UNIVERSAL MIND
enterprise RIA

Effectively using the Cairngorm Extensions

Presented by:

Thomas Burleson,
Principal Architect, Universal Mind

August 13, 2008

UNIVERSAL MIND

Fx

Speaker:

Thomas Burleson, a principal architect at Universal Mind, has mentored, lead development teams, and delivered RIA solutions for companies such as SAP, Oracle, Sherwin Williams, Oppenheimer Funds, and many other commerical software applications.

He is an Adobe Certified Trainer for Flex, the author of the Adobe Cairngorm courseware, the founder of the Cairngorm Extensions, the creator of the Universal Mind Flex Exporter SDK, and a key contributor to the new SpatialKey SDK.

SpatialKey

Agenda:

Why do Flex developers have resistance or objections to using Cairngorm MVC?
How do developers address well-known issues and concerns when using the standard Cairngorm micro-architecture?

Learn about these issues and discover why the Cairngorm extensions were created! Learn about the key components of the “extensions” and how to effectively them within your RIAs. See a road map for future, pending enhancements and possible integration into new Cairngorm open source site.

Audience:

Developers familiar with Cairngorm. Developers who encountered problems with using Cairngorm within “enterprise” applications.

Take Away:

Familiarity with Cairngorm Extensions and understanding of the problems those extensions offer to developers using the Cairngorm MVC framework.

Adobe Cairngorm

#1 Flex MVC framework...

- Recently open sourced @ http://openSource.adobe.com/cairngorm/
- Can be compiled with Enterprise LCDS features or standalone
- Available for Flex 1.5, 2.01 or Flex 3

- Applies design patterns to encourage MVC layers in Flex RIAs
- No centralized documentation, tutorials, samples… yet!

- Has many interfaces without concrete classes
- Developers commonly encounter scalability and anti-patterns.

Cairngorm

Confusion & Slow Adoption…

- No centralized documentation and lack of non-trivial applications.
- Poor industry-awareness of Cairngorm [MVC] benefits.
- Abstract with few concrete classes
- ModelLocator Anti-patterns
- Confusion regarding uses & benefits of Delegates layer
- ViewHelper and ViewLocator complexities/deprecations

Adobe Cairngorm

1. Missing best practices for RPC responses and Services configurations
2. Missing best practices for Delegate layers

3. Concerns with huge quantity of “event-command-delegate” classes
4. Missing event sequencing
5. Missing server-call sequencing

6. Missing centralized fault handling
7. Missing view notifications
8. Missing view-direct data delivery
9. Missing support for modules and sub-MVC apps

Cairngorm Extensions

- Dependency injection and IOC features
- Support for DataManagement and Publish/Subscribe functionality
- Features for view controllers
- Fixes to ServiceLocator singleton constraints

Cairngorm [Future]

Cairngorm Extensions

Enhancements to make Cairngorm MVC easier…

- Open source @ http://code.google.com/p/flexcairngorm/
- Can be compiled with Cairngorm or separately
- Available for Flex 2.01 or Flex 3

- Make using Cairngorm easier
- Addresses some “complaints” and “issues” with Adobe Cairngorm

- Make common MVC patterns easier to use
- Encourages several best practices

Cairngorm Extensions

- Improvements to FrontController for subModules and “direct” listeners
- Improvements to ServiceLocator for timeouts and WSDL error reporting

Minor Enhancements

1. Command aggregation of events; by context
2. Delegates for data transformations or server sequencing
3. View notifications
4. Enable view-direct data delivery
5. Support for centralized fault handling

Major Enhancements

- Event Generator: used to spawn sequences or batches of events
- PollingEventGenerator: used to periodically spawn batches of events

New Features

Foundation for Cairngorm:
Issue #1: Missing Best Practices for RPC

Services.mxml - loginService uses global eventhandlers; assignment purpose unknown

```
<cairngorm:ServiceLocator
    xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:cairngorm="com.adobe.cairngorm.business.*">

    <mx:RemoteObject id="loginService" destination="loginService"
                     result="event.token.resultHandler( event );"
                     fault="event.token.faultHandler( event );">
    </mx:RemoteObject>

</cairngorm:ServiceLocator>
```

