Rich Components with Flash and Flex

By Jonnie Spratley

Creating Rich Components with Flash and Flex have never been so easy, you just never hear about, until now. Many designers and developers dislike using the Flash CS4 as there primary coding IDE when creating applications, website, or even simple banners. Why?

What you will learn…

- Using Flash Symbols in Flex Applications
- Using Flex Builder as a coding environment for Flash Projects
- Creating custom symbols in Flash and bringing them to life in Flex Builder

What you should know…

- Basic Flash
- Novice ActionScript
- Basic Illustator

Level of difficulty

This is because the lack of intelli-sence that the IDE provides. Then there are those designers and developers that dislike the Flex Builder IDE for its lack of drawing tools such as the pen tool. So this usually becomes a problem when starting to create a application, with a rich interface and be able to take a beating from all kinds of users. So what if you could make certain features in Flash and then us them in an existing Flex application? That would be great wouldn't it. Well with a few modifications to your Flash document, you can! Lets get to it!

What we are going to do

Here is the scenario, our client wants us to create a t-shirt application that there employees can use to select a color for there shirt that they will be wearing at there company open house. Other companies around the area they all have matching shirts. But our client wants every shirt to be unique and colorful. With that in mind we wanted to create something that will display a t-shirt, then let the user choose one of the many colors available for that shirt. Instead of using some JavaScript we decided to use Flash for the design, and Flex for the structure. We are going to create

Image Preparation

For this application we were giving a very high quality t-shirt image to use. So for the preparation lets open it up in Adobe Illustrator. After you have opened the document, I took a copy of the original image and made 4 layers from it. Here they are:

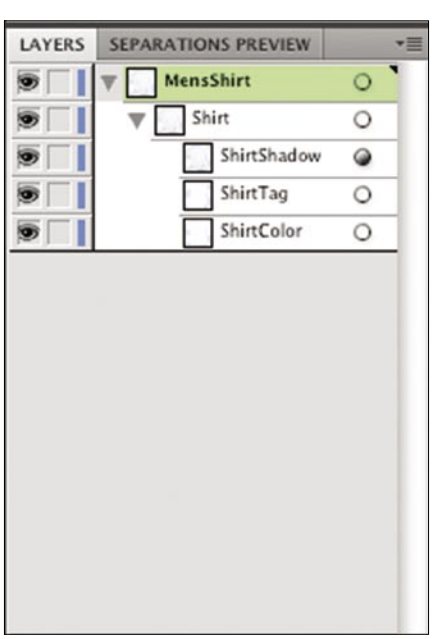

Figure 1. *Illustrator Layers Panel*

- Layer 1 - Shirt Shadows - This is going to keep the realistic feel of the t-shirt.
- Layer 2 - Shirt Tag - This is a small image of the shirts tag, to keep its visual appearance.
- Layer 3 - Shirt Design - (not created yet) - This is going to be where a design if any will go.
- Layer 4 - Shirt Color - This is a solid color of the t-shirt that will give our users the effect that the t-shirt is changing colors.

Here is what my layer panel looks like, see (Figure 1) .

Breaking up the layers is perfect for Flash, because we are going to be using only the Shirt Color layer. This will give us the appearance that the whole shirt is changing every time in the application, which in fact it is only the color layer. Doing this will let us keep the quality of the shirt by leaving the shadows and shirt tag in tack. Then if we were to add the ability for a user to add some sort of design, we could easily allow this by just inserting the design under the shadow layer but above the color layer. After the image is complete, lets open Flash CS4.

