Flex in a Week, Flex 4.5

Video 1.07: Introducing Flex components and controls

You learned in the last video that Spark and MX are libraries in the Flex framework.

In this video, you will learn about the UI components available in both of these libraries.

You will also learn why there are two libraries and the architectural difference between them.

MX components are also called Halo components and were the user interface elements available in Flex 3.

Each MX component has logic that defines its behavior, layout, styles and skin.

Spark components are a new set of components that were made available in Flex 4 and they have been designed specifically to separate behavior, layout, styles and skins into separate classes.

You can use Spark and MX components together because both libraries are based on the same UIComponent class.

When both a Spark and MX component exists, choose the Spark component.

So, the questions flying around in your brain right now are probably something like: What are Flex UI components? What is the practical difference between MX and Spark components? Why did they create a whole new library of components, especially since so many of them seem to be the same? What does it mean that Spark components have separate behavior, layout, styles and skins?

I will attempt to start answering these questions in this video.

You will get further answers throughout the training series.

So, what are Flex UI components?

For both the Spark and MX libraries, there are two categories of components: controls and containers.

Controls are UI elements like TextInput, Button, DataGrid or DropDownList components.

Containers hold content which could be controls or other containers.

Their main purpose is to lay out the content of your application.

Since they are primarily for layout, not all of them have a visual display.

To display images, you should use one of the Spark components that support scaling, smoothing and caching.

BitmapImage is a lightweight, non-skinnable UI component that you will use throughout this series.

Image is a skinnable UI component that is outside the scope of this course.

You can learn more about both by reviewing this article.
A great resource for exploring Flex UI components and Flex in general is the Tour de Flex application.

This is the launch page for it from the Adobe Developer Connection web site.

Tour de Flex is available as a Desktop application, rendered in the AIR runtime, or as a web application.

This is the web version.

At the time of this recording, Flex 4.5 has not officially launched, so the Tour de Flex site may look different when you explore it.

By the time you watch this, you should see final release content if you visit the site.

Note that it contains information about Flex 3, 4, 4.5 and mobile.

I am expanding the Flex 4.5 Preview section and drilling down into Flex Hero Samples > Hero DataGrid.

You can see that a rendered UI view is displayed above a code view.

You can click on other options to view more information about those topics.

Now that you know a little more about Flex components, let me try to answer the questions about why the Spark component library was introduced in Flex 4 and how it differs from the MX component library.

This is a Flex project and an MXML application that I created to illustrate the difference between Spark and MX.

Keep in mind that my purpose right now is to illustrate some basic concepts to you, not teach you a lot of code.

You will learn about all this code as the training series progresses.

For now, you will find that I skim through the code explanations and only focus on particular elements.

I am double-clicking on the Editor tab to expand the code view.

First, let's take a look at this Panel component instance.

It is in the mx namespace, has a title of "MX Panel" and contains one text field, a RichText control that has some filler text.

When I run the application, you see the panel displayed with the title header in the header bar and the text field in the content area, as you might expect.

All Flex UI components have properties.

For instance, this Panel container has a title property while the RichText control has text, width, height and various padding properties, which add some white space around the text content.

I am going to add two more properties to the Panel container.

As I mentioned in an earlier video, this series places one property on each line of code, unless the properties are related, as are these width and height and padding properties.

I'm placing my cursor after the title property, typing the Enter, or Return, key and then typing Control + Spacebar to access the content assist tool.

As I type the word width, the Flash Builder content assist tool quickly locates the property.

I am hitting the Enter, or Return, key on my keyboard to accept the suggestion.

I am setting the value to 150 pixels and then adding a height property with the same value.

When I save the file and run the application, you can see that the panel is bigger.

Notice that when I mouse over a property, Flash Builder's content assist tool shows me help information about the property.

When I mouse over the Panel, it actually tells me that I should not be using the MX Panel container, but rather the Spark Panel container.

So why is the Spark component considered better?