Delegates add handlers manually to dynamic properties (no type checking); rigid relationships

```
    public class LoginDelegate
    {
        public function login( loginVO : LoginVO ): void
        {
            var token : AsyncToken = service.login( loginVO );
                token.resultHandler = responder.onResult;
                token.faultHandler = responder.onFault;
        }
    }
}
```

flash.net.Responder

Foundation for Cairngorm:
Best Practices - Using Responders

Services.mxml is simple registry of RPC with methods for proxy access to server APIs

```
<cairngorm:ServiceLocator
    xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:cairngorm="com.adobe.cairngorm.business.*">

    <mx:RemoteObject id="loginService" destination="loginService">
        <mx:method name="login" concurrency="last" />
    </mx:RemoteObject>

</cairngorm:ServiceLocator>
```

Delegates add responder to EACH service invocation.

```
    public class LoginDelegate
    {
        private var responder : Responder;
        private var service : Object;

        public function LoginDelegate( responder : IResponder)
        {
            this.responder = responder;
            this.service   = ServiceLocator.getInstance().getService( "loginService" );
        }

        public function login( loginVO : LoginVO ): void
        {
            var token : AsyncToken = service.login( loginVO );
                token.addResponder(this.responder);
        }
    }
}
```

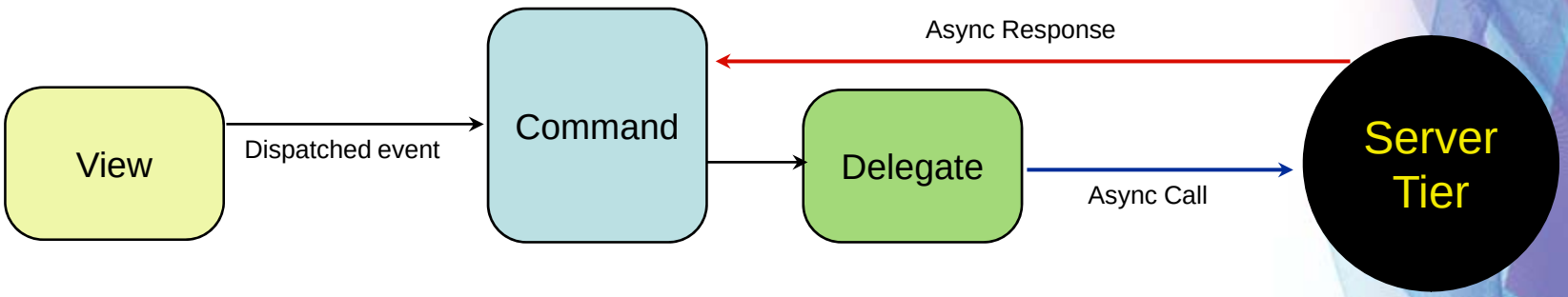
mx.rpc.Responder

Cairngorm Extensions (CE):

Responders in CE

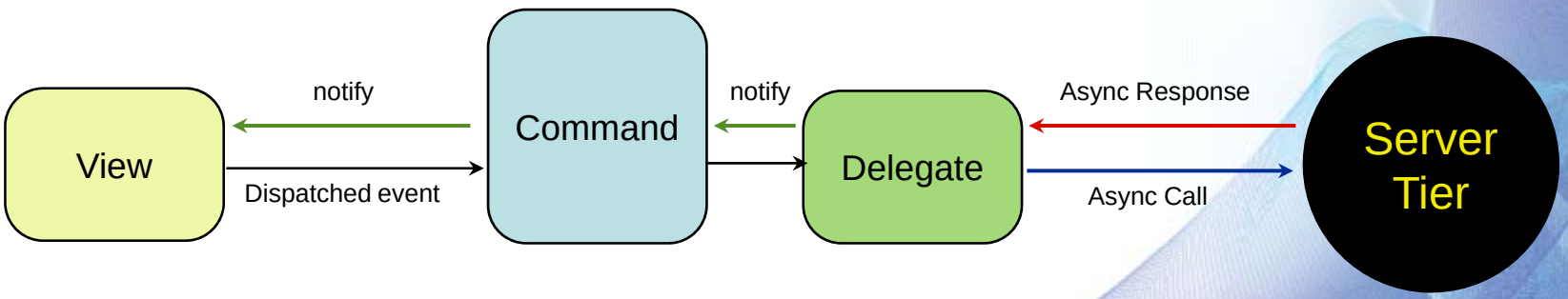
Foundation for Cairngorm Extensions:
Best Practices - Using Responders

Shown below is traditional, “interpreted” usage for event-command-delegate processing in Cairngorm business layers. Here the async response is delivered directly to Command::result() or Command::fault().



