一个数据绑定的，我们在当前的应用里面定义了一个当前的对象，这个当前的对象和它们分别绑定。下面看代码部分的截图：**

**自定义组件类：**

****

**注意：高度设置一下。否则默认为0.其实就是一个对象和lable的绑定，这个lable有select事件。**

****

**定义了几个对象，都装上监听，e.target其实就是我们的item对象。**

****

**这是那几个textinput。**

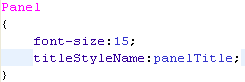
## 第六章 样式表和主题

我们采用样式表通常是这样的5种方式:

**1.直接使用CSS文件,系统默认使用default.css**

**<mx:Style source=“style.css”/>**

**如下:**

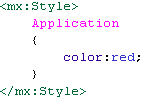
****

**这是我们自己定义的样式表.在程序里面引入**

****

**就可以了.在组件上面有styleName=”引入就可以了”;**

1. **使用<mx:Style>标签**

**这样就可以改变字体的颜色.**

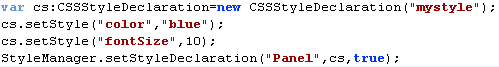
1. **使用style设计器（->MXML）**

**这里只得就是开发工具上面的,有fill的等都可以.**

1. **使用对象的getStyle( )和setStyle( )**

****

1. **使用StyleManager类**

****

**如上.样式属性分为可以继承和不可以继承两种,上级容器可继承的属性，直接作用其包含的组件。**

全局样式：Global Var globaalStyle:CSSStyleDeclaration=StyleManager.getStyleDeclaration(“global”);

**主题:**

主题是一套配置完成的样式风格 Flex提供的几种主题，文件类型.swc

建议使用.swc文件来封装主题 给编译指令添加 -theme xxx.css

在Flex-config.xml中通过修改<theme>标签配置主题

1.通过修改以后的主题文件创建自己的主题内容

2.使用compc编译主题，生成swc文件（见帮助)

-include-file name path

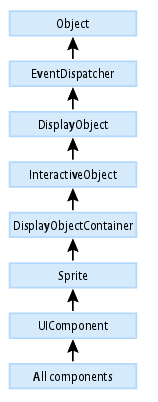
-o output.swc

-include-classes

3.通过配置文件编译主题 compc -load-config myconfig.xml

## 第七章 走进组件

1. 组件类的继承关系如下图:



1. 下面介绍组件我们按如下的分类:

容器类组件（Containers）

常用的可视化组件（General controls ）

数据验证类组件（Validators and formatters ）

行为与特效类组件（Effects,view states,and transitions）

可视化数据组件（Data visualization components）

其他组件

1. 容器类组件:

Application,Canvas ,Panel ,Titlewindow ,Form,FormHeading,FormItem ,Box (VBox、HBox),Dividebox(VDivideBox、HDivideBox), Tile,Grid,GridItem,GridRow ,Accordion,ViewStack,TabNavigator ,Applicationcontrolbar

1. Application

是所有组件最上层的组件,先说一下布局这块,是否有约束布局的话,取决于它的父类容器.布局这块有三种方式:一个是垂直的(放到该容器上面的组件都是按顺序垂直放置),一个是水平方式(放到该容器上面的组件都是按水平方式放置)还有一个是绝对方式(有坐标的方式). Application的使用基本就是布局.

里面的属性还有设置每个组件的距离等很多方法。

1. Applicationcontrolbar

这个使用一般和Application一起使用.它就是一个普通的容器,上面可以放置组件,但有一个属性Dock属性,当它的值为false的时候就可以随便放置,为true的时候就停靠在父窗口的上面。通常是用来做全局导航的来使用。Paddingxxx的属性可以设置四周的距离。

1. Canvas

画布，只有绝对坐标的布局方式，没有其它的两种。

1. Vbox和Hbox（容器）也和Aoolication一样可以设置摆放方式

Vbox放置里面的组件都垂直摆放，Hbox放置里面组件都水平摆放。

1. Dividebox(容器，多了一个分割条)

其中包括VdivideBox，HdivideBox这其中有live draging的属性是设置拖动的效果是实时拖动，还是鼠标左键放下才拖动。（一个属性为true，一个为false）

1. Panel

注意有一个statue的属性,最终显示在右上角的地方。

1. Titlewindow

有一个关闭的按钮，但这个我们可以设置为显示和不显示。点击关闭的时候触发close事件，在这里我们可以处理我们的业务。注意的地方是：Popupmanager是用来管理所有的弹出窗口的（这些窗口都可以移动）。

1. Form(有自己的表头，里面有一些formitems)每个item都有两个值，一个显示的，一个是需要填写的。

FormHeading,FormItem有这些东西在里面

这里提前把校验的东西说一些,通常的校验有字符串，电话号，日期，数字，区域码，邮箱等。在验证的里面有个source的属性(这个是校验那个组件的id),property(“组件的什么属性,一般是text”).

1. Tile平铺

平铺的方式有水平的方式和垂直的方式（它会计算放置厘米组件的宽度和自己的宽度的比例，有点像flowlayout）

1. Grid栅格组件（里面都是griditems）

和java的基本一样，有rowspan和colspan等属性，GridItem,GridRow这些在里面。

1. Accordion用来导航的。

相当我们用的outlook风格的容器，在accordion里面每添加一个容器都相当时一个outlook的子项。

它的selectIndex属性可以设置当前选中的状态。也可以用selectChild来设置，只不过这里是容器的名字。索引的方式是按排列的顺序。

1. ViewStack（相当cardlayout）
2. TabNavigator

选中的状态和Accordion一样用index和selectChild

1. 常规组件：

Button,ButtonBar, CheckBox, LinkBar, LinkButton, PopupButton, RadioButton,RadioButtonGroup ,ToggleButtonBar

Datechooser , Datefield ,Alert,ColorPicker,ComboBox,DataGrid,Horizontallist,HRule,HScrollber,HSlider,List,

NumericStepper,ProgressBar,Spacer,Tabbar,TileList,Tree,VRule,VScrollBar,VSlides,Menu,MenuBar,PopupMenuButton,Label,RichTextEditor,Text,TextArea,TextInput

1. Alert

Alert.show();方法里面可以传很多的参数，text显示的信息，title是标题，flag是指显示的风格（对应的值有1.2.4.8），parents指的是父窗口，clickListenner只要执行的函数，这里我们一般的时候这样写：

Private function clickMethord(enevt:CloseEvent):void{

If(event.detail==Alert.ok)[

}

}

也可以修改Alert.buttonWidth ,Alert.yesLabel,Alert.noLabel等属性。

1. ColorPicker

有selectcolor.toString(16);

1. ComboBox

有一个dataprovide,一般的时候用arrayCollection里面装这样类型的数据,有close关闭的事件。

New ArrayCollection([{[label:xxx,data:data]},{ [label:xxx,data:data]}]);

有selectItem的属性还可以selectitem.label/data

1. Horizontallist

水平给你放置的list(手机)

1. TileList

竖向方向显示的，可以设置每行显示多少个。

<mx:mode>可以在页面定义数据源相当于xml

1. Tree

<mx:Tree id="myTree" width="50%" height="100%" labelField="@label" showRoot="false" dataProvider="{treeData}" change="treeChanged(event)"/>

