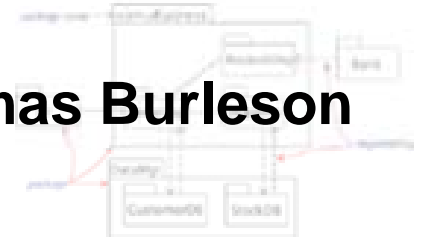




Thomas Burleson



**PRINCIPAL ARCHITECT
CERTIFIED INSTRUCTOR**

Adobe® Flex™



UNIVERSAL MIND

HIGH IMPACT CONSULTING

ThomasB@UniversalMind.com

<http://www.thomasburleson.biz/>

Win a Chumby!

www.universalmind.com/drawing.cfm

Flex & Cairngorm **Event Dispatching**

Solutions and Gotchas

Overview

Without MVC!

Review Event Delegation and Dispatching

Adobe Cairngorm MVC!

Overlay Adobe Cairngorm on standard apps

- 1) Understanding **FrontController**
- 2) Impacts of Adobe Cairngorm on views
- 3) Issues with Adobe Cairngorm

UM Cairngorm MVC!

Understanding UM Cairngorm

- 1) Issues & solutions with events
- 2) Impacts on views
- 3) Gotchas

Overview

Without MVC!

Review Event Delegation and Dispatching

Adobe Cairngorm MVC!

Overlay Adobe Cairngorm on standard apps

- 1) Understanding FrontController
- 2) Impacts of Adobe Cairngorm on views
- 3) Issues with Adobe Cairngorm

UM Cairngorm MVC!

Understanding UM Cairngorm

- 1) Issues & solutions with events
- 2) Impacts on views
- 3) Gotchas

Let us explore some concepts!

What is the event delegation model?

Why are events so cool?

What components can dispatch events?



Event Delegation Model:

Responses and reactions to activity is delegated to other components. Allows loose coupling for responsibilities and roles of components



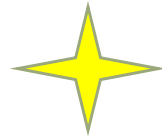
Events: Allows components to be loosely coupled

1. no callbacks to fire on internal activity
2. no “reach-backs” to gather current data/state

Custom events can be used to transport data to the delegates.

Events are everywhere: Flash, Flex, Cairngorm, etc.

Let us explore some concepts!



Many types of Events

1. Framework events: initialize, creationComplete,
2. User Events: click, mouseMove,
3. Custom view-level: hideContactDetails,
4. Business events: addContact, getContactDetails,
5. DataService events: ResultEvent, FaultEvent,



Which events are **asynchronous**:

1. Business events (if desired)
2. Remote DataService Calls respond asynchronously



UIComponents has **asynchronous phases**:

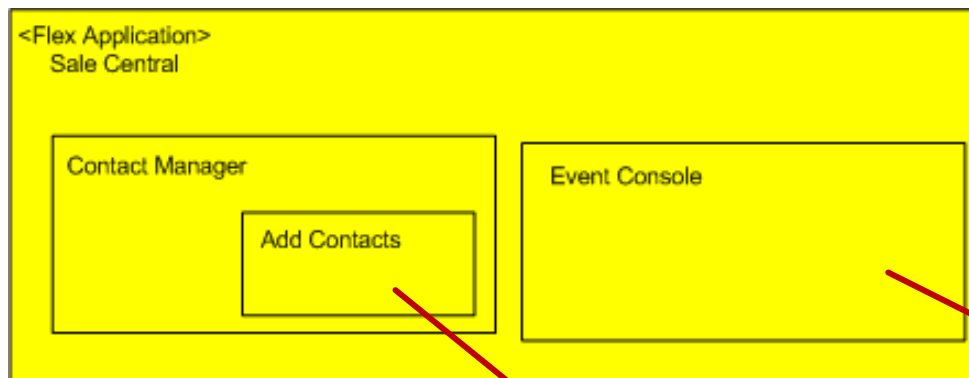
1. Occurs during construction
2. Occurs during invalidation

Many categories of events?

Are all events synchronous?

Why are UIComponents more challenging?

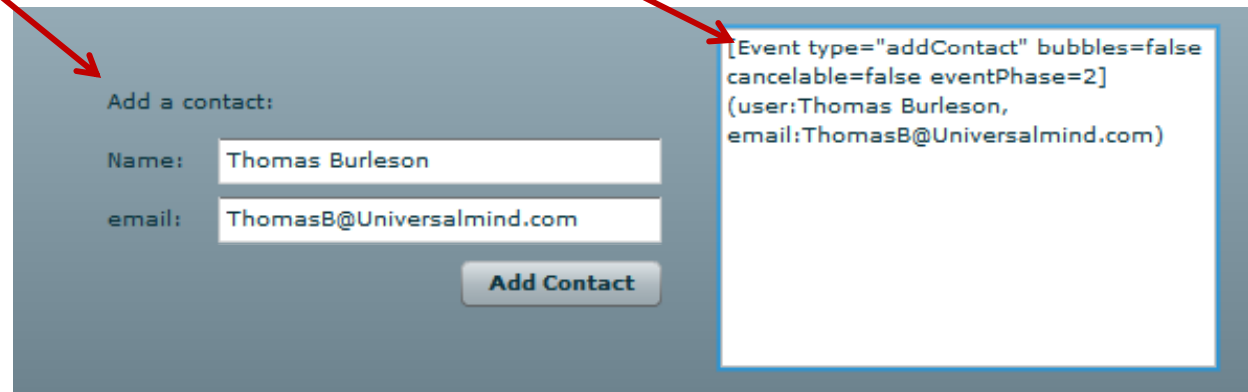
Consider “Sales Central”



<< view hierarchy >>

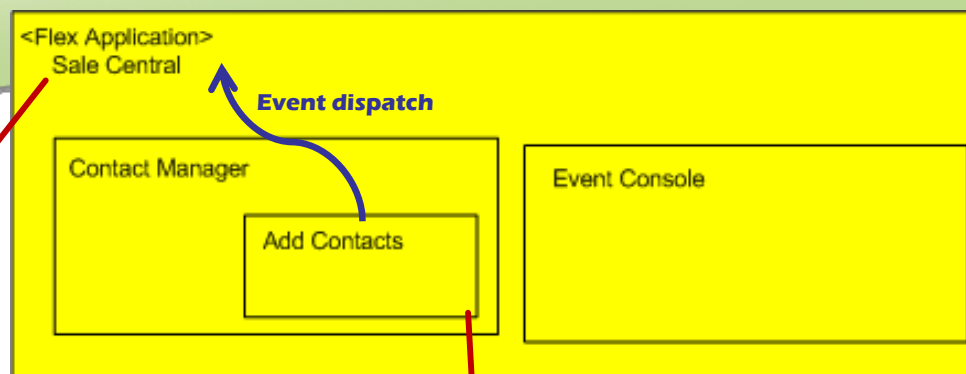
“Sales Central” is a

1. standard Flex application
2. with nested views & partitions of functionality.



The screenshot shows the "Add a contact:" form in the "Sales Central" application. The form has two input fields: "Name:" with the value "Thomas Burleson" and "email:" with the value "ThomasB@Universalmind.com". There is an "Add Contact" button. A red arrow points from the "Add Contacts" button in the view hierarchy diagram to this form. Another red arrow points from the "Event Console" in the view hierarchy diagram to a code block on the right.

```
[Event type="addContact" bubbles=false  
cancelable=false eventPhase=2]  
(user:Thomas Burleson,  
email:ThomasB@Universalmind.com)
```

Flex Development - Cairngorm_AddContact/src/Cairngorm_AddContact.mxml - Eclipse SDK

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application layout="absolute" xmlns:mx="http://www.adobe.com/2006/mxml" >
3
4   <mx:Script>
5     <![CDATA[
6       import com.clientx.salesCentral.control.events.AddContactEvent;
7     ]]>
8   </mx:Script>
9
10  <ns1:AddContact x="10" y="10" width="310" height="162"
11    addContact="txtConsole.text = String(event);"
12    xmlns:ns1="com.clientx.salesCentral.views.contacts.*" />
13
14  <mx:TextArea id="txtConsole"
15    y="10" height="162" right="20" left="338"/>
16
17 </mx:Application>
18
  
```

Flex Development - Cairngorm_AddContact/src/com/clientx/salesCentral/views/contacts/AddContact.mxml - Eclipse SDK

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml" width="100%" height="100%">
3
4   <mx:Metadata>
5     [Event(name='addContact', type='com.clientx.salesCentral.control.events.AddContactEvent')]
6   </mx:Metadata>
7
8   <mx:Script>
9     <![CDATA[
10       import com.clientx.salesCentral.control.events.AddContactEvent;
11
12       private function onAddContact():void {
13         var request : AddContactEvent = new AddContactEvent(txtUserName.text,txtEmail.text);
14         dispatchEvent(request);
15       }
16     ]]>
17   </mx:Script>
18
19   <mx:Label text="Add a contact:" x="33" y="27" />
20   <mx:TextInput id="txtUserName" y="53" right="10" left="88" />
21   <mx:TextInput id="txtEmail" y="81" left="88" right="10" />
22
23   <mx:Label x="33" y="83" text="email:" />
24   <mx:Label x="33" y="55" text="Name:" />
25
26   <mx:Button label="Add Contact" right="10" top="111" click="onAddContact();" />
27
28 </mx:Canvas>
29
30
  
```


Flash & Flex are based on **event delegation**:

All display classes in **flash.*** and **mx.*** are based on EventDispatcher functionality.

UIComponent

[Properties](#) | [Methods](#) | [Events](#) | [Styles](#) | [Effects](#) | [Constants](#)

Package
Class
Inheritance

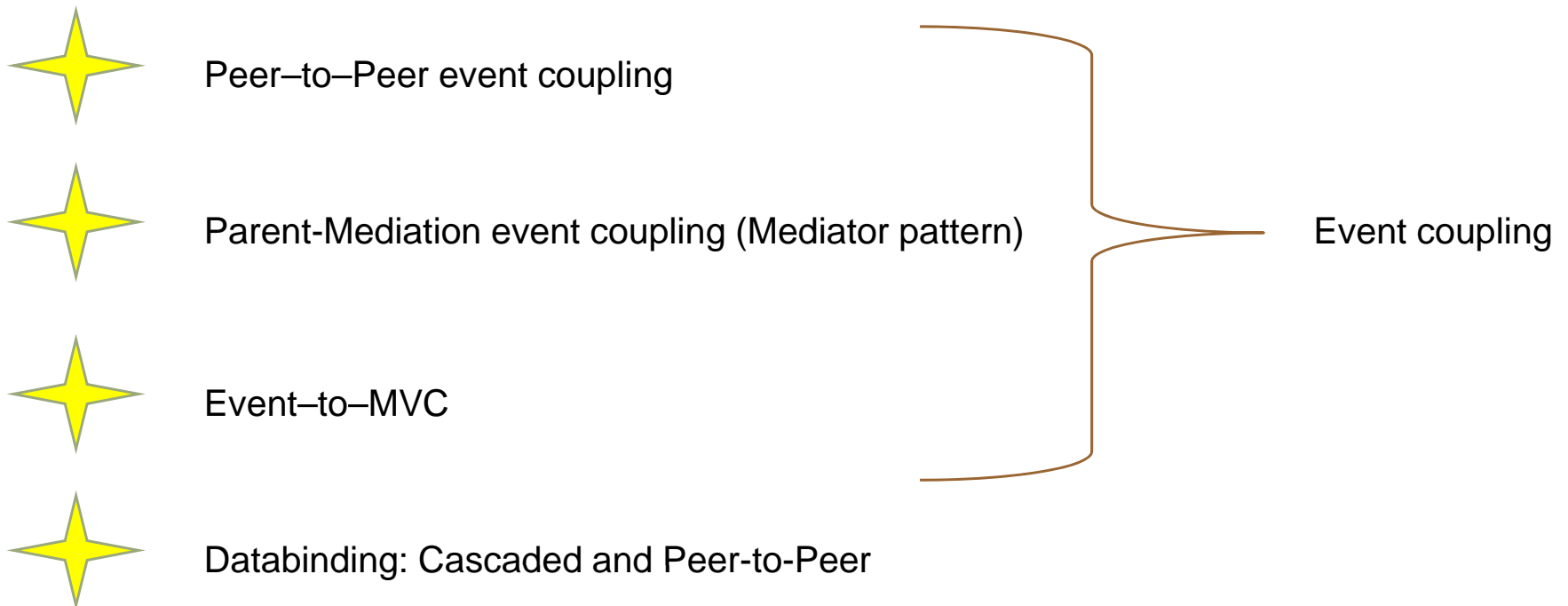
mx.core

public class UIComponent

UIComponent → [FlexSprite](#) → [Sprite](#) → [DisplayObjectContainer](#) → [InteractiveObject](#) → [DisplayObject](#)
→ [EventDispatcher](#) → [Object](#)

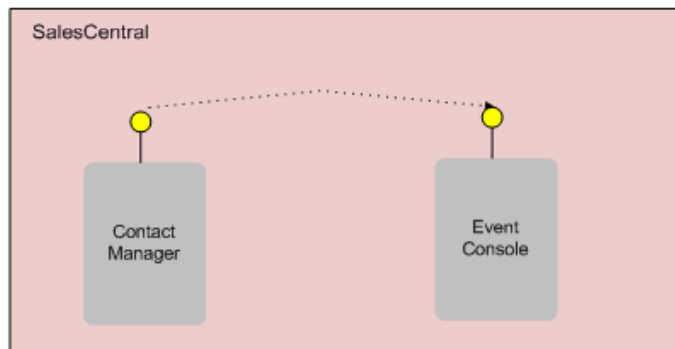
Method	Defined By
addEventListener (type:String, listener:Function, useCapture:Boolean = false, priority:int = 0, useWeakReference:Boolean = false):void Registers an event listener object with an EventDispatcher object so that the listener receives notification of an event.	IEventDispatcher
dispatchEvent (event:Event):Boolean Dispatches an event into the event flow.	IEventDispatcher
hasEventListener (type:String):Boolean Checks whether the EventDispatcher object has any listeners registered for a specific type of event.	IEventDispatcher
removeEventListener (type:String, listener:Function, useCapture:Boolean = false):void Removes a listener from the EventDispatcher object.	IEventDispatcher
willTrigger (type:String):Boolean Checks whether an event listener is registered with this EventDispatcher object or any of its ancestors for the specified event type.	IEventDispatcher

Use events to establish relationships (couple) between components. Four (4) types of component couplings:



Note: databinding uses code injection to configure events to invoke updates and announce changes. Those subjects will not be covered here...

