Flex in a Week, Flex 4.5
Video 1.04: Using Flash Builder

Now that you understand what the Flex framework is and your choices for development environment, let's focus on using Flash Builder.

In this video, you will learn the proper terminology for using this Eclipse-based tool.

You will also learn how to organize your application code into projects, which you can also export to share with others, or import to bring into Flash Builder.

I will show you how to organize your projects in either the default workspace or your own custom workspaces.

Lastly, I will discuss the coding methodology for this series.

When you first open Flash Builder, you will be presented with the Flash Builder Start Page.

My presentation window is somewhat small, so I am double-clicking on the Start Page tab to maximize the display.

You can do this with any tab in Flash Builder.

The Start Page provides you with links to tutorials and other useful resources.

I am double-clicking on the tab to minimize the view again and then clicking on the close icon in the tab to close the Start Page.

You can always access this resource again by selecting Help > Flash Builder Start Page.

This screenshot shows the default Flash Builder environment, which is, in Eclipse terms, called the Workbench.

I have the standalone version of Flash Builder.

If you are using the plug-in version, then your interface will likely look a little different.

Let me start by pointing out some basic terminology.

These panels in the interface are called views.

The main code area is called the Editor and the toggle between the code and visual views are called Source and Design modes.

The way that the views are laid out can be changed.

Each configuration is called a perspective.

Here in the live application, I have a few projects open.

We will talk about projects in a bit.

If you can't see all of your perspectives, you can resize this bar or click on the dropdown.

I am clicking on the Flash Debug perspective so that you can see the views rearrange.

You will use the Flash Debug perspective many times in this training to analyze your application data, and otherwise debug your code.

The main perspective is called the Flash perspective, and I'm switching back to that now.

If you have a preference for your own layout, you can rearrange the views by dragging them around and then save the changes using the Window > Save Perspective As option.

You can reset your display at any time by selecting the Reset Perspective option.

I am opening the exercise 3.02 solution files and expanding the package until I locate the main application file, which I am double-clicking on to open.

Here in the Editor, I can change some settings by right-clicking on the margin bar.

I am turning on line numbers.

I can also switch to Design mode by clicking this button.

Note how the views change in Design mode.

You will use this Components view often during training to layout components.

You can access more views from the Window menu, but the primary ones that you will use are the ones that already appear in the default perspective in both Design and Source mode.

The Package Explorer displays all of the files and folders for your code projects and will automatically refresh each time you add, delete or modify a resource.

The Problems view shows errors and warning messages based on your code.

Here I am introducing an error in the code.

When I save the file, you can see the error appear in the Editor view as well as in the Problems view.

I am undo-ing my code change and then re-saving the file.

The error message disappears.

The Outline view hierarchically presents all of the user interface and code elements in a file. You can click on an element in the view to locate it in the code.

The Package Explorer view shows many projects.

A Flash Builder project is a grouping of resources that make up a Flex application.

The ones that are grayed out are closed.

I can open other projects – one at a time or multiple – by right-clicking on them and choosing Open Project.

I don't have any referenced projects for these, but I will click Yes anyway.

I can drill down into the src directory and then the default package to open up one of the resources.

When I double-clicking on a file you can see that it opens in a second Editor tab.

I am closing these projects again by selecting them and then right-clicking on them and choosing Close Project.

You can, of course, have multiple projects open at one time, but I tend to like to close the ones that I'm not using to save on system resources.

Let me show you how to create a project.

I am selecting File > New > Flex Project.

I am naming this project EmployeeDirectory.

Keep in mind that the project name must only contain numbers and letters, and by common convention, it is often in upper camel case.

It can contain a space, but by convention, most project names do not have spaces in them.

I am going to leave the project location to the default directory created in this workspace.

We will discuss workspaces in more detail in a few minutes.

This is going to be a web application, not an AIR desktop application, so I will leave the Application type set to Web.

