Using Xcode with Unsupported Languages

Author: Mark Szymczyk
Last Update: September 22, 2006

This article demonstrates how to use Xcode with languages it does not directly support by writing a simple Ruby program in Xcode.

Introduction

Xcode comes with built-in support for AppleScript, C, C++, Java, and Objective C. This built-in support comes in the form of project templates, file templates, compilers, and debuggers, which is everything you need to write programs in those languages.

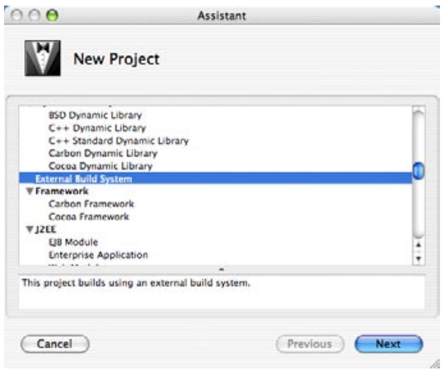
Obviously there are more than just the five computer languages Xcode has built-in support for. Mac OS X ships with Perl, PHP, Python, and Ruby. Languages like Eiffel, Fortran, Lisp, Lua, Pascal, and Smalltalk can be installed on Mac OS X. Can you use Xcode to write programs in these languages?

Yes, you can use Xcode to write and build programs written in unsupported languages, but you can't debug them or run them from Xcode. Writing programs in other languages with Xcode requires you to do two things. First, you must create an external build system project, which supports the building of programs written in other languages. Second, you must tell Xcode to use your language's compiler or interpreter to build your program.

In this article I am going to demonstrate Xcode's support for other languages by writing a simple Hello World program in Ruby. If you want to use a different language, you should not have much trouble following the material. Substitute your language's file extension, source code, and build tool for Ruby's.

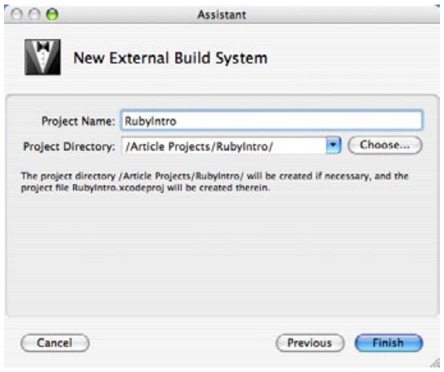
Creating an External Build System Project

The first step when writing programs in Xcode is to create a project. Choose File > New Project. The New Project Assistant window opens.



This project builds using an external build system.

The window contains the list of projects you can create. Select External Build System and click the Next button.



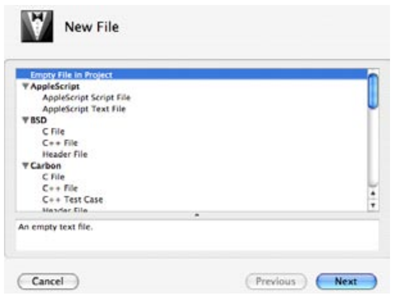
Project Name: RubyIntro
Project Directory: /Article Projects/RubyIntro/

The project directory /Article Projects/RubyIntro/ will be created if necessary, and the project file RubyIntro.xcodeproj will be created therein.

Name your project. I named the project RubyIntro. Click the Choose button to choose where to store the project. Click the Finish button to create the project.

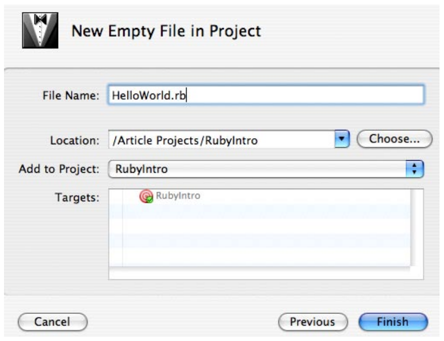
Adding a File to the Project

When you create an external build system project, you will notice the project contains no files. You need to add source code files to your project. The example in this article is very simple. It requires only one file. Choose File > New File to add a new file to the project. The New File Assistant window opens.



An empty text file.

The window contains the list of file types you can create. Select Empty File in Project and click the Next button.



File Name: HelloWorld.rb
Location: /Article Projects/RubyIntro
Add to Project: RubyIntro
Targets: RubyIntro

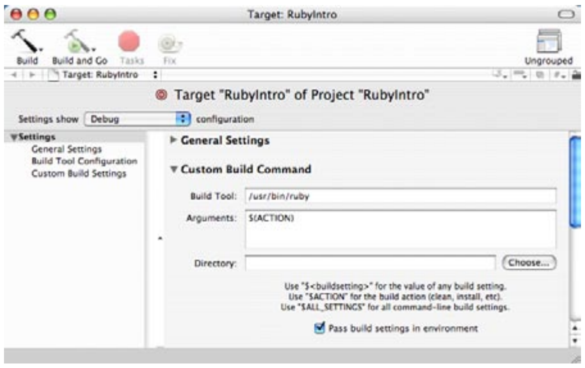
Name the file. I named the file `HelloWorld.rb`. The file should be set to be added to your project, but make sure the Add to Project menu shows the name of your project. Click the Finish button to create the file and add it to your project.