Peer-2-Peer Event Coupling



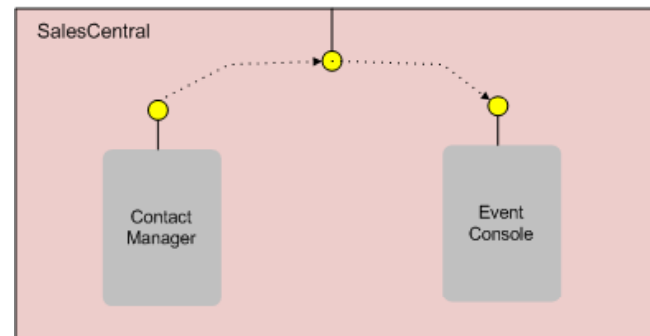
```
<ns1:AddContact addContact="txtConsole.text = String(event);" />
```

Notice how these three (3) types of component coupling use events & event delegation.

Note:

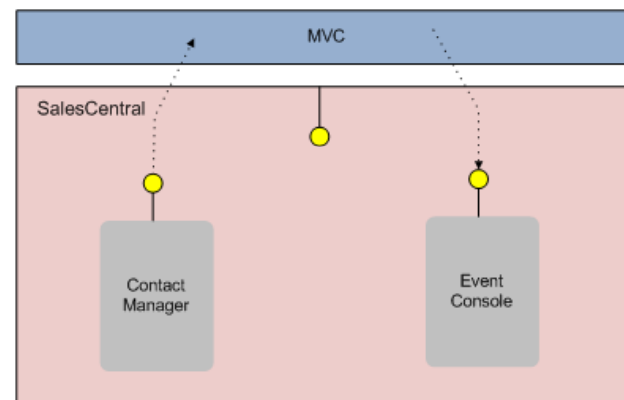
Most likely, you will use all 3 of these solutions within the same application.

Parent Mediation Event Coupling



```
<ns1:AddContact addContact="onAddContact(event);" />
```

Coupling by Event-2-MVC Databinding



Wham!

```
<mx:TextArea id="txtConsole" text="{__model.traceOutput}" />
```

Peer-2-Peer Event Coupling

Flex Development - Cairngorm_AddContact/src/SalesCentral_Peer2Peer.mxml - Eclipse SDK

File Edit Source Navigate Search Project Data Run Window Help

SalesCentral_Peer2Peer.mxml

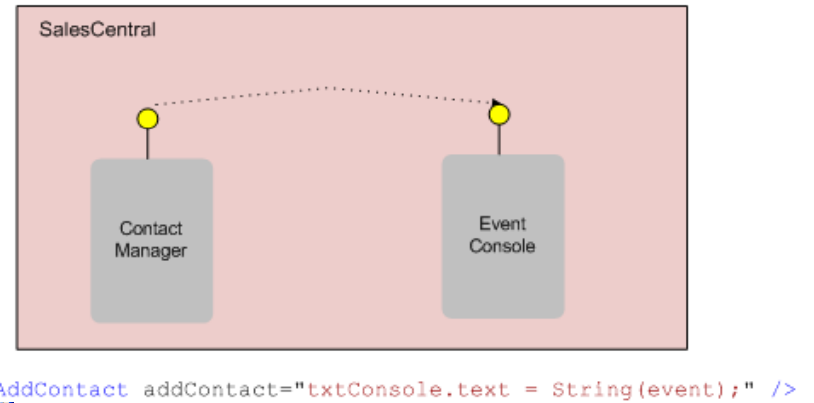
Source Design

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application layout="absolute" xmlns:mx="http://www.adobe.com/2006/mxml" >
3
4
5   <ns1:AddContact x="10" y="10" width="310" height="162"
6     addContact="txtConsole.text = String(event);"
7     xmlns:ns1="com.clientx.salesCentral.views.contacts.*" />
8
9   <mx:TextArea id="txtConsole"
10     y="10" height="162" right="20" left="338"/>
11
12 </mx:Application>
13

```

Writable Insert 5:53

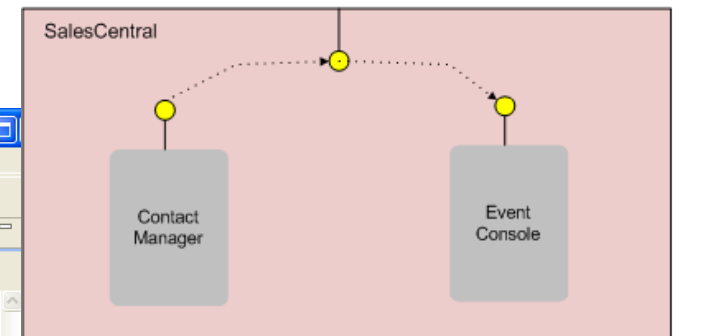


Parent Mediation Event Coupling

```

Flex Development - Cairngorm_AddContact/src/SalesCentral_ParentMediation.mxml - Eclipse SDK
File Edit Source Navigate Search Project Data Run Window Help
SalesCentral_ParentMediation.mxml
Source Design
1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application layout="absolute" xmlns:mx="http://www.adobe.com/2006/mxml" >
3
4   <mx:Script>
5     <![CDATA[
6       import com.clientx.salesCentral.control.events.AddContactEvent;
7
8       private function onAddContact(event:AddContactEvent):void {
9         // Can intercept, prevent, or do additional work here...
10        txtConsole.text = event.toString();
11      }
12    ]]>
13  </mx:Script>
14
15  <ns1:AddContact x="10" y="10" width="310" height="162"
16    addContact="onAddContact(event);"
17    xmlns:ns1="com.clientx.salesCentral.views.contacts.*" />
18
19  <mx:TextArea id="txtConsole"
20    y="10" height="162" right="20" left="338"/>
21
22 </mx:Application>
23

```



```
<ns1:AddContact addContact="onAddContact(event);" />
```

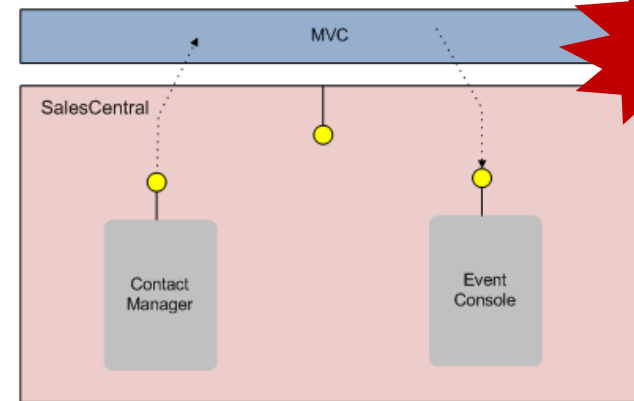
How do we use MVC with our events?

Adobe Cairngorm ?

What Layers ?

Special Events ?

Coupling by Event-2-MVC Databinding