Let's explore both of them side by side.

I am copying the Panel code block and pasting it again.

I am changing the namespace to s for the Spark namespace and changing the title property to Spark Panel.

When I save the file and run the application, you can see that the panels look exactly the same.

So, since the width and height properties had the same effect on the MX and Spark Panel containers, let me try some styles.

Styles are another language element in ActionScript that affect the visual display of the UI component.

I am adding a color style to the MX Panel container.

You can see that the content assist tool even has a different icon to designate a style rather than a property.

You can see that a style is three blue squares, while a property, like width, is a green circle.

I am setting the color style to a value of #006699, which is a shade of blue.

This will change the text color in the MX Panel container.

I am copying the style to the Spark container.

When I save the file and run it, the containers look the same.

Back in Flash Builder, I'm adding a borderColor style to the MX Panel container, also set to the same shade of blue, #006699.

Again, I am copying this style to the Spark container, saving the file and running it, to again see that the panels look the same.

Notice, however, that the both do have drop shadows, so let me get rid of those.

I am now adding the dropShadowVisible style and setting the value to false

Again, I am moving it to the Spark Panel.

When I save the file and run the application, you can see that, again, that the containers look the same.

Alright, so far, there haven't been any differences between the two containers.

These basic properties and styles seem to be holding true to both libraries.

In the MX Panel, I'm adding a headerHeight style with a value of 50 pixels.

When I save the file and run the application, you can see that the header is, indeed, taller.

When I go back to Flash Builder and try to add the same style to the Spark container, however, I run into snag.

It just doesn't exist.

Now, this might make you think that this lack of a headerHeight style is a limitation of the Spark library.

If you start comparing other styles in other Spark and MX components, you will find that the Spark equivalents actually have fewer styles embedded in them than the MX components.

Remember that in an earlier slide I said that MX components each contain behavior, layout, style and skins.

By this, I meant that all of these elements were intrinsically tied together in the component architecture.

By contrast, the Spark components separate their behavior, layout, styles and skins into different classes.

When looking at the fact that the headerHeight style is conveniently in the MX container but missing in the Spark container, you might jump to the conclusion that the MX architecture is better than the Spark architecture.

I would suggest, that for simple changes in styles, the MX architecture may be easier, but for fundamental changes in the look-and-feel of a component, you will find that the Spark architecture is infinitely more flexible.

This is a Flex application that you will build on Day 5.

These three columns of information are actually Panel containers with their header bars and title turned vertical.

You can do this very simply using the Spark component architecture.

It is more time consuming to do it using the MX component architecture.

Let me show you why.

This is an MX skin in Flash Professional CS4.

You can skin MX components in ActionScript, but I am using the visual tool to more quickly make the point.

The layers at the top show you that the panel is composed of a content area, panel base and shadow.

The content area is the white area of the panel.

The gray area is the panel base and the shadow is around the edges.

In this next slide, you can see that I have selected a portion of it.

Which I can simply delete because it is simply a drawn vector image.

That's easy enough, but you may have noticed that there are no text controls displayed here for me to modify placement or font properties of the header or the content text.

That is the crux of the matter.

By modifying the MX Panel skin's visual elements, I am changing the way it looks, but I am really not changing the fundamental behavior and layout of its core elements.

To move the header text and make it vertical, I would have to extend the MX Panel class in ActionScript.

It's easier to change the Spark Panel display because the Spark architecture separates the Panel logic and content from their visual implementation.

This is the ActionScript documentation for the Spark Panel container.

I am clicking on the Skin Parts link.

You can see that a Spark Panel container's skin is actually made up of specific parts.

The Panel content is referenced by the contentGroup identifier and the title text field has an id of titleDisplay.

Back in Flash Builder, I have created a skin class file for my Spark Panel container in the skins folder and named it SampleSkin.mxml.

I am double-clicking on the SampleSkin.mxml tab to maximize the view.

Again, don't get too caught up in all the code right now.