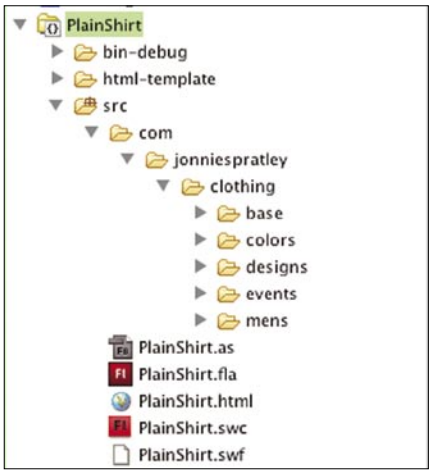

Figure 2. *Flex Builder Project Structure*

Setting Up ActionScript Project/ Flash Project

To correctly configure both IDE's to use both of there powerful features we are going to have to set up a new project in both. Here is how you can do that:

Step 1 - Flex Builder

- Click New->ActionScript Project
- Then for the project name, type PlainShirt, and for the location choose a directory.
- After you have typed the name of the project and clicked finish you should see a new project in the project explorer and a new ActionScript (.as) file named PlainShirt.as. This is going to be our base class that we will use in our Flash project next.
- Its a good time to create any namespace folders for your classes, I went ahead and created my namespace folders, and a few other ones, such as events, base, colors, designs, and mens.
- These folders are going to hold the custom classes that we are about to create.
- Switch to Flash

Step 2 - Flash CS4

- Click New->Flash Project
- Now save the project. For the location, make sure you select the same location as the project you created in Flex builder. But inside of that project folder, select the src folder. This is where you want your Flash Project (.fla) file to be saved.
- After both Flex and Flash projects are created, You now should see something similiar to (Figure 2).

Flash CS4 settings

For every to work correctly you need to change the settings of your Flash project to this:

- Classes Tab - Save classes in: './src'
- Locations Tab - None
- Paths Tab - Flex SDK: '/Applications/Adobe Flash CS4/Common/Configuration/ActionScript 3.0/libs/flex_sdk_3' Folders Containing ActionScript Files: '/Applications/Adobe Flash CS4/Common/First Run/Classes'
- Click OK. Now you are ready to start importing the image that we prepared in Illustrator.

Importing and Linking symbols

You are going to want to import the Illustrator (.ai) file now into Flash CS4, where are going to create a symbol of it, then adjust the linkage so when we publish the Flash document, we will have our custom classes linked directly to our symbol. When importing the file into Flash CS4 you will be prompted with a window like the one at (Figure 3).

You are going to want to import the Illustrator (.ai) file now into Flash CS4, where are going to create a symbol of it, then adjust the linkage so when we publish the Flash document, we will have our custom classes linked directly to our symbol. When importing the file into Flash CS4 you will be prompted with a window like the one at (Figure 3).

From there you want to select each layer one by one and adjust the import settings. Make sure that you check the 'Create Movie Clip' box, then go ahead and give it a instance name, I just used the name of the layer but in camelCase.

