Automator

Your Personal Automation Assistant

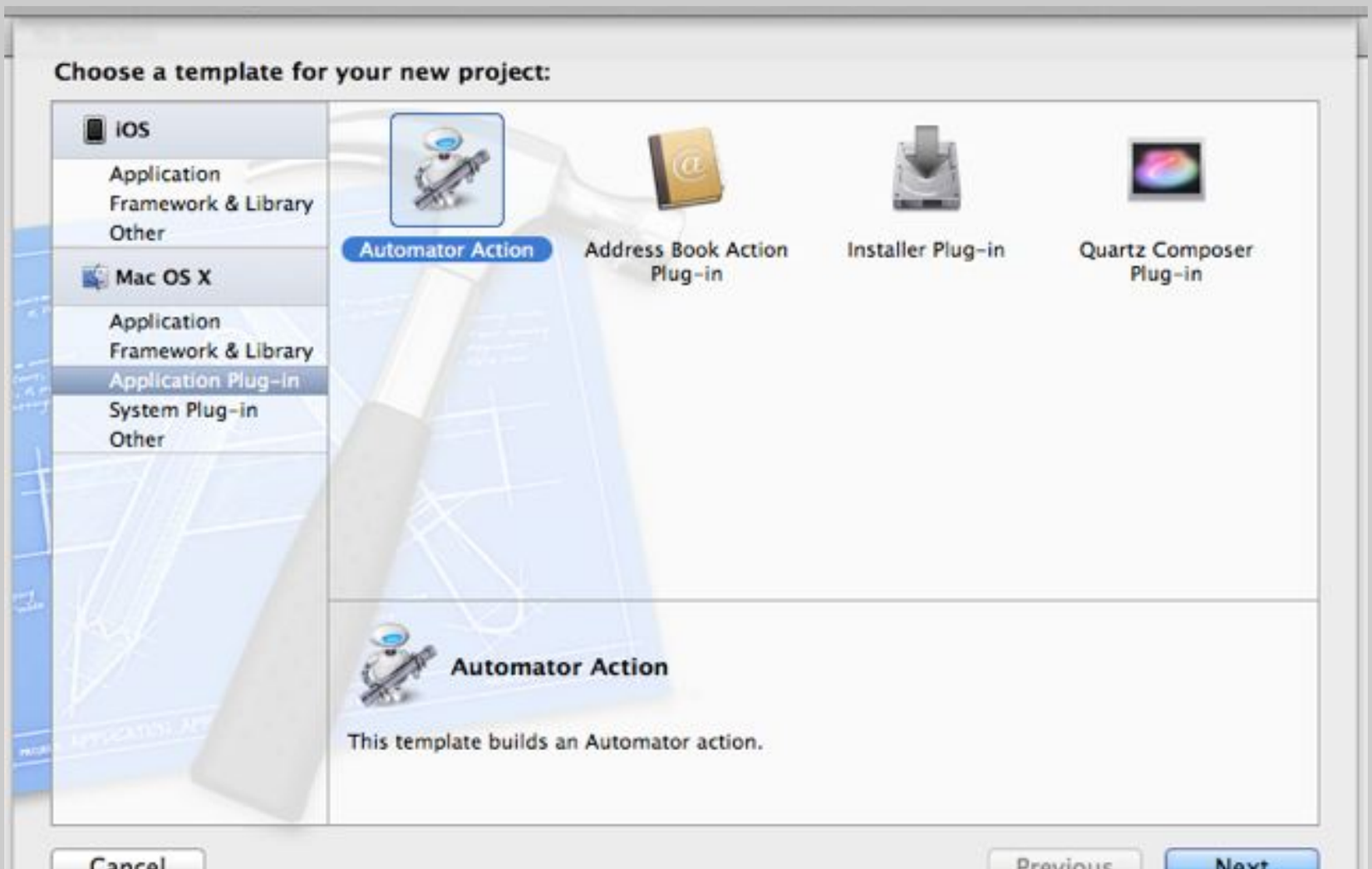
THIS WEBSITE IS NOT HOSTED BY APPLE, INC.