这块是数据源@label是读xml格式的一种方式

<mx:XMLList id="treeData">

<node label="Mail Box">

<node label="Inbox">

<node label="Marketing"/>

<node label="Product Management"/>

<node label="Personal"/>

</node>

<node label="Outbox">

<node label="Professional"/>

<node label="Personal"/>

</node>

<node label="Spam"/>

<node label="Sent"/>

</node>

</mx:XMLList>

1. Datagrid

数据表格

<mx:DataGrid id="dg" width="100%" height="100%" rowCount="5" dataProvider="{employees}">

<mx:columns>

<mx:DataGridColumn dataField="name" headerText="Name"/>

<mx:DataGridColumn dataField="phone" headerText="Phone"/>

<mx:DataGridColumn dataField="email" headerText="Email"/>

</mx:columns>

</mx:DataGrid>

数据源：

<mx:XMLList id="employees">

<employee>

<name>Christina Coenraets</name>

<phone>555-219-2270</phone>

<email>ccoenraets@fictitious.com</email>

<active>true</active>

</employee>

</mx:XMLList>

<mx:form的使用>

<mx:Form width="100%" height="100%">

<mx:FormItem label="Name">

<mx:Label text="{dg.selectedItem.name}"/>

</mx:FormItem>

<mx:FormItem label="Email">

<mx:Label text="{dg.selectedItem.email}"/>

</mx:FormItem>

<mx:FormItem label="Phone">

<mx:Label text="{dg.selectedItem.phone}"/>

</mx:FormItem>

</mx:Form>

1. TabBar

<mx:TabBar itemClick="clickEvt(event);">

<mx:dataProvider>{STATE\_ARRAY}</mx:dataProvider>

</mx:TabBar>

数据源：

[Bindable]

public var STATE\_ARRAY:Array = [{label:"Alabama", data:"Montgomery"},

{label:"Alaska", data:"Juneau"},

{label:"Arkansas", data:"LittleRock"}

];

1. ProgressBar

<mx:ProgressBar id="bar" labelPlacement="bottom" themeColor="#F20D7A"

minimum="0" visible="true" maximum="100" label="CurrentProgress 0%"

direction="right" mode="manual" width="100%"/>

1. Spacer

空白区<mx:Spacer width="100%"/>

<mx:Image source="@Embed('../assets/Nokia6630.png')"/>

1. NumericStepper

<mx:NumericStepper id="ns"

minimum="10.00" maximum="40.00"

stepSize="0.01"

value="20.00"

width="65"/>

<mx:NumericStepper/>这个是默认的属性。

1. Vrule，、Hrule

垂直的一条线

<mx:VRule rollOverEffect="WipeUp" strokeWidth="1" strokeColor="red"/>

1. Vslides，Hslides

<mx:VSlider id="hSlider"

dataTipPlacement="top"

minimum="0" maximum="100" value="100"

tickColor="black"

snapInterval="1" tickInterval="10"

labels="['0%','100%']"

allowTrackClick="true"

liveDragging="true"

change="changeSize();"/>

phoneImage.width=uint(imageWidth\*hSlider.value/100);

1. VscrollBar

就是竖直的一个滚动条

<mx:HScrollBar id="bar" width="100%"

minScrollPosition="0" maxScrollPosition="{panel.width - 20}"

lineScrollSize="50" pageScrollSize="100"

scroll="myScroll(event);"

repeatDelay="1000" repeatInterval="500" />

上面的几个size属性就是分别点击三角号和里面的增量大小。事件类型：

private function myScroll(event:ScrollEvent):void

{ "The maximum scroll position: " + event.currentTarget.maxScrollPosition + '\n' +

"The minimum scroll position: " + event.currentTarget.minScrollPosition ; }

1. Button

<mx:Button id="iconButton" icon="@Embed('../assets/btnicon.png')" label="Button with Icon"

paddingLeft="12" paddingRight="18" labelPlacement="right" color="#993300" click="printMessage(event);"/>

<mx:Button label="Customized Button" color="#993300" toggle="true" selected="true" //toggle是按钮状态

textAlign="left" fontStyle="italic" fontSize="13" width="{iconButton.width}" click="printMessage(event);"/>

<mx:Button label="Default Button" click="printMessage(event);"/>

1. Buttonbar

<mx:ButtonBar itemClick="clickHandler(event);">

<mx:dataProvider>

<mx:Array>

<mx:String>Flash</mx:String>

<mx:String>Director</mx:String>

<mx:String>Dreamweaver</mx:String>

<mx:String>ColdFusion</mx:String>

</mx:Array>

</mx:dataProvider>

</mx:ButtonBar>

事件处理：

private function clickHandler(event:ItemClickEvent):void {

myTA.text="Selected button index: " + String(event.index) +

"\n" + "Selected button label: " + event.label;

}

1. Checkbook

<mx:CheckBox id="milkCB" label="milk" click="modifyCart()"/>

1. Linkbar

<mx:LinkBar color="#0000FF" fontWeight="bold" dataProvider="{myViewStack}"/> 下面是数据源：

<mx:ViewStack id="myViewStack" borderStyle="solid" width="100%" height="80%">

<mx:Canvas id="search" backgroundColor="#FFFFCC" label="Search" width="100%" height="100%">

<mx:Label text="Search Screen" color="#000000"/> </mx:Canvas>

<mx:Canvas id="custInfo" backgroundColor="#CCFFFF" label="Customer Info" width="100%" height="100%"> <mx:Label text="Customer Info" color="#000000"/></mx:Canvas>

<mx:Canvas id="accountInfo" backgroundColor="#FFCCFF" label="Account Info" width="100%" height="100%"> <mx:Label text="Account Info" color="#000000"/> </mx:Canvas>

</mx:ViewStack>

1. Linkbutton

样子和button不一样

1. PopupButton

就是一个下拉列表

<mx:PopUpButton id="popB" label="Edit" creationComplete="initMenu();" width="135" />

private function initMenu():void {

myMenu = new Menu();

var dp:Object = [{label: "New Folder"}, {label: "Sent Items"}, {label: "Inbox"}];

myMenu.dataProvider = dp;

myMenu.selectedIndex = 0;

myMenu.addEventListener("itemClick", itemClickHandler);

popB.popUp = myMenu;

popB.label = "Put in: " + myMenu.dataProvider[myMenu.selectedIndex].label;

}

1. Radiobutton

<mx:RadioButton groupName="year" id="option1" label="1942"/>

<mx:RadioButton groupName="year" id="option2" label="1952"/>

<mx:RadioButton groupName="year" id="option3" label="1962"/>

<mx:RadioButton groupName="year" id="option4" label="1972"/>

<mx:Button label="Check Answer" click="Alert.show(option4.selected?'Correct Answer!':'Wrong Answer', 'Result')"/>

1. RadioButtonGroup

<mx:RadioButtonGroup id="cardtype" itemClick="handleCard(event);"/>

<mx:RadioButton groupName="cardtype" id="american" value="AmEx" label="American Express" width="150" />

<mx:RadioButton groupName="cardtype" id="masterCard" value="MC" label="MasterCard" width="150" />

<mx:RadioButton groupName="cardtype" id="visa" value="Visa" label="Visa" width="150" />

事件：

private function handleCard(event:ItemClickEvent):void {

if (event.currentTarget.selectedValue == "AmEx") {

Alert.show("You selected American Express")

