

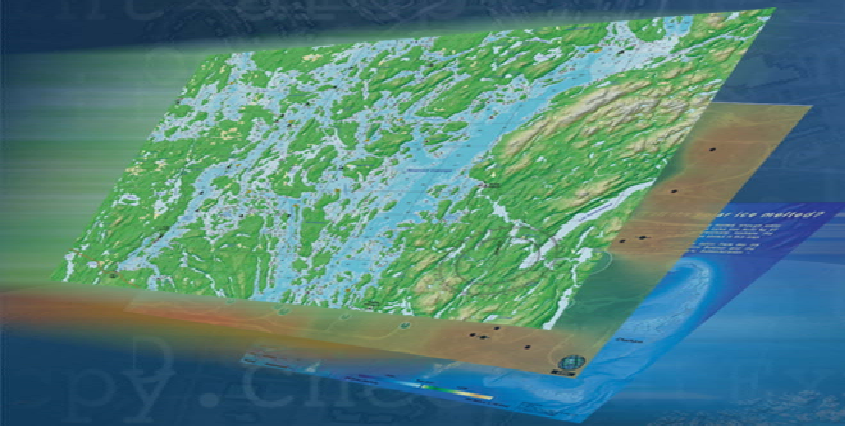
# ESRI Developer Summit

March 22–25, 2010  
Palm Springs, CA

## Patterns and Best Practices for Building Applications with ArcGIS API for Flex

*Antony Jayaprakash*

*Mansour Raad*



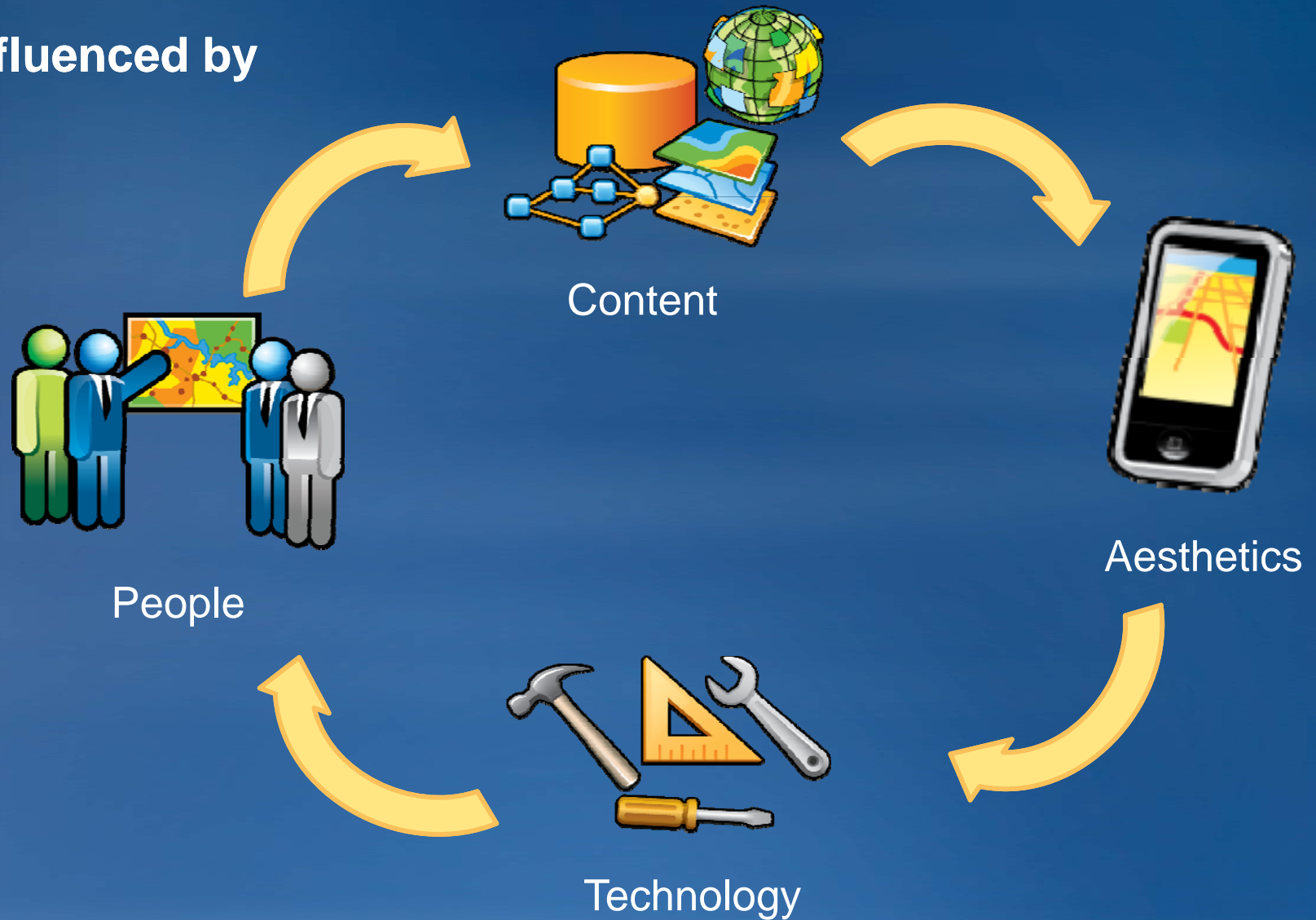
# Agenda

- Today we will cover
  - User experience
  - Implementation
  
- We will answer questions at the end of the session

*Please complete the session survey!*

# Great Web Applications

- Influenced by



**U**ser **E**Xperience

# Paths to Good UX

- **Concepts in Flex**
  - Styling
  - Skinning
  - Effects
  - States
  - Layouts
  - Custom components
  
- **Tools**
  - Flash Builder
  - Flash Catalyst
  - Adobe Creative Suite (CS4)

# New Packages in Flex 4

## Flex 3

### mx Packages

`mx.controls.*`  
`mx.containers.*`  
`mx.effects.*`  
`mx.events.*`  
`mx.graphics.*`  
`mx.managers.*`  
...