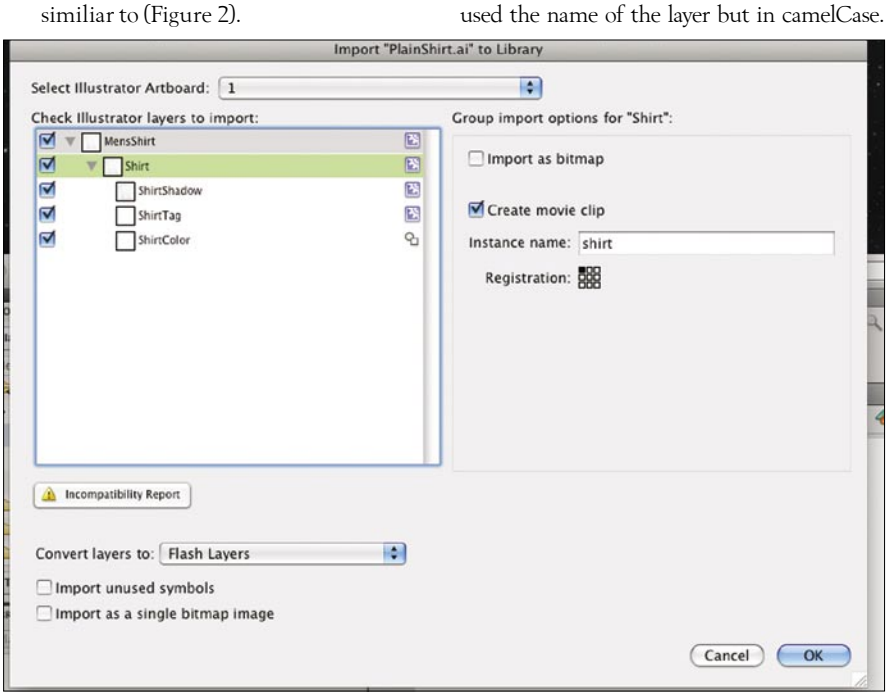

Figure 3. *Flash Symbol Import*

Once you do that to each layer you will then have 5 new symbols in your Flash project library along with the assets that hold the actual image. Here is what just got created.

- MensShirt - Symbol that holds another symbol
- Shirt - Symbol that currently holds 3 other symbols, and will hold our design symbol.

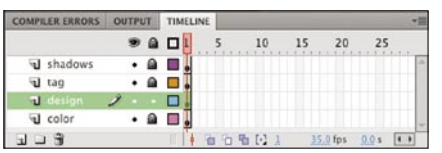

Figure 4. *Flash Timeline Layers*

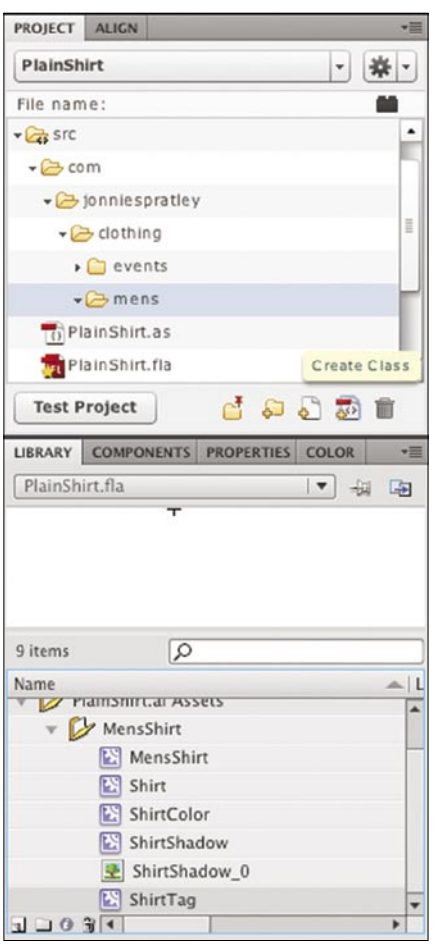

Figure 5. *Flash Project Panel*

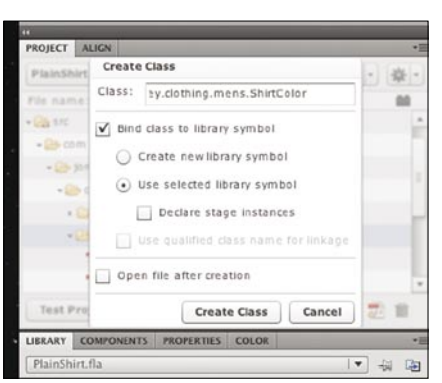

Figure 6. *Component Class Dialog*

- ShirtHighlights - Symbol that holds nothing.
- ShirtTab - Symbol that holds nothing.
- ShirtColor - Symbol that holds nothing.

I know that Flash when importing images might group some of the images, for instance if you were to add the Shirt symbol to the stage, then double click it, you will be inside of the shirt symbol, but selecting the shirt once more, in the properties panel, you will notice that it does not say you have selected the ShirtShadow symbol, it will say 'Group', just press cmd+shift+G to ungroup those 3 symbols that are inside of the Shirt symbol. Now we need to create 4 layers on the timeline, make them as in (Figure 4).