Create a Shell Script Automator Action

This tutorial describes how to create an Automator action using the standard Shell Script Automator Action Xcode template. The action will rotate the image files passed to it as input, using the rotation direction chosen by the user.

STEP 01 – New Xcode Project

DO THIS ▶ Open a new project in Xcode, and in the project template picker window, choose **Application Plug-in** as the category and **Automator Action** as the project template (⬇ SEE BELOW). Name the project: Rotate Image



PROJECT FILES

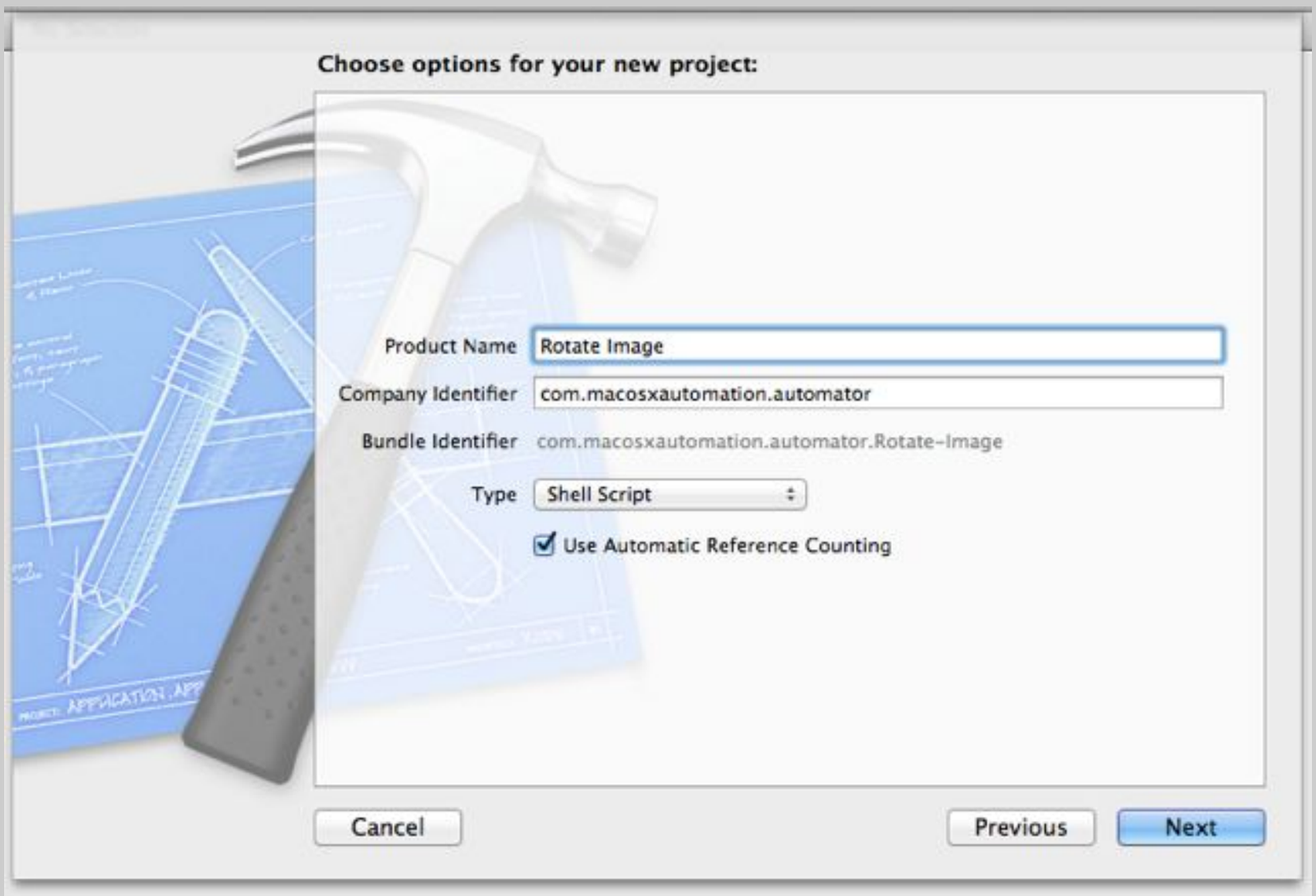
DOWNLOAD the completed Xcode project files.