```
<mx:TextArea id="txtConsole" text="{__model.traceOutput}" />
```

Overview

Without MVC!

Review Event Delegation and Dispatching

Adobe Cairngorm MVC!

Overlay Adobe Cairngorm on standard apps

- 1) Understanding events in Cairngorm
- 2) Introducing the **FrontController**
- 3) Impacts of Adobe Cairngorm on views
- 4) Issues with Adobe Cairngorm

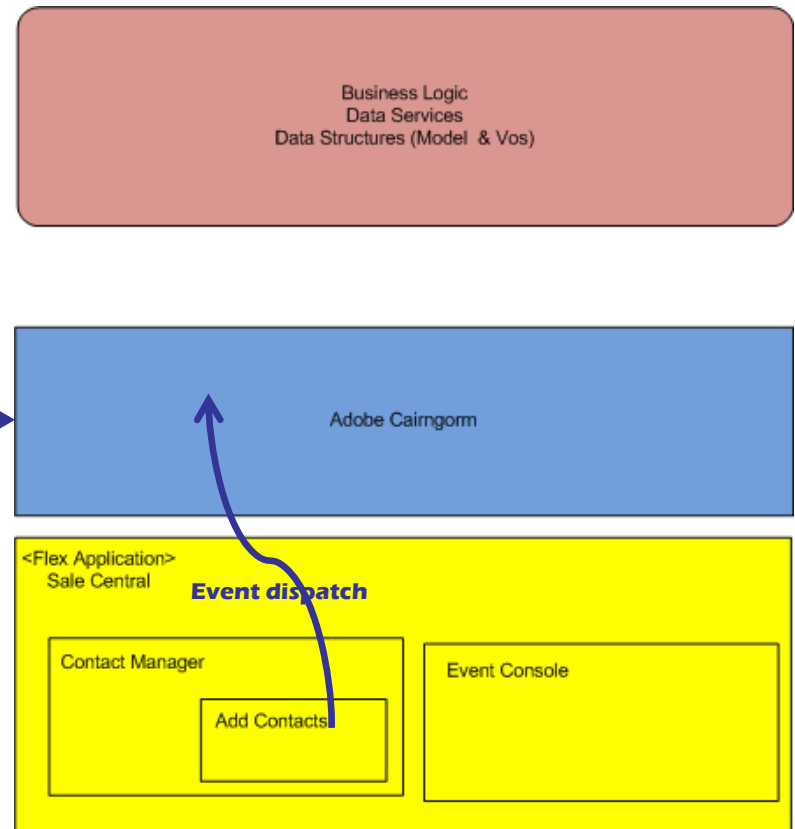
UM Cairngorm MVC!

Understanding UM Cairngorm

- 1) Issues & solutions with events
- 2) Impacts on views
- 3) Gotchas

Let's layer **Cairngorm MVC** on top of our application code and move our non-UI code to the business layer(s).

Events & **Event Delegation** to the rescue!

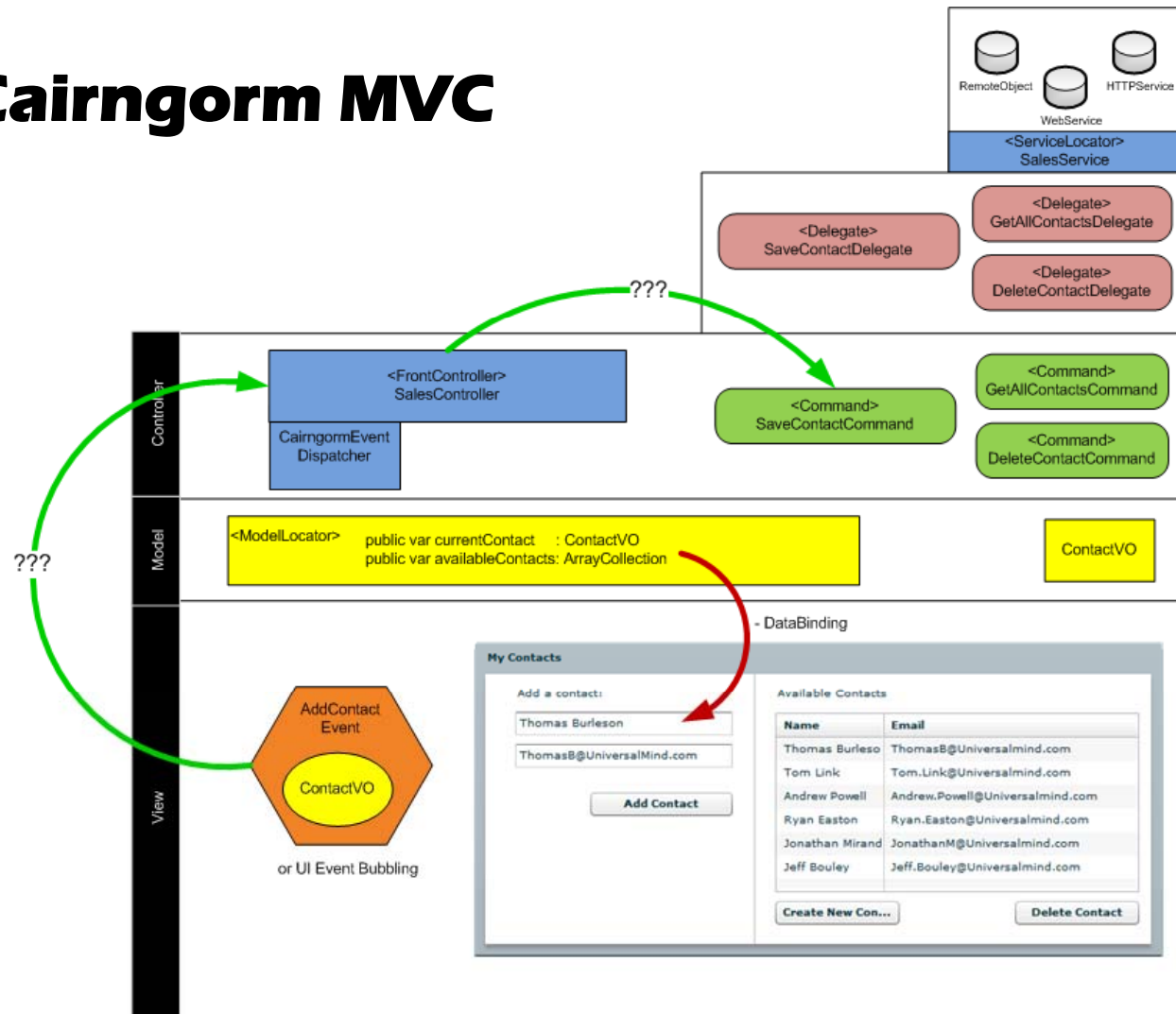


Cairngorm MVC

Event-2-MVC Coupling

1. **How** are events dispatched?
2. **How** are event listeners configured?

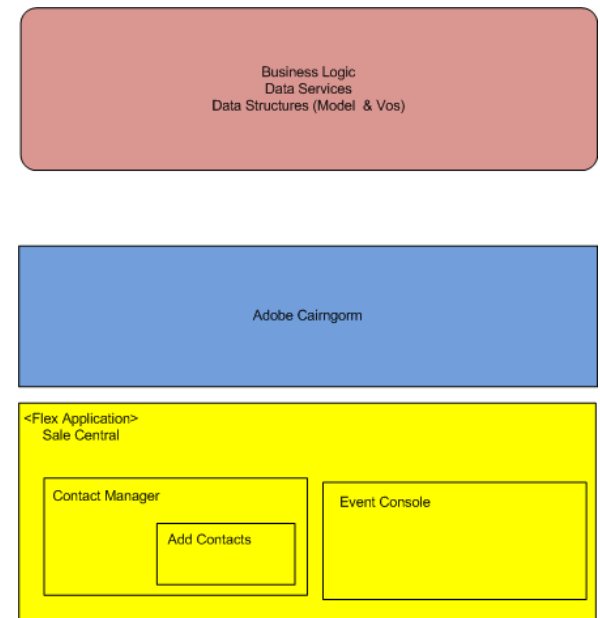
Remember, events are dispatched and listeners are "added" to respond to the event delegation...



Layering **Adobe Cairngorm** on your Application(s)

1. **Modify** your event coupling.
2. Move business logic to non-ui classes.
3. Bind to data... modified by biz logic

Steps 2 & 3 will not be covered here...



Cairngorm MVC

Event-2-MVC Coupling

1) How are events dispatched?

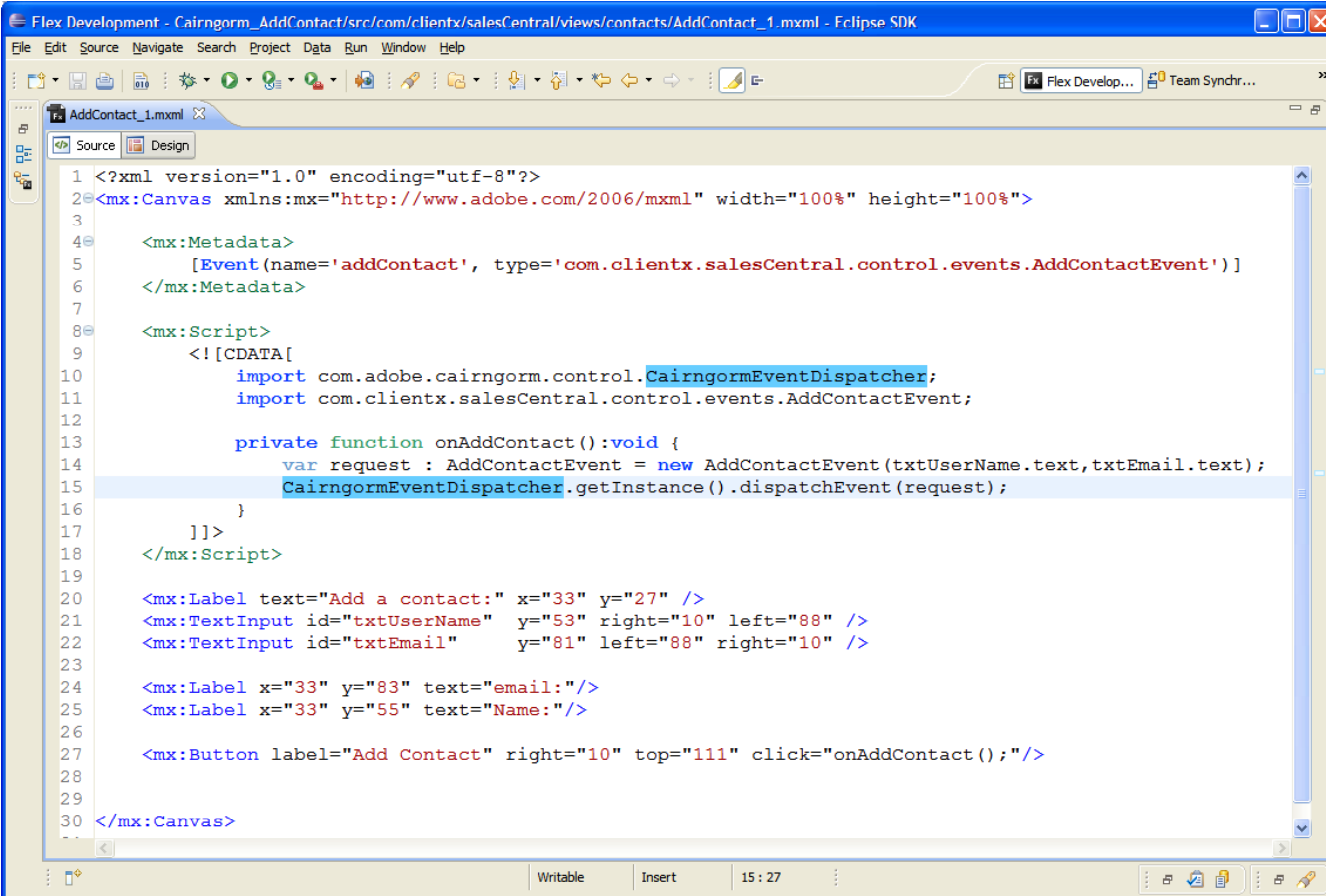
All business events are dispatched
FROM the CairngormEventDispatcher
 and not from the view [display object].

So...

this.dispatchEvent(request);

Becomes ...

CairngormEventDispatcher.getInstance().dispatchEvent(request);