These layers hold the different symbols that we have in our library. On the 'design' layer draw a box the size of what a logo would be, and convert it into a symbol, make sure you use a dark color for the rectangle so we can identify it. After you have created the rectangle design symbol, edit it and make the color alpha '0%', so the color will be invisible, but ill still be able to hold our design. If you have any favorite design for testing purposes just import it to your library and create a symbol of it. We are going to use one that

Listing 1. *ClothingItem.as*

```
package com.jonniespratley.clothing.base
{
    import com.jonniespratley.clothing.events.ClothingEven;
    import flash.display.Sprite;
    import flash.geom.ColorTransform;

    [Event(name='clothingColorChanged', type='com.jonniespr
                    atley.clothing.events.ClothingEvent'
                    )]
    [Event(name='clothingDesignChanged', type='com.jonniesp
                    ratley.clothing.events.ClothingEvent
                    ')]

    public class ClothingItem extends Sprite
    {
        private var _clothingColor:Number;
        private var _clothingDesign:DisplayObject;

        public function ClothingItem()
        {
            super();
        }

        /* ==========================Clothing Color======
                    ====================================
                    */

        [Bindable( 'clothingColorChanged' )]
        public function get clothingColor():Number
        {
            return _clothingColor;
        }

        public function set clothingColor( value:Number ):
                    void
        {
            if ( value )
            {
                _clothingColor = value;
                setColor( value );
            }
            var clothingObj:Object = { color: value };

            dispatchEvent( new ClothingEvent(
                    ClothingEvent.CLOTHING_COLOR_CHANGED,
                    clothingObj ) );
            trace( 'Dispatching ' + ClothingEvent.CLOTHING_
                    COLOR_CHANGED );
        }

        private function setColor( color:Number ):void
        {
            var colorTrans:ColorTransform = new ColorTransform();
                colorTrans.color = color;

            transform.colorTransform = colorTrans;
            trace( 'Setting Color: ' + color );
        }

        [Bindable( 'clothingDesignChanged' )]
        public function get clothingDesign():DisplayObject
        {
            return _clothingDesign;
        }

        public function set clothingDesign( value:DisplayObject
                    ):void
        {
            if ( value )
            {
                _clothingDesign = value;
                setDesign( value );
            }
            var clothingObj:Object = { design: value };

            trace( 'Dispatching ' + ClothingEvent.CLOTHING_
                    DESIGN_CHANGED );
            dispatchEvent( new ClothingEvent(
                    ClothingEvent.CLOTHING_DESIGN_CHANGED,
                    clothingObj ) );
        }

        public function setDesign( design:DisplayObject ):void
        {
            if ( this.numChildren > 0 )
            {
                for ( var i:int; i < this.numChildren; i++ )
                {
                    this.removeChildAt( i );
                }
            }
            this.addChild( design );
        }
    }
}
```

I created for testing. Now we are going to create our custom ActionScript classes that will be linked to our symbols. In the project panel in Flash I expanded my src folder. Make sure you select the correct folder that our first symbol will go into, see (Figure 5). This is going to open a prompt that looks like in (Figure 6).

You want to select the checkbox 'bind class to library symbol', and then the checkbox 'use selected library symbol', hopefully you have selected both the folder the symbol is going to be created in, and the symbol that you are creating the class for. After you have that done, click 'Create Class' and you will notice that your library symbol isn't in its correct place anymore, it is going to be in a folder structure that matches the folders in your project panel, If you select the symbol you just created and click edit, you will see that your symbol is a instance of the class we just had created for us. You want to do this for each of the symbols we have in your library. Its a good thing to start from the deepest symbol of them all, In this case we have a shirt symbol that holds 3 other symbols, shadows/tag, design, color.

So when we create our Shirt class that is linked to the Shirt symbol last, we can select the checkbox labeled 'Declare stage instances', this will create our Shirt class and it will have automatically imported and created instances of our other symbol classes inside of it. Now its time to modify the class to hold some functions we can really put to use in Flex Builder.

