Universal Mind™

Enterprise ActionScript

Writing clean code fast with Swiz 1.0

Universal Mind™

Ben Clinkinbeard

Technical Architect
Universal Mind

Universal Mind™

WTF is a Swiz?

When proven patterns and clean code love each other very much...

Tao of Swiz

What does Swiz believe in?

Swiz is unobtrusive

Imposes no patterns on your code

Swiz is unobtrusive

Requires minimal framework references

Universal Mind™

Swiz makes you productive

Virtually zero boilerplate

Swiz makes you productive

Expedites tedious tasks

Swiz is extensible

It doesn't do everything. On purpose.

Swiz gets out of your way

Focus on problems, not a framework

What is clean code?

- Separation of concerns
- Decoupled

Models

Models

Hold state

Models

Fairly dumb

Models

Broadcast system events

Models

Are not all created equal

Domain Models

Represent the real world

Application Models

Maintain system state

Universal Mind™

Presentation Models

Support views

Views

Views

Look pretty

Views

Handle user gestures

Views

Rely on presentation models

Controllers

Controllers

Manage one functional area

Controllers

Handle system events

Controllers

Interact with services (via delegates)

Controllers

Update models

Controllers

Can dispatch system events

Delegates

Delegates

Shield controllers from service implementations

Universal Mind™

Delegates

Great place to mock services

Delegates

Perform marshalling if needed

Universal Mind™

Swiz terminology

What does it all mean, Basil?

Universal Mind™

Swiz instance

Unit of awesomeness

Swiz instance

```
<swiz:Swiz>
    <swiz:config>
        <swiz:SwizConfig
            eventPackages="com.foo.events"
            viewPackages="com.foo.views.*" />
    </swiz:config>

    <swiz:beanProviders>
        <config:OrdersServiceBeans />
        <!--config:OrdersDelegateBeans /-->
        <config:OrdersMockDelegateBeans />
        <config:OrdersModelBeans />
        <config:OrdersControllerBeans />
    </swiz:beanProviders>
</swiz:Swiz>
```

Universal Mind™

SwizConfig

He made a slide for this?

Bean

An object managed by Swiz

BeanProvider

Provider of beans

BeanProvider

Declares non-view beans

Universal Mind™

Prototype

Bean++

Dependency

[Inject] marks the spot

Dispatcher

Shared event bus, injected by Swiz

[Dispatcher(“global”)]

Maps to root Swiz instance

Universal Mind™

[Dispatcher(“local”)]

Maps to *your* Swiz instance

Universal Mind™

Mediator

System event handler

[Inject]

Inject by type

```
[Inject]
public var model:UserModel;

[Inject]
public var delegate:IUserDelegate;
```

Inject by name

```
[Inject( “userService” )]
public var service:RemoteObject;

...

<mx:RemoteObject id=”userService” />
```

Inject bean property

```
[Inject( “userModel.currentUser” )]
public var currentUser:User;
```

Inject bean property

```
[Inject( “userModel.currentUser”, bind=”true” )]
public var currentUser:User;
```

Inject bean property

```
[Inject( “userModel.currentUser”, twoWay=”true” )]
public var currentUser:User;
```

Setter injection

[Inject]
public function setModel(model:UserModel):void
{
  this.model = model;
}

Class level injection

```
[Inject( source="userModel.currentMode",
destination="modeViewStack.selectedIndex" )]
```

Universal Mind™

[Mediate]

Mediator basics

```
[Mediate( “com.foo.events.UserEvent.ADD_USER” )]
public function addUser( event:UserEvent ):void{...}
```

Mediator basics

```
[Mediate( event = "UserEvent.ADD_USER" )]
public function addUser( event:UserEvent ):void{...}

...

<swiz:SwizConfig eventPackages="com.foo.events" />
```

Mediator basics

```
[Mediate( “UserEvent.CLEAR_ALL_USERS” )]
public function clearUsers():void{...}
```

Mediator scope

```
[Mediate( “UserEvent.CLEAR_ALL_USERS”, scope=“local” )]
public function clearUsers():void{...}
```

Mediator hotness

```
[Mediate( “UserEvent.ADD_USER”, properties=”user” )]
public function addUser( user:User ):void{...}
```

Mediator hotness

```
[Mediate( “UserEvent.ADD_USER”, properties=”user” )]
[Mediate( “UserEvent.EDIT_USER”, properties=”user” )]
[Mediate( “UserEvent.DELETE_USER”, properties=”user” )]
public function manageUser( user:User ):void{...}
```

Mediator hotness

```
[Mediate( “UserEvent.*”, properties=”user” )]
public function manageUser( user:User ):void{...}
```

Mediator options

priority, useCapture, stop(Immediate)Propagation

Go forth and kick ass

Bean lifecycle

[PostConstruct]
[PreDestroy]
BeanEvent

Service layer

ServiceHelper
URLRequestHelper
MockDelegateHelper

ServiceHelper

```
[Mediate( “UserEvent.ADD_USER”, properties=”user” )]
public function addUser( user:User ):void
{
    sh.executeServiceCall( delegate.insertUser( user ),
                           handleInsertUserResult, handleInsertUserFault,
                           [ user ] );
}

private function handleInsertUserResult( data:Object, user:User ):void
{
    user.isInserted = true;
}
```

Universal Mind™

MockDelegateHelper

```
public function getUsers():AsyncToken
{
    mdh.createMockResult( getFakeUsers(), 1000 );
}

private function getFakeUsers():ArrayCollection
{
    var users:ArrayCollection = new ArrayCollection();
    users.addItem( new User( “Ben” ) );
    users.addItem( new User( “Jeff” ) );
    return users;
}
```

MockDelegateHelper

```
public function getUsers():AsyncToken
{
    mdh.createMockResult( AMFUtil.getAMF3Data( MyMocks.USERS ), 1000 );
}
```

Modules

Custom metadata processors

[URLMapping(“login”)]

[URLMapping(“hello/{0}”, title=”Hello, {0}”)]

Custom metadata processors

[MediateSignal(“galleryUpdatedSignal”)]

Custom metadata processors

[Resource(key=”title”, bundle=”example”)]

Custom metadata processors

```
[Scheduled( delay="2000", repeatCount="5" )]
```

Universal Mind™

Custom metadata processors

[Logger]
[Bind]
[YourAwesomeTagThatDoesCoolStuff]

Universal Mind™

Chaining API

Universal Mind™

CommandMap

Tell me more!

Site / blog

http://swizframework.org

Wiki

http://wiki.swizframework.org

Bugs

http://bugs.swizframework.org

Mailing list

http://groups.google.com/group/swiz-framework