```

1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml" width="100%" height="100%">
3
4   <mx:Metadata>
5     [Event(name='addContact', type='com.clientx.salesCentral.control.events.AddContactEvent')]
6   </mx:Metadata>
7
8   <mx:Script>
9     <![CDATA[
10       import com.adobe.cairngorm.control.CairngormEventDispatcher;
11       import com.clientx.salesCentral.control.events.AddContactEvent;
12
13       private function onAddContact():void {
14         var request : AddContactEvent = new AddContactEvent(txtUserName.text,txtEmail.text);
15         CairngormEventDispatcher.getInstance().dispatchEvent(request);
16       }
17     ]]>
18   </mx:Script>
19
20   <mx:Label text="Add a contact:" x="33" y="27" />
21   <mx:TextInput id="txtUserName" y="53" right="10" left="88" />
22   <mx:TextInput id="txtEmail" y="81" left="88" right="10" />
23
24   <mx:Label x="33" y="83" text="email:" />
25   <mx:Label x="33" y="55" text="Name:" />
26
27   <mx:Button label="Add Contact" right="10" top="111" click="onAddContact();" />
28
29 </mx:Canvas>
  
```

Cairngorm MVC

Event-2-MVC Coupling

`CairngormEventDispatcher.getInstance().dispatchEvent(request);`

Broadcasts to all (0-n) listeners the current event by invoking the event handler for each listener.

2) How are event listeners configured?

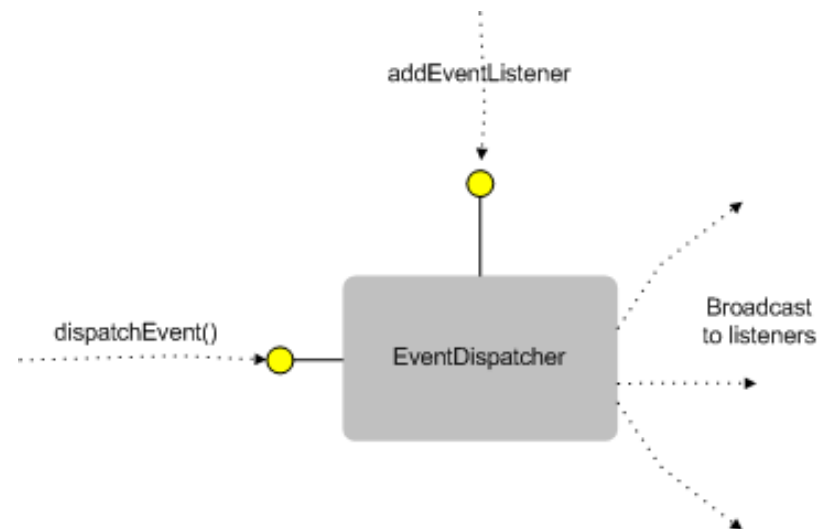
Goals:

- i) Want a central place to **register** and cross-reference each business event and the command that should **"handle"** that event.
- ii) Want this registry to automatically add "listeners" to the CairngormEventDispatcher.
- iii) Want to insure that the compiler **"links"** the byte code for each command. [If the linker finds no dependencies on a "command" class, that class will not be linked into the SWF]
- iv) Want this registry to be **common** to all business events and **"easy"** to use within a Flex application.

Solution:

Use the **FrontController** pattern.

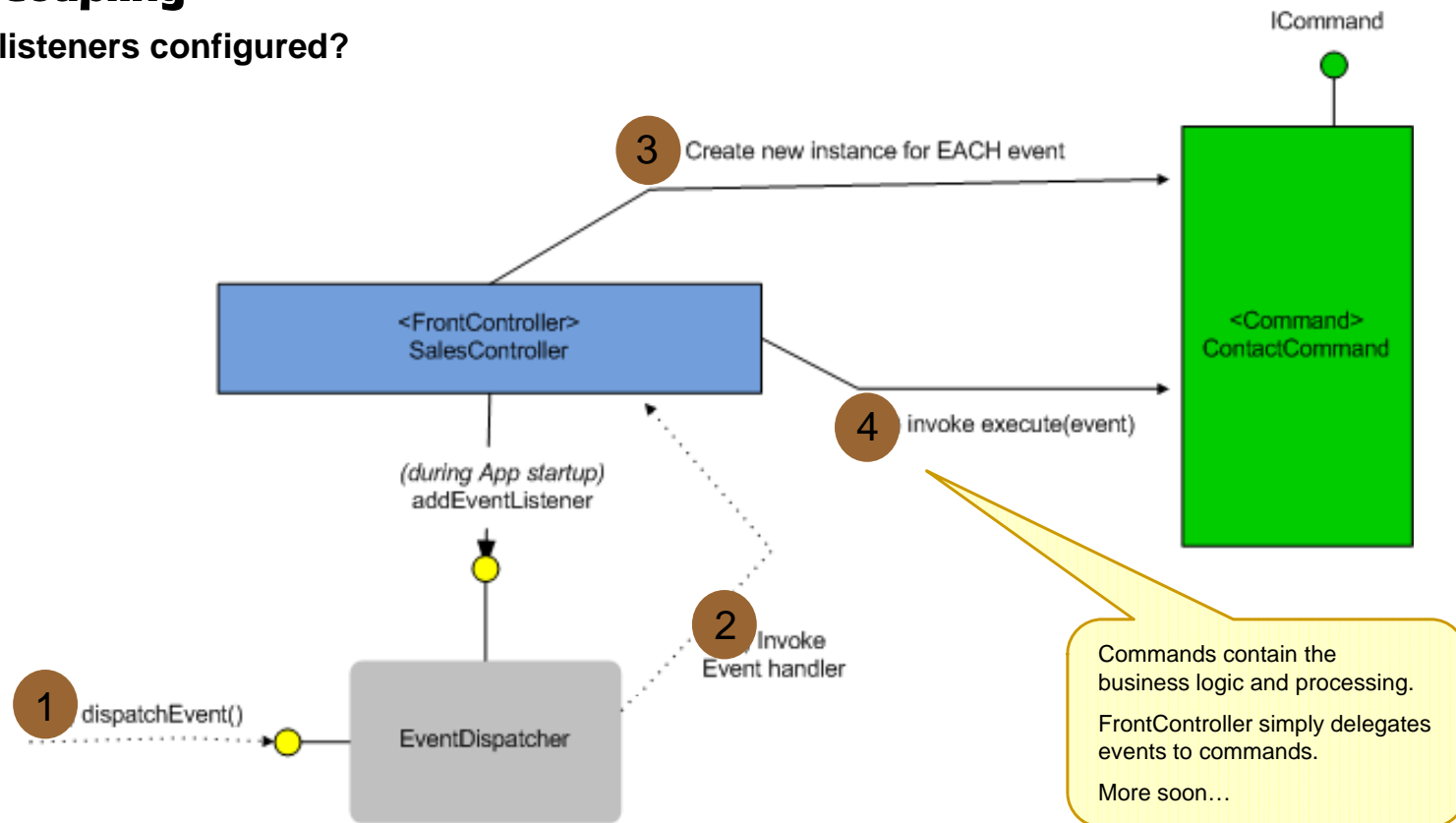
Views dispatch to CairngormEventDispatcher & events are "registered" with specific commands.



Introducing the Cairngorm **FrontController**

Event-2-MVC Coupling

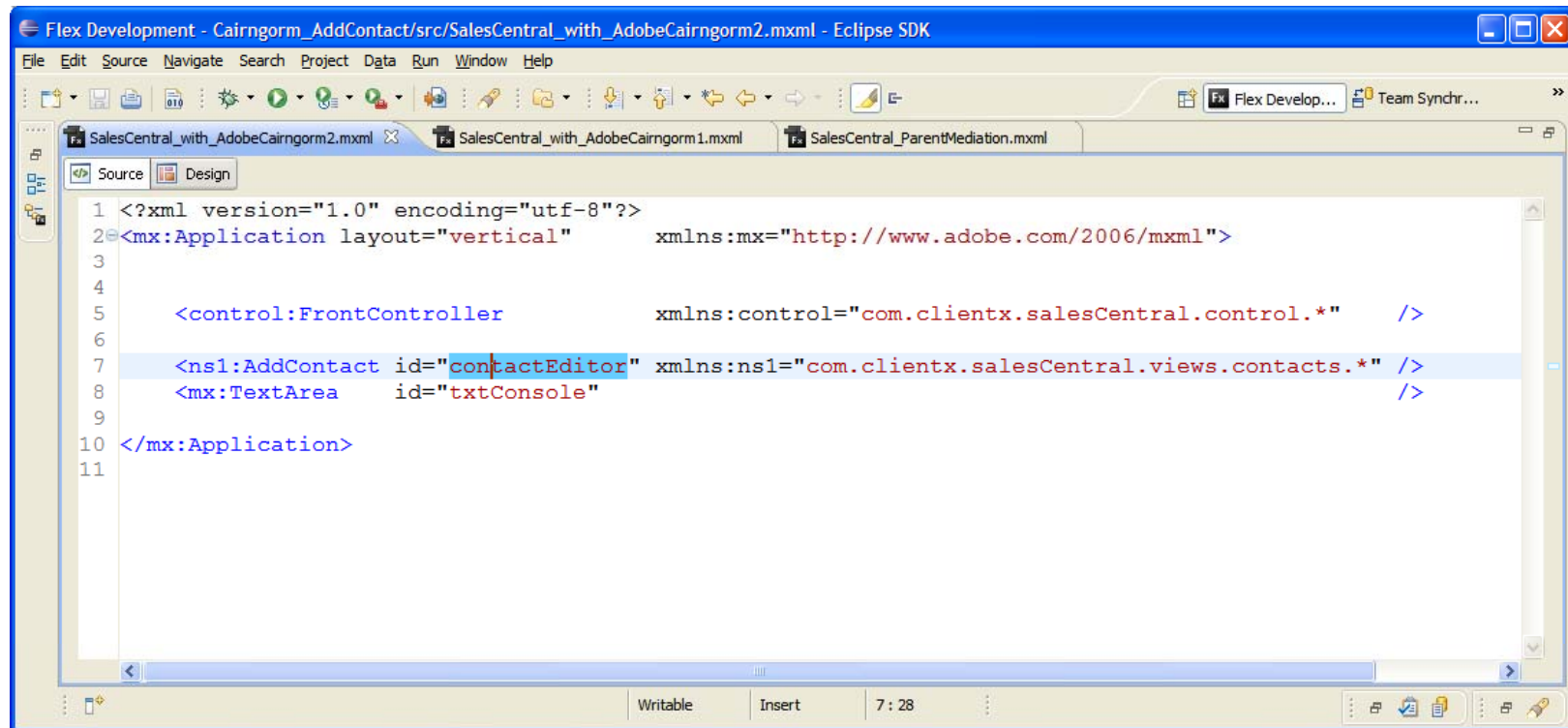
2) **How** are event listeners configured?



Cairngorm FrontController

Event-2-MVC Coupling

How are event listeners configured?



The screenshot shows the Eclipse IDE with the 'Flex Development' window open. The editor displays the 'SalesCentral_with_AdobeCairngorm2.mxml' file. The code is as follows:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application layout="vertical" xmlns:mx="http://www.adobe.com/2006/mxml">
3
4
5     <control:FrontController xmlns:control="com.clientx.salesCentral.control.*" />
6
7     <ns1:AddContact id="contactEditor" xmlns:ns1="com.clientx.salesCentral.views.contacts.*" />
8     <mx:TextArea id="txtConsole" />
9
10 </mx:Application>
11
```

Above, we instantiate a custom FrontController using a declarative tag. It is never directly used after instantiation.

Cairngorm FrontController

Event-2-MVC Coupling

How are event listeners configured?

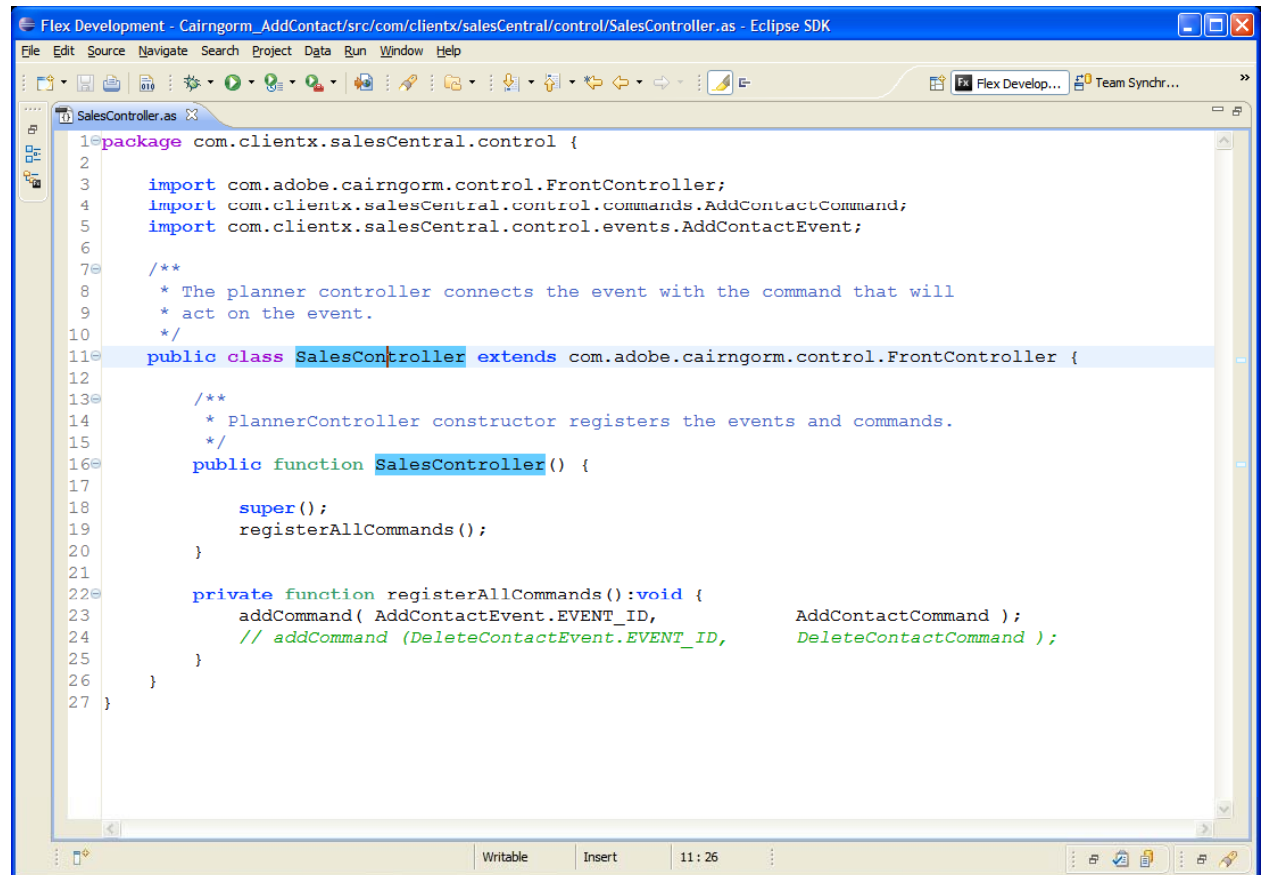
Register all events and the command that should “handle” each event.

Notes:

The FrontController is not a subclass of UIComponent but still can be used in a MXML tag form...

An event should only be registered for one (1) command.