} else {

if (event.currentTarget.selectedValue == "MC") {

Alert.show("You selected MasterCard")

} else {

Alert.show("You selected Visa")

}} }

1. ToggleButtonBar

就是有深陷的效果：

<mx:ToggleButtonBar itemClick="clickHandler(event);">

<mx:dataProvider>

<mx:Array>

<mx:String>Flash</mx:String>

<mx:String>Director</mx:String>

<mx:String>Dreamweaver</mx:String>

<mx:String>ColdFusion</mx:String>

</mx:Array>

</mx:dataProvider>

</mx:ToggleButtonBar>

1. Datechooser

日期组件：

<mx:DateChooser id="dateChooser1" yearNavigationEnabled="true"

change="displayDate(DateChooser(event.target).selectedDate)"/>

事件里面的到参数就是date的类型。

<mx:DateChooser id="dateChooser2" yearNavigationEnabled="true"

disabledRanges="{[ {rangeEnd: new Date(2006, 5, 1)} ]}"/>

<mx:DateFormatter id="df"/>格式化日期的一个组件。

1. DateField

就是日期的一个按钮。

<mx:DateField id="dateField1" yearNavigationEnabled="true"

change="dateChanged(DateField(event.target).selectedDate)" />

<mx:DateField id="dateField2" yearNavigationEnabled="true"

disabledRanges="{[ {rangeEnd: new Date(2006, 5, 1)} ]}" />

1. MenuBar

<mx:MenuBar labelField="@label" itemClick="menuHandler(event);" dataProvider="{menuBarCollection}" />

public var menuBarCollection:XMLListCollection;

private var menubarXML:XMLList =

<> <menuitem label="Menu1" data="top">

<menuitem label="MenuItem 1-A" data="1A"/>

<menuitem label="MenuItem 1-B" data="1B"/>

</menuitem>

<menuitem label="Menu2" data="top">

<menuitem label="MenuItem 2-A" type="check" data="2A"/>

<menuitem type="separator"/>

<menuitem label="MenuItem 2-B" >

<menuitem label="SubMenuItem 3-A" type="radio"

groupName="one" data="3A"/>

<menuitem label="SubMenuItem 3-B" type="radio"

groupName="one" data="3B"/>

</menuitem>

</menuitem> </>;

private function initCollections():void {

menuBarCollection = new XMLListCollection(menubarXML);

}

private function menuHandler(event:MenuEvent):void {

if (event.item.@data != "top") {

Alert.show("Label: " + event.item.@label + "\n" +

"Data: " + event.item.@data, "Clicked menu item");

}

}

1. Menu

private var point1:Point = new Point();

private var myMenu:Menu;

private function showMenu():void {

myMenu= Menu.createMenu(panel, myMenuData, false);

myMenu.labelField="@label"

myMenu.addEventListener("itemClick", menuHandler);

point1.x=mybutton.x;

point1.y=mybutton.y;

point1=mybutton.localToGlobal(point1);

myMenu.show(point1.x + 25, point1.y + 25);

}

private function menuHandler(event:MenuEvent):void {

Alert.show("Label: " + event.item.@label, "Clicked menu item");

}

1. PopUpMenuButton

<mx:PopUpMenuButton id="p2" dataProvider="{treeDP2}" labelField="@label" itemClick="itemClickHandler(event);"/>

<mx:XMLList id="treeDP2">

<node label="Inbox"/>

<node label="Calendar"/>

<node label="Deleted Items"/>

</mx:XMLList>

public function itemClickHandler(event:MenuEvent):void {

Alert.show("Menu label: " + event.label + " \n " + event.index);

}

1. Label

<mx:Label id="simpleLabel" text="This Label displays plain text."/>

<mx:Button id="Display" label="Click to display HTML Text" click="displayHTML();"/>

<mx:Button id="Clear" label="Click to display plain text" click="displayText();"/>

private var htmlData:String="<br>This label displays <b>bold</b> and <i>italic</i> HTML-formatted text.";

private function displayHTML():void {simpleLabel.htmlText= htmlData; }

private function displayText():void {simpleLabel.text="This Label displays plain text."; }

1. Text

<mx:Text width="100%">

<mx:text>

This is a multiline, non-editable text component.

If you need an editable multiline component, use TextArea.

</mx:text>

</mx:Text>

<mx:Text width="100%">

<mx:htmlText>

<![CDATA[This is <font color="#FF0000">HTML text</font> in a <b>Text component</b>. Using the <u>htmlText attribute</u> of the <font color="#008800">Text component</font> you can use basic HTML markup.]]>

</mx:htmlText>

</mx:Text>

1. TextArea

<mx:TextArea width="400" height="100">

<mx:text> This is a multiline, editable TextArea control. If you need

a non-editable multiline control, use the Text control.

</mx:text>

</mx:TextArea>

<mx:TextArea width="400" height="100">

<mx:htmlText><![CDATA[This is <font color="#FF0000">HTML text</font> in a <b>TextArea control</b>. Use the <u>htmlText property</u> of the <font color="#008800">TextArea control</font> to include basic HTML markup in your text.]]></mx:htmlText>

</mx:TextArea>

1. TextInput

<mx:TextInput id="src" text="Hello World!"/>

<mx:Button label="Copy Text" click="dest.text = src.text"/>

<mx:TextInput id="dest"/>

1. RichTextEditor

<mx:RichTextEditor id="rte" title="RichTextEditor" height="75%" text="Enter text into the RichTextEditor control, then click a button to display your text as plain text, or as HTML-formatted text."/> <mx:TextArea id="rteText" width="80%" height="25%"/>

<mx:Button label="Show Plain Text" click="rteText.text=rte.text;"/>

<mx:Button label="Show HTML Markup" click="rteText.htmlText=rte.htmlText;"/>

1. 数据验证类

Validators

Credit card validator, Currency validator,Data validator,Email validator,Number validator,Phone number validator,Regexpvalidator, Social security validator,String validator,Validator,Zipcode validator

. Formatters

Currency Formatter,Date Formatter,Formatter,Number Formatter,Phone Formatter,Switch Symbol Formatter,Zipcode Formatter

1. EmailValidator

<mx:EmailValidator source="{email}" property="text"

trigger="{myButton}" triggerEvent="click"

valid="Alert.show('Validation Succeeded!');"/>

<mx:TextInput id="email" width="100%"/>

<mx:Button id="myButton" label="Validate" />

1. DateValidator

<mx:DateValidator source="{dob}" property="text" allowedFormatChars="/"

trigger="{myButton}" triggerEvent="click" valid="Alert.show('Validation Succeeded!');"/>

<mx:TextInput id="dob" width="100%"/>

<mx:Button id="myButton" label="Validate" />

1. PhoneNumberValidator

<mx:PhoneNumberValidator source="{phone}" property="text"

trigger="{myButton}" triggerEvent="click" valid="Alert.show('Validation Succeeded!');"/>

<mx:TextInput id=" phone " width="100%"/>

<mx:Button id="myButton" label="Validate" />

1. SocialSecurityValidator社会保险号

<mx:SocialSecurityValidator source="{ssn}" property="text"

trigger="{myButton}" triggerEvent="click" valid="Alert.show('Validation Succeeded!');"/>

