Flex in a Week, Flex 4.5

Video 3.01: Creating an event type and dispatching the event object

In the second day of training, you learned how to handle user and system events for pre-built Flex framework component.

You have also learned how to create MXML custom components.

In this exercise you learn how to create your own custom event types for those custom components.

You will find that handling these events in the main application is no different than handling the pre-built Flex framework events.

As a quick review, here is the application you built at the beginning of Day 2.

It has two DateChooser components.

I have chosen a date in April as the car pickup date.

When I choose a return date, the application validates that the pickup date is before the return date.

If it's not, it shows you an appropriate message.

In the code, you created this functionality by registering a change event on the DateChooser components, which calls the event handler function.

By the way, you can hold down the Control key and hover over the function name to get a link that will redirect you to the function code.

When I click, I find the function code and you can see that function performs the conditional check to display the error message.

This is the application that I will demonstrate in this video.

Initially it only displays the MXML custom component, named Choose.mxml, on the left.

The idea is that an administrator of the Employee Portal would access this page to select the Employee of the Month and write a congratulations message.

When the administrator clicks on this Preview button in the Choose component, the application, turns on the second custom component, named Preview, to allow the administrator to preview the display of the Employee of the Month panel for the Employee Portal.

If you look at the code for this application, you can see that Preview button is inside of the Choose.mxml custom component class.

If I handle a click event on the button and generate an event handler, all of this code is still maintained within the Choose.mxml component.

How do I send the fact that this button has been clicked in the Choose component over to the Preview component so that it knows to become visible?

In this video, you will learn how to create an event type for your custom components.

The event will dispatch from your custom component, the Choose.mxml file in this case, to the main application, which will then pass the information on to the other custom component, the Preview.mxml file.

This is the starter application for this video.

You can see that the Choose and Preview custom components appear by default and that the Preview custom component does not display any information about the Employee of the Month.

You will learn about displaying this data in the next video.

For now, I will focus on simply turning on the Preview panel when the administrator clicks on the Preview button in the Choose custom component.

In Flash Builder, let's quickly explore the code in the main application file.

You can see that the main application file defines an HTTPService call to retrieve data from the employees.xml file in the data folder and then populates a class property named employeeData with the returned data from the server.

The employeeData object is then passed to the chooseEmployee instance of the Choose custom component.

Note that the employeeData object is named the same as the employeeData property of the custom component.

This is not required, of course, but is a common practice.

Again, I can use the Flash Builder shortcut of holding down the Control key to click code.

When I do this for the employeeData property of the Choose custom component, Flash Builder opens the Choose.mxml custom component file and locates the property definition.

I am scrolling down to show you that this property is bound to the DropDownList control.

That code logic is what populates this DropDownList control displayed in the application.

Remember that, by default, I want the Preview custom component to be invisible.

I have located that component instance in the main application code and adding the visible property with a value of false to it.

When I save the file and run the application, you can see that the Preview custom component is invisible by default.

I will trigger the application to make it visible again when a user clicks on the Preview button in the Choose custom component.

There are four steps to create and dispatch a custom event.

In your custom component, you will first define the name of the event type using the Event metadata tag.

The second step is to trigger and handle the user or system event.

Next, you instantiate the event object and dispatch it to the parent component, which is often the main application.

You handle this custom component event as you would any other Flex framework event.

In this case, I will modify the Preview custom component.

This illustration shows the first step.

Within the Choose dispatching component, I will use a Metadata tag set to tell the Flex compiler that I am creating a new event type for the Event class.

Remember that you have encountered event types before.

For instance, you have used the click event type for a Button control and the change event for the DropDownList control.

In this case, I am using the Event compiler directive to create a custom event type named showPreview for the flash.events.Event class.

In Flash Builder, I am looking at the Choose.mxml file.

I am locating the Metadata comment and creating a Metadata tag block below it.

Within the tag block, I am typing [Ev to invoke the content assist tool and I am selecting the Event directive. This adds the Event directive to my code.

I am modifying the directive so that the name of the event type is showPreview.

I am leaving the default, flash.events.Event class, as the event type.

The Metadata tag block does not compile into executable code, but it does register this new custom event type with the application in Flash Builder.

The Event metadata tag makes the custom event public so that the MXML compiler recognizes it as an MXML tag attribute.

I am saving the file to maintain my changes.

Remember that I just created this showPreview event type in the Choose custom component file.

When I switch back to the main application and locate the instance of the Choose component, I can press return and then CTRL + to see that the showPreview event is now listed.

It does not yet function, but the Metadata tag has registered the new event type with the application.

Back in the Choose custom component, my second step is to set up the environment to dispatch the event.

I mentioned before that I want the Preview component to display when I click on the Preview button.

So I will use a click event on the Button control to run an event handler function.
Here is the Preview button in the Choose component.

I am adding a click event to the Button control and then using the Generate Click Handler wizard to create the associated function.

Remember that the function is named based on the button id property, in this case preview, plus the name of the event, in this case click, and the word handler.

Here is the generated event handler function in the Script block.

Within the handler, the next step is to create an Event instance of type showPreview.
In the Choose component, I am removing the comment stub and creating a local variable that I'm arbitrarily naming eventObject and data typing to the Event class.

Remember that classes in the flash.events package do not explicitly need to be imported.

They are available in the Flash Player.

Next I am instantiating the object using the new keyword and calling the constructor for the Event class and passing in the one required property, the type.

In this case, the type is showPreview, which is the custom type that I defined earlier in the Metadata tag block.

Now that the event object has been created, you need to pass it out of the component back into the main application using the dispatchEvent() method.

Within the event handler, below the eventObject variable instantiation, I am using the dispatchEvent() method to dispatch the eventObject.

Let's recap what I have done to create this custom event type.

First, I created a new event type named showPreview for the flash.events.Event class using the Metadata tag block.

Next, I determined that I want to create the event when the user clicks on the Preview button, so I registered a click event for that button with an associated event handler.

Lastly, I instantiated an Event object with the showPreview event type and dispatched it from the custom component to the main application using the dispatchEvent() method.

The only thing left to do is to handle the event in the main application.

I am first saving my file to maintain my changes.

You handle a custom event type for your custom components the same way that you handle any event in the Flex framework.

When the Choose custom component, dispatches the showPreview event to the main application, you handle the event by registering the showPreview event with the Choose component instance and running an associated event handler function.

In my example, I will handle the click on the Preview button in the Choose component by making the previewEmployeeOfTheMonth custom component instance turn visible.
In the main application file, I have located the instances of the Choose and Preview custom components.

Notice that their instance names are chooseEmployee and previewEmployeeOfTheMonth, respectively.

I am using code hinting to register a listener for the showPreview event on the Choose component that I added earlier and then using the Generate event handler wizard to generate the associated function.

I am going to use the Control+click method to locate that method.

Now I can delete the generated stub comment.

I am setting the previewEmployeeOfTheMonth instance's visible property equal to true.

When I save the file and run the application, you can see that the Preview component instance is initially invisible.

When I click the Preview button in the Choose custom component, all of the code that I just wrote will come into play.

The click event for the button will fire and run the associated handler which creates the new custom showPreview event object that I created in the Metadata directive and then dispatch the object to the main application file, which, as I click, will make the Preview custom component visible.

The Preview component is currently empty, but you will learn how to populate it with information about the selected Employee from the Choose component in the next video.

For your next step, work through the exercise titled "Creating an event and dispatching the event object".