This class is NOT to be used to “fire” sequence of events. Only to configure responders/handlers for events.



```

1 package com.clientx.salesCentral.control {
2
3     import com.adobe.cairngorm.control.FrontController;
4     import com.clientx.salesCentral.control.commands.AddContactCommand;
5     import com.clientx.salesCentral.control.events.AddContactEvent;
6
7     /**
8      * The planner controller connects the event with the command that will
9      * act on the event.
10    */
11    public class SalesController extends com.adobe.cairngorm.control.FrontController {
12
13        /**
14         * PlannerController constructor registers the events and commands.
15        */
16        public function SalesController() {
17
18            super();
19            registerAllCommands();
20        }
21
22        private function registerAllCommands():void {
23            addCommand( AddContactEvent.EVENT_ID,      AddContactCommand );
24            // addCommand (DeleteContactEvent.EVENT_ID, DeleteContactCommand );
25        }
26    }
27 }
  
```

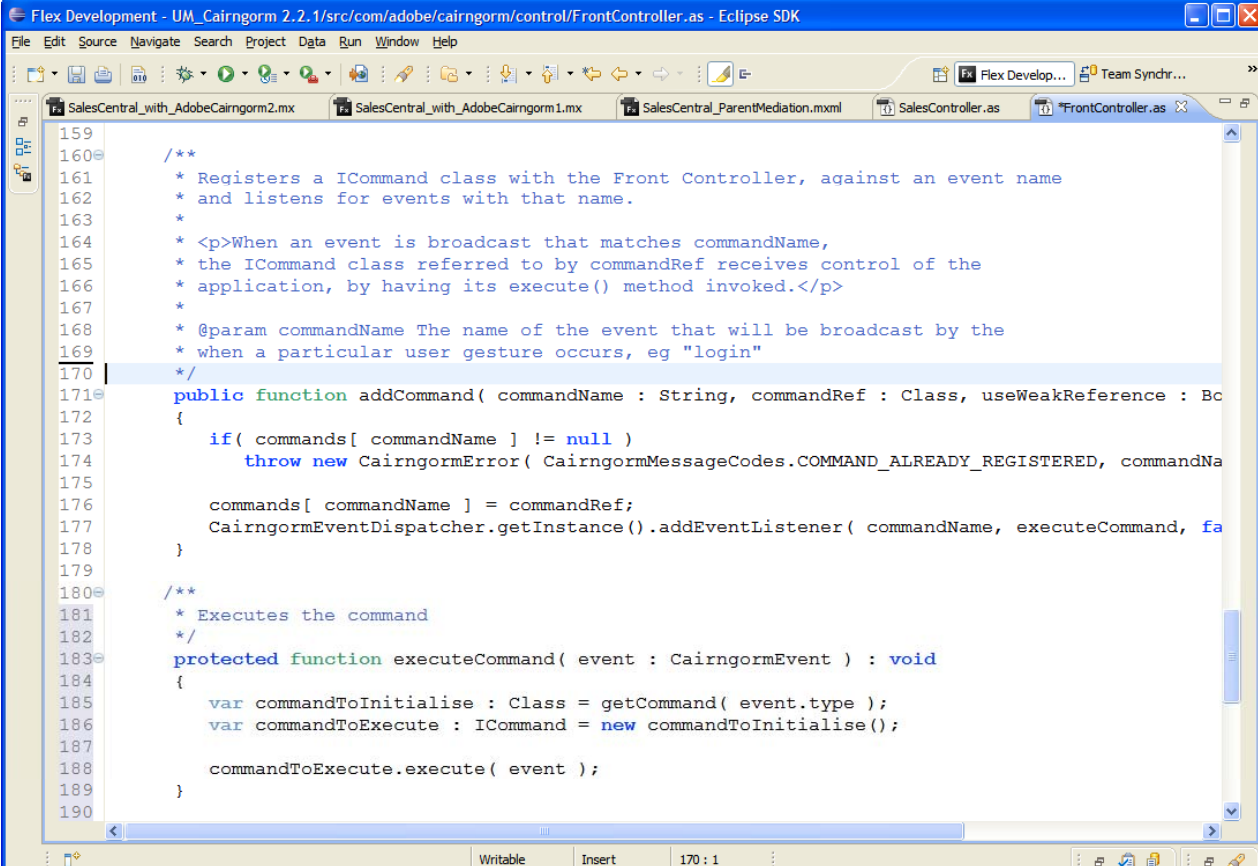
Cairngorm FrontController

Event-2-MVC Coupling

How are event listeners configured?

- i) Each event is used to add the same event handler to the CairngormEventDispatcher.
- ii) The common handler “executeCommand” creates an instance of the command based on the event type/ID.
- iii) An instance of command is created for EACH event instance
- iv) This command is then asked to “execute” or process the event.

Thus the FrontController is a **centralized event delegator** to business processors [aka Commands].



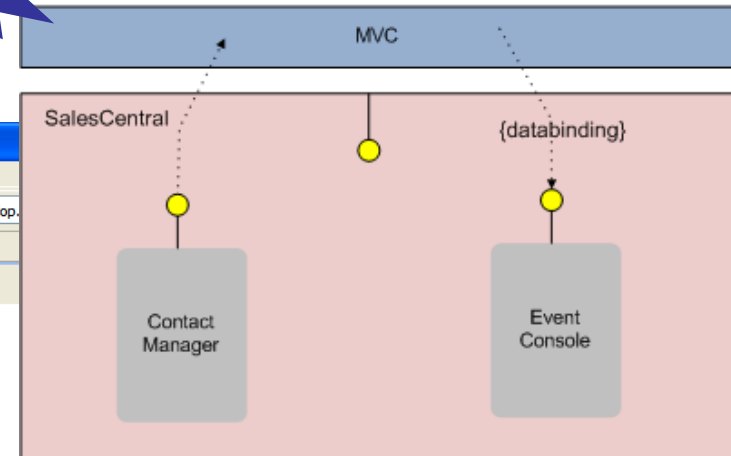
```

159
160
161 /**
162  * Registers a ICommand class with the Front Controller, against an event name
163  * and listens for events with that name.
164  *
165  * <p>When an event is broadcast that matches commandName,
166  * the ICommand class referred to by commandRef receives control of the
167  * application, by having its execute() method invoked.</p>
168  *
169  * @param commandName The name of the event that will be broadcast by the
170  * when a particular user gesture occurs, eg "login"
171  */
172 public function addCommand( commandName : String, commandRef : Class, useWeakReference : Boolean ) : void
173 {
174     if( commands[ commandName ] != null )
175         throw new CairngormError( CairngormMessageCodes.COMMAND_ALREADY_REGISTERED, commandName );
176
177     commands[ commandName ] = commandRef;
178     CairngormEventDispatcher.getInstance().addEventListener( commandName, executeCommand, false );
179 }
180
181 /**
182  * Executes the command
183  */
184 protected function executeCommand( event : CairngormEvent ) : void
185 {
186     var commandToInitialise : Class = getCommand( event.type );
187     var commandToExecute : ICommand = new commandToInitialise();
188
189     commandToExecute.execute( event );
190 }

```

Wham!

Coupling by Event-to-MVC



```

Flex Development - Cairngorm_AddContact/src/SalesCentral_with_AdobeCairngorm2.mxml - Eclipse SDK
File Edit Source Navigate Search Project Data Run Window Help
SalesCentral_with_AdobeCairngorm2.mxml
Source Design
1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application layout="vertical" xmlns:mx="http://www.adobe.com/2006/mxml">
3
4 <mx:Script>
5 <![CDATA[
6     import com.clientx.salesCentral.model.ModelLocator;
7
8     [Bindable] private var model : ModelLocator = ModelLocator.getInstance();
9
10 ]]>
11 </mx:Script>
12 <control:SalesController xmlns:control="com.clientx.salesCentral.control.*" />
13
14 <ns1:AddContact id="contactEditor" xmlns:ns1="com.clientx.salesCentral.views.contacts.*" />
15 <mx:TextArea id="txtConsole" text="{model.output}" />
16
17 </mx:Application>
18
    
```

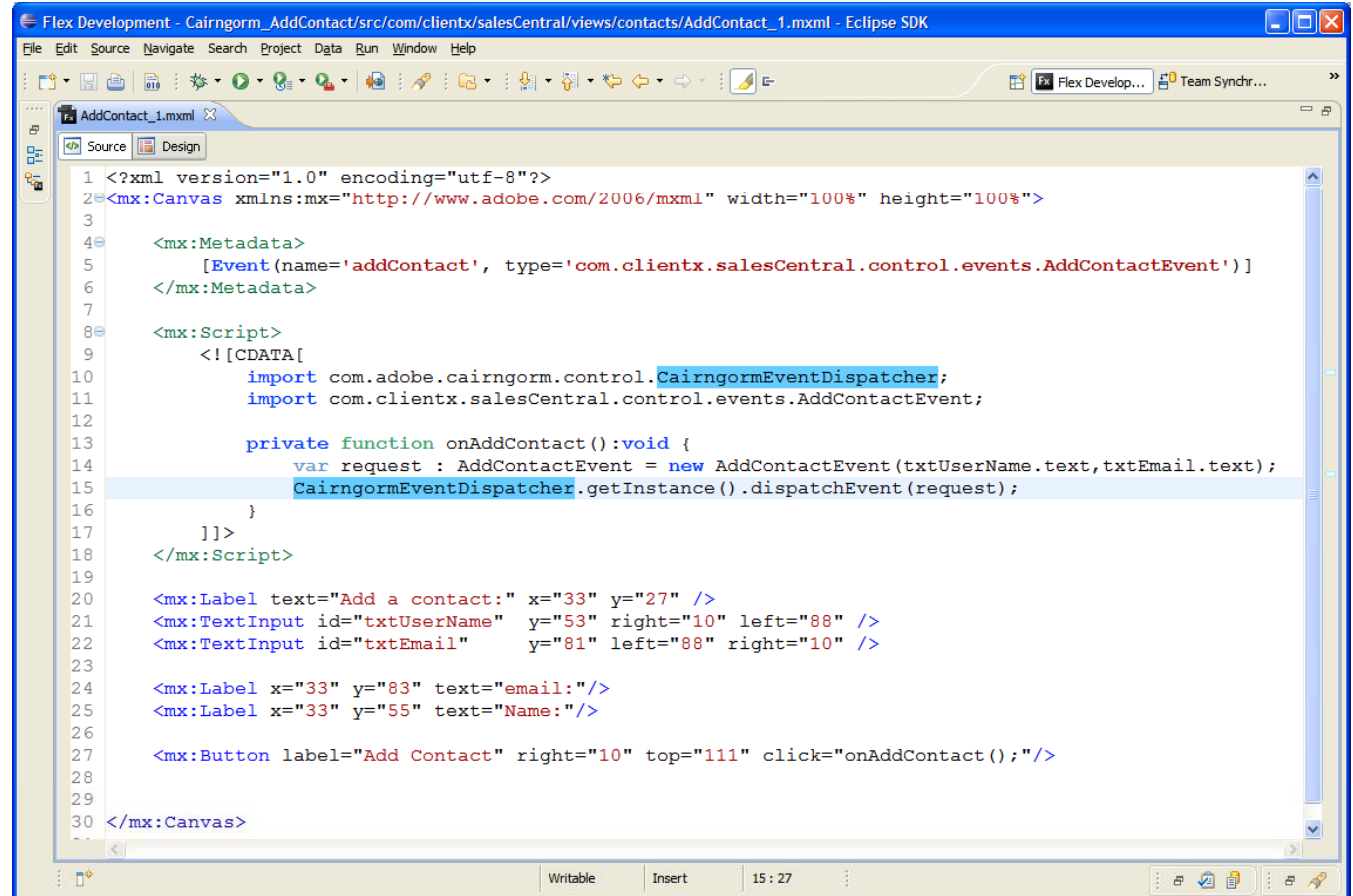
What are the Adobe Cairngorm **Impacts** on your views ?

Coupling...

- 1) To dispatch business events, all views must **import** CairngormEventDispatcher.
- 2) This means your views **must** be used within a MVC application. **YIKES!**



Crash



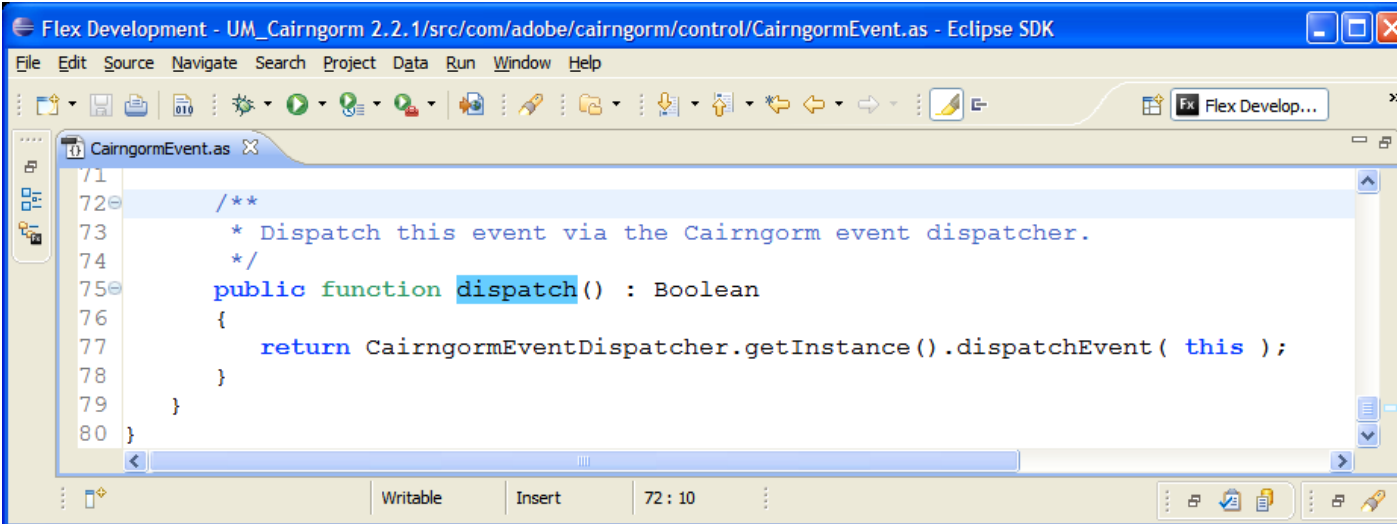
```

1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml" width="100%" height="100%">
3
4   <mx:Metadata>
5     [Event(name='addContact', type='com.clientx.salesCentral.control.events.AddContactEvent')]
6   </mx:Metadata>
7
8   <mx:Script>
9     <![CDATA[
10       import com.adobe.cairngorm.control.CairngormEventDispatcher;
11       import com.clientx.salesCentral.control.events.AddContactEvent;
12
13       private function onAddContact():void {
14         var request : AddContactEvent = new AddContactEvent(txtUserName.text,txtEmail.text);
15         CairngormEventDispatcher.getInstance().dispatchEvent(request);
16       }
17     ]]>
18   </mx:Script>
19
20   <mx:Label text="Add a contact:" x="33" y="27" />
21   <mx:TextInput id="txtUserName" y="53" right="10" left="88" />
22   <mx:TextInput id="txtEmail" y="81" left="88" right="10" />
23
24   <mx:Label x="33" y="83" text="email:" />
25   <mx:Label x="33" y="55" text="Name:" />
26
27   <mx:Button label="Add Contact" right="10" top="111" click="onAddContact();" />
28
29
30 </mx:Canvas>
  
```

Adobe Cairngorm solution to this issue:

Cairngorm (v2.2) Enhancement!