<mx:TextInput id=" ssn " width="100%"/>

<mx:Button id="myButton" label="Validate" />

1. CurrencyValidator货币验证

<mx:CurrencyValidator source="{priceUS}" property="text" precision="2" //精确度

trigger="{myButton}" triggerEvent="click" valid="Alert.show('Validation Succeeded!');"/>

<mx:TextInput id=" priceUS " width="100%"/>

<mx:Button id="myButton" label="Validate" />

1. NumberValidator

<mx:NumberValidator source="{age}" property="text" integerError="Enter Integer value"

minValue="18" maxValue="50" domain="int" trigger="{myButton}" triggerEvent="click"

valid="Alert.show('Validation Succeeded!');"/>

<mx:TextInput id=" age " width="100%"/>

<mx:Button id="myButton" label="Validate" />

1. StringValidator

<mx:StringValidator source="{fname}" property="text"

tooShortError="This string is shorter than the minimum allowed length of 4. "

tooLongError="This string is longer than the maximum allowed length of 20."

minLength="4" maxLength="20" trigger="{myButton}" triggerEvent="click"

valid="Alert.show('Validation Succeeded!');"/>

<mx:TextInput id=" fname " width="100%"/>

<mx:Button id="myButton" label="Validate" />

1. ZipCodeValidator

<mx:ZipCodeValidator source="{zip}" property="text"

trigger="{myButton}" triggerEvent="click" valid="Alert.show('Validation Succeeded!');"/>

<mx:TextInput id=" zip " width="100%"/>

<mx:Button id="myButton" label="Validate" />

1. CreditCardValidator银行卡的校验

<mx:CreditCardValidator id="ccV"

cardTypeSource="{creditcard}" cardTypeProperty="cardType"

cardNumberSource="{creditcard}" cardNumberProperty="cardNumber"

trigger="{myButton}" triggerEvent="click" cardTypeListener="{cardTypeCombo}"

cardNumberListener="{cardNumberInput}"valid="Alert.show('Validation Succeeded!');"/>

<mx:Model id="creditcard">

<card>

<cardType>{cardTypeCombo.selectedItem.data}</cardType>

<cardNumber>{cardNumberInput.text}</cardNumber>

</card>

</mx:Model>

<mx:Form id="creditCardForm">

<mx:FormItem label="Card Type">

<mx:ComboBox id="cardTypeCombo">

<mx:dataProvider>

<mx:Object label="American Express" data="American Express"/>

<mx:Object label="Diners Club" data="Diners Club"/>

<mx:Object label="Discover" data="Discover"/>

<mx:Object label="MasterCard" data="MasterCard"/>

<mx:Object label="Visa" data="Visa"/>

</mx:dataProvider>

</mx:ComboBox>

</mx:FormItem>

<mx:FormItem label="Credit Card Number">

<mx:TextInput id="cardNumberInput"/>

</mx:FormItem>

<mx:FormItem>

<mx:Button id="myButton" label="Check Credit"/>

</mx:FormItem>

</mx:Form>

1. Regexpvalidator正则的校验

<mx:RegExpValidator id="regExpV"

source="{regex\_text}" property="text" flags="g" expression="{regex.text}"

valid="handleResult(event)" invalid="handleResult(event)"

trigger="{myButton}" triggerEvent="click"/>

<mx:FormItem label="Enter text: ">

<mx:TextInput id="regex\_text" text="xxxxABC4xxx" width="100%"/>

</mx:FormItem>

<mx:FormItem label="Enter regular expression: ">

<mx:TextInput id="regex" text="ABC\d" width="100%"/>

</mx:FormItem>

<mx:FormItem label="Results: ">

<mx:TextInput id="reResults" width="100%"/>

</mx:FormItem>

<mx:FormItem >

<mx:Button id="myButton" label="Validate"/>

</mx:FormItem>

private function handleResult(eventObj:ValidationResultEvent):void {

if (eventObj.type == ValidationResultEvent.VALID)

{var xResult:RegExpValidationResult;

reResults.text="";//匹配成功的结果

for (var i:uint = 0; i < eventObj.results.length; i++)

{xResult = eventObj.results[i];

reResults.text=reResults.text + xResult.matchedIndex + " " +

xResult.matchedString + "\n";}}else{reResults.text="";

}

}

1. Number Formatte格式化

<mx:Script>

<![CDATA[

import mx.events.ValidationResultEvent;

private var vResult:ValidationResultEvent;

private function Format():void

{vResult = numVal.validate();

if (vResult.type==ValidationResultEvent.VALID) {

formattedNumber.text= numberFormatter.format(inputVal.text);

}

else {

formattedNumber.text= "";

}

}

]]>

</mx:Script>

<mx:NumberFormatter id="numberFormatter" precision="4"

useThousandsSeparator="true" useNegativeSign="true"/>

<mx:NumberValidator id="numVal" source="{inputVal}" property="text"

allowNegative="true" domain="real"/>

<mx:Panel title="NumberFormatter Example" width="75%" height="75%"

paddingTop="10" paddingLeft="10" paddingRight="10" paddingBottom="10">

<mx:Form>

<mx:FormItem label="Enter number:">

<mx:TextInput id="inputVal" text="" width="50%"/>

</mx:FormItem>

<mx:FormItem label="Formatted number (precision=4): ">

<mx:TextInput id="formattedNumber" editable="false" width="50%"/>

</mx:FormItem>

<mx:FormItem>

<mx:Button label="Validate and Format" click="Format();"/>

</mx:FormItem>

</mx:Form>

</mx:Panel>

1. PhoneFormatter

<mx:PhoneFormatter id="phoneFormatter"

formatString="(###) ###-####" validPatternChars="#-() "/>

<mx:PhoneNumberValidator id="pnVal" source="{phone}" property="text"

allowedFormatChars=""/>

1. SwitchSymbolFormatter

var switcher:SwitchSymbolFormatter=new SwitchSymbolFormatter('#');

formattedSCNumber.text = switcher.formatValue("Formatted Social Securty number: ###-##-#### ", scNum.text);

1. ZipCodeFormatter

<mx:ZipCodeFormatter id="zipFormatter" formatString="#####-####"/>

<mx:ZipCodeValidator id="zcVal" source="{zip}" property="text" allowedFormatChars=""/>

private function Format():void

{vResult = zcVal.validate();

if (vResult.type==ValidationResultEvent.VALID) {

formattedZipcode.text= zipFormatter.format(zip.text);

}

else {

formattedZipcode.text= "";

}

}

1. CurrencyFormatter金融货币

<mx:CurrencyFormatter id="usdFormatter" precision="2"

currencySymbol="$" decimalSeparatorFrom="."

decimalSeparatorTo="." useNegativeSign="true"

useThousandsSeparator="true" alignSymbol="left"/>

<mx:NumberValidator id="numVal" source="{priceUS}" property="text"

allowNegative="true" domain="real"/>

1. DateFormatter日期

<mx:DateFormatter id="dateFormatter" formatString="month: MM, day: DD, year: YYYY"/>

<mx:DateValidator id="dateVal" source="{dob}" property="text" inputFormat="mm/dd/yyyy"/>

private function Format():void

{ vResult = dateVal.validate();

if (vResult.type==ValidationResultEvent.VALID) {

formattedDate.text=dateFormatter.format(dob.text); }

else { formattedDate.text= "";}

}

## 第八章 行为类和特效类

行为类和特效类:

Effects:

Add Item Action Effect, Animate Property,Blur,Dissolve,Effect,Fade,Glow,Iris,Move,Parallel,Pause,Remove Item Action Effect,Resize,Rotate,Sequence, Sound effect,Wipedown,Wipeleft,Wiperight,Wipeup,Zoom

View states:

State

Transitions:

Transition

1. 模糊组件Blur

<mx:Blur id="blurImage" duration="1000" blurXFrom="0.0" blurXTo="10.0" blurYFrom="0.0" blurYTo="10.0"/>

<mx:Blur id="unblurImage" duration="1000" blurXFrom="10.0" blurXTo="0.0" blurYFrom="10.0"blurYTo="0.0"/> paddingRight="10" paddingBottom="10">

<mx:Image id="flex" source="@Embed(source='../assets/Nokia6630.png')"

mouseDownEffect="{blurImage}" mouseUpEffect="{unblurImage}"/>

1. Fade淡入淡出

<mx:Fade id="fadeOut" duration="1000" alphaFrom="1.0" alphaTo="0.0"/>

<mx:Fade id="fadeIn" duration="1000" alphaFrom="0.0" alphaTo="1.0"/>

<mx:Label text="Nokia 9930"

fontFamily="MyMyriad" fontSize="14"

visible="{cb1.selected}"

hideEffect="{fadeOut}" showEffect="{fadeIn}"/>

<mx:Image source="@Embed(source='../assets/Nokia\_6630.png')"

visible="{cb1.selected}"

hideEffect="{fadeOut}" showEffect="{fadeIn}"/>

<mx:CheckBox id="cb1" label="visible" selected="true"/>

1. Glow发光效果

<mx:Glow id="glowImage" duration="1000"

alphaFrom="1.0" alphaTo="0.3" blurXFrom="0.0" blurXTo="50.0"

blurYFrom="0.0" blurYTo="50.0" color="0x00FF00"/>

<mx:Glow id="unglowImage" duration="1000"

alphaFrom="0.3" alphaTo="1.0" blurXFrom="50.0" blurXTo="0.0"

blurYFrom="50.0" blurYTo="0.0" color="0x0000FF"/>

<mx:Image source="@Embed(source='../assets/Nokia\_6630.png')"

mouseDownEffect="{glowImage}" mouseUpEffect="{unglowImage}"/>

1. Iris彩虹效果,变大变小

<mx:Iris id="irisOut" duration="1000" showTarget="true"/>

<mx:Iris id="irisIn" duration="1000" showTarget="false"/>

<mx:Image id="flex" source="@Embed(source='../assets/Nokia\_6630.png')"

visible="{cb1.selected}" showEffect="{irisIn}" hideEffect="{irisOut}"/>

<mx:CheckBox id="cb1" label="visible" selected="true"/>

1. Dissolve溶解效果

<mx:Dissolve id="dissolveOut" duration="1000" alphaFrom="1.0" alphaTo="0.0"/>

<mx:Dissolve id="dissolveIn" duration="1000" alphaFrom="0.0" alphaTo="1.0"/>

<mx:Image source="@Embed(source='../assets/Nokia\_6630.png')"

visible="{cb1.selected}" hideEffect="{dissolveOut}" showEffect="{dissolveIn}"/>

1. Move效果

<mx:Canvas id="canvas" width="100%" height="100%" mouseDown="moveImage();">

<mx:Image id="img" source="@Embed(source='../assets/Nokia\_6630.png')"/>

</mx:Canvas>

<mx:Move id="myMove" target="{img}"/>

private function moveImage():void {

myMove.end();

myMove.xTo=mouseX-60;

myMove.play();

}

1. Resize从新绘制大小

<mx:Resize id="expand" target="{img}" widthTo="100" heightTo="200"/>

<mx:Resize id="contract" target="{img}" widthTo="30" heightTo="60"/>

<mx:Image id="img" width="30" height="60" source="@Embed(source='../assets/Nokia\_6630.png')"/> <mx:ControlBar>

<mx:Button label="Expand" click="expand.end(); expand.play();"/>

<mx:Button label="Contract" click="contract.end(); contract.play();"/>

</mx:ControlBar>

1. Rotate旋转

<mx:Rotate id="rotate" angleFrom="{angle-45}" angleTo="{angle}" target="{myVB}"/>

<mx:VBox id="myVB" width="50%" horizontalAlign="center">

<mx:Label text="Nokia 9930" fontFamily="MyMyriad" fontSize="14"/>

<mx:Image id="img" source="@Embed(source='../assets/Nokia\_6630.png')"/>

</mx:VBox>

<mx:Button label="Rotate 45 Degrees" click="rotateImage();"/>

1. Zoom缩放

<mx:Image id="img" source="@Embed(source='../assets/Nokia\_6630.png')" scaleX=".5" scaleY=".5" rollOver="doZoom(event)" rollOut="doZoom(event)"/>

<mx:Zoom id="zoomAll" zoomWidthTo="1" zoomHeightTo="1" zoomWidthFrom=".5" zoomHeightFrom=".5" />

public function doZoom(event:MouseEvent):void {

if (zoomAll.isPlaying) {

zoomAll.reverse();

} else {

zoomAll.play([event.target], event.type == MouseEvent.ROLL\_OUT ? true : false);

}

}

1. Sound特效声音

<mx:Image id="flex" source="@Embed(source='../assets/Nokia\_6630.png')"

mouseDownEffect="{mySounds}"/>

<mx:SoundEffect id="mySounds" source="@Embed(source='../assets/jazz.mp3')"/>

1. WipeRight右擦除(还有左擦除,上擦除,下擦除)用法一样这里就介绍一个.

<mx:Image source="@Embed(source='../assets/Nokia\_6630.png')"

visible="{cb1.selected}" hideEffect="{wipeOut}" showEffect="{wipeIn}"/>

<mx:CheckBox id="cb1" label="visible" selected="true"/>

<mx:WipeRight id="wipeOut" duration="1000"/>

<mx:WipeRight id="wipeIn" duration="1000"/>

1. AnimateProperty拉伸动画效果

<mx:Image id="flex" source="@Embed(source='../assets/Nokia6630.png')"

mouseDownEffect="{animateScaleXUpDown}"/>

<mx:Sequence id="animateScaleXUpDown" >

<mx:AnimateProperty property="scaleX" fromValue="1" toValue="1.5" duration="1000" />

<mx:AnimateProperty property="scaleX" fromValue="1.5" toValue="1" duration="1000" />

</mx:Sequence>

1. Sequence

<mx:Sequence id="movePauseMove">

<mx:Move xBy="150" duration="2000" easingFunction="Bounce.easeOut"/>

<mx:Pause duration="2000"/>

<mx:Move xBy="-150" duration="2000" easingFunction="Bounce.easeIn"/>

</mx:Sequence>

<mx:Image source="@Embed(source='../assets/Nokia\_6630.png')" mouseDownEffect="{movePauseMove}"/>

