Flex in a Week, Flex 4.5
Video 5.05: Styling the Form container

You have already worked with the Spark Form container in earlier videos, but you have only used the default settings.

In this video, I will show you have to use the Form stacked layout skin and how to add sequence labels, text prompts and custom icons for the required and error elements.

Here is the completed Employee Portal: Vehicle Request Form application for this video.

Previously, this application was laid out in the default horizontal layout.

In this video, I will change the layout of the application to use the stacked layout of the Form container, where the label sits above the controls.

In addition to changing the layout of the Form container, I will add sequence labels for each of the form items in the application.

I will also show you how to add a text prompt for the Office Phone TextInput control.

Lastly, I will show you how to customize the default required icon with this custom icon and how to customize the error icon – I'm clicking on the form's submit button – with the one here on the Mobile Phone field.

The Spark Form container lays out its children in three ways.

There are two built-in layouts.

The horizontal layout is the default layout of the Form container and only uses one row for all the FormItem elements.

In other words, it displays all labels, controls, required indicators, help and error messages in one row.

The stacked layout, stacks all of the Form's children in a vertical display.

This layout uses two rows for each FormItem container.

The first row contains the FormItem labels, while the second row displays the required indicator and help content.

In the stacked layout, error message display above all FormItem containers, at the very top of the form.

This exercise will focus on the stacked layout of the Form container.

The last layout is a customized grid-based constraint layout that is outside the scope of this training.

This is the starter application that you customized in the third day of this training series.

You can see that this Form is displayed using the default horizontal layout, with the labels to the left of the control.

Now, I will show you how to apply the stacked layout skin to the form.

I'm returning to the main application file in Flash Builder.

Below the Styles comment and above the existing Style instance, I am adding a Style block.

Note that the application namespaces are automatically added to the Style block.

After the namespace definitions, I am adding a CSS style selector for the Spark FormItem container.

In the FormItem selector block, I am adding the skinClass property and referencing the spark.skins.spark.StackedFormItemSkin class.

I am saving the file and running the application.

You can see that I get an error.

This is because I also have to apply the stacked skin to the FormHeading control so that its columns match the stacked FormItem controls.

I'm return to the main application file.

Between the namespace definitions and above the selector for the FormItem control, I am adding a selector for the Spark FormHeading control.

In the selector block, I am using the skinClass property again to reference the spark.skins.spark.StackedFormHeadingSkin.

When I save the file and run the application, you can see that the FormItem labels appear above the input controls.

I am clicking the Submit Request button.

Notice that the error icons appear, but the error messages do not.

This occurs because, in a horizontal layout, the error message displays to the right of the TextInput controls.

In a stacked layout, the error message displays at the top of the container.

The error message will not display in the application without the Form selector, so back in the main application file, I am adding the Form selector under the namespace definitions.

To the selector block, I am using the skinClass property to reference the spark.skins.spark.StackedFormSkin class.

When I save the file, run the application and click the Submit Request button, you can see that the error messages now display at the top of the Form.

In a previous exercise, you customized the FormItem container by adding the required and helpContent properties.

Now, I will show you how to apply sequence labels to the application.

Sequence labels are displayed to the left of the FormItem container label.

It is defined with the sequenceLabel property and can contain any text as its value.

Now I am going to show you how to add sequence labels that display to the left of the FormItem containers.

I'm CTRL + clicking on the VehicleRequestForm custom component that you created in an earlier day of this training.

I am locating the FormItem container with the Employee label.

To this control, I am adding the sequenceLabel property and giving it a value of 1).

Now I am going to add this property to the rest of the FormItem containers within the form and using consecutive numbers.

When I save the file and run the application, you can see that the sequence labels have been added in front of each FormItem label.

Text prompts are used in a form to describe the input controls within the form.

The prompt property is supported by the Spark TextInput and TextArea controls.

Text prompts appear when the input control is create, disappear when the input control gets focus, and reappears when the input control loses focus and no value is entered.

Back in the VehicleRequestForm.mxml file, I'm locating the TextInput control in the Office Phone FormItem container.

To the TextInput control, I am adding the prompt property and giving it a value of "Select employee".

I'm saving the file and running the application.

You can see that the Office Phone field now shows the text prompt.

When I click in the Office Phone input area, the prompt disappears.

I'm leaving the field blank and clicking in the Mobile Phone field.

The text prompt reappears when the TextInput control loses focus.

The Spark FormItem container supports customization of the required and error fields.

You can use a custom icon image and change the background color, text and text properties.

In this exercise, I will show you how use a custom icon image to replace the default required asterisk and error icon using the requiredIndicatorSource and errorIndicatorSource properties.

I am returning to the VehicleRequestForm.mxml file.

To the opening tag of the Mobile Phone FormItem container, I am adding the requiredIndicatorSource property and assigning the value to point to the required.png file in the assets package.

I am saving the file and running the application.

You can see the newly customized required icon next to the Mobile Phone field.

Without entering a value in the Mobile Phone field, I am clicking the Submit Request button.

This yellow triangle icon is the default error icon.

Just as you added a custom icon for the required icon, you can also customize the default error icon.

Back in the VehicleRequestForm file, I am adding the errorIndicatorSource property to the FormItem container for the Mobile Phone, and assigning the value to point to the invalid.png file in the assets package.

When I save the file, run the application, and click the Submit Request button, you can see the newly customized error icon.

For your next step, work through the exercise titled “Styling the Form container”.