## Flex 4

### `spark.components.*`

Application  
Border  
Button  
ButtonBar  
CheckBox  
Drop  
Gro  
List  
Skin  
...

### `spark.layouts.*`

BasicLayout  
HorizontalLayout  
TileLayout  
VerticalLayout  
...

### `spark.effects.*`

Animate  
AnimateColor  
Fade  
Move  
Move3D  
Rotate  
Rotate3D  
Resize  
...

### `spark.primitives.*`

BitmapImage  
Ellipse  
Graphic  
Line  
Path  
Rect  
VideoElement  
...

# Namespaces in Flex 4

- MXML 2006 Language namespace  
URI <http://www.adobe.com/2006/mxml>  
Default Prefix: mx
- MXML 2009 Language namespace  
URI <http://ns.adobe.com/mxml/2009>  
Default Prefix: fx
- Spark component namespace. Use with MXML 2009 Language namespace  
URI <library://ns.adobe.com/flex/spark>  
Default Prefix: s
- MX component namespace. Use with MXML 2009 Language namespace  
URI <library://ns.adobe.com/flex/halo>  
Default Prefix: mx

# Spark Component Model

Code

Component

Behavior  
Logic  
Data



Markup

Skin

Graphics  
Layout  
Animation  
Parts  
States





# Styles

- Type selectors need namespaces
- Spark components don't support Halo theme

```
<fx:Style>
  @namespace s "library://ns.adobe.com/flex/spark";
  @namespace mx "library://ns.adobe.com/flex/mx";
  s | Button {
    color: #FF0000;
  }
  mx | DateChooser {
    color: #FF0000;
  }
</fx:Style>
```

# Declarative Graphics

- New graphic primitives for drawing lines, ellipses and curves
- Two types of graphics
  - Static / Optimized FXG (.fxg files)
  - Runtime FXG / MXML Graphics

```
<s:Rect width="50" height="50" >  
  <s:fill>  
    <s:SolidColor color="0x2D92C7" />  
  </s:fill>  
  <s:stroke>  
    <s:SolidColorStroke color="#cccccc" weight="1"/>  
  </s:stroke>  
</s:Rect>
```

# States

- Help represent different visual behavior
- Use attributes `includeIn` & `excludeFrom`

```
<s:Button label="Submit" includeIn="completeState" />  
<s:Button label="Clear" excludeFrom="resetState" />
```

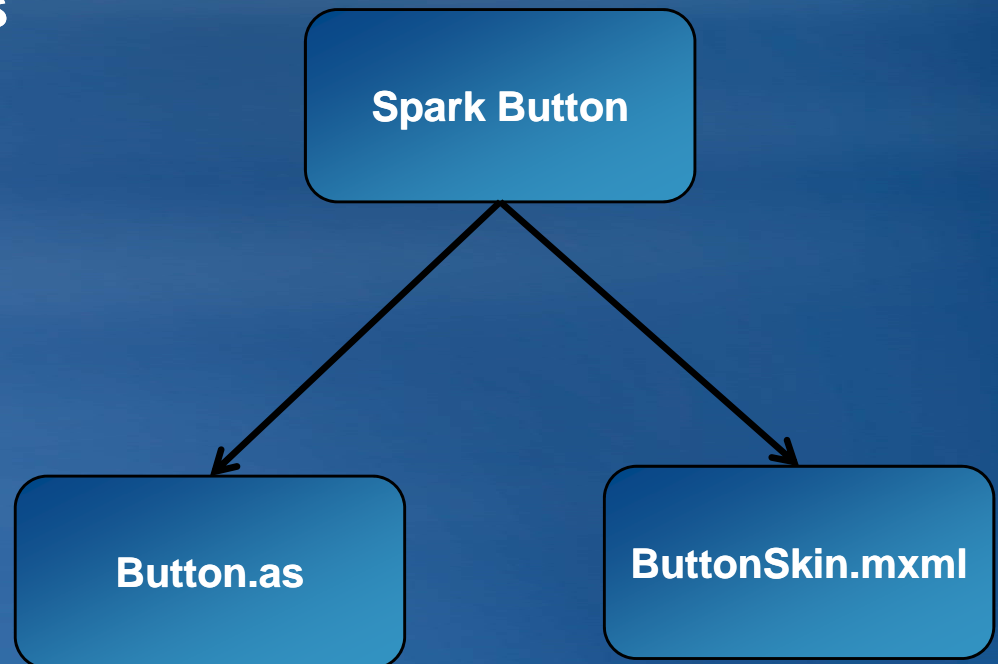
- New dot syntax

```
<s:TextInput color.errorState="0xFF0000"  
             color.validState="0x000000" />  
<s:Button click.submitState="submit()"  
          click.clearState="resetForm()" />
```

- States are used in Skin files

# Skins

- **Components all have separate skin files (e.g. ButtonSkin.mxml)**
  - Skin parts drawn with MXML graphics
  - Skins define states
- **To customize components**
  - Styles
  - Custom Skin