XCODE ISSUE

Xcode has an issue with build settings for Automator action projects. Project build settings need to be adjusted in order for the completed project to launch Automator and be installed in its default action library for testing. Step-by-step instructions for making this simple adjustment are here.

DISCLAIMER

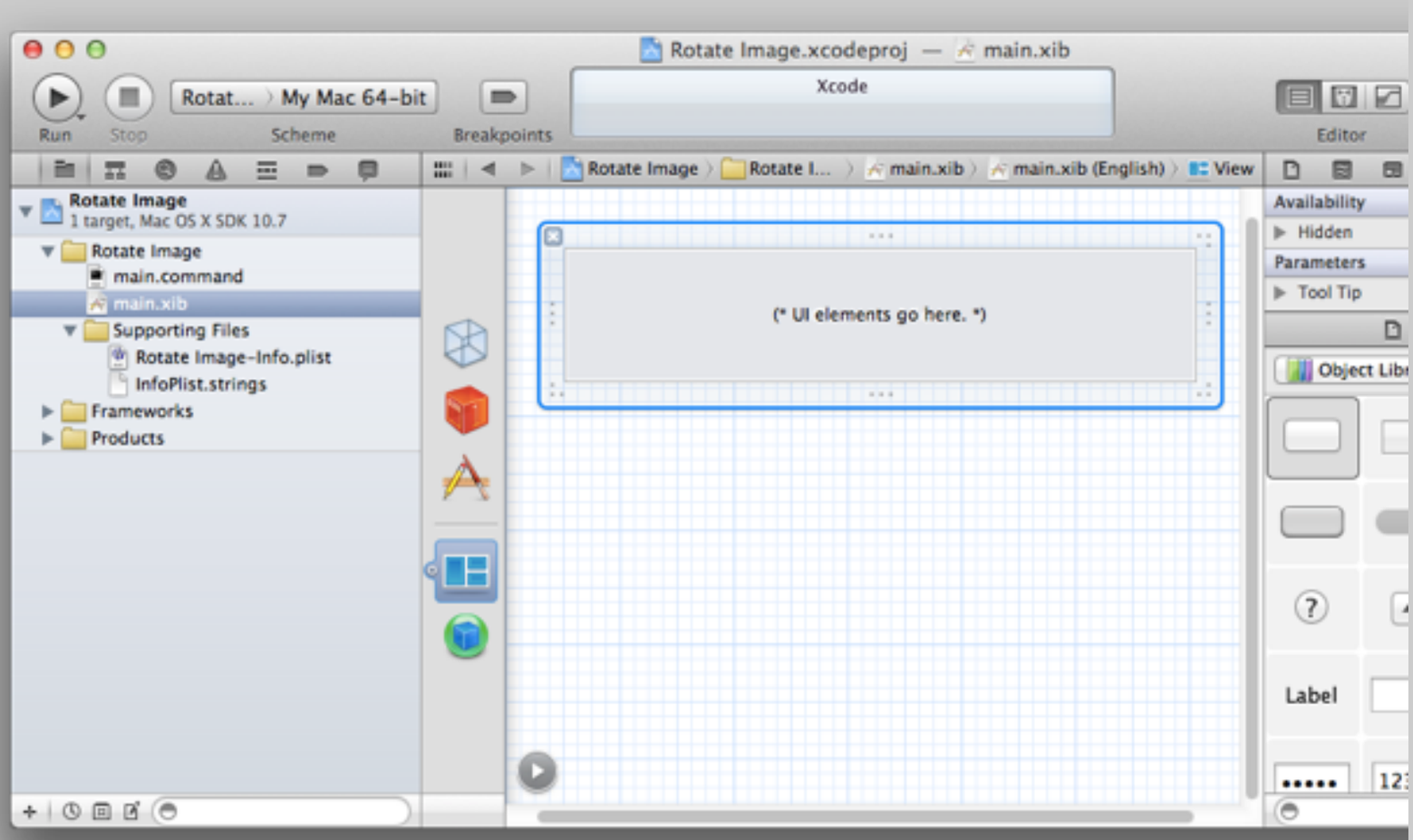
DO THIS ▶ In the forthcoming pane, name the action **Rotate Image** and set the type popup to be **Shell Script**. There are three types of Automator action projects: Cocoa, AppleScript-Objective-C, and Shell Script