1. State通常和transitions一起使用

<mx:states>

<mx:State name="Register">

<mx:AddChild relativeTo="{loginForm}" position="lastChild">

<mx:target>

<mx:FormItem id="confirm" label="Confirm:">

<mx:TextInput/>

</mx:FormItem>

</mx:target>

</mx:AddChild>

<mx:SetProperty target="{loginPanel}" name="title" value="Register"/>

<mx:SetProperty target="{loginButton}" name="label" value="Register"/>

<mx:SetStyle target="{loginButton}" name="color" value="blue"/>

<mx:RemoveChild target="{registerLink}"/>

<mx:AddChild relativeTo="{spacer1}" position="before">

<mx:target>

<mx:LinkButton id="loginLink" label="Return to Login" click="currentState=''"/>

</mx:target>

</mx:AddChild>

</mx:State>

<mx:State name="newstate">

<mx:SetProperty target="{loginPanel}" name="height" value="287"/>

<mx:RemoveChild target="{formitem1}"/>

<mx:AddChild relativeTo="{loginForm}" position="lastChild">

<mx:FormItem label="输入邮箱：" fontSize="12">

<mx:TextInput/>

</mx:FormItem>

</mx:AddChild>

<mx:AddChild relativeTo="{loginForm}" position="lastChild" >

<mx:FormItem label="Label">

<mx:Button label="Button" id="newBtn"/>

</mx:FormItem>

</mx:AddChild>

<mx:SetProperty target="{loginPanel}" name="title" value="newstate"/>

<mx:SetEventHandler target="{registerLink}" name="click" handler="currentState=''"/>

<mx:SetEventHandler target="{newBtn}" name="click" handlerFunction="newClick"/>

</mx:State>

</mx:states>

<mx:LinkButton id="registerLink" label="Need to Register?" click="currentState='newstate'"/>

1. transitions

<mx:transitions>

<mx:Transition id="toRegister" fromState="\*" toState="Register">

<mx:Sequence targets="{[loginPanel, registerLink, confirm, loginLink, spacer1]}">

<mx:RemoveChildAction/>

<mx:SetPropertyAction target="{loginPanel}" name="title"/>

<mx:Pause duration="2000"/>

<mx:SetPropertyAction target="{loginButton}" name="label"/>

<mx:Pause duration="2000"/>

<mx:SetStyleAction target="{loginButton}" name="color"/>

<mx:Pause duration="2000"/>

<mx:Resize target="{loginPanel}" duration="3000"/>

<mx:Pause duration="2000"/>

<mx:AddChildAction/>

</mx:Sequence>

</mx:Transition>

## 第九章 数据可视化组件

Chart Control:

Area Chart,Axis Renderer,Bar Chart,Bubble Chart,Candle Stick Chart,CategoryAxis,Column Chart,

Date Time Axis,Gridlines, HLoc chart,Legend,Linear Axis ,Line Chart,Logaxis,Pie Chart,Plot Chart

Chart Series:

Area Series,Bar Series,Bubble Series, Candle Stick Series,

Column Series, HLoc Series ,Line Series,Pie Series, Plot Series

Chart Effects:

Series Inter Polate,Series Slide,Series Zoom

1. Area Chart

import mx.collections.ArrayCollection;

定义数据源

[Bindable]

private var expensesAC:ArrayCollection = new ArrayCollection( [

{ Month: "Jan", Profit: 2000, Expenses: 1500, Amount: 450 },

{ Month: "Feb", Profit: 1000, Expenses: 200, Amount: 600 },

{ Month: "Mar", Profit: 1500, Expenses: 500, Amount: 300 },

{ Month: "Apr", Profit: 1800, Expenses: 1200, Amount: 900 },

{ Month: "May", Profit: 2400, Expenses: 575, Amount: 500 } ]);

]]>

定义颜色(填充的时候用)

<mx:SolidColor id="sc1" color="blue" alpha=".3"/>

<mx:SolidColor id="sc2" color="red" alpha=".3"/>

<mx:SolidColor id="sc3" color="green" alpha=".3"/>

自定义线的方式

<mx:Stroke id = "s1" color="blue" weight="2"/>

<mx:Stroke id = "s2" color="red" weight="2"/>

<mx:Stroke id = "s3" color="green" weight="2"/>

<mx:LineChart id="linechart" height="100%" width="45%" paddingLeft="5" paddingRight="5"

showDataTips="true" dataProvider="{expensesAC}">

<mx:horizontalAxis> <mx:CategoryAxis categoryField="Month"/> </mx:horizontalAxis>

<mx:series>

<mx:LineSeries yField="Profit" form="curve" displayName="Profit" lineStroke="{s1}"/>

<mx:LineSeries yField="Expenses" form="curve" displayName="Expenses" lineStroke="{s2}"/>

<mx:LineSeries yField="Amount" form="curve" displayName="Amount" lineStroke="{s3}"/>

</mx:series>

</mx:LineChart>

<mx:Legend dataProvider="{linechart}"/>//图例

<mx:AreaChart id="Areachart" height="100%" width="45%" paddingLeft="5" paddingRight="5"

showDataTips="true" dataProvider="{expensesAC}">

<mx:horizontalAxis><mx:CategoryAxis categoryField="Month"/> </mx:horizontalAxis>

<mx:series>

<mx:AreaSeries yField="Profit" form="curve" displayName="Profit" areaStroke="{s1}" areaFill="{sc1}"/>

<mx:AreaSeries yField="Expenses" form="curve" displayName="Expenses" areaStroke="{s2}" areaFill="{sc2}"/>

<mx:AreaSeries yField="Amount" form="curve" displayName="Amount" areaStroke="{s3}" areaFill="{sc3}"/>

</mx:series>

</mx:AreaChart>

<mx:Legend dataProvider="{Areachart}"/>

1. HLoc Series

<mx:HLOCChart id="hlocchart" height="100%" width="100%"

paddingRight="5" paddingLeft="5" showDataTips="true" dataProvider="{stockDataAC}">

<mx:verticalAxis>

<mx:LinearAxis id="vaxis" baseAtZero="false" title="Price"/>

</mx:verticalAxis>

<mx:horizontalAxis>

<mx:CategoryAxis id="haxis" categoryField="Date" title="Date"/>

</mx:horizontalAxis>

<mx:horizontalAxisRenderers>

<mx:AxisRenderer axis="{haxis}" canDropLabels="true"/>

</mx:horizontalAxisRenderers>

<mx:series>

<mx:HLOCSeries openField="Open" highField="High" lowField="Low"

closeField="Close" stroke="{s1}" openTickStroke="{s1}" closeTickStroke="{s1}"

openTickLength="7" closeTickLength="7"/>

</mx:series>

</mx:HLOCChart>

1. Bar Series

和我们常用的一样

1. Bubble Series

气泡的样子图例

1. Candle Stick Series

蜡烛图

1. Date Time Axis

日期坐标

1. Line Series

正常的线

1. Gridlines

背景带格

1. Logaxis

对数的

1. Pie Chart

饼图

1. Plot Chart

用一些离散的点来表示

1. Series Inter Polate

特效，缓慢的变换

1. Series Slide

数据先隐藏，在出来

1. Series Zoom

有个缩放的过程

## 第十章 其它组件

1. Repeat组件

重复的时候用

1. Image

图片的时候用embed元数据

1. Swfloader

加载swf

1. VideoDisplay

播放flv(这样的格式文件小，adobe公司发布的)