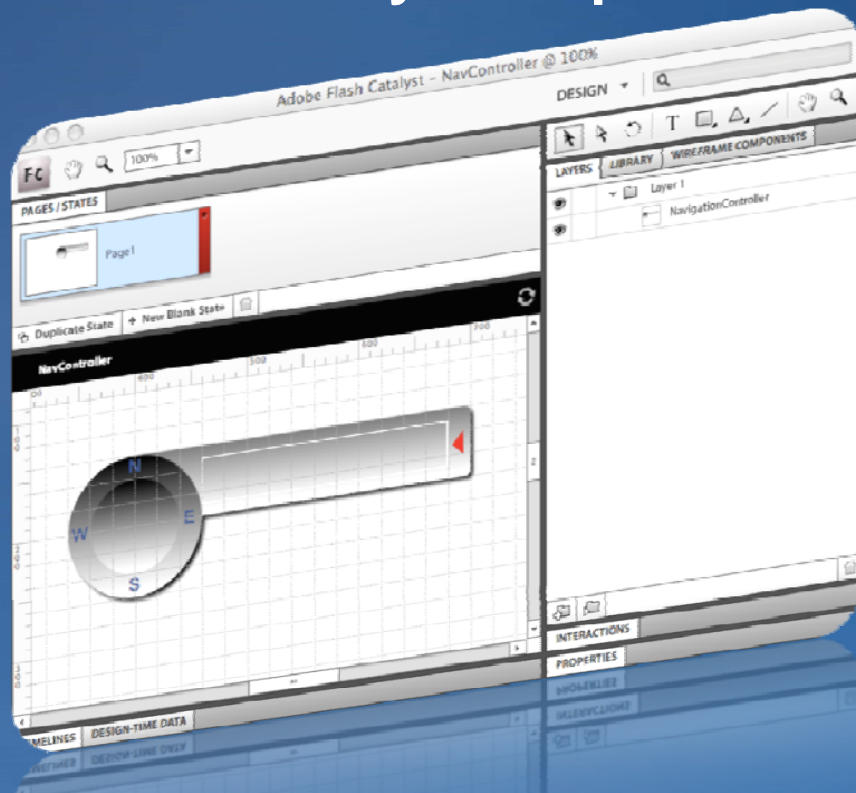


# Skinning Workflow

- Copy skin class
- Rename that skin
- Make modifications
- Skins should have
  - HostComponent defined
  - States
  - MXML Graphics
- Use the skinClass property to assign the custom skin

# Custom Components

- Wraps reusable functionality
- Division of labor
- Enhances maintainability
- Flash Catalyst simplifies component creation



# Summary

- **Enhance and create unique user experience**
  - Custom skins
  - Custom components
- **Do not design in isolation**
- **Consider interconnections with**
  - Context
  - Content
  - Users

# Implementation Design



# Debugging

```
trace( "m_value =", m_value);
```

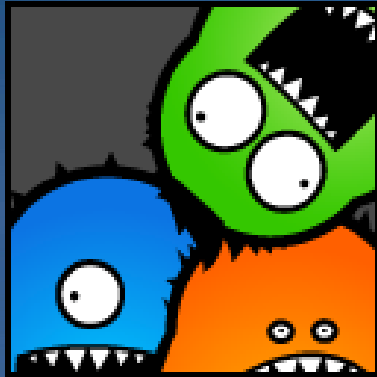
# Debugging

- De Monster Debugger

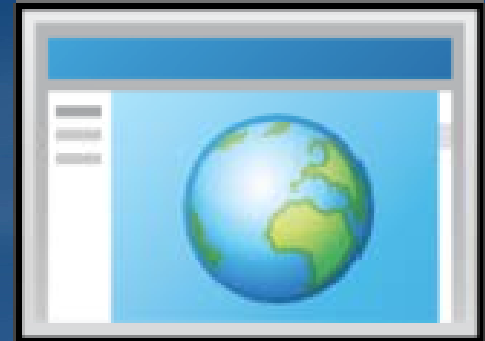


<http://www.demonsterdebugger.com/>

# Debugging



Local Connection



# Debugging

**De MonsterDebugger**

**LIVE APPLICATION**

- (SimpleMarkerSymbol)
  - alpha (Number) = 0.5
  - angle (Number) = 0
  - color (uint) = 16711680
  - outline (SimpleLineSymbol)
  - pattern (Array)
    - size (Number) = 20
    - style (String) = circle
    - xoffset (Number) = 0
    - yoffset (Number) = 0

**INSPECTOR**

**PROPERTIES** | **METHODS**

Name	Value
alpha	0.5
angle	0
color	
outline	
pattern	
size	
style	
xoffset	
yoffset	

**TRACES** Filter

#	Time	Target
1	20:28:42.196	DeMonsterApp0.myMap
2	20:29:17.539	DeMonsterApp0.myMap
3	20:29:21.53	DeMonsterApp0.myMap

De Monsters are hiring Adobe Flash and AIR developers!

