XML - Namespaces - Flex 4[*]

R.G. (Dick) Baldwin

Abstract

Learn about namespace differences and some of the other differences between Flex 3 and Flex 4.

NOTE: **Click** Namespace02 [1] to run the Flex program from this lesson. *(Click the "Back" button in your browser to return to this page.)*

1 Table of Contents

*Version 1.1: Jun 12, 2010 7:55 pm -0500
†http://creativecommons.org/licenses/by/3.0/
1http://cnx.org/content/m34602/latest/Namespace02.html

2 Preface

2.1 General

This tutorial lesson is part of a series of lessons dedicated to programming using Adobe Flex.

> NOTE: The material in these lessons is based on Flex version 3 and Flex version 4. A distinction between the two will usually be made in those situations where that distinction is important.

A previous lesson in this series titled XML - Namespaces - Flex 3 [2] concentrated on teaching the XML concept of *namespaces* and illustrated the concept using a program written in Flex version 3.

Differences in namespaces between Flex 3 and Flex 4

Some of the first things that one is likely to notice when comparing Flex version 3 to Flex version 4 [3] are some obvious differences in the use of namespaces. Therefore, this is an opportune place in the series to introduce Flex version 4 and to explain some of the differences between the two versions.

2.2 Viewing tip

I recommend that you open another copy of this document in a separate browser window and use the following links to easily find and view the figures and listings while you are reading about them.

2.2.1 Figures

2.2.2 Listings

2.3 Supplemental material

I recommend that you also study the other lessons in my extensive collection of online programming tutorials. You will find a consolidated index at www.DickBaldwin.com [4] .

3 General background information

3.1 Historical perspective

Adobe's Flex is an XML-based programming language that is used to create programs that execute in the Adobe Flash Player [5] .

Teaching XML using Flex

[2]http://cnx.org/content/m34600/latest/
[3]http://www.adobe.com/devnet/flex/articles/flex3and4_differences.html
[4]http://www.dickbaldwin.com/toc.htm
[5]http://www.adobe.com/products/flashplayer/

In the Spring semester of 2010, I introduced Adobe's *Flex version 3* and the *Flex Builder 3 IDE* into a course named **Introduction to XML** that I had been teaching for several years at Austin Community College in Austin, TX. The concept of using Flex as the programming vehicle to teach XML was well received by the students.

During that same semester, Adobe released *Flex version 4* and replaced *Flex Builder 3* with a new IDE named *Flash Builder 4* . The new IDE supports both Flex 3 and Flex 4.

A fortunate circumstance

This is a fortunate circumstance insofar as the concept of using Flex to teach XML is concerned. Flex 4 is similar to, but very different from, and somewhat more complicated than Flex 3. The availability of the two versions of Flex makes it possible for the students to gain experience with two similar but different flavors of XML, both supported by the same IDE and both supported by similarly formatted documentation.

3.2 What is Flex?

As mentioned above, Flex is an XML-based programming language that is used to create programs that execute in Adobe's Flash Player. In order to understand Flex, and particularly the differences between Flex 3 and Flex 4, we need to start with the Flash Player and work backwards to Flex.

What is the Flash Player?

According to the Flash Player [6] website:

> *Adobe Flash Player is a cross-platform browser-based application runtime that delivers uncompromised viewing of expressive applications, content, and videos across screens and browsers. Flash Player delivers breakthrough web experiences to over 98% of Internet users.*

Flash Player is widely available

Many of the popular websites that people frequently visit require that the Flash Player be installed on the local computer in order to view the material on the website.

Typically if you visit a website that requires the Flash Player and you don't have it installed on your computer, you will be guided through the installation process. Therefore, a very large percentage of computers already have the Flash Player installed.

An execution engine

In short, the Flash Player is an execution engine that is used to execute or *play* programs that are written in the *ActionScript* programming language. *(See Baldwin's ActionScript programming website [7] .)*

What is ActionScript?

According to the ActionScript Technology Center, [8]

> *"Adobe ActionScript is the programming language of the Adobe Flash Platform. Originally developed as a way for developers to program interactivity, ActionScript enables efficient programming of Adobe Flash Platform applications for everything from simple animations to complex, data-rich, interactive application interfaces.*
>
> *First introduced in Flash Player 9, ActionScript 3.0 is an object-oriented programming (OOP) language based on ECMAScript – the same standard that is the basis for JavaScript – and provides incredible gains in runtime performance and developer productivity."*

What is the Adobe Flash Platform?