- 1) Cairngorm events can now dispatch themselves to the business layers.



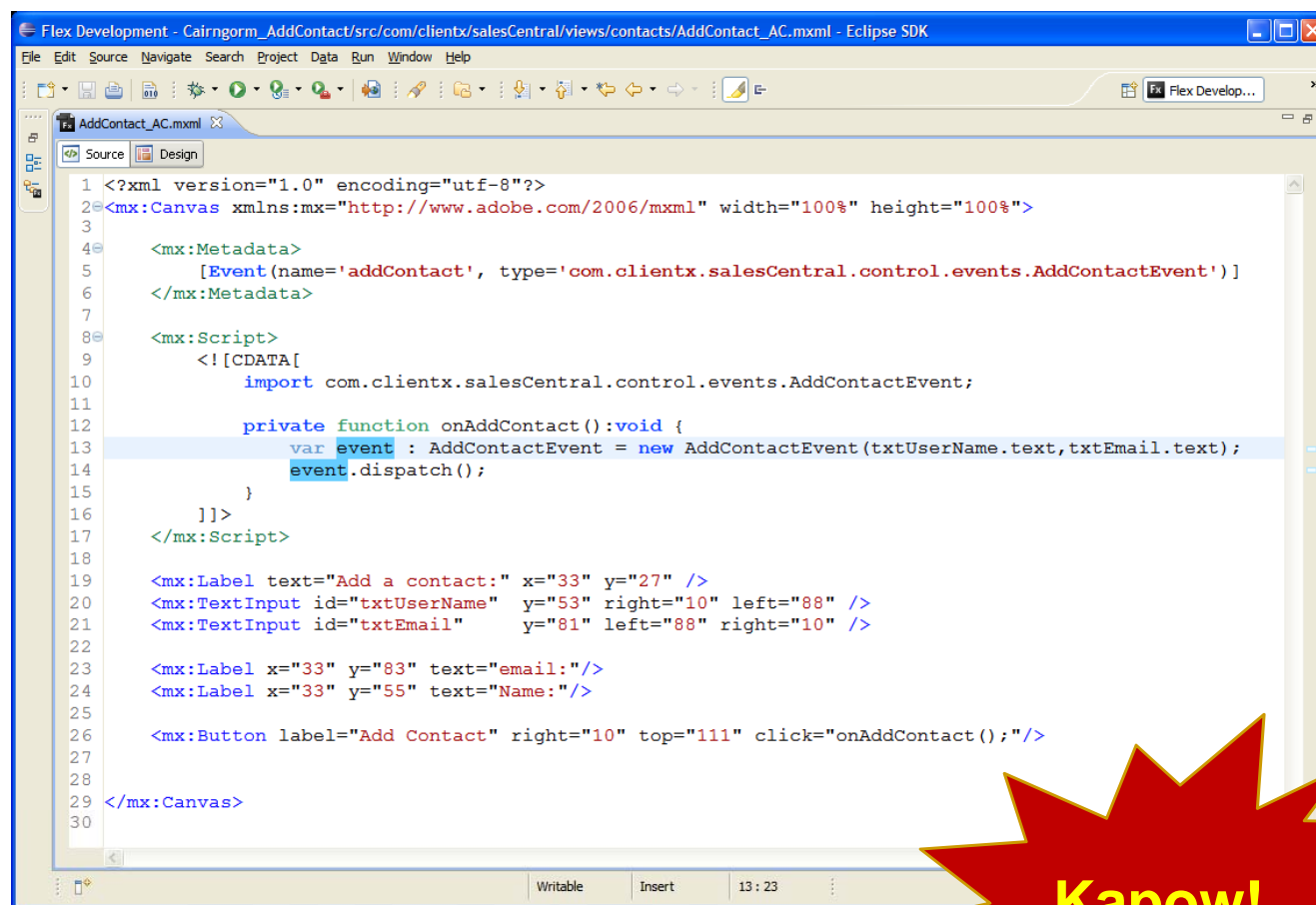
```
Flex Development - UM_Cairngorm 2.2.1/src/com/adobe/cairngorm/control/CairngormEvent.as - Eclipse SDK
File Edit Source Navigate Search Project Data Run Window Help
CairngormEvent.as
72 /**
73  * Dispatch this event via the Cairngorm event dispatcher.
74  */
75 public function dispatch() : Boolean
76 {
77     return CairngormEventDispatcher.getInstance().dispatchEvent( this );
78 }
79
80 }
```


Adobe Cairngorm solution to this issue:

Self-Dispatching

- 1) Cairngorm events can now dispatch themselves to the business layers.
- 2) **No** more **import** needed.

No other event knows how to self-dispatch... this is not a good solution.



```

1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml" width="100%" height="100%">
3
4   <mx:Metadata>
5     [Event(name='addContact', type='com.clientx.salesCentral.control.events.AddContactEvent')]
6   </mx:Metadata>
7
8   <mx:Script>
9     <![CDATA[
10       import com.clientx.salesCentral.control.events.AddContactEvent;
11
12       private function onAddContact():void {
13         var event : AddContactEvent = new AddContactEvent(txtUserName.text,txtEmail.text);
14         event.dispatch();
15       }
16     ]]>
17   </mx:Script>
18
19   <mx:Label text="Add a contact:" x="33" y="27" />
20   <mx:TextInput id="txtUserName" y="53" right="10" left="88" />
21   <mx:TextInput id="txtEmail" y="81" left="88" right="10" />
22
23   <mx:Label x="33" y="83" text="email:" />
24   <mx:Label x="33" y="55" text="Name:" />
25
26   <mx:Button label="Add Contact" right="10" top="111" click="onAddContact();" />
27
28 </mx:Canvas>
  
```

Kapow!

What other MVC Event-solutions are available?



Direct dispatch to CairngormEventDispatcher



Bad coupling



Event self-dispatches



None conformant!



Event Bubbling to Application



Depth & Popup Manager Issues



Event Hook



Best available solution

Has 1 issue... which we will see soon!

Overview

Without MVC!

Review Event Delegation and Dispatching

Adobe Cairngorm MVC!

Overlay Adobe Cairngorm on standard apps

- 1) Understanding events in Cairngorm
- 2) Introducing the **FrontController**
- 3) Impacts of Adobe Cairngorm on views
- 4) Issues with Adobe Cairngorm

UM Cairngorm MVC!

Understanding UM Cairngorm Extensions

- 1) Solutions with events**
- 2) Impacts on views**
- 3) Gotchas**

Solutions with **UM** Cairngorm **Extensions**

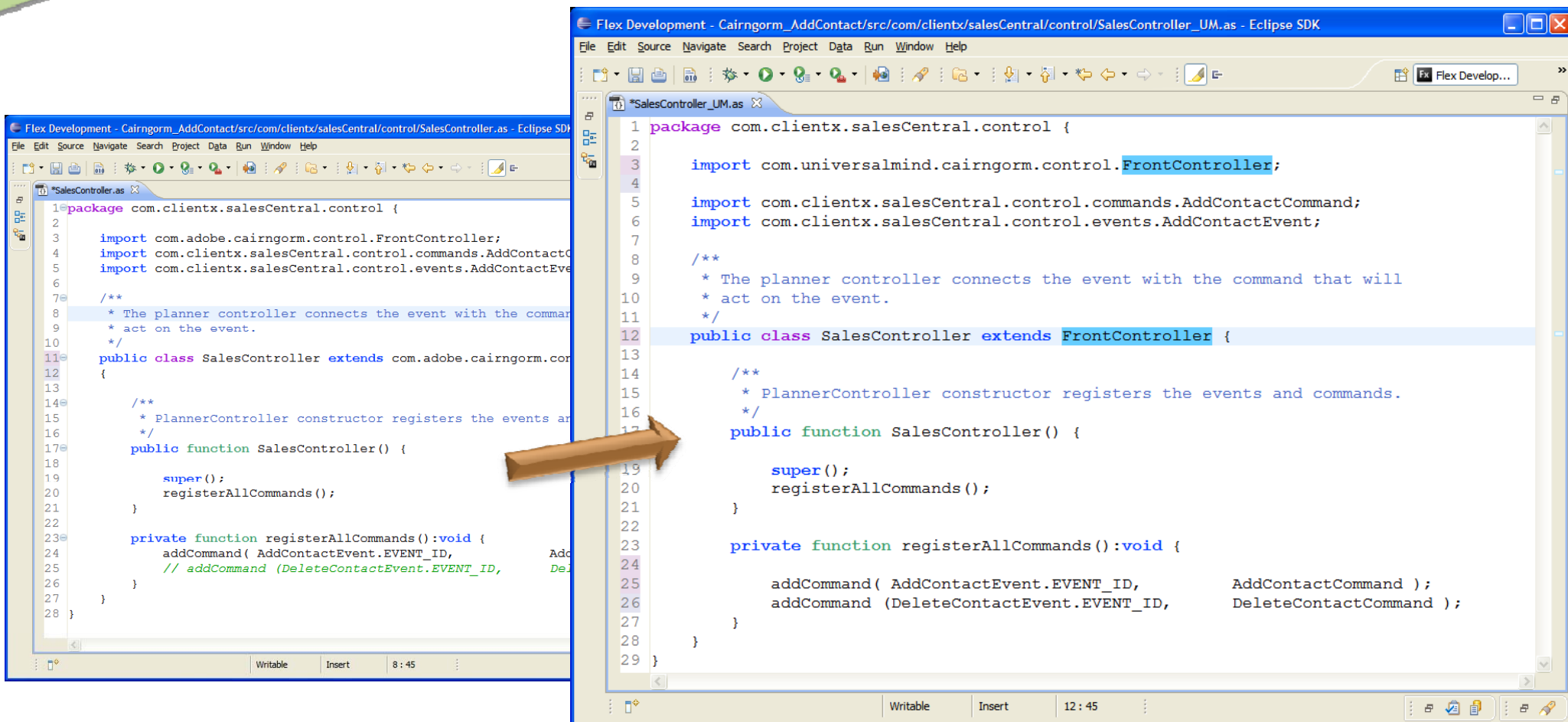
Adobe Cairngorm

1. Uses MVC pattern
2. Poor support of Webservices initializations
3. Enterprise issues view notifications
4. Modules not supported

Universal Cairngorm Extensions

1. Support for view responders
 2. Support for command event dispatching
 3. Superior support for delegate queues
 4. Support for batch events (parallel and/or sequential)
 5. Support for delegate-level data transformations
 6. Support for command logic aggregations
 7. Mini-MVC apps/modules supported
 8. Usable for full non-ui **FlexUnit** testing
9. Event **Hooks** for Event Dispatching from views

UM Cairngorm Event Bubbling



Flex Development - Cairngorm_AddContact/src/com/clientx/salesCentral/control/SalesController.as - Eclipse SDK

```

1 package com.clientx.salesCentral.control {
2
3     import com.adobe.cairngorm.control.FrontController;
4     import com.clientx.salesCentral.control.commands.AddContactCommand;
5     import com.clientx.salesCentral.control.events.AddContactEvent;
6
7     /**
8      * The planner controller connects the event with the command that will
9      * act on the event.
10    */
11    public class SalesController extends com.adobe.cairngorm.control.FrontController {
12
13        /**
14         * PlannerController constructor registers the events and commands.
15        */
16        public function SalesController() {
17
18            super();
19            registerAllCommands();
20        }
21
22        private function registerAllCommands():void {
23            addCommand( AddContactEvent.EVENT_ID,
24                      // addCommand (DeleteContactEvent.EVENT_ID,
25
26        }
27    }
28 }