**DeMonsterApp.html**

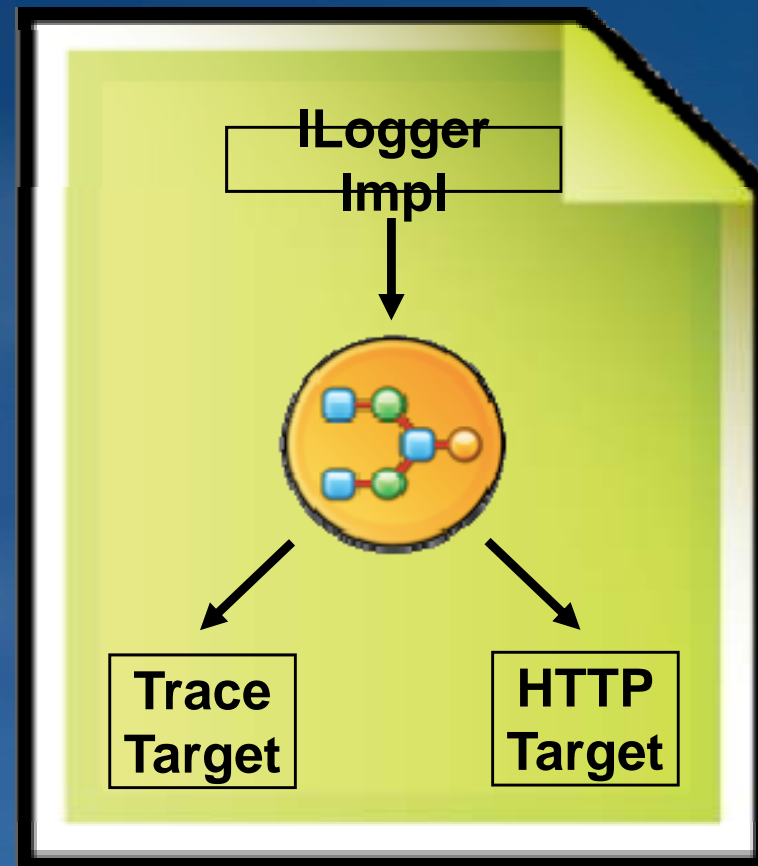
file:///Users/mraad/Flex3Works Google

Map interface showing a red dot, a scale bar (5000 km / 4000 mi), and a "POWERED BY ESRI" logo.

# Logging

# Logging

- Log4J
- Log4Net
- ILogger Implementation
- Log Mediator
- Log Target



# Logging

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="horizontal">
3   <mx:Script>
4     <![CDATA[
5       import mx.logging.ILogger;
6       import mx.logging.Log;
7
8       private var m_logger:ILogger = Log.getLogger("MyLogApp");
9
10      private function logInfo_clickHandler():void
11      {
12        if (Log.isInfo())
13        {
14          m_logger.info("This is an info {0} {1}", "message", 1);
15        }
16      }
17    ]]>
18  </mx:Script>
19  <mx:TraceTarget/>
20  <mx:Button label="Log Info" click="logInfo_clickHandler()"/>
21 </mx:Application>
```

# Logging

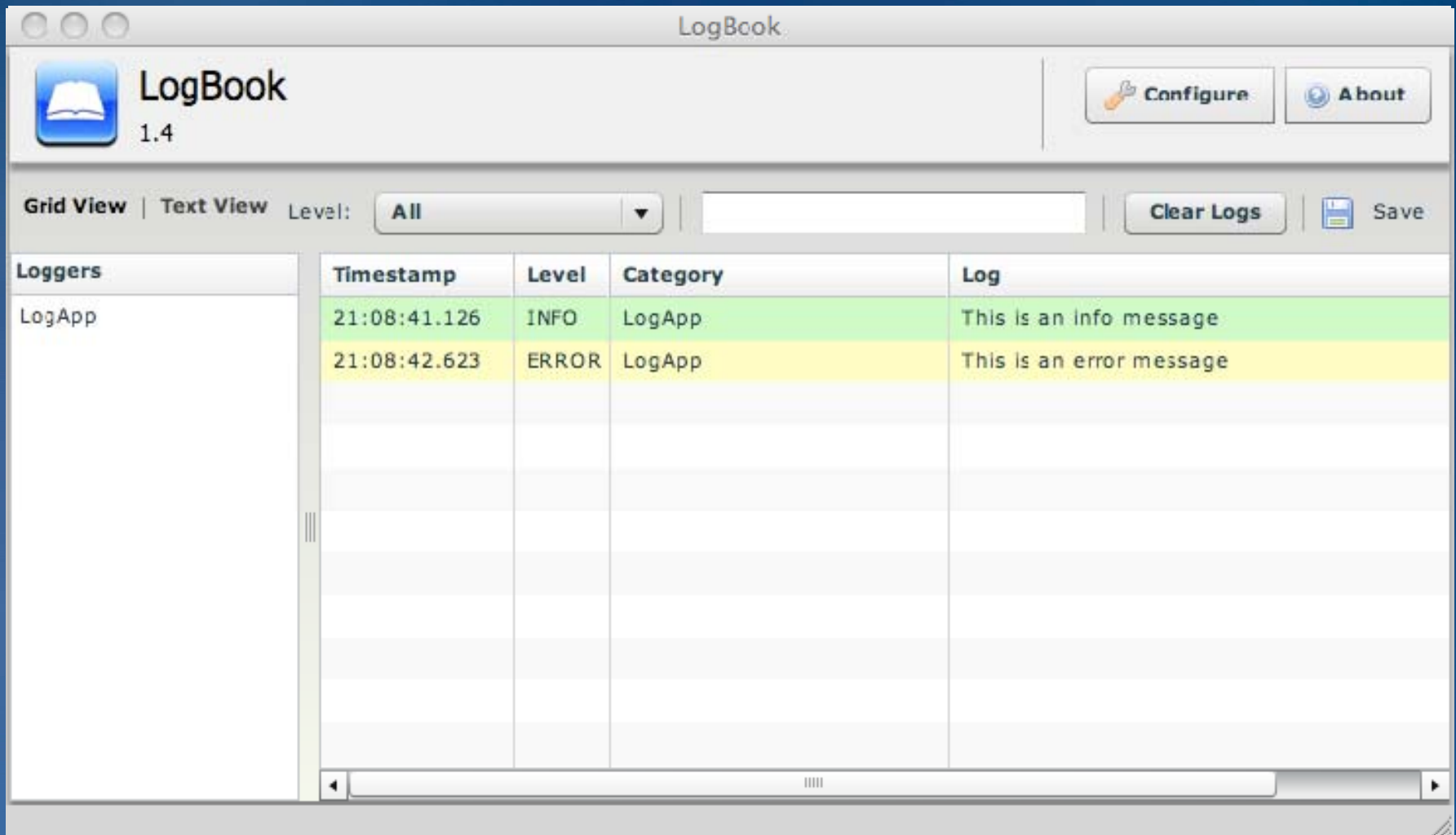
The screenshot shows a window titled "LogApp.html" with a header "Log App". Below the header is a table with two columns: "Category" and "Message". The table contains three rows of data, with the third row highlighted in yellow. Below the table is a control panel with three buttons: "Info", "Debug", and "Error". To the right of these buttons is a text input field containing "My Error Meeasge" (note the typo) and a "Save" button.

Category	Message
LogInfoCommand	[Message]
MainApplication	[Message]
MainApplication	My Error Meeasge

Info   Debug   Error   My Error Meeasge   Save



# Logging

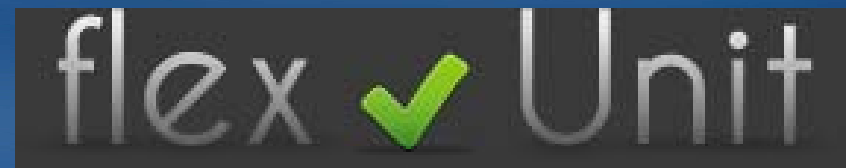


<http://code.google.com/p/cimlogbook/>

# Unit Testing

# Testing

- **TestMethod**
  - Smallest Unit Of Execution
  - Assert Condition At End Of Execution
  - [Test(description="My Test Description")]
- **Test Case**
  - Collection of TestMethods
  - [BeforeClass]
  - [Before]
  - [Test]
  - [After]
  - [AfterClass]
- **Test Suite**
  - Collection of Test Cases and or Suites
  - [Suite]



<http://www.flexunit.org>

# Testing

**[Test]**