According to Adobe Flash Platform [9] ,

[6] http://www.adobe.com/products/flashplayer/
[7] http://www.dickbaldwin.com/tocActionScript.htm
[8] http://www.adobe.com/devnet/actionscript/
[9] http://www.adobe.com/flashplatform/

> *"The Adobe Flash Platform is an integrated set of technologies surrounded by an established ecosystem of support programs, business partners, and enthusiastic user communities. Together, they provide everything you need to create and deliver the most compelling applications, content, and video to the widest possible audience."*

The primary delivery mechanisms for applications built with the Adobe Flash Platform are the Adobe Flash Player [10] and Adobe Air [11] .

What is Adobe Air?

According to Adobe Air [12] ,

> *"The Adobe AIR runtime lets developers use proven web technologies to build rich Internet applications that run outside the browser on multiple operating systems."*

Once again, what is Flex?

Flex is an *XML-based* programming language that can be used to create *ActionScript* programs for execution in the *Flash Player* . When you compile a Flex project, it is first converted into an ActionScript program and the ActionScript program is compiled into a form suitable for execution by the Flash Player.

According to The Adobe Flash Builder 4 and Flex 4 Bible [13] by David Gassner,

> *When you compile a Flex application, your MXML code is rewritten in the background into pure ActionScript 3. MXML can be described as a "convenience language" for ActionScript 3 that makes it easier and faster to write your applications that if you had to code completely in ActionScript.*

Easier and faster is debatable

In my opinion, as a person with many years of object-oriented programming experience, it is debatable whether coding ActionScript programs in Flex is *easier and faster* than coding them in pure ActionScript. Any program that can be coded in Flex can also be coded in pure ActionScript, but the reverse is not true.

XML, not ActionScript

In any event, the purpose of the lessons in this series is to teach XML and not to teach ActionScript programming. *(ActionScript OOP is a different course that I teach at the college.)* Therefore, insofar as practical, the lessons in this series will concentrate on Flex programming and not on ActionScript programming.

However, to understand the differences between Flex 3 and Flex 4, it will sometimes be necessary to refer to ActionScript, particularly insofar as the documentation is concerned.

4 Preview

Run the Flex program named Namespace02

If you have the Flash Player plug-in *(version 10 or later)* installed in your browser, click here (p. 1) to run the program that I will explain in this lesson.

If you don't have the proper Flash Player installed, you should be notified of that fact and given an opportunity to download and install the Flash Player plug-in program.

Namespaces is an XML concept

The concept of **namespaces** is an XML concept. It is not a concept that is exclusive to Flex. However, because Flex is an XML-based programming language, Flex makes heavy use of namespaces.

I explained the concept of XML namespaces in the earlier lesson titled XML - Namespaces - Flex 3 [14] . I also presented and explained a relatively simple Flex program that illustrated the use of XML namespaces to resolve name conflicts.

[10]http://www.adobe.com/products/flashplayer/?promoid=DJDWD
[11]http://www.adobe.com/products/air/?promoid=DJDTL
[12]http://www.adobe.com/products/air/?promoid=DJDTL
[13]http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470488956.html
[14]http://cnx.org/content/m34600/latest/

Some of the differences between Flex 3 and Flex 4

In this lesson, I will present a somewhat broader view of namespaces and will also present and explain a program that illustrates some of the differences between Flex 3 and Flex 4.

In explaining the differences between Flex 3 and Flex 4, I will need to dig a little more deeply into the Flex programming language than was the case in the earlier lesson.

The program that I explained in the earlier lesson was written exclusively using Flex 3. The program that I will explain in this lesson was written exclusively in Flex 4. The new Flex 4 program approximates the look and feel of the Flex 3 program from the earlier lesson.

5 Discussion and sample code

Two ways to create Flex projects

As I explained in the earlier lesson, Flex projects can be created using nothing more than a text editor and a Flex software development kit *(SDK)* that is freely available from the Adobe website. However, to make the development of Flex projects a little easier, Adobe previously sold a product named *Flex Builder 3* and now sells a replacement product named *Flash Builder 4* , which includes the Flex 3 and Flex 4 SDKs along with a visual project editor.