You will learn more about skins in a later video of this day as well as in Day 5.

For now, just know that this code at the top is required and specifically ties the skin to the Spark Panel container.

Now, down in the UI components section, you can see that there is a Group tag named contentGroup.

Remember that this is one of the Spark skin parts for the Panel container and it represents the content between the opening and closing Panel tags in

the main application file.

In this case, it represents the placement and display of the RichText control.

Above the contentGroup skin part is the titleDisplay skin part, which represents the title text in the Panel header bar.

The titleDisplay is positioned 22 pixels from the top of the Panel container and 10 pixels from the left.

The contentGroup is positioned 52 pixels from the top and 5 pixels from the left and right sides of the Panel container.

You will see that this translates to the same position as the text fields in the MX Panel container.

Back in the main application file, I am applying this skin to the Spark Panel container by using the skinClass style, and pointing it to skins folder, and then to the SampleSkin file.

Note that I don't use the mxml file extension.

When I save the file and run the application, you can see that the skin was applied, but does not have any border or header bar.

Above the titleDisplay, I am pasting a code block that represents a rectangle for the Panel border.

You can see that it has a height of 100% and width of 100%.

This means that it will always expand to 100% of the Panel size, which is currently set to 150 pixels wide and 150 pixels high.

The rectangle also has a solid blue stroke color.

When I save the file and run the application, you can now see the blue border around the Panel container.

The MX Panel container actually has a lighter a blue border around it, and this is because the MX Panel, by default, is set to an alpha transparency of 50%.

Notice that the panel text also extends outside the rectangle. We will fix this in a bit.

Back in my code, I'm going to add an alpha property to my rectangle and set it to .5, which is 50% transparent.

When I save the file and run the application, you can see that the border colors are the same.

I have added a second rectangle block to represent the header bar for the Spark container.

I've set the height to 50 pixels and the width to 100%.

The rectangle fill is a light gray with a slightly darker gray stroke.

Now, when I save the file and run the application, the Spark Panel container looks the same as the MX Panel container except that the header text is not bolded.

I am returning to my code to add a fontWeight property set to bold to the titleDisplay Label control.

When I save the file and run the application, now the Spark and MX containers are equivalent.

This must seem like a lot of work on the Spark container to get exactly the same look you had before, so let's get on to the real fun of the Spark architecture.

Remember that my goal is to create a vertical header bar for the Panel container.

I am changing the rectangle that represents the header bar so that it has a width of 50 pixels and a height of 100%.

I am also going to add a right property with a value of zero, which will tie the rectangle to the right side of the Panel container.

When I save the file and run the application, you can see that the Panel header is now vertically placed on the right side.

Back in the code, I am setting the rotation property for the titleDisplay skin part to 90 degrees.

I am also setting a right property to 25 pixels and changing the top property to 10 pixels.

This will constrain the text to 25 pixels from the right margin and 10 pixels from the top margin.

I am removing the left property entirely, because I want the title text to be tied to the right side of the Panel container, not the left side.

When I save the file and run the application, you can see that I now have a Panel with a vertical header, although the content text is overlapping the header.

That's simple enough to fix, I am locating the contentGroup skin part and setting its right property to 50 pixels so that it will always leave 50 pixels to the right available for the header bar.

I'm also changing its top property to 5 pixels and leaving the left property set to 5 pixels.

This will leave a nice margin around the Panel content.

When I save the file and run the application, you can see that in just a few steps, I've easily created a Panel with a vertical header that has accounted for all of the text elements.

This is the power of the Spark architecture.

Because all of the Spark components are built in a modular fashion, you can move the parts around to easily create your own custom look-and-feel.

Note that you aren't just tied to creating your skins in MXML code, as I've shown you here.

The Adobe Flash Catalyst tool allows you to visually create UI skins that can be imported into Flash Builder for your Flex applications.

For your next step, watch the video titled "Laying out a form in Flash Builder Design mode".