```

Flex Development - Cairngorm_AddContact/src/com/clientx/salesCentral/control/SalesController_UM.as - Eclipse SDK

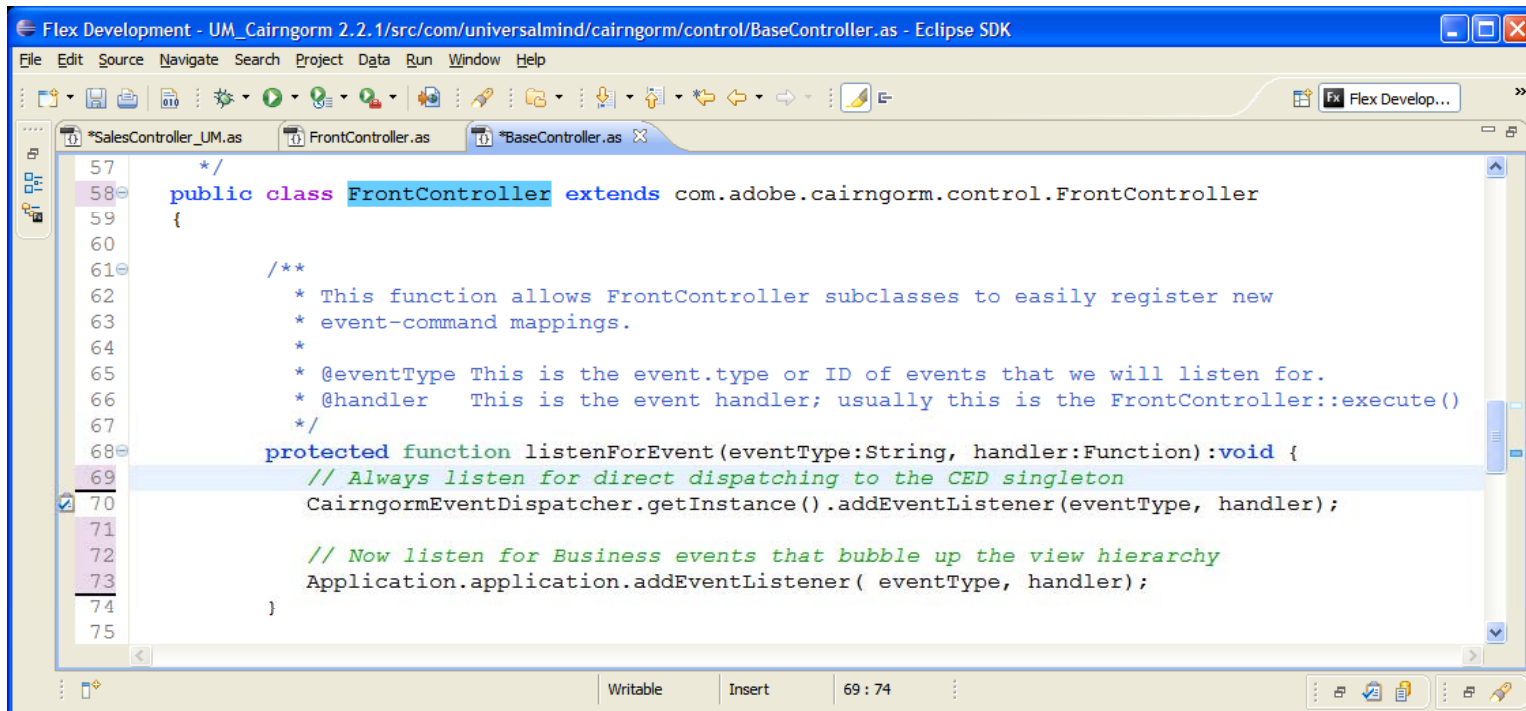
```

1 package com.clientx.salesCentral.control {
2
3     import com.universalmind.cairngorm.control.FrontController;
4
5     import com.clientx.salesCentral.control.commands.AddContactCommand;
6     import com.clientx.salesCentral.control.events.AddContactEvent;
7
8     /**
9      * The planner controller connects the event with the command that will
10    * act on the event.
11    */
12    public class SalesController extends FrontController {
13
14        /**
15         * PlannerController constructor registers the events and commands.
16        */
17        public function SalesController() {
18
19            super();
20            registerAllCommands();
21        }
22
23        private function registerAllCommands():void {
24
25            addCommand( AddContactEvent.EVENT_ID,      AddContactCommand );
26            addCommand (DeleteContactEvent.EVENT_ID,  DeleteContactCommand );
27
28        }
29    }

```

Use the UM FrontController (v.9)

UM Cairngorm^{v.9} Event Bubbling



```

Flex Development - UM_Cairngorm 2.2.1/src/com/universalmind/cairngorm/control/BaseController.as - Eclipse SDK
File Edit Source Navigate Search Project Data Run Window Help
[Icons] Flex Develop...

*SalesController_UM.as *FrontController.as *BaseController.as
57 */
58 public class FrontController extends com.adobe.cairngorm.control.FrontController
59 {
60
61 /**
62  * This function allows FrontController subclasses to easily register new
63  * event-command mappings.
64  *
65  * @eventType This is the event.type or ID of events that we will listen for.
66  * @handler This is the event handler; usually this is the FrontController::execute()
67  */
68 protected function listenForEvent(eventType:String, handler:Function):void {
69     // Always listen for direct dispatching to the CED singleton
70     CairngormEventDispatcher.getInstance().addEventListener(eventType, handler);
71
72     // Now listen for Business events that bubble up the view hierarchy
73     Application.application.addEventListener(eventType, handler);
74 }
75
    
```



Requires all business events to set “bubbles = true” and bubble up the view hierarchy

Listens at the Application.application level ONLY.

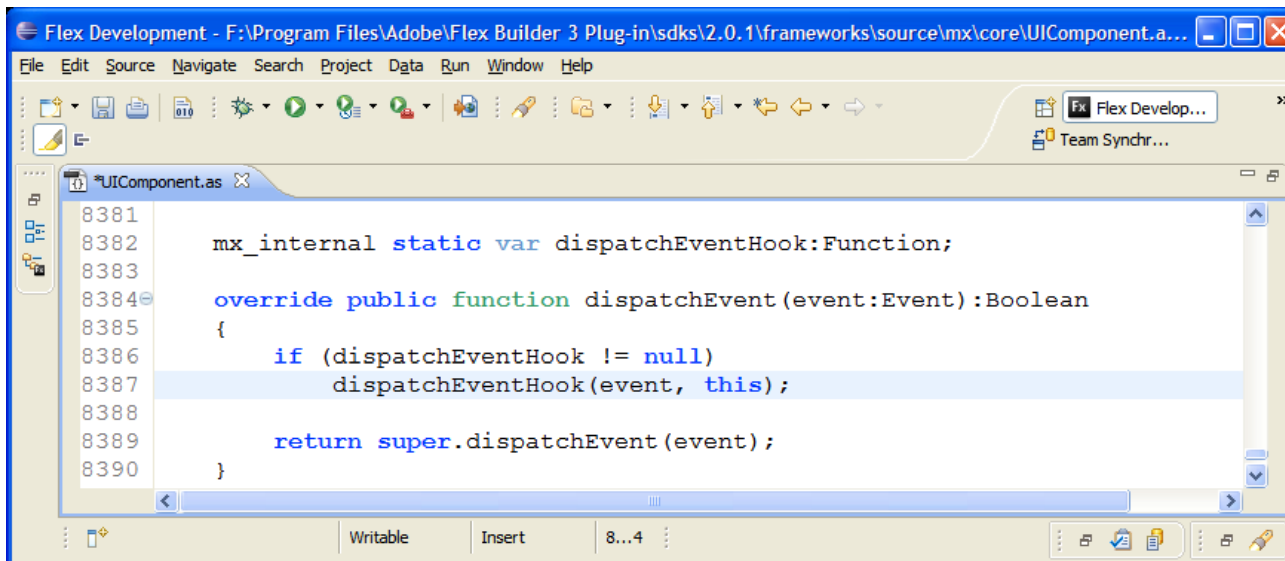
PopupManager dispatches are not supported with custom work.

UM Cairngorm^{v1.0} Event Hooks



A single function can be automatically called for every `UIComponent.dispatchEvent()`

This is a “Head-hook” callback that is called BEFORE the normal dispatch and bubbling.



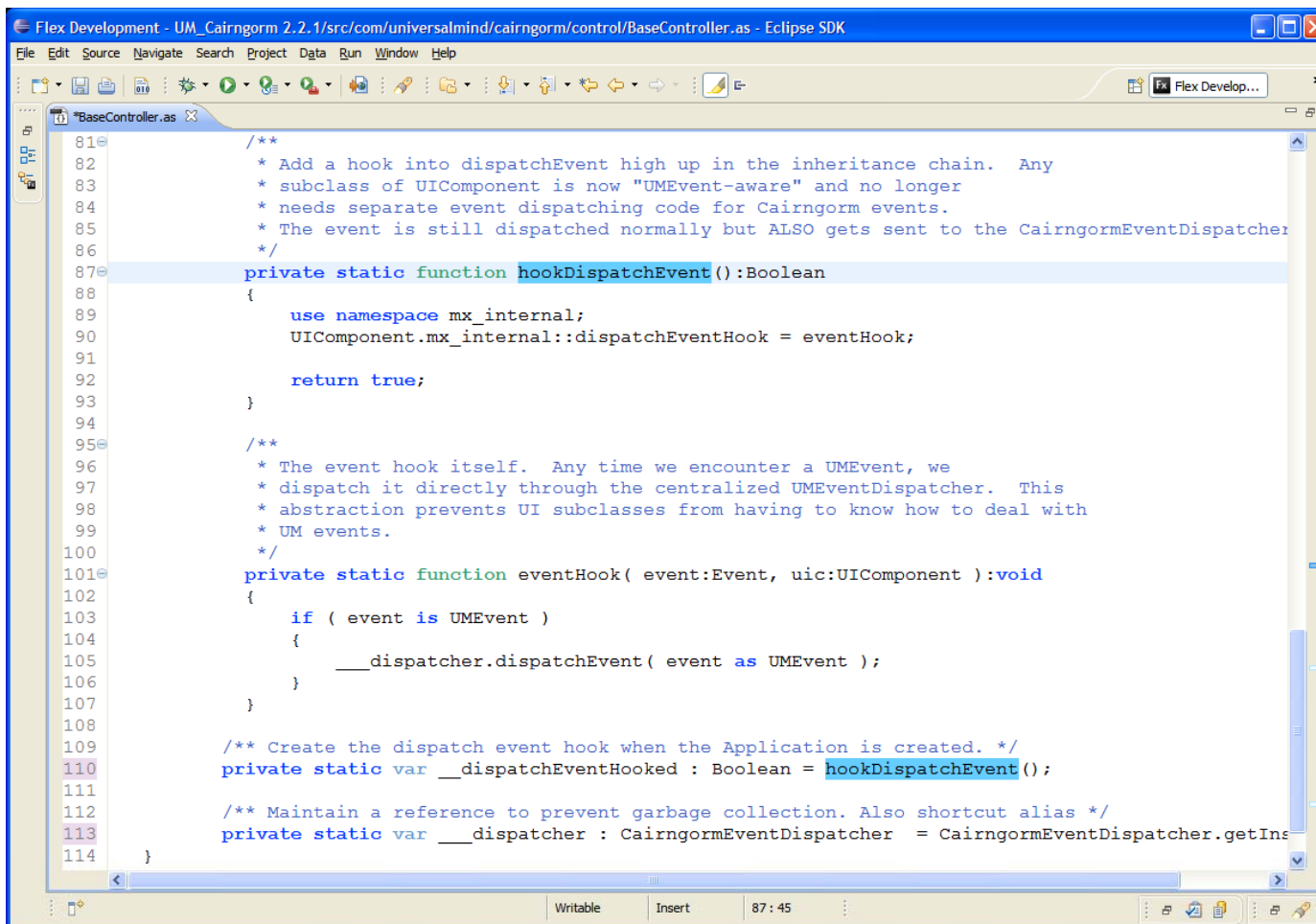
```
8381
8382     mx_internal static var dispatchEventHook:Function;
8383
8384     override public function dispatchEvent(event:Event):Boolean
8385     {
8386         if (dispatchEventHook != null)
8387             dispatchEventHook(event, this);
8388
8389         return super.dispatchEvent(event);
8390     }
```

Doug Knudsen discovered this in the `mx.core.UIComponent` source...

Darron Schall has a blog discussing this cool feature and its tradeoffs...