The project that I explained in the earlier lesson was created using Flex Builder 3. The project that I will explain in this lesson was created using Flash Builder 4.

Free for educational use

As of June 2010, Adobe provides free copies [15] of Adobe Flash Builder 4 Standard to:

- Students, faculty and staff of eligible educational institutions
- Software developers who are affected by the current economic condition and are currently unemployed
- Event attendees who receive a special promotional code at their event

5.1 Skeleton mxml code and namespaces

Skeleton mxml code for a new Flex 3 project

When you create a new Flex 3 project in Flex Builder 3 or Flash Builder 4, a skeleton of the required mxml file is created for you. Listing 1 shows the contents of such a skeleton mxml file for a Flex 3 project.

Listing 1: Skeleton mxml code for a new Flex 3 project.

```
    <?xml version="1.0" encoding="utf-8"?>

<mx:Application
  xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">

</mx:Application>
```

Created using Flex Builder 3

The skeleton code shown in Listing 1 was created using Flex Builder 3, but the skeleton code for a Flex 3 project is essentially the same regardless of whether it is created using Flex Builder 3 or Flash Builder 4. *(Flash Builder 4 inserts a couple of relatively insignificant size attributes that are not inserted by Flex Builder 3.)*

The namespace (xmlns) attribute

[15] http://www.adobe.com/devnet/flex/free/index.html

In the earlier lesson, I explained the concept of the *root element* , and I explained that the term **xmlns** is the required name for a *namespace* attribute. *(This is true for XML in general and not just for Flex mxml.)* While it isn't necessary in general to include a namespace attribute in the root element, when a namespace attribute is included in the root element, it becomes the *default namespace* for the entire document.

Namespace is always required for a Flex project

Even though it isn't necessary to include a namespace attribute in the root element of a general XML document, it is always necessary to include the namespace attribute shown in Listing 1 in the root element of the main mxml document for a Flex 3 project. That is why Flex Builder 3 includes it in the skeleton code for the project.

What does this mean?

The inclusion of the default namespace attribute shown in Listing 1 means that all elements with names that refer to components from the standard Flex 3 library of components must be prefixed with **"mx:"** .

Skeleton mxml code for a new Flex 4 project

As with a Flex 3 project, when you create a new Flex 4 project in Flash Builder 4, a skeleton of the mxml file is created for you. Listing 2 shows the contents of such a skeleton mxml file for a Flex 4 project.

Listing 2: Skeleton mxml code for a new Flex 4 project.

```
    <?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               minWidth="955"
               minHeight="600">
  <fx:Declarations>
    <!-- Place non-visual elements (e.g., services,
    value objects) here -->
  </fx:Declarations>
</s:Application>
```

More namespace attributes in the root element

If you compare Listing 2 with Listing 1, you will see that the namespace attributes in Listing 2 are different from those in Listing 1, and there are more of them in Listing 2.

Listing 1 has only one namespace attribute while Listing 2 has three namespace attributes.

Mix or match Flex components

You can use Flash Builder 4 to create projects that

- use Flex 3 exclusively
- use Flex 4 exclusively
- use a mixture of the two

Must specify compiler version for project

When you create a new project in Flash Builder 4, you must specify whether the project is to be compiled using the Flex 3 compiler or the Flex 4 compiler.

Different versions of the skeleton code

If you specify the Flex 3 compiler, the skeleton code will look like Listing 1 *(with a couple of additional sizing attributes)* . For that case, you must use Flex 3 components exclusively.

If you specify the Flex 4 compiler, the skeleton code will look like Listing 2. In that case, you can use Flex 3 components, Flex 4 components, or a mixture of the two.

What do these namespace attributes mean?

Building on what I explained earlier, the inclusion of the namespace attributes with the name **"mx"** in Listing 1 and Listing 2 means that all elements with names that refer to components from the Flex 3 library of components must be prefixed with **"mx:"** . *(You will see examples of this in code fragments later in this lesson.)*

The inclusion of the namespace attribute with the name **"s"** in Listing 2 means that all elements with names that refer to the new components from the Flex 4 library of components must be prefixed with **"s:"** . *(You will also see examples of this in code fragments later in this lesson.)*

Resolution of duplicate names

The Flex 3 library and the Flex 4 library contain many components with the same names, such as **Label** and **Button** . Therefore, the name of the component alone is not sufficient to identify which of two components having the same name is to be used at a particular location in the program. The **"mx:"** prefix and the **"s:"** prefix are the mechanisms by which you identify the correct component to the compiler.

