Flex in a Week, Flex 4.5

Video 5.10: Implementing skin states

So far you have only created skins with two component states.

In this video, you learn how to define additional states in the component skin.

You will also implement animated transitions between three states.

This is the completed Employee Portal application for this video.

Notice the curved upper right corner of the blue header bar.

You can see that the Button in the Employee Directory panel at the bottom of the application follows the same theme with a curve in the upper right corner.

The button’s up state is blue.

When I mouse over the button, it turns green and grows wider.

When I click the button, it remains the same size and changes color to a dark gray.

When I release the button, the color returns to green, and when I mouse off of it, it slowly resizes back to its original size and color.

In this video, I will show you how to handle the additional Button component states in the skin and how to create the animated transitions between them.

This is the Language Reference for the Spark Button control.

Not all controls have multiple states, but the Button control does, as you can see here.

They are: disabled, down, over and up.

I am opening the ButtonWithStatesSkin.mxml file from the skins directory.

Note that it has a HostComponent directive that makes a contract with the Spark Button class.

It also contains a Label control that represents the labelDisplay skin part that handles the label text on the Button control.

You learned about skin parts in the last video.

In the Properties of the parent section of the skin class, you can see the states block with the four defined states: up, over, down and disabled.

Next, I want to create the rectangle background for the button with different fill colors for each state.

Below the UI components comment, I am creating an instance of the Rect MXML tag and giving it an id property with a value of buttonColor.

I will use this identifier when I create the transitions to animate the button.

I am also giving the Rect instance a left property of zero and right, top and bottom properties of zero as well.

This will ensure that the rectangular background always fills the entire button surface.

I am adding the topRightRadiusX property with a value of 3. This will round the upper right corner of the button skin and make it resemble the shape of the header skin.

I'm turning the Rect tag into a tag block, and then, between the Rect tags, I am creating a fill property.

Inside the fill property I am creating a SolidColor instance with a color.up value set to blue, 0x0D86B8.

This represents the color displayed when the button is in the up state.

I am adding a color.over property set to green, 0x64BC48, and a color.down property set to dark gray, 0x555555.

I missed one character in the color.up property value, so I'm adding that now.

Next, I want to apply the skin to the Button instance.

I am saving the file and then going back to the main application file and opening the EmployeeDirectory file by CTRL + clicking on it.

Here is the Button control that I want to skin.

I am adding the skinClass property and referencing the skins.ButtonWithStatesSkin class.

I am saving the file and running the application.

You can see that the skin has been applied to the button and that the states are also applied.

At the top of the skin file, I am creating a transitions property below the states property.

Within the property I am creating one Transition instance with a fromState set to up and a toState set to over.

When the button transitions from its up state to its over state, I want the button to grow wider.

Between the Transition tags, I am adding a Resize effect to target the buttonColor instance, which is the rectangular shape that I created earlier.

I am adding a widthBy property set to 25 pixels.

This Resize effect will run for 1 second, which is the default duration for the animation.

As it runs, the effect will continue to resize the button by 20 pixels at a time.

When I save the file and run the application, you can see that the Button control grows when I mouse over it, but jumps back to its original width after the animation plays.

I am locating the Rect instance and setting a width.over property value to 90 pixels.

When I save the file and run the application, and mouse over the button, the animation plays and the Button control remains large, as expected.

However, when I click the Button control, the size changes abruptly back to its original size.

Back in the skin file, I am adding the autoReverse property to the Transition tag and setting its value to true.

This will reverse the button's animation when you mouse away from the Button control.

I'm saving the file and running the application.

When I mouse over and then mouse away from the Submit button, the Button control resizes appropriately, but notice that the width of the down state is not the same as the width of the over state.

Back in the Rect instance, I am adding the width.down property and setting its value to 90.

When I save the file, run the application and click the Submit button, you can see that the width of the Button control is the same for the over and down states.

When I mouse away from the Submit button, the Submit button returns to its original size.

For your next step, work through the exercise titled “Animating Button component states”.