```
Public function addition():void {  
    Assert.assertEquals(12, simpleMath.add(7, 5));  
}
```

**[Test]**

```
Public function subtraction():void {  
    Assert.assertEquals(9, simpleMath.subtract(23, 3));  
}
```

# Testing

## **[BeforeClass]**

```
Public static function runBeforeClass():void {  
    //run for one time before all test cases  
}
```

## **[AfterClass]**

```
Public static function runAfterClass():void {  
    //run one time after all test cases  
}
```

# Testing

```
[Suite]
[RunWith("org.flexunit.runners.Suite")]
Public class MySuite {
    public var t1:BasicMathTest;
    public var t2:MyTheory;
}
```

# Testing

```
[DataPoints]
[ArrayElementType("String")]
public static var stringValues:Array = ["one","two","three","four","five"];
```

```
[DataPoint]
public static var values1:int = 2;
[DataPoint]
public static var values2:int = 4;
```

```
[DataPoints]
[ArrayElementType("int")]
public static function provideData():Array {
    return [-10, 0, 2, 4, 8, 16 ];
}
```

```
[Theory]
public function testDivideMultiply( value1:int, value2:int ):void {
    assumeThat( value2, greaterThan( 0 ) );
    var div:Number = simpleMath.divide( value1, value2 );
    var mul:Number = simpleMath.multiply( div, value2 );
    Assert.assertEquals( mul, value1 );
}
```