NOTE: For those with knowledge of ActionScript or Java programming, this is analogous to using a package name to identify a class in those programming languages.

You can read more on the topic of required namespaces here [16] .

5.2 The sample program named Namespace02

Figure 1 shows the output from the Flex 3 program named **Namespace01** that I explained in the earlier lesson on this topic.

Output from Namespace01.

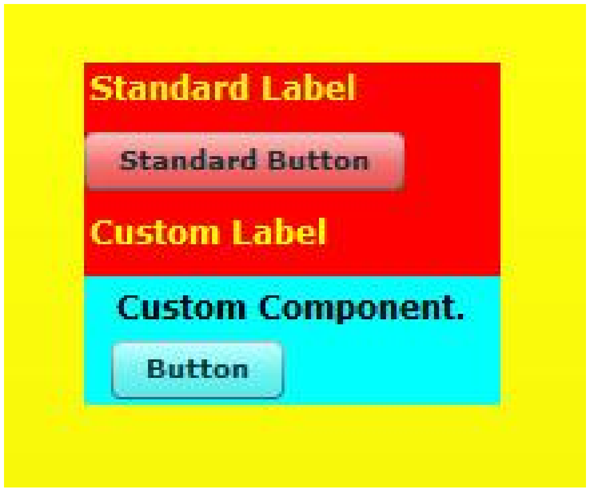

Figure 1: Output from Namespace01.

Figure 2 shows the output from the Flex 4 program named **Namespace02** that I will explain in this lesson.

[16] http://www.adobe.com/devnet/flex/articles/flex3and4_differences_03.html

Output from Namespace02.

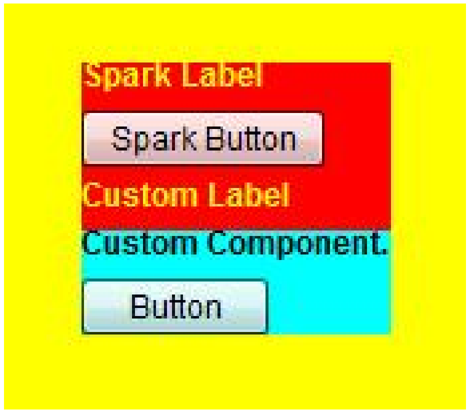

Figure 2: Output from Namespace02.

Mostly default look and feel

In both programs, the top portion of the output was purposely colored red and the bottom portion was purposely colored cyan. Otherwise, the colors, sizes, positions, and shapes of the components in both programs were allowed to take on default values.

The project tree for the project named Namespace02

The project tree for the Flex 4 project named **Namespace02** is shown in Figure 3.

Project tree for the project named Namespace02.