STEP 02 – Edit the Action Interface

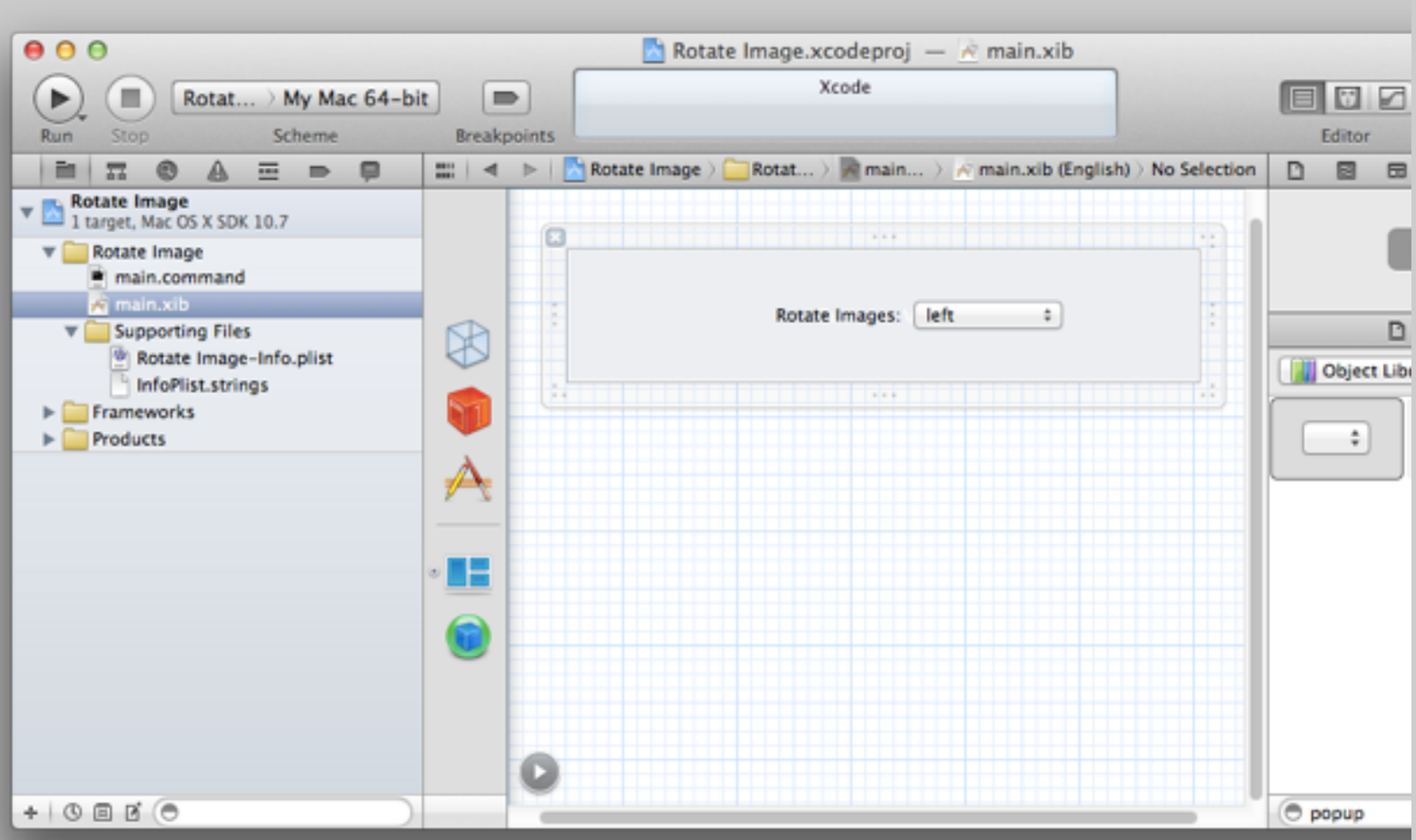
Every Automator Action project includes, by default, an interface nib that can be used to display parameter controls to the user during workflow creation or at the time the workflow is run.