```
[Theory]
public function testStringIntCombo( value:int, stringValue:String ):void {
    //call some method and do something
}
```

# Testing

FlexUnitApp.html

FlexUnit 4.0.0rc1 Runner powered by Adobe Consulting

### TEST CASES

Q Type filter All results

Case	Result	Expected	Actual
▼ Theory...	✔	0.00	
test...	✔	0	-
▼ CalcTest	✔	0.00	
test...	✔	0	-

### TEST SUITE

✔

- Tests run 2
- Time taken
- Errors 0
- Failures 0
- Ignored 0

### RESULT DETAILS

Case	CalcTest
Function	testCalc
Expected	-
Actual	-
Location	-
Message	-

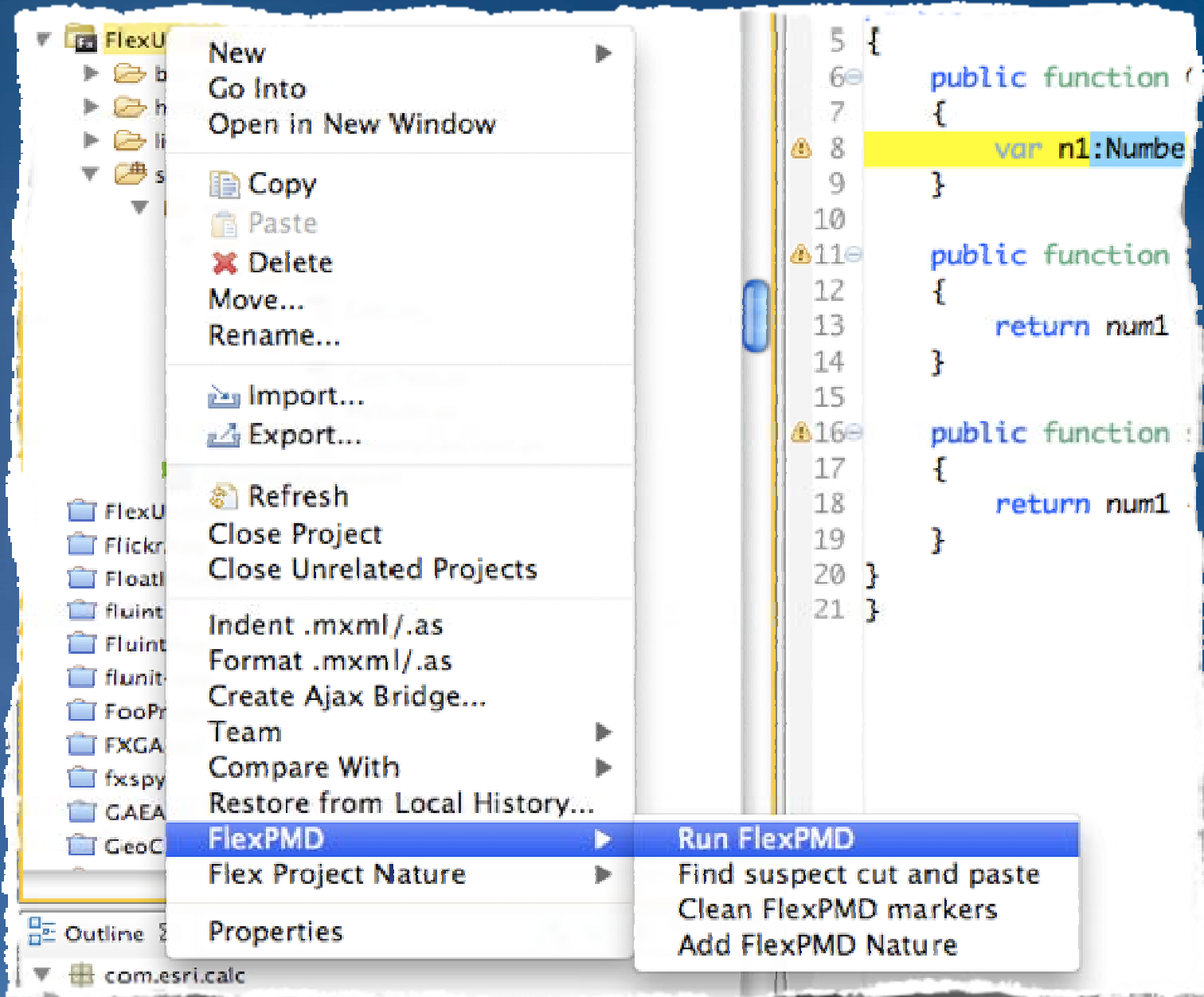


# Testing - “Programming Mistake Detector”

- **Possible Bugs**
  - empty try/catch
- **Dead Code**
  - unused variables
- **Suboptimal Code**
  - heavy constructor
- **Overcomplicated Expressions**
  - nested loops
- **Duplicated Code**
  - copy & paste



# Testing - “Programming Mistake Detector”



# Code Coverage

# Code Coverage

The screenshot shows the CoverageViewer application interface. At the top, there are buttons for 'Show details', 'Update', 'Search By:', 'Load Files...', 'Save Report...', and 'Reset Recording'. Below these is an 'Auto' checkbox. The main area is divided into two panes: 'Branches' and 'Lines'. The 'Branches' pane contains a tree view of the project structure with a table of coverage data. The 'Lines' pane shows the source code with line numbers and coverage indicators.

Name	Branch Coverage	Uncovered
[Application]	87.50% (14/16)	2
[top level]	100.00% (3/3)	0
com.esri.calc	75.00% (3/4)	1
Calc	75.00% (3/4)	1
add	100.00% (1/1)	0
Calc	100.00% (1/1)	0
mul	0.00% (0/1)	1
sub	100.00% (1/1)	0
com.esri.test	88.88% (8/9)	1
CalcTest	100.00% (4/4)	0
MySuite	0.00% (0/1)	1
MySuite	0.00% (0/1)	1
TheoryCalcTest	100.00% (4/4)	0
beforeTest	100.00% (1/1)	0
provideData	100.00% (1/1)	0
testAddSub	100.00% (1/1)	0
TheoryCalcTest	100.00% (1/1)	0