YES

Using Responders is fundamental to “**flex**”ible event handling. Cairngorm Extensions use responders at the view, command, and delegate layers. Here the delegate “intercepts” the server response, then notifies its caller [command], which in turn notifies its caller [if available].



Cairngorm Extensions:
Using Responders in Delegates

```
12  public class GalleryDelegate extends Delegate
13  {
14      /**
15       * GalleryDelegate Constructor
16       * @param responder
17       */
18      public function GalleryDelegate(responder:IResponder=null)
19      {
20          super(responder, GALLERY_SERVICE);
21      }
22
23      /**
24       * Load the Photo XML layer data using photo UUID
25       * @param photoID
26       */
27      public function loadThumbnails(session:SessionVO ):void {
28          var intercept : Callbacks  = new Callbacks(onResults_loadThumbnails);
29          var token     : AsyncToken = service.loadGalleryPhotos(session);
30
31          prepareHandlers(token,intercept);
32      }
33
34      private function onResults_loadThumbnails(event:ResultEvent):void {
35          var results    : Array   = [];
36          var items      : XMLList = (event.result as XML).photo;
37          for each (var it:XML in items) {
38              var thumbnail : GalleryThumbnailVO = new GalleryThumbnailVO().initialize(it);
39              results.push( thumbnail );
40          }
41
42          notifyCaller(results);
43      }
44
45      private static const GALLERY_SERVICE        : String = "galleryServices";
46
47  }
48 }
```

Uses CallBacks implements mx.rpc.IResponder

Here the delegates “intercepts” the response to transform incoming data to formats Expected by the RIA. Delegates are great places to:

1) Transform outgoing data to formats expected by server; e.g. restful or JSON outputs
2) Transform incoming data to formats expected by RIA; e.g. bindable Vos
3) Can internal sequence multiple calls to the server.

Cairngorm Extensions:
Using Responders in Commands

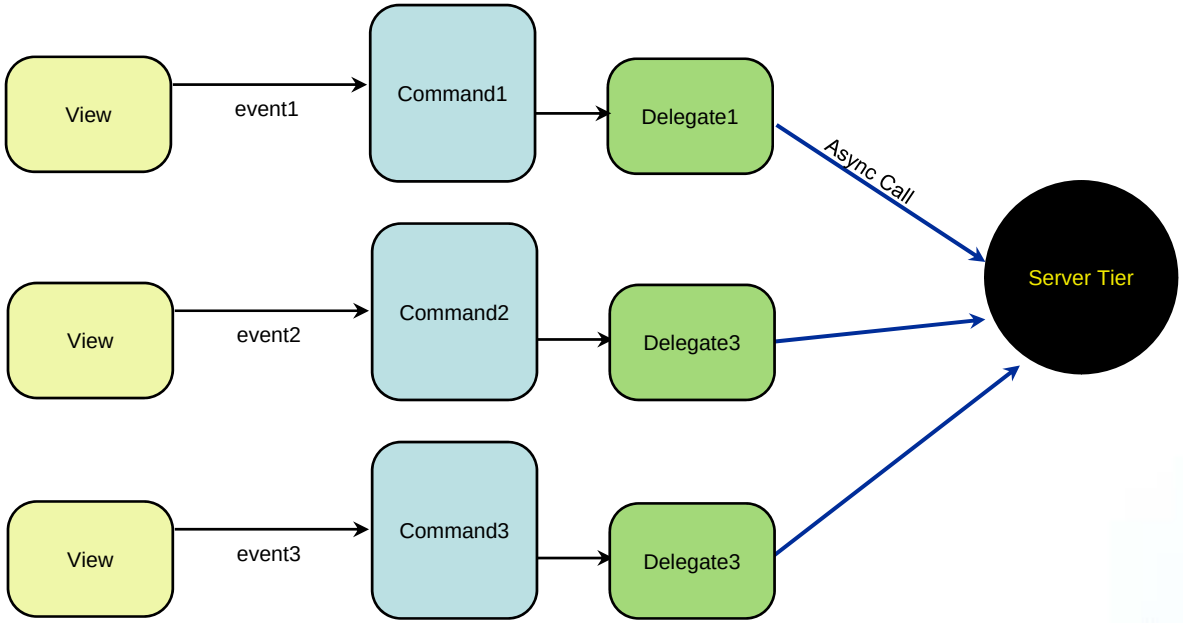
```
14  public class PatientVisitsCommand implements ICommand {
15
16      public function execute(event:CairngormEvent):void {
17          switch(event.type) {
18              case SavePatientRequestEvent.EVENT_ID : saveVisitRequest(event as SavePatientRequestEvent);     break;
19
20              default                               : break;
21          }
22      }
23
24      // ************************************************************************
25      // Forward request to Delegate for remote, server-tier business services
26      // This is an asynchronous feature
27      // ************************************************************************
28
29      private function saveVisitRequest(event:SavePatientRequestEvent):void {
30          var responder : IResponder            = new Responder(onResults_saveVisitRequest,onFault);
31          var delegate  : PatientVisitsDelegate = new PatientVisitsDelegate(responder);
32
33           delegate.saveVisitationRequest(event.patientID,event.firstName,event.lastName);
34      }
35
36      // ************************************************************************
37      // Event Handlers for asynchronous calls to server.
38      // ************************************************************************
39
40      private function onResults_saveVisitRequest(event:ResultEvent):void {
41          Alert.show("Your Appointment Request has been Processed!");
42      }
43
44      private function onFault(event:FaultEvent):void {
45          Alert.show(event.fault.faultCode + "\n"
46              + event.fault.faultString + "\n" + event.fault.faultDetail,
47              "Error");
48      }
49
50  }
51 }
```