1. Print

打印的一些操作

## 第十一章 FABridge技术详解

1. Flex-Ajax Bridge技术是Adobe公司提供的一种服务技术。通过Flex-Ajax Bridge技术可达使用JavaScript控制Flex应用程序。
2. Flex-Ajax Bridge技术包括两个文件FABridge.as和FABridge.js文件。FABridge.as文件中定义了Flex客户端的各种属性和方法。FABridge.js文件定义了JavaScript客户端的各种属性和方法。

D:\Program Files\Adobe\Flex Builder 3\sdks\3.0.0\frameworks\javascript\fabridge\src\bridge

1. 如何嵌套swf文件？

是通过<object>的方式。

1. 如何获得flex应用程序？

flashvars='bridgeName=example'这个名字是应用的名字,我们可以采取

var flexApp = FABrige.example.root();flexApp.getSlider().getValue

是通过上面的方式得到组件和相应的属性。

命名大小写敏感

使用getCompenentID()获取组件

使用getPropertyName()获取属性值

使用setPropertyName()设置属性值

1. Js动态创建组件例子？
2. 动态添加一个DataGrid？

var flexApp = FABridge.example.root();

var grid = FABridge.example.create("mx.controls.DataGrid");

var col1 = FABridge.example.create("mx.controls.dataGridClasses.DataGridColumn");

col1.setDataField("apples");

var col2 = FABridge.example.create("mx.controls.dataGridClasses.DataGridColumn");

col2.setDataField("oranges");

grid.setColumns( [col1, col2] );

grid.setWidth(300);

grid.setDataProvider( [ { apples: 12, oranges: 32 }, { apples: 14, oranges:21 } ] );

flexApp.getPanel().addChild(grid);

grid.addEventListener("change",function(event){trace("apples: " + event.getTarget().getSelectedItem().apples); } );

1. 动态添加一个图表？

var flexApp = FABridge.example.root();

var chart = FABridge.example.create("mx.charts.ColumnChart");

chart.setName("chart");

var s1 = FABridge.example.create("mx.charts.series.ColumnSeries");

s1.setYField("apples");

var s2 = FABridge.example.create("mx.charts.series.ColumnSeries");

s2.setYField("oranges");

chart.setSeries( [s1, s2] );

chart.setWidth(300);

chart.setHeight(200);

flexApp.getPanel().addChild(chart);

1. 给图表增加特效？

var flexApp = FABridge.example.root();

var chart = flexApp.getPanel().getChildByName("chart");

var dp = [];

for(var i=0; i < 30; i++) {

dp.push( {apples: Math.random()\*100, oranges: Math.random()\*100} );

}

chart.setDataProvider(dp);

var flexApp = FABridge.example.root();

var chart = flexApp.getPanel().getChildByName("chart");

var series = chart.getSeries();

var effect = FABridge.example.create("mx.charts.effects.SeriesZoom");

effect.setMinimumElementDuration(300);

series[0].setStyle("hideDataEffect",effect);

series[0].setStyle("showDataEffect",effect);

series[1].setStyle("hideDataEffect",effect);

series[1].setStyle("showDataEffect",effect);

1. 绘制矩形？用到精灵（精灵就是有动画效果）

var flexApp = FABridge.example.root();

var spr = FABridge.example.create("flash.display.Sprite");

trace(spr);

var g = spr.getGraphics();

g.beginFill(0xFF0000);

g.lineStyle(2,2);

g.drawRect(-100,-100,200,200);

g.endFill();

flexApp.addChild(spr);

spr.setX(300);

spr.setY(300);

var flexApp = FABridge.example.root();

var spr = FABridge.example.create("flash.display.Sprite");

trace(spr);

var g = spr.getGraphics();

g.beginFill(0xFF0000);

g.lineStyle(2,2);

g.drawRect(-100,-100,200,200);

g.endFill();

flexApp.addChild(spr);

spr.setX(300);

spr.setY(300);

var speed = Math.random() \* 13;

flexApp.getStage().addEventListener("enterFrame", function(e) {

spr.setRotation(spr.getRotation() + speed);

});

<fab:FABrige xmlns=”brige.\*” />

## 第十二章 DataGrid高级应用

1. 可编辑的表格,放置组件editable, itemEditor

<mx:ArrayCollection id="arrColl">

<mx:source>

<mx:Array>

<mx:Object label="中国" score="1" />

<mx:Object label="美国" score="0" />

<mx:Object label="韩国" score="0" />

<mx:Object label="肯尼亚" score="0" />

</mx:Array>

</mx:source>

</mx:ArrayCollection>

<mx:DataGrid id="dataGrid"dataProvider="{arrColl}" editable="true"

rowCount="8" fontSize="12" fontFamily="Arial" width="324" height="334">

<mx:columns>

<mx:DataGridColumn dataField="label" headerText="国家" editable="false" />

<mx:DataGridColumn dataField="score" headerText="金牌数" editable="true"

editorDataField="value" itemEditor="mx.controls.NumericStepper"

labelFunction="labelFun" />

</mx:columns>

</mx:DataGrid>

1. itemRenderer

自定义单元的渲染

<mx:Script>

<![CDATA[

import mx.core.ClassFactory;

public function initCellEditor():void {

myList.itemRenderer=new ClassFactory(LinkRenderer);

}

]]>

</mx:Script>

<mx:List id="myList" variableRowHeight="true" height="180" width="250"

backgroundColor="white" initialize="initCellEditor();" >

<mx:dataProvider>

<mx:Object label="Alaska" data="Juneau" webPage="http://www.state.ak.us/"/>

<mx:Object label="Alabama" data="Montgomery" webPage="http://www.alabama.gov/" />

<mx:Object label="Arkansas" data="Little Rock" webPage="http://www.state.ar.us/"/>

</mx:dataProvider>

</mx:List>

LinkRender

<mx:HBox xmlns:mx="http://www.adobe.com/2006/mxml" width="140"

height="32" horizontalAlign="center" verticalAlign="middle" initialize="init();">

<mx:LinkButton label="{data.label}" themeColor="#03FA09" color="#B50AF4"/>

</mx:HBox>

用系统的组件

<mx:DataGridColumn dataField="score" headerText="产生金牌数"editable="true" itemRenderer="mx.controls.TextInput" />

1. labelFunction格式化数据的输出

<mx:DataGridColumn dataField="score" headerText="金牌数" editable="true"

editorDataField="value" itemEditor="mx.controls.NumericStepper"

labelFunction="labelFun" />

internal function labelFun(item:Object,column:DataGridColumn):String

{column.editable=false;

if (item.hasOwnProperty("score"))

return item.score+"枚";

else

return item.score=null;

}

1. sortCompareFunction排序

<mx:DataGrid id="dataGrid" dataProvider="{arrColl}">

<mx:columns>

<mx:DataGridColumn dataField="index" sortCompareFunction="index\_sortCompareFunc" />

<mx:DataGridColumn dataField="value" sortCompareFunction="value\_sortCompareFunc" />

</mx:columns>

</mx:DataGrid>

private function index\_sortCompareFunc(itemA:Object, itemB:Object):int {

if (!itemA.hasOwnProperty("index")) { itemA.index = null; }

if (!itemB.hasOwnProperty("index")) { itemB.index = null; }

return ObjectUtil.numericCompare(itemA.index, itemB.index);