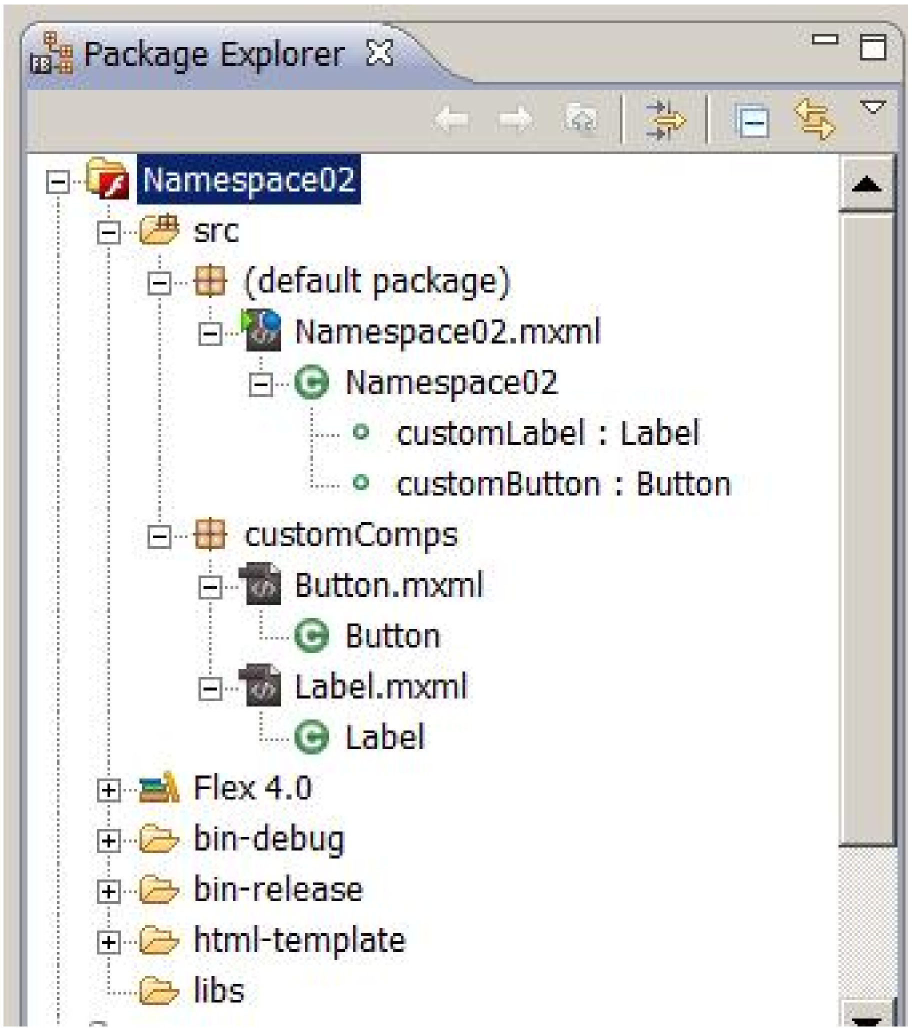

Figure 3: Project tree for the project named Namespace02.

A comparable image for the Flex 3 project named **Namespace01** was provided in the earlier lesson. If you compare the two, you will see that more information is displayed in the project tree for the Flex 4 project in Figure 3.

Major items of interest

For purposes of this lesson, we will be primarily interested in the following items showing in Figure 3. Those are the items that I had to create in order to create the project.

- The file named Namespace02.mxml
- The folder named customComps
- The file named Button.mxml

- The file named Label.mxml

Two buttons, three labels, etc.

As I explained in the earlier lesson, the project named **Namespace01** creates a GUI with two buttons and three labels in **VBox** containers with red and cyan backgrounds as shown in Figure 1.

All are mx components

Because that project was created exclusively using Flex 3, all of the components shown in Figure 1 are Flex 3 components. I will sometimes refer to them as **"mx"** components because of the name of the namespace attribute shown in Listing 1.

No VBox components in Namespace02

Because the Flex 4 program named **Namespace02** was intended to replicate **Namespace01** , it also contains two buttons and three labels. However, as you will see later, they are not in **VBox** containers because there is no **VBox** container in Flex 4. Instead, they are in containers named **Group** and **VGroup** .

All are Spark components

Because **Namespace02** was created exclusively using Flex 4, all of the components are Flex 4 components. I will sometimes refer to them as **"Spark"** components on the basis of the last word in the value of the namespace attribute named **"s"** in Listing 2.

NOTE: The names "mx" and "Spark" actually derive from ActionScript package names, but an explanation of that is beyond the scope of this lesson.

The main mxml file for Namespace01

Listing 3 shows the code in the main mxml file for the Flex 3 project named **Namespace01** .

Listing 3: The main mxml file for Namespace01.

```
    <?xml version="1.0"?>
<!--
Namespace01
Illustrates the use of namespaces to avoid name conflicts.
-->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
                xmlns:MyComps="customComps.*"
                backgroundColor="#FFFF00">

  <!--Add a standard VBox container-->
  <mx:VBox backgroundColor="#FF0000">

    <mx:Label text="Standard Label"
      color="#FFFF00"
      fontSize="12"
      fontWeight="bold"/>
    <mx:Button label="Standard Button" />

    <MyComps:Label id="customLabel"/>
    <MyComps:Button id="customButton"/>
  </mx:VBox>

</mx:Application>
```

The Application element

I will explain **mxml** syntax in more detail in future lessons, so I'm not going to go into syntax issues at this point in time. Suffice it to say that the **Application** element in Listing 3 represents the entire program. The behavior as well as the look and feel of the program is defined by the attributes and the content of the **Application** element. Everything in the program is part of the attributes or the content of the **Application** element.

