Flex in a Week, Flex 4
Video 1.05: Compiling and viewing the application

In this video, you will learn how to run your Flex application from within Flash Builder.

This process will compile a SWF file and generate a convenient HTML wrapper file which you can modify for your own browser display needs.

You will also learn about some basic compiler settings and deal with a couple common compiler and browser issues that you might encounter during development.

In Flash Builder's Package Explorer, I have three projects currently open.

The one I want to focus on right now is this solution project for an application that you will build in Day 5.

I am going to select the other two projects and right-click to close them.

I'm expanding the remaining project to look in the src > default package directory for the main application file, which I'm double-clicking on to open.

Bear with me, please, in these first few videos as I
spend a little extra time emphasizing the Flash Builder features and terminology that you learned in the earlier video on using Flash Builder.

I will be less descriptive of my actions as the series progresses.

Let's focus for a minute on this main application file.

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

This is the MXML code that handles the user interface display.

I am double-clicking again on the Editor tab to minimize the view and then expanding the skins folder.

I am opening the skins PanelSkin.mxml file.

You will build this in a later exercise and it is a skin for the Panel container.

I'm going to maximizing the view, and as I scroll down, you can see some ActionScript code to handle the Panel header bar color.

You will learn more about all of this code throughout the course.

For now, focus on the fact that all of this code, plus the code in these other custom skin and component files, will be compiled into a SWF file.

It is the actual SWF file that is moved to the production environment for download and interaction with an end user.

Run_application.camrec
I am clicking on the Run button, here in the menu bar, to have Flash Builder display the Flex application in a browser.

Note two things.

First, by default, Flash Builder will use your system's default web browser. In this case, mine is set to Firefox.

You can change the default browser in Flash Builder by selecting Window > Preferences > General > Web Browser.

Also consider the URL path to the application.

The URL points to the default Adobe Flash Builder 4 workspace that I discussed in an earlier video.

Then it points to the exercise 5.08 solution project and then to the bin-debug directory, which I'll discuss in a minte.

And then finally to an HTML file.

Note that the name of the HTML file is based on the name of the main application file, which is in this case EmployeePortal.

Bin_debug.camrec

Return to Flash Builder and expand the bin-debug directory.
Show them the SWF and HTML file.

Back in Flash Builder, I'm locating the bin-debug directory in the Package Explorer view.

Remember that when you create a project, as I'm showing you here, one of the settings defines the output folder for the compiled SWF.

By default, this is defined as the bin-debug directory.

You can change the folder name, but it is common practice to leave it alone.

I'm going to close this wizard.

When I expand the bin-debug directory, you can see the HTML file that was referenced in the browser URL as well as the compiled SWF file.

I'm going to double-click on this view to expand it so we can see a little better and I'm also going to close up a couple of these folders.

Later in this video, I will discuss the other files and folder in the bin-debug directory.

However, for now, note that the bin-debug folder does not include the components or skins folder since all the code in these source files are compile into the SWF.

Build_automatically.camrec

I'm double-clicking on this view to return the environment back to the default display.

So by default, if you look in the Project menu you will see that there is an option called Build Automatically that's checked.

This option tells Flash Builder to attempt to compile the application each time the project files are saved and this can be quite convenient.

Let's say that I accidentally mis-type some code.

So, let me go down here and mess up some of my code.

And then when I save it, you will see the compile error is immediately shown here in the Problems view.

The feature, however, is not always convenient, since sometimes the compile process can take a while and you may not want that to happen each time you save.

Flex_Compiler_settings.camrec

I've corrected my syntax error and re-saved the file so the error message is gone.

Let’s now spend some time looking at the Flex compiler settings, which you can find by selecting
Project > Properties > Flex Compiler.

The settings are organized into these four sections:

Flex SDK version
Adobe Flash Player options
Compiler options
HTML wrapper

SDKs.camrec

Remind them of the sdks dir in application installation dir
By default, your Flex applications will be compiled for the Flex 4 SDK.

If you want to use an earlier framework, you can select this radio button and choose an SDK.

The options are Flex 3.5 and 4.0.

In an earlier video, I showed you the installation directory for Adobe Flash Builder 4.

Note that the sdks listed here match the ones listed in the Flex Compiler settings.

In a later video, you will learn about some of the new Flex components available in Flex 4.

If you switch to an earlier SDK, just note that you will not be able to use these new components in your application.

SWF_Version.camrec

Open the HTML wrapper file

var swfVersionStr = "10.0.0";

In the Adobe Flash Player options section, you can explicitly target a version of the Flash Player.

Let me show you how this relates to the HTML wrapper file generated by Flash Builder for the SWF display.

I am closing the Flex Compiler settings for a moment.

Now, if I double-click on the HTML file, it opens in the internal web browser, which I don't want.

So let me close that and right-click on the file to select Open With > Text Editor.

I am double-clicking on the Editor tab to expand the window and then scrolling down to locate the JavaScript variable, right here, that declares the minimum Flash Player target to be 10.0.0 for this application.

If the end-user's browser does not have this version of the Flash Player, the user will be prompted to upgrade.

I am double-clicking on the Editor tab again, to minimize this view.

