Flex in a Week, Flex 4.5

Video 2.04: Introducing ArrayCollection and other data types

In this video, I will first discuss the top-level ActionScript classes Object and Array.

I will then introduce you to the ArrayCollection class and the benefits it contributes for handling data and binding to components.

As an experienced programmer, this series assumes that you have worked with complex data objects like arrays and associative arrays – also known as hashes or structures in some languages.

In ActionScript, the associative array is implemented as the Object class and arrays are created from the Array class.

Both are top-level ActionScript classes.

Here is the language reference for the Object class.

I am scrolling down the page to click on the View examples link.

As an associative array, each Object class instance is a logical collection of name/value pairs.

You will continue to learn how to write ActionScript code throughout this series, but note how this Object instance is created with the new operator and then populated with dot syntax to create multiple name/value pairs.

This brief introduction does not do this powerful class justice – just look at all of its subclasses! - but it does give you enough information so that you understand that we can use the Object class to create and manage name/value pairs of data.

Here is the language reference for the Array class.

As you know, an array can also contain complex data, but the values in the data are indexed, rather than named.

For instance, in this syntax example, you populate oneArray with a simple list of strings.

Each of these values is then referenced by a number.

ActionScript is a zero-based language, so 0 represents the first element in the array.
This code overwrites the first element in the array, which is a, with the value z.

Again, don't worry about the ActionScript syntax right now if it looks foreign to you.

You will learn more about it later.

While the Array and Object classes are immensely valuable, in Flex, you will often choose to use the ArrayCollection class to manage data in your applications.

The ArrayCollection class is a wrapper class for the Array class.

It provides methods and properties that are not available in the Array class to manipulate and sort data.

These methods and properties are available through the ICollectionView and IList interfaces.

Most importantly, the ArrayCollection class automatically monitors and updates data bindings to all values in an instance.

In other words, if you bind Array or Object instances to a UI control and then change one of the values in the instance, the UI control does *not* update the value in the display.

The data binding is only triggered when the application starts or when the instance itself is updated.

As a result, the ArrayCollection class is the recommended class for you to use as the data provider for your components since it is constantly monitoring all of its individual elements.

Remember, however, that you must import the class in ActionScript to use it.

This is the main application file for the Employee Portal: Vehicle Request Form that you have been working with through most of this series.

When you requested the XML data from the remote server, you bound the returned data to the UI controls using the lastResult property of the HTTPService object.

Remember that you bound the data specifically to the repeating node of the property, in this case, employees.employee.

Here in the Network Monitor, you can see that it is the employees.employee node that repeats.

This node is essentially an array of employee data.

Flex actually converts the data into an ArrayCollection instance, by default, when it is returned to the application.

Using the lastResult property of the service object is a very quick way to display data.

However, if you want to manipulate the data first, or place it into a reusable variable, you should use a result handler to handle the data instead of directly binding it to the lastResult property.

In the earlier videos and exercises of this Day, you learned how to handle both user and system events.

The result event of the service object, in this case HTTPService, is a system event that is triggered when the data that you requested is returned from the server.

I am adding the result event to the HTTPService object and then using the code assist tool to generate the result event for me.

Holding down the Control key, I can hover over the generated function name to get a link.

When I click the link, Flash Builder relocates me to the generated function.

Note that the event type is ResultEvent.

I'm scrolling up so that you can see that Flash Builder has automatically imported this class for me.

Inside the result event handler, the result data is stored in this event object and I can access the repeating node by typing event.result.employees.employee.

However, I need a way for the data to exist outside of this function code.

Below the import statements, I am going to create an ArrayCollection instance in ActionScript.

The property that I'm creating should be available to the entire MXML file but not to other files, so I am using the private access modifier.

You learn about access modifiers in an earlier video.

Then I am using the var keyword to create the variable.

I'm naming the variable employees and then data typing it to the ArrayCollection class by typing a colon and then the name ArrayCollection.

Note that Flash Builder automatically added the associated import statement for me since I used content assist.

Remember that a semicolon ends ActionScript statement.

Now, inside of the event handler, I am typing employees and then the equal sign before the repeating node reference to the data.

When the data is returned from the server and dispatched to this event handler, the event handler will populate the employees ArrayCollection instance with the data.

I am updating the reference to the lastResult property in the DropDownList control with a reference to the employees ArrayCollection instance.

However, when I save the file you will see a warning message stating that data binding will not be able to detect changes to this variable.

When you worked with data binding on Day 1, you were binding two MXML components to each other.

In order to bind an MXML component to an ActionScript variable, you have to add the [Bindable] metadata directive to the ActionScript variable.

Now when I save the file, the warning message disappears.

When I run the file, the application populates the DropDownList control, as before, however I now have the ability to manipulate the data in the result event, if I choose to do so.

You may have noticed that ActionScript also contains a class named ArrayList.

ArrayList was introduced in Flex 4, while ArrayCollection was available in Flex 3.

Both the classes can be used in data bindings, however the ArrayList class has less overhead because it does not contain all the sorting or filtering functionality, as does the ArrayCollection class.

In this training, we will tend to use the ArrayCollection class, but you can choose the ArrayList class in many instances.

For your next step, watch the video titled “Verifying data retrieval with the Debugger and Network Monitor”.