Use mx.rpc.Responder to "proxy"

Cairngorm Extensions:

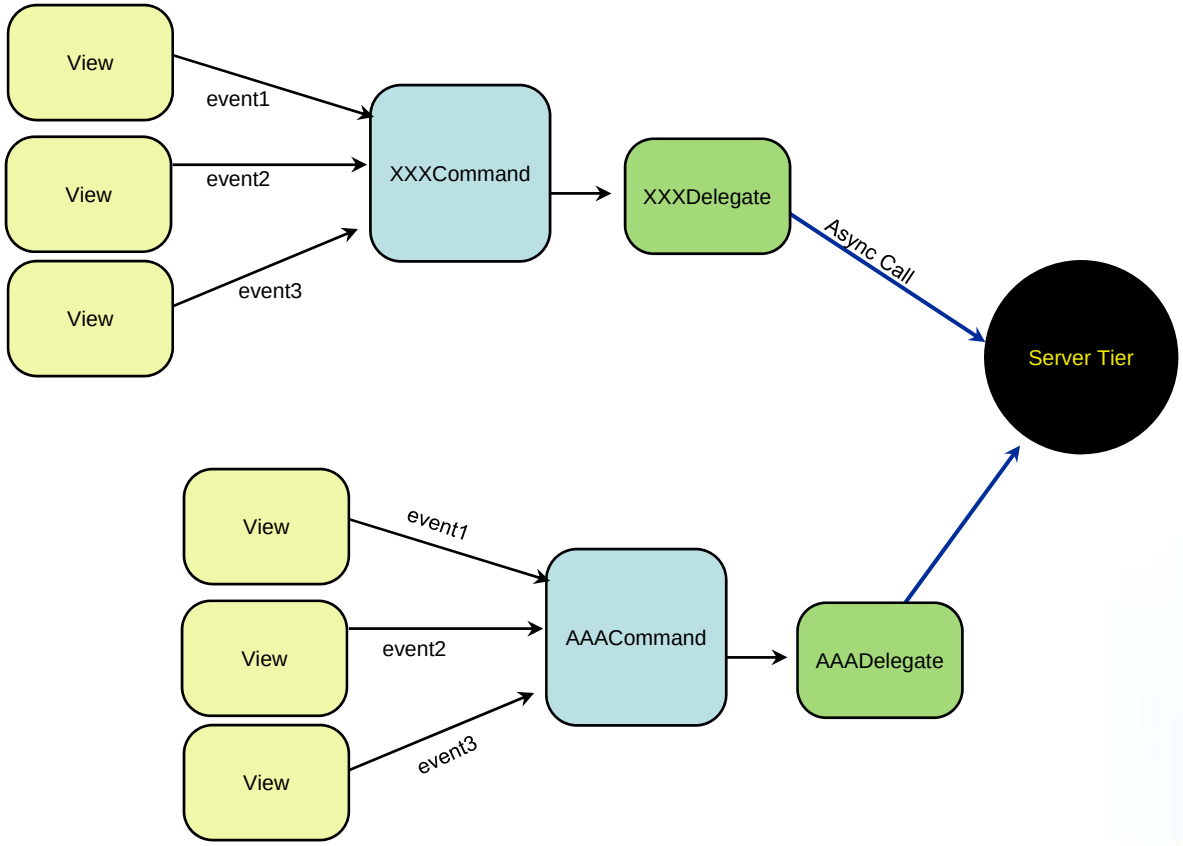
Command-Event Aggregation

Cairngorm Extensions:
Issue #3: Too many classes…



Traditionally, each event is processed by unique command and delegate classes (not instances). This quickly becomes unmanageable… 100 events → 300 classes.

Cairngorm Extensions:
Command aggregation of Events (by context)

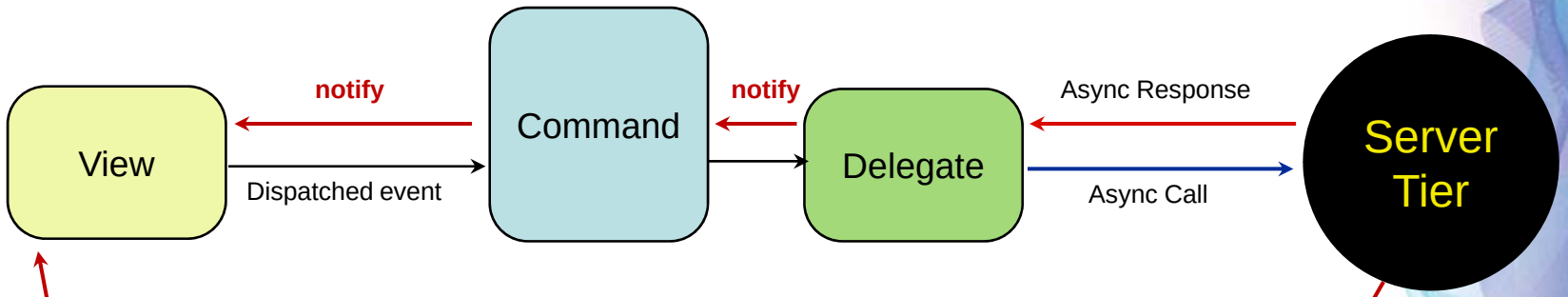


Use Responders inside of Commands to have individual METHODS serve has handlers!

Cairngorm Extensions:

View Notifications using Responders

Cairngorm Extensions:
Using Responders for View Notifications




```
14  public class PatientVisitsCommand implements ICommand {
15
16      public function execute(event:CairngormEvent):void {
17          switch(event.type) {
18              case SavePatientRequestEvent.EVENT_ID : saveVisitRequest(event as SavePatientRequestEvent);     break;
19
20              default                               : break;
21          }
22      }
23
24      // ********************************************************************
25      // Forward request to Delegate for remote, server-tier business services
26      // This is an asynchronous feature
27      // ********************************************************************
28
29      private function saveVisitRequest(event:SavePatientRequestEvent):void {
30          var responder : IResponder            = new Responder(onResults_saveVisitRequest,onFault);
31          var delegate  : PatientVisitsDelegate = new PatientVisitsDelegate(responder);
32
33           delegate.saveVisitationRequest(event.patientID,event.firstName,event.lastName);
34      }
35
36      // ********************************************************************