}

private function value\_sortCompareFunc(itemA:Object, itemB:Object):int {

if (!itemA.hasOwnProperty("value")) {itemA.value = null; }

if (!itemB.hasOwnProperty("value")) { itemB.value = null; }

return ObjectUtil.stringCompare(itemA.value, itemB.value, checkBox.selected);

}

1. 排序

<mx:DataGrid id="dataGrid" dataProvider="{xmlListColl}" width="300"

verticalScrollPolicy="on" sortableColumns="true">

<mx:columns>

<mx:DataGridColumn dataField="old" labelFunction="dataGridCol\_labelFunc"

sortCompareFunction="test2\_compareFunc" />

<mx:DataGridColumn dataField="new" labelFunction="dataGridCol\_labelFunc"

sortCompareFunction="test3\_compareFunc" />

</mx:columns>

</mx:DataGrid>  
private function dataGridCol\_labelFunc(item:XML, col:DataGridColumn):String {

return item.\*.(@name == col.dataField).text();

}

private function test2\_compareFunc(itemA:XML, itemB:XML):int {

var valueA:String = itemA.test2.text();

var valueB:String = itemB.test2.text();

return ObjectUtil.stringCompare(valueA, valueB);

}

private function test3\_compareFunc(itemA:XML, itemB:XML):int {

var valueA:String = itemA.test3.text();

var valueB:String = itemB.test3.text();

return ObjectUtil.stringCompare(valueA, valueB);

}

1. 动态排序

private function sortOld():void {

var sortField:SortField = new SortField();

sortField.compareFunction = test2\_compareFunc;

sortField.descending = checkBox.selected;

var oldSort:Sort = new Sort();

oldSort.fields = [sortField];

xmlListColl.sort = oldSort;

xmlListColl.refresh();

}

private function sortNew():void {

var sortField:SortField = new SortField();

sortField.compareFunction = test3\_compareFunc;

sortField.descending = checkBox.selected;

var sort:Sort = new Sort();

sort.fields = [sortField];

xmlListColl.sort = sort;

xmlListColl.refresh();

}

private function resetSort():void {

xmlListColl.sort = null;

xmlListColl.refresh();

}

1. 数据过滤

通过对数据源进行过滤，实现对datagrid的过滤。解决的要点是如何对每一行数据的每一个字段进行过滤.

<FilterControl xmlns:mx="http://www.adobe.com/2006/mxml"

xmlns="com.custom.ctrls.\*"

layout="vertical" applicationComplete="init()">

<mx:HBox width="100%">

<mx:Label text="Filter:"/>

<mx:TextInput id="keyword" width="100%"/>

<mx:ComboBox id="cmbCol" dataProvider="{dpColumns}"/>

</mx:HBox>

<mx:HBox width="100%" horizontalAlign="right">

<mx:Button label="Filter" id="btnFilter" click="filterResults()" width="150"/>

<mx:Button label="Reset" click="resetFilter()" width="150"/>

</mx:HBox>

<mx:DataGrid width="100%" height="100%" dataProvider="{dpRows}" variableRowHeight="true">

<mx:columns>

<mx:DataGridColumn dataField="type" headerText="Type" width="80"/>

<mx:DataGridColumn dataField="name" headerText="Name" width="180"/>

<mx:DataGridColumn dataField="url" headerText="URL"/>

<mx:DataGridColumn dataField="desc" headerText="Description"/>

<mx:DataGridColumn dataField="tags" headerText="Tags" width="200"/>

</mx:columns>

</mx:DataGrid>

</FilterControl>

下面就是FilterControl

[Bindable]

public var dpColumns:ArrayCollection;

[Bindable]

public var dpRows:ArrayCollection;

public var keyword:TextInput;

public var btnFilter:Button;

public var cmbCol:ComboBox;

public function init():void {

dpColumns = new ArrayCollection();

dpColumns.addItem({data:'any', label:'Any'});

dpColumns.addItem({data:'type', label:'Type'});

dpColumns.addItem({data:'name', label:'Name'});

dpColumns.addItem({data:'url', label:'URL'});

dpColumns.addItem({data:'desc', label:'Descriptiom'});

dpColumns.addItem({data:'tags', label:'Tags'});

dpRows = new ArrayCollection();

var item:Object = new Object();

item = new Object();

item.type = "Tutorial";

item.name = "Quick Outline in FlexBuilder - Ctrl+O";

item.url = "http://www.franto.com/blog2/quick-outline-in-flexbuilder-ctrlo";

item.desc = "Speed up your development with Quick Outline in FlexBuilder";

item.tags = "Flex, AIR, Tutorial";

dpRows.addItem(item);

}

public function resetFilter():void {

keyword.text = "";

filterResults();

}

public function filterResults():void {

dpRows.filterFunction = \_sortRows;

dpRows.refresh();

}

private function \_sortRows(item:Object):Boolean {

var col:String = cmbCol.selectedItem.data as String;

var key:String = keyword.text;

key = key.toLowerCase();

if (key != ""){

if (col != "any"){

var value:String = item[col];

value = value.toLowerCase();

if (value.indexOf(key) >= 0) {

return true;

}

} else {

for (var o:String in item) {

value = item[o];

value = value.toLowerCase();

if (value.indexOf(key) >= 0) {

return true;

}

}

}

} else {

return true;

}

return false;

}

1. 导出为Excel

将DataGrid输出为对应的Html字符串，DataGrid对应为<table></table>，DataGridColumn的headerText属性对应<th></th>，DataGrid的一行对应一个<tr></tr>，DataGrid的一个单元格对应一个<td></td>

使用URLVariables 类和URLLoader类将字符串变量发送到服务器端脚本。

在服务器端我们只是简单地加一个文件头，再将Flex端发送的字符串在客户端打印出来。如果使用asp.net，请validateRequest设定为false；

contentType="application/vnd.ms-excel“

("Content-Disposition", "attachment;filename=test.xls");控制下载格式

## 第十三章 低数据量的数据通信

**1. 低数据量的数据通信**

1）HTTPService form表单的提交

<mx:HTTPService showBusyCursor="true" id="loginSrv" result="doResult();" method="GET"

url="http://localhost:800/BOOK/flexlogin.aspx">

<mx:request>

<username> {txtname.text} </username>

<userpassword> {txtpwd.text}</userpassword>

</mx:request>

</mx:HTTPService>

2）Web Service

<mx:WebService id="Service"

wsdl="http://www.webxml.com.cn/WebServices/TranslatorWebService.asmx?wsdl" useProxy="false">

<mx:operation name="getEnCnTwoWayTranslator" resultFormat="object"

result="getTrans\_result(event);" fault="getTrans\_fault(event);">

<mx:request>

<Word>{txtWord.text}</Word>

</mx:request>

</mx:operation>

</mx:WebService>

<mx:WebService id="LocalService"

wsdl="http://localhost:800/BOOK/flexwebservice.asmx?WSDL"

useProxy="false">

<mx:operation name="ReverseWord"resultFormat="object"

result="getReverse\_result(event);" fault="getReverse\_fault(event);">

<mx:request><word>{txtStr.text}</word></mx:request>

</mx:operation>

</mx:WebService>

3）URLLoader

URLLoader+URLRequest+URLVariables

看例子就可以了.

4）FileReference

**2.大数据量的数据通信**

1）AMFPHP

2）LiveCycle Data Services

## 第十四章 用BlazeDS数据通信