A blank editor window opens after adding the file. Add the following line of code and save the file:

```
puts “Hello World”
```

Telling Xcode to Use Your Build Tool

Before you can build the project, you must tell Xcode what program to use to build the project. Double-click the name of the target in the Groups and Files list to open the target’s settings window. You should see a section named Custom Build Command with three settings: Build Tool, Arguments, and Directory.



Target "RubyIntro" of Project "RubyIntro"

Settings show: Debug configuration

General Settings

Custom Build Command

Build Tool: /usr/bin/ruby

Arguments: $(ACTION)

Directory:

Use "$<buildsetting>" for the value of any build setting.
Use "$ACTION" for the build action (clean, install, etc).
Use "$ALL_SETTINGS" for all command-line build settings.

[✓] Pass build settings in environment

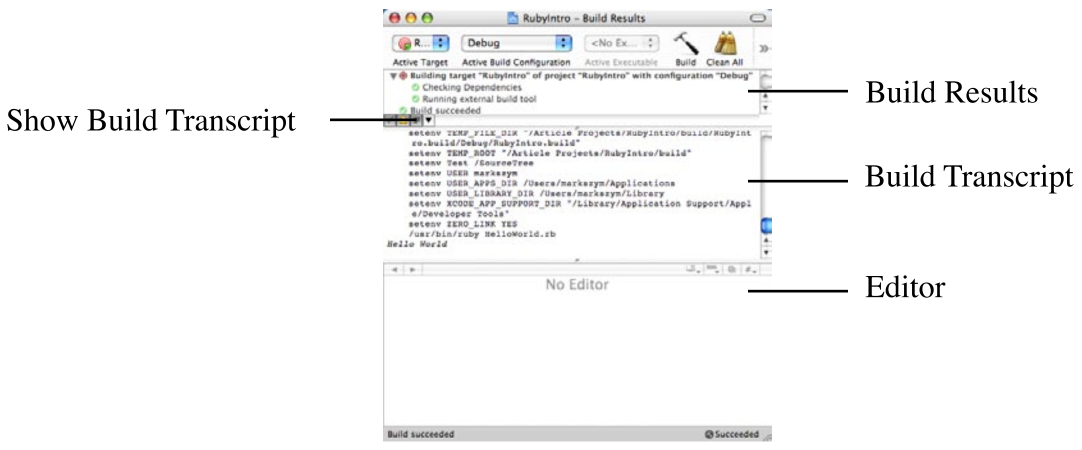
The default build tool is `make`. I want to use the Ruby interpreter to build the project so I have to enter the following path for the Build Tool setting:

```
/usr/bin/ruby
```

The Arguments setting is where you specify any flags or options you want to use to build the project. If you are following my example and don’t want to run the Terminal application to see the output of the program, add the name of your Ruby source code file to the Arguments setting. You shouldn’t have to modify the Directory setting.

Building the Project

Now it is time to build the project. Open the build results window by choosing Build > Build Results. The build results window has three areas. At the top is the build results area. In the center is the build transcript, and at the bottom is the editor. The build results area shows the high-level build steps. The build transcript shows the low-level build steps. The editor shows source code files when build errors occur. When building programs using unsupported languages, the build transcript is the most interesting area.



```
Building target "RubyIntro" of project "RubyIntro" with configuration "Debug"
    Checking Dependencies
    Running external build tool
Build succeeded
```

```
    setenv TEMP_FILE_DIR "/Article Projects/RubyIntro/build/RubyInt
     ro.build/Debug/RubyIntro.build"
    setenv TEMP_ROOT "/Article Projects/RubyIntro/build"
    setenv Test /SourceTree
    setenv USER markszym
    setenv USER_APPS_DIR /Users/markszym/Applications
    setenv USER_LIBRARY_DIR /Users/markszym/Library
    setenv XCODE_APP_SUPPORT_DIR "/Library/Application Support/Appl
     e/Developer Tools"
    setenv ZERO_LINK YES
    /usr/bin/ruby HelloWorld.rb
Hello World
```

Build succeeded — Succeeded

The first time you open the build results window, you will see only two areas. The build transcript is initially invisible. Click the Show Build Transcript button to show the build transcript.

Click the Build button in the build results window toolbar to build the project. You will see the steps Xcode takes to build the project in the build transcript. If you read the build transcript you will see that Xcode invoked the Ruby interpreter.

If you added `HelloWorld.rb` to the Arguments setting, you will see Hello World appear in the build transcript. If you didn't, you have to run the Terminal application to run the program. Navigate to the directory where your source code file is and run the following command:

```
ruby HelloWorld.rb
```

Conclusion

If you worked through the Ruby example, you noticed that Xcode's build capabilities are not too useful for interpreted languages like Ruby and Python. Interpreted languages require you to run the interpreter in order to run the program. There is nothing for Xcode to build when building a Ruby project. Xcode is normally used just to write programs in interpreted languages, not build them.

Xcode's build capabilities are more useful for compiled languages such as Pascal and Fortran. The compiler converts your source code into an executable file. Launch the executable file to run the program. Xcode can invoke the compiler and build executable files for compiled languages.