Flex in a Week, Flex 4.5
Video 1.06: Understanding namespaces

In this video, you will begin to learn about the MXML application code.

First, I will show you the XML declaration statement, which is the first line of the application.

Next, you will learn about the fundamental topic of namespaces, which you use to reference the library of Flex framework classes in your applications.

You will also explore how to use a custom namespace to access custom components that you create yourself.

This is the Employee Portal: Vehicle Request Form main application file from an exercise in Day 2.

I am double-clicking on the Editor tab to expand the code view.

The first line of code in this file is the XML declaration that defines this document as an XML file.

Remember that MXML is the XML-based, declarative language in the Flex framework that is primarily used for visual layout.

The XML declaration must be the very first line of code in the document and cannot include any characters before it, not even whitespace.

Note that if I add a space before this code and save the file, the Editor tab displays a red icon which indicates that there is an error in the code.

I am double-clicking on the Editor tab again to see the Problems view and expanding the error message so I can read it.

I am correcting the error, removing the extra space, and saving the file again.

The error indicators all disappear.

The next element of code in the MXML file is the Application tag, which is required for all MXML application files.

The first three properties of the Application tag define three XML namespaces that reference three libraries of code in the Flex framework.

Before I explain what is in each of these libraries, let me discuss namepaces.

If you have not worked with namespaces before, you should be aware that the xmlns property is a reserved XML attribute that is used to declare a namespace.

The values after the colon, in this case, fx, s and mx, are the namespace prefixes.

You use prefixes before each MXML tag – like the s in front of Application and the fx in front of Style - to identify the library you are referencing and to avoid any ambiguity if two libraries have tags that are named the same.

Each namespace has an associated Uniform Resource Identifier, or URI, which is a string that is used to uniquely identify each namespace.

Don't confuse the fact that the URI looks like a URL or path with thinking that the libraries are stored in that location.

In fact, a URI is really just a string that the implementation uses to ensure a unique value for the namespace mapping.

I will explain the association between the namespaces and the class files in just a bit.

First, let me explain what is in each of these libraries represented by the namespaces.

The mx namespace references classes from Flex 3 that include the mx package and data visualizing components.

The fx namespace was first introduced in Flex 4 and includes the top-level ActionScript language elements like Object, Number, Boolean and Array.

It also includes built-in compiler tags like Script, Declarations and Style.

Also introduced in Flex 4, the s, or Spark, namespace contains all of the new Spark components and text framework classes available in Flex.

You will also reference the data services components for web service, HTTP service and remote object calls from the Spark namespace.

Lastly, the Spark namespace avoids confusion between the Flex 3 MX components and the Flex 4.5 Spark components, which you will learn about in the next video.

For now, just understand that the Spark and MX libraries include many components that have the same names, so placing them in different libraries avoids confusion.

Each namespace is mapped to manifest files that list all of the MXML tags that are part of that namespace.

In the file system, I have located the Adobe Flash Builder 4.5 install directory and now I am opening the sdks > 4.5.0 > frameworks > flex-config.xml file in Firefox, which has a built-in XML parser.

When I scroll down in the flex-config.xml file to the namespaces section, you can see that each URI is associated with a manifest file.

Note that there is one additional namespace defined for the Flex framework that was not in the Application tag.

The 2006 namespace definition exists for backwards compatibility with previous versions of the framework.

You can find the manifest files in the same directory as the flex-config.xml file.

I am opening the spark-manifest.xml file in Firefox.

You can see that this lists the spark classes in the Spark component package.

When I scroll down to the rpc section, you can see that the class files all reference the mx package.

However, the data services are available in the Spark namespace for convenience.

When I open the mxml-2009-manifest.xml file associated with the fx namespace, you can see that it does not list the compiler tags, like Declarations, Script or Style, which I said were in this namespace.

These tags are built directly into the compiler and so are not referenced in the manifest file.

Let's go back to the solution code from the Day 2 exercise I showed you at the beginning of this video.

I will discuss custom components in more detail later, but for now I just want to review some of the code that is in this project's custom component so that you get a sense of namespaces in action.

You can see that the Group container is in the Spark namespace while the Script tag is in the fx namespace.

Down in the UI components section of my code, you can see that the Form container, DropDownList control, TextInput control and FormItem control are all the Spark namespace.

The DateChooser control, however, is in the MX namespace.

I will show you how to identify the component namespaces and available components throughout this series.

Back in the project's main application file, in opening Application tag, you can see that there is a components namespace.

This is a custom namespace that could be named whatever you want.

Its value, components.*, refers to all of the files in the components directory of this project.

In other words, it points to the library of your own custom components.

The VehicleRequestForm.mxml file in the components directory is a custom component that you will create later.

To use this custom component in your main application, you refer to the namespace, components, and then reference the name of the component, VehicleRequestForm, without the MXML file extension.

You should always use components in the Spark namespace over those in the MX namespace whenever possible.

If you are able to avoid the MX namespace entirely, you could remove the namespace from the Application container. However, that will still leave the MX library reference in your project.

To completely remove the MX library reference, select Project > Properties >Flex Build Path > Library path.

When I expand the Flex 4.5 reference, you can see all the SWC files that represent the libraries that are used in this project.

Note the mx.swc file.

When I change the Component set option to Spark only, you will see that the mx.swc file, along with several others, is removed from the project.

For your next step, watch the video titled “Introducing Flex components and controls”.