37      // Event Handlers for asynchronous calls to server.
38      // ********************************************************************
39
40      private function onResults_saveVisitRequest(event:ResultEvent):void {
41          notifyCaller(event.result);
42      }
43
44      private function onFault(event:FaultEvent):void {
45          Alert.show(event.fault.faultCode + "\n"
46              + event.fault.faultString + "\n" + event.fault.faultDetail,
47              "Error");
48      }
49
50  }
51 }
```

```
12  public class GalleryDelegate extends Delegate
13  {
14      /**
15       * GalleryDelegate Constructor
16       * @param responder
17       */
18      public function GalleryDelegate(responder:IResponder=null)
19      {
20          super(responder, GALLERY_SERVICE);
21      }
22
23      /**
24       * Load the Photo XML layer data using photo UUID
25       * @param photoID
26       */
27      public function loadThumbnails(session:SessionVO ):void {
28          var intercept : Callbacks  = new Callbacks(onResults_loadThumbnails);
29          var token     : AsyncToken = service.loadGalleryPhotos(session);
30
31          prepareHandlers(token,intercept);
32      }
33
34      private function onResults_loadThumbnails(event:ResultEvent):void {
35          var results    : Array   = [];
36          var items      : XMLList = (event.result as XML).photo;
37          for each (var it:XML in items) {
38              var thumbnail : GalleryThumbnailVO = new GalleryThumbnailVO().initialize(it);
39              results.push( thumbnail );
40          }
41
42          notifyCaller(results);
43      }
44
45      pri    static const GALLERY_SERVICE        : String = "galleryServices";
46
47  }
48 }
```