Listing 2. *ShirtColor.as, ShirtDesign.as, ShirtShadow.as, ShirtTag.as*

```
package com.jonniespratley.clothing.mens
{
    import com.jonniespratley.clothing.base.ClothingItem;

    public class ShirtColor extends ClothingItem
    {
        public function ShirtColor()
        {
            super();
        }
    }
}

package com.jonniespratley.clothing.mens

    import com.jonniespratley.clothing.base.ClothingItem;

    public class ShirtDesign extends ClothingItem
    {
        public function ShirtDesign()
        {
            super();
        }
    }
}

package com.jonniespratley.clothing.mens
{
    import com.jonniespratley.clothing.base.ClothingItem;

    public class ShirtShadow extends ClothingItem
    {
        public function ShirtShadow()
        {
            super();
        }
    }
}

package com.jonniespratley.clothing.mens
{
    import com.jonniespratley.clothing.base.ClothingItem;

    public class ShirtTag extends ClothingItem
    {
        public function ShirtTag()
        {
            super();
        }
    }

    }
```

Coding

We need to create our methods inside of our custom classes, I created a base class titled 'ClotingItem.as'. This class is going to extend UIMovie clip, which is necessary for using our Flash symbols inside of Flex Builder. Inside of that class we need to create some private variable to hold the values of the color and design then we are going to create some getters and setters for those private variables. So we can have data binding of colors and designs. The class is as follows in (Listing 1). Now that we have our base class created open up the other 5 classes. ShirtTag.as, ShirtShadow.as, ShirtColor.as, ShirtDesign.as and Shirt.as. Inside ShirtTag.as, ShirtShadow.as, ShirtColor.as, ShirtDesign.as make them all extend the ClothingItem.as class, and a call to super() in the constructor. This is how they should look, see (Listing 2). Now open the Shirt.as class, you will see that there are 3 public variables inside of this class, those hold the references to the symbols, and the name of the variables are the same name as the stage instances. Thats how Flash is going to map them to each other. Here is how the Shirt.as class should be, see (Listing 3).

Final Class will be the main class in the src folder, it should have the same name as the Flash Project, PlainShirt.as.

Note: You may notice some other classes in there, I created a simple event class, colors class and an design class that holds

a design for testing. You can find those files at *http://jonniespratley.com/code*. Make your class look as follows in (Listing 4). After that is complete in Flash CS4 goto 'Publish Settings' and make sure you check the box that reads 'Export SWC'. This is going to be our library of symbols and classes.

Flex Builder

Create a new project and name it PlainShirtDesigner, click finish. This is where we take the .SWC file that was published by Flash CS4 and copy it into the libs folder of our new Flex Builder project. Once that is finished we are ready to start coding the application. The application is going to do the following.

Once the application is loaded it will be the data provider for a TileList which is using the Shirt we just created in Flash as the item renderer. Making a beautiful list of every color in a thumbnail shirt. Upon selecting on a thumbnail the color will then be transformed on to our main Shirt component `PlainShirtDesigner.mxml` see (Listing 5).

Listing 3. *Shirt.as*

```
package com.jonniespratley.clothing.mens
{
    import flash.display.DisplayObject;
    import com.jonniespratley.clothing.designs.*;

    import mx.flash.UIMovieClip;

    [Event(name='clothingColorChanged', type='com.jonniespratley.clothing.events.ClothingEvent')]
    [Event(name='clothingDesignChanged', type='com.jonniespratley.clothing.events.ClothingEvent')]

    public class Shirt extends UIMovieClip

        public var shirtColor:ShirtColor;
        public var shirtShadow:ShirtShadow;
        public var shirtTag:ShirtTag;
        public var shirtDesign:ShirtDesign;

        private var _color:Number;
        private var _design:DisplayObject;

        public function Shirt()
        {
            super();
            //color = 0xffff00;
            //design = new DesignAirRelease();
        }

        [Bindable( 'clothingColorChanged' )]
        public function get color():Number
        {
            return this._color;
        }

        public function set color( value:Number ):void
        {
            this.shirtColor.clothingColor = value;
        }

        [Bindable( 'clothingDesignChanged' )]
        public function get design():DisplayObject
        {
            return this._design;
        }

        public function set design( value:DisplayObject ):void
        {
            this.shirtDesign.clothingDesign = value;
        }
    }

    }
```

