Flex in a Week, Flex 4.5

Video 2.05: Verifying data retrieval with the Debugger and Network Monitor

In previous videos, I have used the Network Monitor to show you the data request from the Flex application to the server and then the response from the server with data.

In this video, I will outline some of the other information that you can gather from the Network Monitor.

Earlier today in Day 2, you used the Flash Debugger to look at the event object that is dispatched in an event.

In this video, you will learn how to use the Debugger to view returned data from the server.

The Network Monitor can provide you with a detailed audit trail of all data that is passed between your local Flex application and the server.

You can see the elapsed time between the request and the response, the packet size of the request and response data, and of course, the type and content of the data.

The information is available in three views: tree, raw and binary, or hex, view.

Here is the application that I modified in the last video.

Remember that I have implemented an HTTPService object that requests a remote XML file.

The employeeService.send() method does the actual requesting of data when the Application object is instantiated and fully created.

A result event is registered on the HTTPService object.

When the data is successfully retrieved, the event dispatches to this event handler function - which I will Control + click on – to populate an ArrayCollection class property with the repeating node of the XML data.

This ArrayCollection class property is bound to the DropDownList control, so the data will automatically be updated when the data comes back from the server.

I am selecting the Network Monitor view and then enabling the tool.

When I run the application, the Network Monitor will audit all traffic between the Flex application and the server.

I am expanding the view by double-clicking on it.

The left pane will show all of the network traffic.

You can see that the information includes the request and response times as well as the calculated elapse time.

The right pane shows the actual data that is sent and received.

You can look at this data in three views.

The first is Tree View, and as its name implies, this means that you can expand the nodes to see the details.

The second view is Raw View, which shows you the same data in plain text.

I have found that this view is useful when I have a long error string that is not easy to view in Tree View.

The last view is Hex view which shows the data in hexadecimal format.

This view can be useful if you are working with binary data.

Now let’s work with the Flash Debugger.

I am placing a breakpoint on the point where the employees ArrayCollection instance is populated with data in the result handler.

Note that you can right-click on a breakpoint and choose Breakpoint Properties to set conditional breakpoints.

This is outside the scope of this training but I wanted to show you this feature of Flash Builder.

I am canceling out of that dialog and then clicking the Debug button.

The browser opens fairly quickly because the first breakpoint is hit right away.

I am switching to the Debug perspective when prompted, but before I look at the event object in the Variables tab, I want to point out that the Console view can often hold valuable information, especially if there is an error.

I am double-clicking on the Variables view and expanding the this property, which contains all of the data for this application.

The Breakpoints view, lists my breakpoint.

I am double-clicking to minimize the view.

Note that you can use this toolbar to perform common debugging tasks like Step Into, Step Over and Step Return.

I am clicking the Resume button to run the application to the next point, which might be another breakpoint, but, in this case, is to the end of the code.

Flash Builder has a wide range of features that improve your development productivity and help you debug and tune your application.

Some of these features include:

The memory and performance Profiler,

ASDocs support to add MXML and ActionScript editor comments for your own components,

FlexUnit support for unit testing,

And a refactoring engine that allows you to quickly navigate and restructure your application by renaming references in your code.

All of these features are outside the scope of this training, but be sure to check them out.

For your next step, work through the exercise titled “Populating an ArrayCollection with retrieved data using the result event”.