I want to use the newest features of Flex 4.5 so I will leave the Flex SDK version set to Flex 4.5.

You can choose an earlier SDK if you are creating an application that you want to be compatible with a Flash Player earlier than Flash Player 10.

I am clicking the Next button.

I will not be connected to a server technology for this first project, so I am leaving that value set to the default, which is None.

By default, Flash Builder will compile my Flex application into a directory called bin-debug within my project folder.

By convention, most developers do not change this folder.

You will learn more about the compiled application and the associated files in later videos.

For now, I will click Next.

You can decide whether your project's component set includes only MX, only Spark, or MX + Spark components. Choose MX only for Flex 3 development and choose MX + Spark or Spark only for Flex 4.5 development.

For this project, I will leave the default MX + Spark component set.

If you decide later that you will not use MX components, then go back into your project settings and update this selection to avoid unnecessarily linking the MX library.

The Main application file is automatically named the same as the project name.

In this case, the file is EmployeeDirectory.mxml.

I am leaving that setting and pressing Finish.

The EmployeeDirectory project is now available in the Package Explorer view and the EmployeeDirectory.mxml file is open in Editor view.

The file contains some MXML code that you will learn more about in future videos.

Next, I will explain how projects are grouped into workspaces.

A Workspace is different than a Workbench.

Remember, a Workbench is the Flash Builder development environment.

A Workspace is a file system directory that contains all the files and folders that track a group of projects in Eclipse.

You can see the default location of a Flash Builder workspace by selecting the File menu, then Switch Workspace and then Other.

In this example, the default workspace is in the Windows Documents and Settings folder.

I am copying this path and then locating it in my file system.

You can see that all the projects that you saw in the Package Explorer view are actually folders in this workspace.

The folders with periods in front of them are Flash Builder folders to manage the workspace.

Creating workspaces is a very useful way to logically organize your Flex projects for different clients or other purposes.

Let me show you how to create a workspace.

First, I am creating a folder on my C drive that I am naming adobeTraining.

Within that folder, I am creating a subfolder named fiaw, which stands for Flex in a Week, the name of this training series.

To switch to the new workspace in Flash Builder, I am selecting File > Switch Workspace > Other and browsing to the fiaw directory.

I'll click OK and then when I click the OK again, Flash Builder closes and opens back up in the new workspace.

Notice that there are no projects listed in the Package Explorer view, but there is now a metadata settings folder in the workspace directory.

On the Flex in a Week landing page, there are links under each Day heading for the entire Day's project files.
Each zip is a workspace and I will show you now how to set it up.

This is the Day 1 zip file that I downloaded.

I am unzipping the contents to my C drive.

First note that you might see a Mac folder in this zip file on the Windows OS.

This was automatically created on the Mac that zipped the file.

You can safely ignore it or delete it, which is what I am doing.

Note that the unzipping process has created an extra level of folder, which I am removing.

Not all programs do this, but be careful of what your system does.

You want to reference the folder that contains all the projects.

On my C drive, you now see the FiaW_Day1_ProjectArchive folder with project folders, as well as some video transcripts and a ReadMe file.

The .metadata folder at the top should be a familiar sight to you.

It tells you that this is a Flash Builder workspace.

Back in Flash Builder, I am selecting File > Switch Workspace > Other and then navigating to that new folder.

Be sure to select the folder that contains all the projects.

Flash Builder will close.

When it re-opens, you can see all of the project files for Day1.

Note that there are both starter and solution projects.

You will begin each exercise with the starter files and you can use the solution files for reference or comparison with your own work.

Again, my practice is to close all projects that I'm not using to save on system resource.

I am selecting all the projects except for one and right-clicking on them to close them.

I am drilling down into the one open project and then opening the main application file.

I want to show you some common Flash Builder preferences that you might want to change.

I am selecting Window > Preferences to open the Flash Builder Preferences dialog.

To change the font size in the code editor, you select General > Appearance > Colors and Fonts > Basic > Text Font > and then click the Edit button.

