Flex in a Week, Flex 4.5

Video 1.09: Introducing styling and skinning

In this video, you will have a quick introduction to various techniques for changing the look-and-feel of your application.

First you will learn how to change the application appearance using Cascading Style Sheets, or CSS.

You will also watch me apply a Spark skin to the application and use the Theme Browser to change the component theme.

This is the main application file that I modified in the last video.

The first task I will show you in this video is how to create and use a CSS style in Flash Builder's Design mode.

I am switching to Design mode and then selecting the Label control that reads Employee Portal: Vehicle Request Form.

In the Properties view you can see all of the Text properties that I assigned earlier.

In the Size and Position segment, I am giving the Label control a width value of 690 and a height value of 40.

I am saving the file.

Now I am clicking the Convert to CSS button to have Flash Builder turn all of the properties currently assigned to this text field into a CSS style.

In the New Style Rule dialog, I am clicking the New button next to the Define style in drop-down list.

I am making sure that the src folder is selected, and then typing Styles for the CSS filename and pressing the Finish button.

Now I am selecting All components with style name for the Selector type and then typing titleHeader for the Name field, which will be the name of the CSS selector.

When I click OK you can see that the Styles.css file opens up and that it has the titleHeader custom selector created with all of the properties defined for the text field.

To the titleHeader selector, I am adding a background color property with a value of #000000, a color property with a value of #FFFFFF, a paddingLeft property with a value of 20, and a verticalAlign property with a value of middle.

I am saving the file.

Back in the main application file, in Design mode, you can see that the new style has been applied to the Label control and the control now has its

Style property set to titleHeader.

When I switch over to Source mode, you can see that the control now has the styleName property set to titleHeader.

There is also a Style tag that has been added to my MXML application file which points to the Styles.css file which is now in the source default package.

If you want to change the look and feel of a component, you can either style it, like you have done to the Label control, or you can skin it.

In this video I will not go into depth about skinning; this topic will be discussed in great detail in Day 5 of this training.

However, I will discuss a few key elements necessary for you to implement a skin.

When you create a skin, you base it on a Flex framework class named SparkSkin.

Most components have at least two states: a normal and a disabled state.

Other components, for instance Button controls, have additional states like up, down and over.

Next you define a HostComponent that states what you are skinning.

In this case, I am creating a skin for the Application container.

This Rect tag represents the background for the Application tag and defines a rectangle shape.

Note that the horizontalCenter property is a constraint that directs the Flash Player to render this background image in the center of the application; in other words, zero pixels from the horizontal center of the application.

The width property sets the width of the rectangle to 750 pixels, as you might expect.

The height property ensures that the background is always as tall as the application window.

The fill and stroke properties set the rectangle to a light-gray color with a darker gray border.

Lastly, the Group container declares that the content children of the application, represented by the skin part named contentGroup, will be displayed vertically starting 20 pixels from the top of the browser and consistently centered in the browser horizontally.

The Spark skin that I will apply to this application is in the skins directory of my project and is named AppSkin.mxml.

Back in the main application file, I am locating the opening Application tag and adding a skinClass property to it.

This is the property I will use to tie the skin to the application.

Note that Flash Builder lists the available skins, including the AppSkin custom skins that I created.

When I select the AppSkin option from the code assist dropdown, Flash Builder puts the full skin path, skins.AppSkin, into the skinClass property.

When I save the file and run the application, you can see the same application as before, but now with a gray background.

You can also see that the application content isn't displayed in the center of the browser.

This is because the application defines minWidth and minHeight properties, which are forcing the application to be larger than its current content.

I am deleting the two properties then saving the file and running the application again.

The skin is now centered in the browser.

Back in the AppSkin.mxml file, I am changing the fill color to a light yellow by typing #FFFBCF.

I am also adding a radiusX property with a value of 10 pixels to the Rect tag, to round the rectangle's corners.

Next, I am adding the top and bottom properties to the Rect tag with a value of 20.

This will add 20 pixels between the top of the rectangle and the top browser window and 20 pixels between the bottom of the rectangle and the bottom of the browser window.

When I save the file and run the application, you can see that the application background color is now yellow with rounded corners and more space above and below the rectangle.

It still does have a gray border and when I resized the browser, you probably noticed that the bottom of the rectangle resizes with the browser height so that there's still 20 pixels below it, but does not account for the size of the content.

I am returning to the AppSkin.mxml file. Notice that the height property in the rectangle is set to 100%. If you set this property to 100%, the bottom of the rectangle resizes based on the browser, but the contentGroup does not resize with the browser.

I am changing the Rect tag's height property so that it is bound to the height of the contentGroup and I am cutting the bottom property from the Rect tag and pasting it in the contentGroup.

I am saving the file and running the application.

When I resize the browser, you can see that the application resizes appropriately.

Since I'm working with a limited viewing area, I can't show you the full behavior of the bottom of the browser window, but you will be able to see this when you work through the associated exercise.

The last feature of Design mode that I will show you in this video is the Theme Browser.

Back in the main application, I am switching back to Design mode and then selecting the Appearance tab.

When I click on the Spark link, Flash Builder opens a dialog with many theme options.

I am expanding the last set of options and selecting Sky.

I am clicking OK and then running the application.

You can see that the application's controls are now styled using the Sky – blue – theme and the background of the application is also blue.

For your next step, work through the exercise titled "Creating a user interface".