```
1 package com.esri.calc
2 {
3
4 public class Calc
5 {
6 26 public function +26Calc()
7 26 {
8 26     var n1:Number = 2;
9 26 }
10
11 26 public function +26add(num1:Number, num2:Number):Number
12 26 {
13 26     return num1 + num2;
14 26 }
15
16 0 public function +0mul(num1:Number, num2:Number):Number
17 0 {
18 0     return num1 * num2;
19 0 }
20
21 25 public function +25sub(num1:Number, num2:Number):Number
22 25 {
23 25     return num1 - num2;
24 25 }
25 }
26 }
```

# Code Coverage

Source  
Code



Coverage  
Compiler



Instrumented  
Code



Coverage  
Report



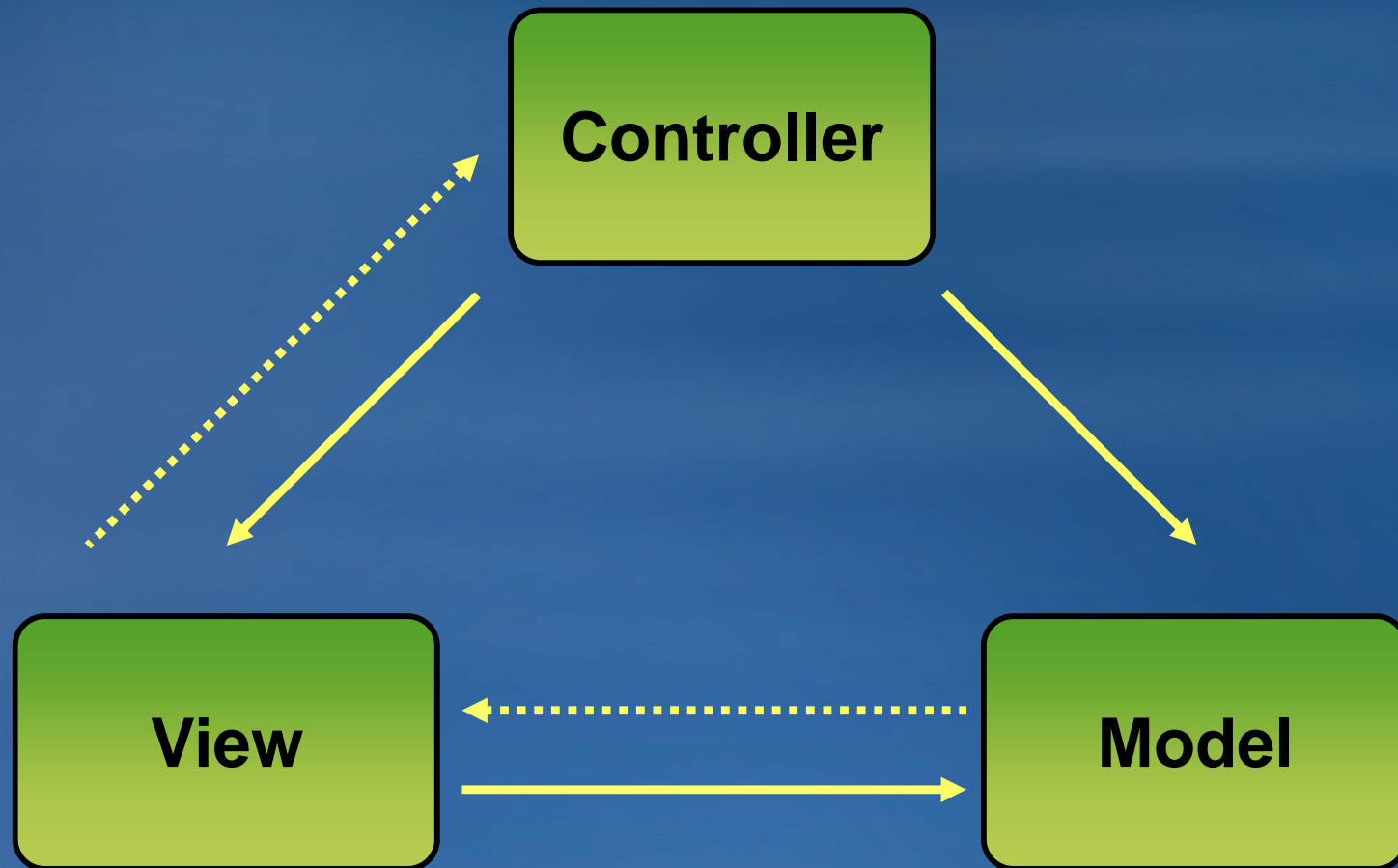
# Frameworks

# Lots Of Frameworks

- Cairngorm
- PureMVC
- Swiz
- Robotleg
- SpringActionScript
- Mate
- Your Own

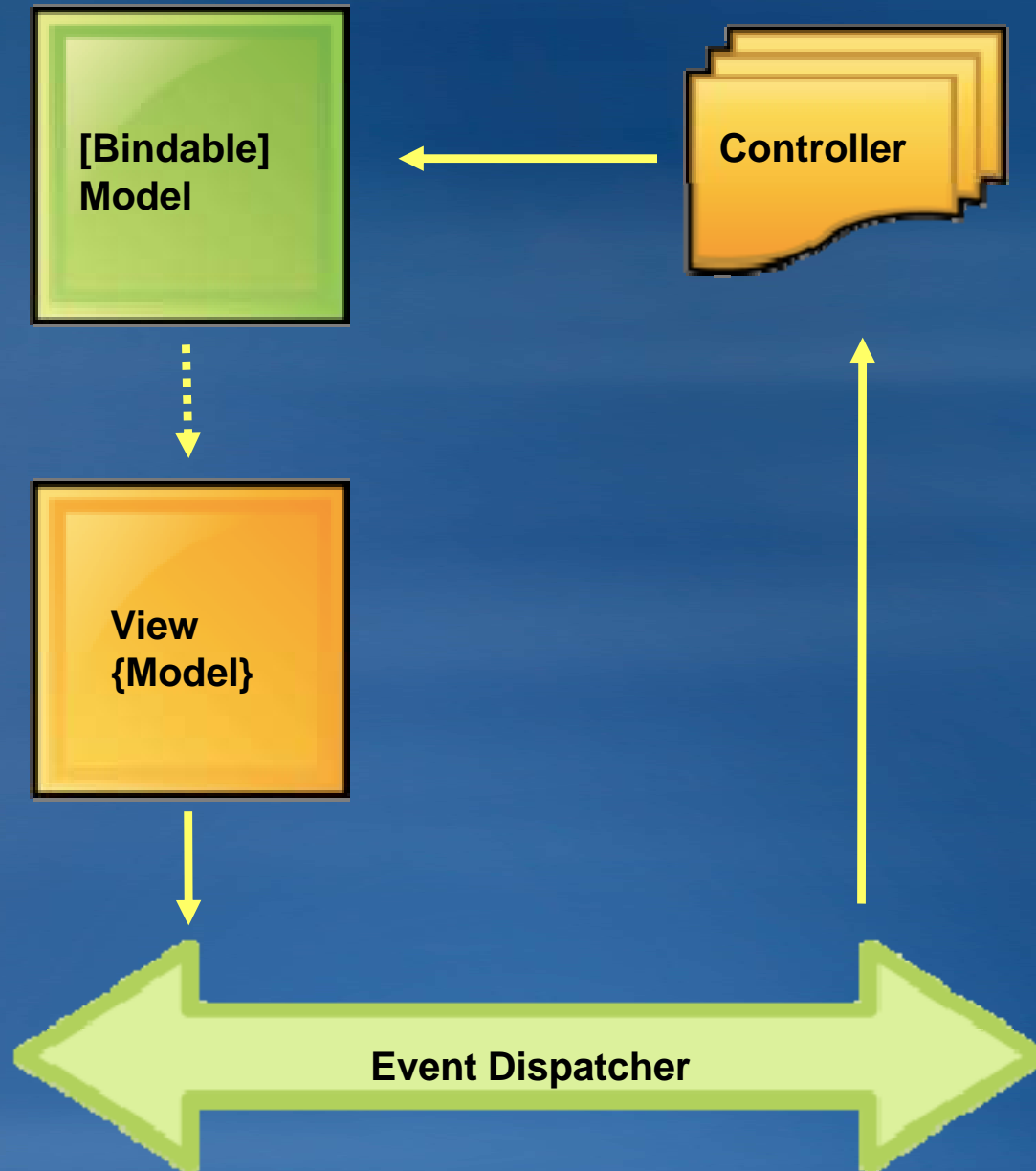
# Model View Controller

→ Associated  
.....▶ Notifies

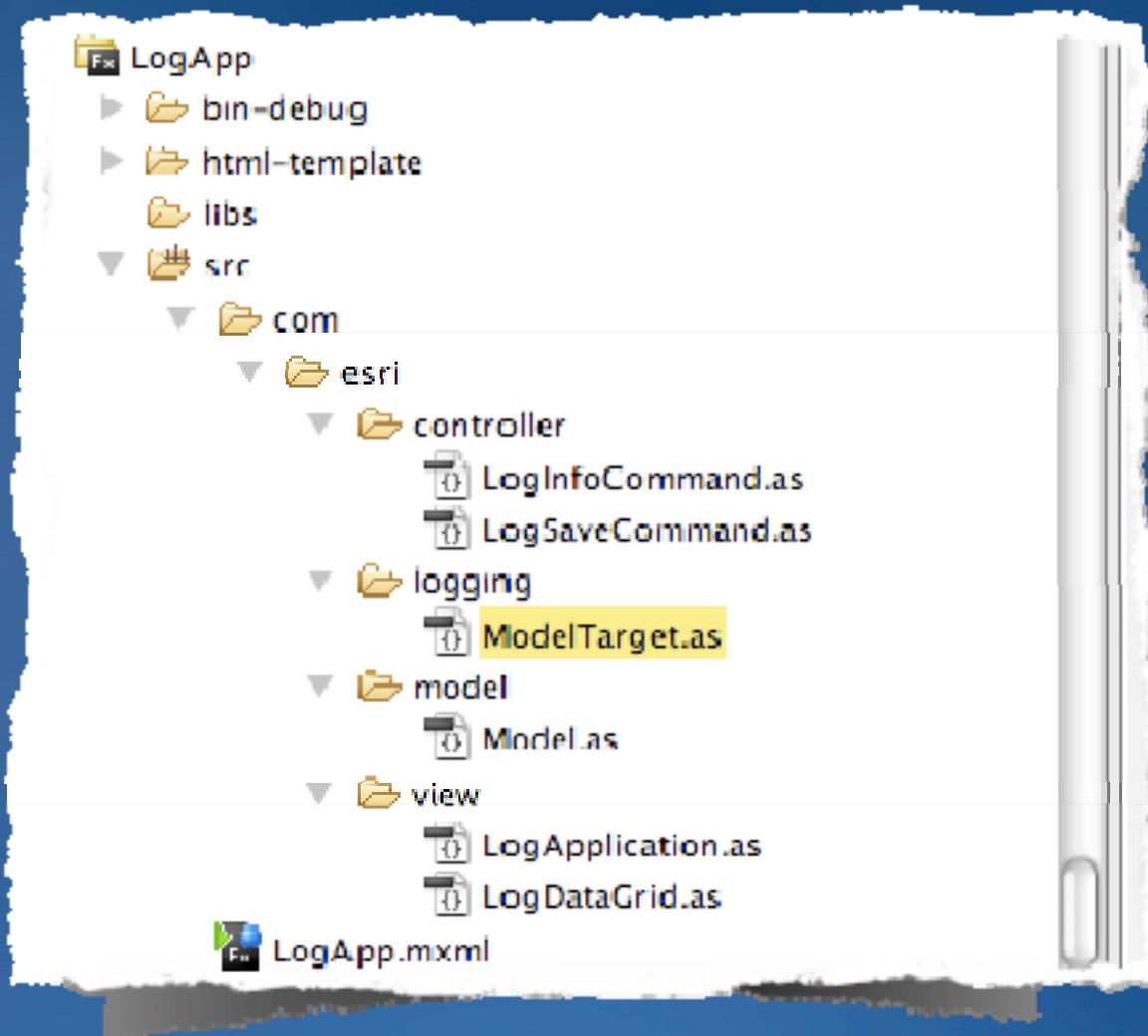




# Model View Controller



# Model View Controller



# Questions