I am changing the font size to 14 and clicking OK.

I can also change default indentation for code by selecting Flash Builder > Editors > ActionScript Code > Indentation.

You can see in the Preview window that all function arguments are automatically indented to the function name.

If you prefer that they aren't indented so far, you can uncheck the Align parameters in function declarations.

Note the change in the code display.

You can also select MXML from the Editor menu and then Indentation to uncheck Align attributes.

Again, the Preview window shows me the changes.

Because I want my code in these videos to match the code in the associated exercises, I will reverse these indentation changes.

Flash Builder 4.5 also includes Code Templates, which are definitions for blocks of code that you can activate by typing CTRL + Space in the Editor.

When you type that keyboard shortcut and then start typing a name from one of these lists, Flash Builder will provide you with options for inserting the code.

Note that you can click the New button to create your own Code Templates.

When I click OK in the Preferences dialog you can see that the font size in the editor has been updated.

Flash Builder preferences that you set in one workspace are specific to that workspace.

I am selecting File then Switch Workspace to switch back to the default workspace.

When Flash Builder opens back up in the default workspace, you can see that the font size is the smaller default size since the changes I made earlier to the preferences were to a different workspace.

If you don't want to download an entire Day's worth of exercises, you can import individual projects for each exercise.

On the exercise page, locate the starter and solution zip files.

You cannot directly import the ZIP files into Flash Builder.

First unzip them to access the FXP file, which is a Flash Builder project archive.

Then in Flash Builder, select File > Import Flash Builder Project.

Next to the File field, click the Browse button and locate the FXP file you want to import.

Click Open and then Finish.

But if you notice, my new project is going to be called ex1_02_starter_1.

I already have a project in my workspace called exercise 1_02 starter so Flash Builder is automatically appending the _1 to the project name.

Now the project is available in Flash Builder.

If you work in a team environment or need to send a Flex project to someone, you can create a Flex project archive by right-clicking on the project and selecting Export.

Note that this does not lock the file or in any way implement version control.

Flash Builder does support version control, but that is outside the scope of this video series.

I am selecting Flash Builder > Flash Builder Project and click Next.

Then simply create the FXP file in your location of choice.

I’m renaming mine to be testExport and then clicking Finish.

Here is the exported project on my desktop.

Let's take a look at the code in the main application solution file for exercise 3.02.

I am double-clicking on the Editor tab so that we can see the code better.

You can see a series of HTML-like comments in the file.

Flex MXML comments are the same as HTML comments: they must start with an angle bracket, an exclamation point and two dashes.

They end with two dashes and a closing angle bracket.

The first comment just tells you which topic is covered for the exercise.

The others orient you to the order in which we will be placing the code as you build the application.

You will learn about adding ActionScript code to your MXML applications in the Script tag block later, but for now, note that ActionScript comments follow the standard double forward slash convention.

Placing the MXML and ActionScript code in this order is not an absolute practice or even a best practice.

It's merely a coding convention that I have decided to follow to ensure consistency in the presentation and exercises.

However, this code order was shown to the Flex engineering team and met their approval.

You may have noticed that each property for an MXML tag is on a separate line in the code file.

You can place all of the code for an MXML tag on one line, but it is a common practice to separate each property to its own line for easy review.

It's really a trade-off between horizontal scrolling and vertical scrolling of your code and is a purely a personal preference.

For this video series, I will stick to one property per line since I have limited horizontal space to work with while recording.

The only exception to the practice of one property per line is when the properties are related.

Since x and y are related, they can share one line of code.

Also, if one of my components has an id property, which is a unique identifier for the element, then I will place that property on the same line as the MXML tag name.

You will continue to learn much more about Flash Builder throughout the rest of the training.

Some of the highlights include Design mode, code assist, quick assist, the Debugger, the Network Monitor and data-centric development tools.

For your next step, watch the video titled "Compiling and viewing the application".