Compiler_options.camrec

The third section of the compiler settings allows you to set some common compiler options.

The first checkbox declares whether you want the Flex 3 MX components to use the new Flash Text Engine.

You will learn more about MX components and the FTE in later videos.

The second option tells the compiler to copy all of your assets, like images, into the bin-debug folder.

You will need to do this if your assets are being called at runtime from the SWF file.

To learn more about the Generate accessible SWF file option, go to the Flex accessibility web site here:

http://www.adobe.com/accessibility/products/flex/

Enable strict type checking enforces strict data typing during compilation and Enable warnings enables warning messages in Flash Builder, like the one you saw when I created the error in my code.

Lastly, you can type additional compiler arguments into this text field.

Compiler_HTML_Wrapper.camrec

The HTML wrapper section defines some settings in the HTML wrapper file.

If you have the Use Express Install checkbox selected the wrapper file will check whether the user has JavaScript enabled to perform Flash Player detection.

The browser also must have at least version 6.0.65 of the Flash Player installed so that it can initiate the update process that installs the latest version of the Player.

The Enable integration with browser navigation selection will allow the user to use the back and next buttons in the browser to navigate the Flex application.

This is also known as deep linking.

Back in the bin-debug directory, note that the PlayerProductInstall.swf file is necessary for Express install and the history directory is related to the deep linking feature.

Embedding_SWF_in_HTML.camrec

Embedding the SWF in the HTML wrapper

- The HTML wrapper implements SWFObject 2
  - Must deploy swfobject.js

```
<script type="text/javascript" src="swfobject.js"></script>
```

- Abstracts implementation details about Flash Player detection, embedding, and other features so that you only need to call a single method to embed your SWF file

```
swfobject.embedSWF(
        "[ApplicationName.swf]", "flashContent",
        "990", "100%",
```

swfVersionStr, xiSwfUrlStr,
flashvars, params, attributes);

Here is the HTML wrapper file code again.

Notice that this HTML wrapper implements a JavaScript library named swfobject.

This is actually version 2 of the SWFobject library, which is a standards-based library that embeds SWF files in HTML pages.

This library also abstracts the implementation details about the Flash Player detection, the SWF embedding and more.

Here is the code to embed the SWF file which is generated for you by Flash Builder.

You can take this HTML file and modify it any way you want for deployment in your application.

RSLs.camrec

Understanding the runtime shared libraries

- Flex provides Flex framework runtime shared libraries (RSLs)
  - Allows Flex applications using the same version of the framework to share the same Flex libraries
    - Prevents users from needing to download the libraries for every Flex application they encounter
- Framework RSLs come in two versions: signed and unsigned
  - Signed framework RSLs
    - Have the extension SWZ
    - Only Adobe can create signed RSLs
    - Cached in a special Player cache and in the browser cache
    - Can be accessed by any application regardless of that application's originating domain
    - Only need to be downloaded to the client once
    - When browser's cache is cleared, signed framework RSLs persist in the special Player cache
  - Unsigned framework RSLs
    - Cached in the browser cache and can only be used by applications that have access to the RSL's domain
    - When the browser's cache is cleared, unsigned framework RSLs are removed just like any other file in the cache
  - By default, all Flex applications use the signed framework RSLs
  - The framework RSLs are located in the {FlexBuilderInstallDir}/sdks/4.0.0/frameworks/rsls
- Local debug RSL are created in the bin-debug directory as SWF files

The last files that we need to discuss in the bin-debug directory are these SWF files that have the specific Flex framework version numbers appended to them.

These files are related to the Flex framework's implementation of runtime shared libraries or RSLs.

Consider the fact that most Flex applications will use the core Flex framework classes and perhaps some of the other libraries.

Every time a user accesses a Flex application on the web, they will have to download all of these libraries, which is a waste of bandwidth.

If a user has already accessed one Flex application, then they ideally would not need to download all the Flex framework elements again.

Runtime shared libraries help accomplish this goal.

An in-depth discussion of RSL is outside the scope of this course, but take a look at the dialog when I select Project > Properties > Flex Build Path.

By default, the Framework linkage dropdown is set to Use SDK default (runtime share library).

In other words, by default, this Flex application is set to use the RSLs on the end-user's computer if they already have them.

If the user doesn't already have them, the Flex application will deliver these SWF files to their computers.

These files are, in other words, backup.

Project_clean.camrec

Lastly, I want to alert you to a couple of issues you may encounter during development.

If you swap out assets, or if for some reason Flash Builder doesn't properly re-compile an application, you can force the project to re-compile by selecting the Project > Clean menu.

You can clean all the projects or select specific ones to clean.

This will force the projects to be rebuilt and can sometimes clear up some unexpected behavior.

One last point before we move on to the next video.

You may occasionally encounter this error that specifies that the SWF cannot access a local resource.

If this happens, close all instances of your browser and then run your application again.

Browsers set their security when they start up and sometimes that session is not prepared to run your Flex application.

Simply restarting the session resets the browser security.

Next_step.camrec

Exercise: Setting up Flash Builder and your project files

For your next step, work through the exercise titled “Setting up Flash Builder and your project files”.