Flex in a Week, Flex 4.5

Video 3.07: Implementing two-way binding

You have used data bindings extensively throughout this training.

You have bound data between MXML components and between MXML components and ActionScript variables.

All of the bindings have been one-way bindings.

In this video, you will learn about two-way bindings and how to implement them.

Here is the example application that I will use in this video and that you will modify in the associated exercise.

I am displaying a ColumnChart control that has salary information about each employee.

When you click on one of the chart items, the form below the chart is populated with that employee’s information, including the salary.

When I make a change to the salary, nothing changes in the chart.

However, after I apply a two-way binding to this salary form field, then, when I make changes, that value is immediately reflected in the chart.

Here is the main starter application file.

It includes the data handling and value objects that you implemented in Day 2.

This HTTPService object accesses a remote XML file with the employee data.

The return data will be placed in the employees ArrayCollection instance, which is a class property and is bound to chart as its dataProvider property value.

The returned data, which is an ArrayCollection of generic object, is converted into an ArrayCollection of Employee value objects, using the Employee ActionScript class in the valueObjects package.

Each Employee value object is added to the employees class property on the loop iteration.

I want the user to be able to click a chart record, to populate the form.

Each time the user clicks on a record in the chart, I want to access the data for that employee.

In this Script block, after the employees variable declaration, I am typing selectedEmployee and pressing CTRL+1 to invoke the quick assist tool and selecting the Create instance variable option.

This creates a private variable named selectedEmployee data typed to the Object class.

I am changing the variable so that it is data typed to the Employee class.

Above the selectedEmployee variable I am typing [B to invoke the content assist tool and pressing Enter to insert a Bindable metadata tag into my code.

Next, I am adding an itemClick event to the ColumnChart tag and then using Flash Builder's code assist tool to generate the handler for me.

The event object has a hitData.item property that contains the data that was clicked.

I am going to assign this data to the selectedEmployee variable I just created by typing selectedEmployee equals event.hitData.item.

When I save the file, you can see that I get an implicit coercion error because the event.hitData.item property is a generic object instance and the selectedEmployee variable is typed to the Employee value object class.

I am adding the as Employee statement to cast the item to the Employee class as well.

Now when I save the file, the error goes away.

When a record in the chart is selected the selectedEmployee variable will hold the data that I want to use to populate the form.

To save time, I am pasting a form with TextInput controls for the First Name, Last Name, Location and Salary data.

Note that each text property is bound to the associated data that is in the selectedEmployee object.

I am saving the file and running the application.

When I click on a column representing the employee, the form is populated with data from the employee record I selected.

However, when I change the salary value, there is no change to the chart.

This is because the bindings are one-way data bindings.

One-way data binding allows one object to watch for changes in the property of another object.

In my example, the TextInput controls are set to watch for changes in the selectedEmployee object but the selectedEmployee object is not watching the TextInput controls.

A two-way data binding allows two objects to watch for changes in a property of the other object.

Note that two-way data binding is not available before Flex 4.

Back in Flash Builder, I am locating the TextInput control for the salary data.

Note that the id property is salaryInput.

Each object in a two-way binding must have a unique identifier that the application can watch.

I am adding the @ character at the beginning of the binding assignment.

This tells the compiler that this is a two-way data binding.

At this point, though, the binding still will not work.

Note the warning messages for each of the selectedEmployee properties.

In the valueObjects directory, I am opening the Employee.as class file and adding a Bindable metadata directive on the entire class.

When I save the file you can see that the warning messages go away.

I am saving the main application file and running the application.

Now, when I click on an employee in the chart and change his salary value, you can see that the chart reflects the changes in real time.

For your next step, work through the exercise titled “Using a two-way binding”.