Listing 4. *PlainShirt.as*

```
package
{
    import com.jonniespratley.clothing.base.ClothingItem;
    import com.jonniespratley.clothing.events.ClothingEvent;
    import com.jonniespratley.clothing.colors.AmericanAppare
                    l;
    import com.jonniespratley.clothing.mens.Shirt;
    import com.jonniespratley.clothing.mens.ShirtColor;
    import com.jonniespratley.clothing.mens.ShirtDesign;
    import com.jonniespratley.clothing.mens.ShirtShadow;
    import com.jonniespratley.clothing.mens.ShirtTag;
    public class PlainShirt extends Sprite
    {
        public function PlainShirt()
        {
            com.jonniespratley.clothing.base.ClothingItem;
            com.jonniespratley.clothing.events.ClothingEvent
                        ;
            com.jonniespratley.clothing.colors.AmericanAppar
                        el;
            com.jonniespratley.clothing.mens.Shirt;
            com.jonniespratley.clothing.mens.ShirtColor;
            com.jonniespratley.clothing.mens.ShirtShadow;
            com.jonniespratley.clothing.mens.ShirtTag;
            com.jonniespratley.clothing.mens.ShirtDesign;
        }
    }

    }
```

Listing 5. *PlainShirtDesigner.mxml*

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
    width="100%"
    height="100%"
    creationComplete="init()"
    xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:mens="com.jonniespratley.clothing.mens.*">
    <mx:Script>
        <![CDATA[
            import mx.collections.ArrayCollection;

            import com.jonniespratley.clothing.colors.Americ
                        anApparel;
            import com.jonniespratley.clothing.events.Clothi
                        ngEvent;
            private function init():void
            {
                var colorCollection:ArrayCollection = new
                        ArrayCollection();
                var colorArray:Array = AmericanApparel.AMERICAN_
                        APPAREL_COLORS;

                for ( var i:int = 0; i < colorArray.length; i++ )
                {
                    colorCollection.addItem( colorArray[ i ] );
                }
                color_list.dataProvider = colorCollection;
            }
        ]]>
    </mx:Script>
    <mx:Panel id="panel_options"
        width="250"
        height="100%"

        <mx:Label
            text="Select a Color:"
            fontWeight="bold"/>

        <mx:TileList id="color_list"
            width="100%"
            height="100%"
            columnCount="4">
            <mx:itemRenderer>
                <mx:Component>
                    <mx:Box>
                        <mens:Shirt
                            color="{ data.color }"
                            height="50"
                            width="50"/>
                    </mx:Box>
                </mx:Component>
            </mx:itemRenderer>
        </mx:TileList>

    </mx:Panel>

    <mx:Panel id="panel_shirt"
        width="100%"
        height="100%"
        title="Shirt Display">

        <mens:Shirt id="shirt"
            color="{ color_list.selectedItem.color }" />

    </mx:Panel>

</mx:Application>
```

Figure 7. *Final*

Conclusion

With these new ways of integrating Flash and Flex the sky is the only limit. Now you can have confidence when integrating Flash and Flex to create rich components that you can use and use again in many of your applications. This is just a taste of what Flex and Flex can do together, if we had more time we could create a whole clothing library for the application keeping it rich, clean, and interactive along the way (Figure 7).

JONNIE SPRATLEY
Flex/PHP Developer
http://jonniespratley.com
Have you checked out CodeGen?
http://code.google.com/p/flex-codegen/