UM Cairngorm^{v1.0} Event Hooks

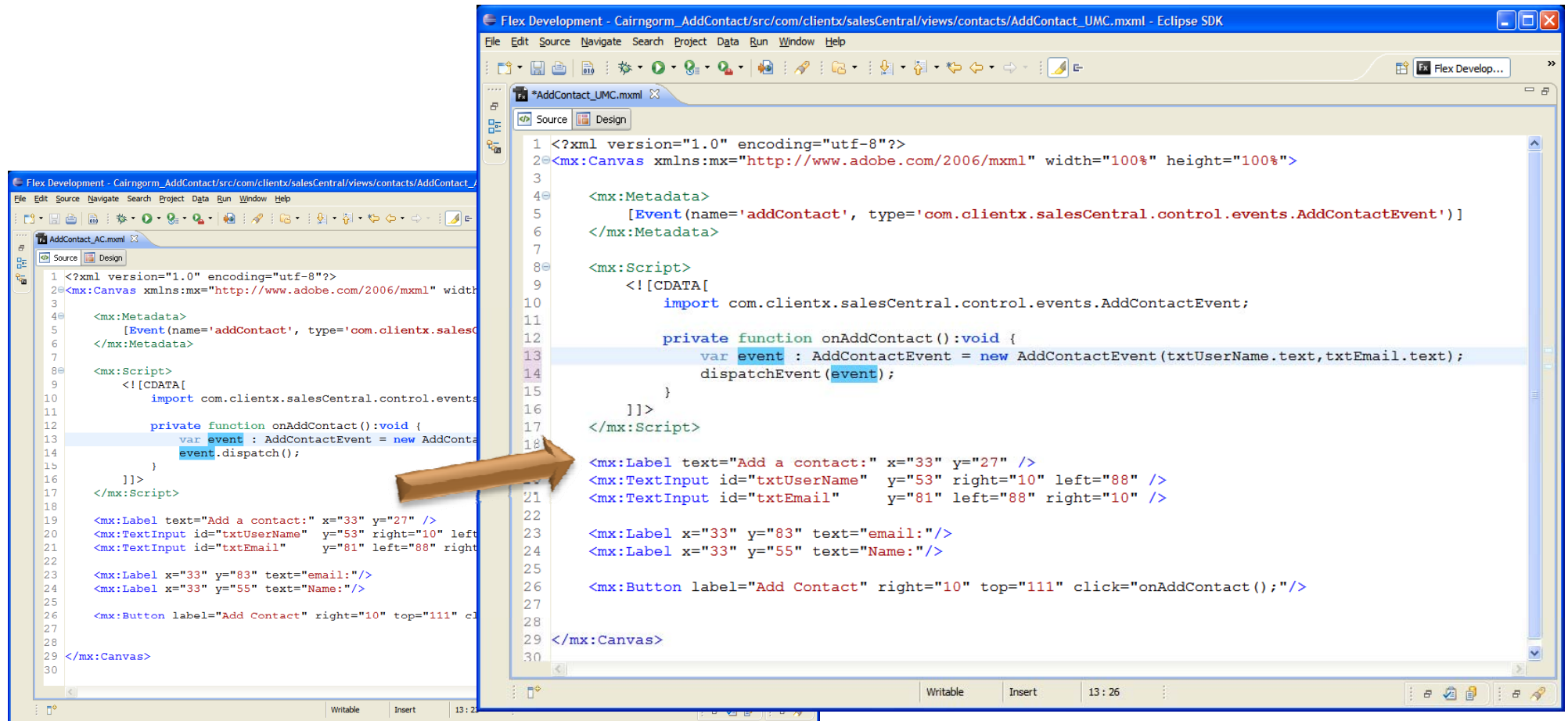
Automatic configured simply by instantiating a UM FrontController subclass.



```
Flex Development - UM_Cairngorm 2.2.1/src/com/universalmind/cairngorm/control/BaseController.as - Eclipse SDK
File Edit Source Navigate Search Project Data Run Window Help

81  /**
82   * Add a hook into dispatchEvent high up in the inheritance chain. Any
83   * subclass of UIComponent is now "UMEvent-aware" and no longer
84   * needs separate event dispatching code for Cairngorm events.
85   * The event is still dispatched normally but ALSO gets sent to the CairngormEventDispatcher.
86   */
87  private static function hookDispatchEvent():Boolean
88  {
89      use namespace mx_internal;
90      UIComponent.mx_internal::dispatchEventHook = eventHook;
91
92      return true;
93  }
94
95  /**
96   * The event hook itself. Any time we encounter a UMEvent, we
97   * dispatch it directly through the centralized UMEventDispatcher. This
98   * abstraction prevents UI subclasses from having to know how to deal with
99   * UM events.
100  */
101  private static function eventHook( event:Event, uic:UIComponent ):void
102  {
103      if ( event is UMEvent )
104      {
105          __dispatcher.dispatchEvent( event as UMEvent );
106      }
107  }
108
109  /** Create the dispatch event hook when the Application is created. */
110  private static var __dispatchEventHooked : Boolean = hookDispatchEvent();
111
112  /** Maintain a reference to prevent garbage collection. Also shortcut alias */
113  private static var __dispatcher : CairngormEventDispatcher = CairngormEventDispatcher.getInst
114  }
```

UM Cairngorm ^{v1.0} Event Hooks



```

1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml" width="100%" height="100%">
3
4   <mx:Metadata>
5     [Event(name='addContact', type='com.clientx.salesCentral.control.events.AddContactEvent')]
6   </mx:Metadata>
7
8   <mx:Script>
9     <![CDATA[
10       import com.clientx.salesCentral.control.events.AddContactEvent;
11
12       private function onAddContact():void {
13         var event : AddContactEvent = new AddContactEvent(txtUserName.text,txtEmail.text);
14         dispatchEvent(event);
15       }
16     ]]>
17   </mx:Script>
18
19   <mx:Label text="Add a contact:" x="33" y="27" />
20   <mx:TextInput id="txtUserName" y="53" right="10" left="88" />
21   <mx:TextInput id="txtEmail" y="81" left="88" right="10" />
22
23   <mx:Label x="33" y="83" text="email:" />
24   <mx:Label x="33" y="55" text="Name:" />
25
26   <mx:Button label="Add Contact" right="10" top="111" click="onAddContact();" />
27
28 </mx:Canvas>
29
30
  
```


UM Cairngorm^{v1.0} Event Hooks Gotchas

This sounds perfect...



If the target dispatching the event is NOT a UIComponent subclass, this does NOT work!

VO subclasses, that are [bindable], may manually dispatch; events from these will NOT be delivered to the MVC business layers.

Sprite subclass or MovieClip instances will NOT dispatch events to the business layers; only UIComponent subclasses.



Both of these situations require either

- (a) event.dispatch()
- (b) CairngormEventDispatcher.getInstance().dispatch(event)

...if the event is a CairngormEvent subclass

UM Cairngorm^{v1.0} MVC

What if your event dispatch never reaches your command . . .

Debugging steps:

- ✦ Are you actually dispatching a **CairngormEvent** subclass ?
- ✦ Are you instantiating a **FrontController** at the mx:Application level ?
- ✦ Have you **registered** – in the FrontController – the Command that should handle the event ?
- ✦ Are you dispatching the event from a **UIComponent** subclass ?

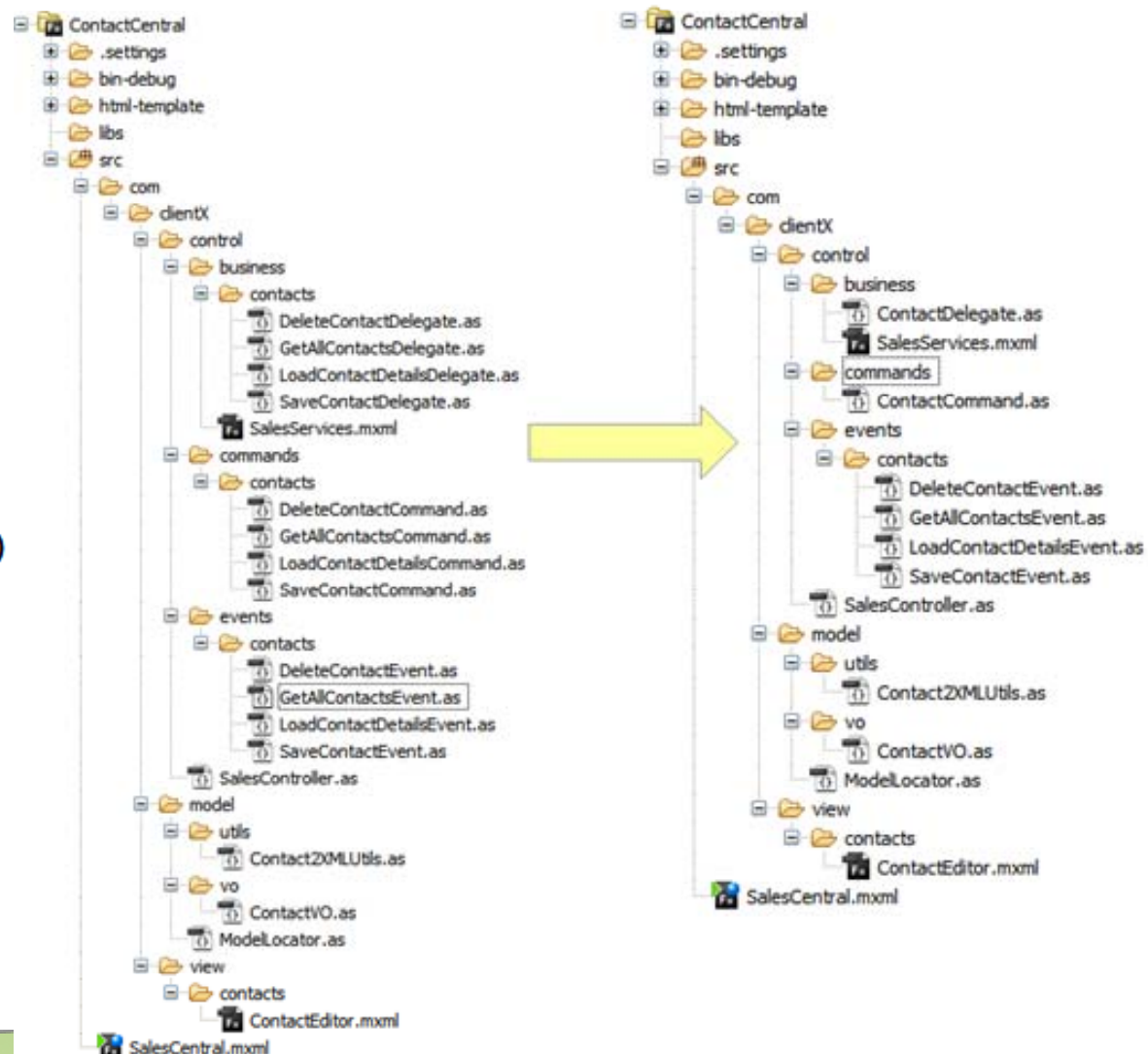
Other aspects of **UM Cairngorm Extensions**

Adobe Cairngorm

1. Uses MVC pattern
2. Poor support of Webservices initializations
3. Enterprise issues view notifications
4. Modules not supported

Universal Cairngorm Extensions

1. Support for view responders
2. Support for command event dispatching
3. Superior support for delegate queues
4. Support for batch events (parallel and/or sequential)
5. Support for delegate-level data transformations
6. Support for command logic aggregations
7. Mini-MVC apps/modules supported
8. Usable for full non-ui **FlexUnit** testing
9. Event **Hooks** for Event Dispatching from views



Resources for Continuous Testing

- **Cairngorm Frameworks**

<http://code.google.com/p/as3flexunitlib/>

<http://www.darronschall.com/weblog/archives/000216.cfm/> [Getting Started w/ FlexUnit]

- **Blogs**

http://subversion.tigris.org/getting_subversion.html

<http://cruisecontrol.sourceforge.net/>

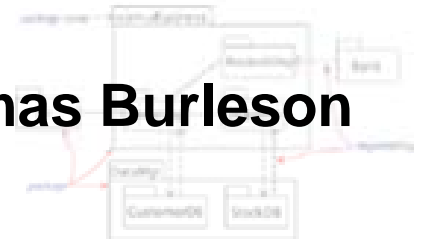
<http://ant.apache.org/>

- **Source Code**

http://subversion.tigris.org/getting_subversion.html



Thomas Burleson



**PRINCIPAL ARCHITECT
CERTIFIED INSTRUCTOR**

Adobe® Flex™



UNIVERSAL MIND

HIGH IMPACT CONSULTING

ThomasB@UniversalMind.com

<http://www.thomasburleson.biz/>