An mx:VBox element

From what you already know about XML, you can see that an element named **mx:VBox** is part of the content of the **Application** element. Very briefly, in Flex, an **mx:VBox** element is a container element that can contain other elements. Couched in visual terms such as Figure 1, an **mx:VBox** object can contain other components such as labels and buttons.

Note that the **mx:VBox** element name has an **mx** prefix, meaning that it represents a component from the Flex 3 library as explained earlier.

The backgroundColor attribute of the mx:VBox element

Also note that the **mx:VBox** element has an attribute named **backgroundColor** with a value of **"#FF0000"** . In a future lesson, I will explain that this is a hexadecimal value that represents the color red at maximum intensity. This attribute produces the red background color that you see in the upper portion of Figure 1.

> NOTE: The lower portion of Figure 1 also has a red background color, but it is covered by another mx:VBox element with an opaque cyan background color.

The backgroundColor attribute for the Application element

While we are discussing background colors, it is also worth mentioning that the **application** element has an attribute named **backgroundColor** with a value of **"#FFFF00"** . This is the hexadecimal value for yellow and causes the background color of the entire Flash Player window to be yellow.

Contents of the mx:VBox element

The **mx:VBox** element contains the following four elements:

- mx:Label
- mx:Button
- MyComps:Label
- MyComps:Button

In the earlier lesson, I explained that the first two of these four elements represent components from the standard Flex 3 library. *(Hence the "mx:" prefix.)* The last two represent custom components that were constructed using components from the standard Flex 3 library.

The MyComps:Button element

If you go back to the earlier lesson [17] and examine the code for the custom component named **MyComps:Button** , you will see that it has an **mx:Label** and an **mx:Button** in an **mx:VBox** container with a backgroundColor value of **"#00FFFF"** *(cyan)* . This produces the cyan rectangle containing the label and the button in the bottom portion of Figure 1.

The main mxml file for Namespace02

The main mxml file for the Flex 4 project named **Namespace02** is shown in Listing 4.

Listing 4: The main mxml file for Namespace02.

```
    <?xml version="1.0" encoding="utf-8"?>

<!--File: Namespace02.mxml
```

[17] http://cnx.org/content/m34600/latest/#Listing_4

```
This is a Flex 4 version of the Flex 3 program
named Namespace01-->

<!--Declare a namespace as the folder named customComps,
which contains a custom label component and a second
custom component consisting of a Spark Label and a
Spark Button. Then declare the three namespaces required
by Flex 4. Finally cause the background to be yellow.-->
<s:Application xmlns:MyComps="customComps.*"
               xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               backgroundColor="0xFFFF00">

  <!--Put a Spark Label and a Spark Button along with two
  custom components in a Spark Group with a red background
  color.-->
  <s:Group horizontalCenter="0" verticalCenter="0">

    <!--Create a red rectangle to serve as the background
    color for the Group-->
    <s:Rect width="100%" height="100%">
      <s:fill>
        <s:SolidColor color="0xFF0000" />
      </s:fill>
    </s:Rect>

    <!--Add a Spark VGroup to contain the components and
    cause them to be laid out vertically.-->
    <s:VGroup>
      <!--Add two Spark components to the VGroup-->
      <s:Label text="Spark Label"
               color="#FFFF00"
               fontSize="12"
               fontWeight="bold"/>
      <s:Button label="Spark Button" />

      <!--Add two custom components to the VGroup-->
      <MyComps:Label id="customLabel"/>
      <MyComps:Button id="customButton"/>
    </s:VGroup>

  </s:Group>

</s:Application>
```

Lots of comments

As you can see, I included lots of comments in Listing 4 in an attempt to make it as self-explanatory as possible.

In this lesson, I will concentrate on the differences between this Flex 4 project and the Flex 3 project named **Namespace01** that arise from creating the two projects using different versions of Flex.

Order of attributes is not important

Let me begin by explaining that in XML, the order in which you write the attributes for an element doesn't matter so long as they are all there with the correct syntax, the correct names, and the correct values.

More and different namespace attributes

As I explained earlier, a Flex 4 project often has three required namespace attributes and almost always has two. *(Because I didn't use any mx components in this program, I could have removed the namespace attribute named mx from Listing 4.)*

Other than the namespace attributes, the **application** element in Listing 4 has the same attribute names and values as the **application** element in Listing 3.