Uses cached responder to deliver response

Cairngorm Extensions:

Event Generators

Cairngorm Extensions:
Using Event Generators

```
<generator:EventGenerator       id="startUpEvents"   failCount="0"
                                trigger="{EventGenerator.TRIGGER_SEQUENCE}"
                                xmlns:generator="com.universalmind.cairngorm.events.generator.*">
    <ev1:LoadGalleryConfigurationsEvent             xmlns:ev1="com.eshots.opm.control.events.gallery.*"/>
    <ev2:LoadGalleryPhotosEvent                     xmlns:ev2="com.eshots.opm.control.events.gallery.*" />
    <ev3:LoadEditorConfigurationsEvent              xmlns:ev3="com.eshots.opm.control.events.editor.*"/>
</generator:EventGenerator>
```

```
// Start the EventGenerator
startupEvents.dispatch();
```

Notifications at Batch level:

```
<generator:EventGenerator id="batch"
          result="handleResult( event )"
          fault="handleFault( event )"
          trigger="sequence,parallel">
```

Notifications at Event level

```
<generator:events>
        <events:LoadConfigurationEvent
          callbacks="{ new Callbacks( result,
fault ) }" />
```

Cairngorm Extensions:
Nesting Event Generators

```
<!-- Event generator to run a app STARTUP -->
<generator:EventGenerator    trigger="sequence" id="startUpEvents"   xmlns:generator="com.universalmind.cairngorm.events.generator.*">
   <ev1:LoadConfigurationEvent    xmlns:ev1="com.xxx.app1.controller.events.*" />
   <ev2:LoadUserFromCacheEvent    xmlns:ev2="com.xxx.app1.controller.events.user.*" />

   <!-- Now run the following events in PARALLEL -->
   <generator:EventGenerator  trigger="parallel" xmlns:generator="com.universalmind.cairngorm.events.generator.*">

      <ev4:LoadInventoryCodesEvent xmlns:ev4="com.xxx.app1.controller.events.stocks.*"/>

      <!-- Run these 2 events in sequence -->
      <generator:EventGenerator  trigger="sequence"  xmlns:generator="com.universalmind.cairngorm.events.generator.*">
        <ev6:LoadMicroNewsEvent    criteria="{ __model.criteria }"          xmlns:ev6="com.xxx.app1.controller.events.news.*"/>
        <ev7:LoadVolumeChartEvent  criteria="{ __model.volumeDetails }"     xmlns:ev7="com.xxx.app1.controller.events.chart.*"/>
      </generator:EventGenerator>

      <ev5:LoadMainNewsEvent     criteria="{ __model.criteria }"          xmlns:ev5="com.xxx.app1.controller.events.news.*"/>

   </generator:EventGenerator>
</generator:EventGenerator>
```

Cairngorm Extensions:
Features still missing…

- Dependency injection and IOC features
- Centralized Configuration MXML file for full IOC
- Support for DataManagement and Publish/Subscribe functionality
- Features for view controllers
- Fixes to ServiceLocator singleton constraints

PRINCIPAL ARCHITECT
CERTIFIED INSTRUCTOR
Adobe® Flex™

- ThomasB @ UniversalMind.com
- http://www.linkedin.com/profile?key=8912571
- Blogs
  - http://www.technicallyrandom.com/
  - http://www.gridlinked.info/Developer/Blog/Blog.html
  - http://blog.universalmind.com/
- Courseware
  - Introduction to Cairngorm - http://www.adobe.com/devnet/flex/articles/introducing_cairngorm.html
  - Architecting Applications with Cairngorm – not publically available (yet).