No VBox element in Namespace02

The next thing to notice is that there is no **mx:VBox** element in Listing 4. Instead, there is an **s:Group** element *(a Flex 4 Spark component)* that replaces the **mx:VBox** element and serves as a container for the labels and the buttons.

No backgroundColor attribute

The **s:Group** element has two positioning attributes that cause it to appear in the center of the Flash Player window, but it does not have an attribute named **backgroundColor** . Like many of the Spark components, and unlike many of the mx components, the **s:Group** element does not have built-in attributes that are used to control its appearance. Instead, other ways must be found to control the appearance of many Spark components.

A red rectangle

In this case, Listing 4 causes the **s:Group** element to appear to have a red background by causing it to contain a red rectangle of exactly the right dimensions to completely fill the **s:Group** element. This produces the red background color in the upper portion of Figure 2.

NOTE: As with Figure 1, the lower portion of Figure 2 also has a red background color, but it is covered by another smaller rectangle with an opaque cyan color.

Add an s:VGroup container

Then Listing 4 adds a Spark **s:VGroup** container to serve essentially the same purpose as the **mx:VBox** container in Listing 3 *(except that it doesn't control the red background color)* . The following elements are added to the **s:VGroup** element in Listing 4 In a manner very similar to Listing 3:

- s:Label
- s:Button
- MyComps:Label
- MyComps:Button

The first two elements in the above list are Spark elements having similar characteristics to the mx elements having the same names.

The last two elements in the above list are custom components having similar characteristics to the custom components having the same names in the earlier program.

Contents of the file named Label.mxml

The contents of the custom component file named **Label.mxml** are shown in Listing 5.

Listing 5: Contents of the file named Label.mxml.

```
   <?xml version="1.0" encoding="utf-8"?>

<!--Create a custom label by putting a Spark Label in
a Spark Group-->
<s:Group xmlns:MyComps="customComps.*"
```

```
         xmlns:fx="http://ns.adobe.com/mxml/2009"
         xmlns:s="library://ns.adobe.com/flex/spark"
         xmlns:mx="library://ns.adobe.com/flex/mx">

  <s:Label
    text="Custom Label"
    color="#FFFF00"
    fontSize="12"
    fontWeight="bold"/>

</s:Group>
```

Contents of the file named Button.mxml

The contents of the custom component file named **Button.mxml** are shown in Listing 6.

Listing 6: Contents of the file named Button.mxml.

```
    <?xml version="1.0" encoding="utf-8"?>

<!--Create a custom component by putting a Spark Label
and a Spark Button in a Spark VGroup inside of a Spark
Group with a Cyan background color.-->
<s:Group  xmlns:MyComps="customComps.*"
          xmlns:fx="http://ns.adobe.com/mxml/2009"
          xmlns:s="library://ns.adobe.com/flex/spark"
          xmlns:mx="library://ns.adobe.com/flex/mx">

  <!--Fill the entire group with a cyan rectangle-->
  <s:Rect width="100%" height="100%">
    <s:fill>
      <s:SolidColor color="0x00FFFF" />
    </s:fill>
  </s:Rect>

  <!--Put a Spark VGroup in the Group and put a Spark
  Label and a Spark Button in the VGroup-->
  <s:VGroup>
    <s:Label
      text="Custom Component."
      color="#000000"
      fontSize="12" fontWeight="bold"/>

    <s:Button
      label="Button"/>
  </s:VGroup>
</s:Group>
```

No further explanation needed

Assuming that you understand the contents of the files named **Label.mxml** and **Button.mxml** in the Flex 3 program in the earlier lesson, and assuming that you understood the explanation of the differences between the two main mxml files given above, the comments in Listing 5 and Listing 6 should serve as a sufficient explanation of the code in Listing 5 and Listing 6.

6 Run the program

I encourage you to run (p. 1) this program from the web. Then copy the code from Listing 4 through Listing 6. Use that code to create your own projects. Compile and run the projects. Experiment with the code, making changes, and observing the results of your changes. Make certain that you can explain why your changes behave as they do.

7 Resources

I will publish a list containing links to Flex resources as a separate document. Search for Flex Resources in the Connexions search box.

8 Miscellaneous

This section contains a variety of miscellaneous materials.

NOTE: **Housekeeping material**

- Module name: XML - Namespaces - Flex 4
- Files:
  - Flex0086a\Connexions\FlexXhtml0086a.htm

-end-