DO THIS ▶ To edit the default view, click the main.nib resource icon in the Xcode project window to open the nib for editing. By default, the action view for the project will be displayed.

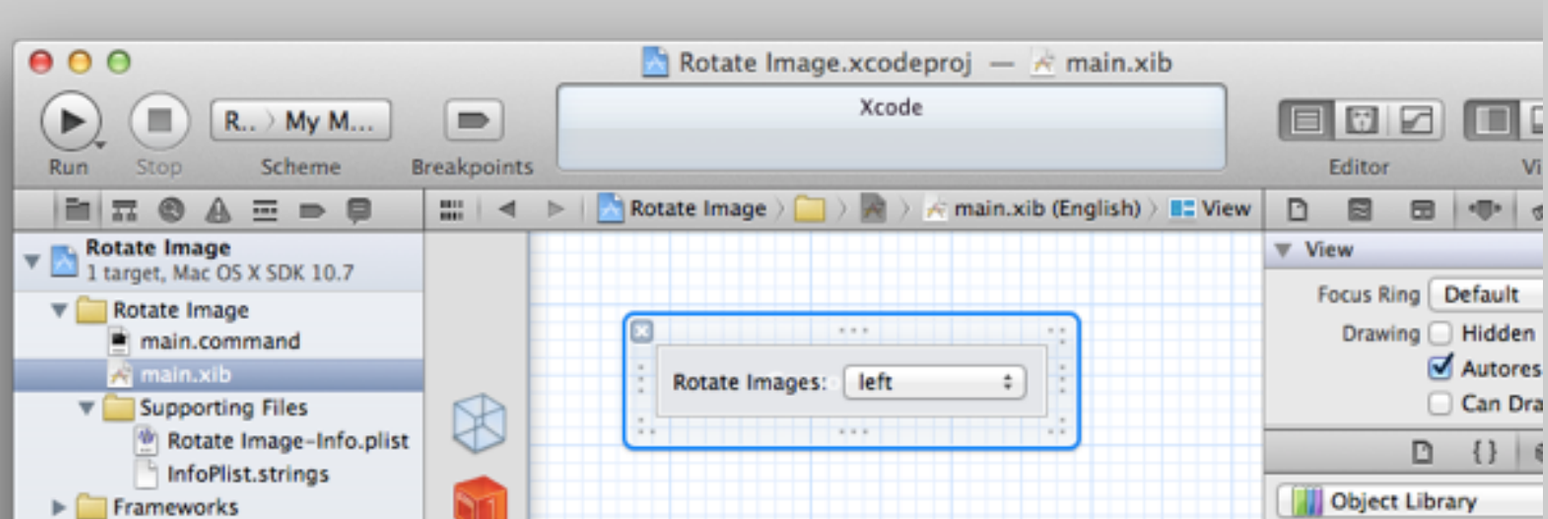


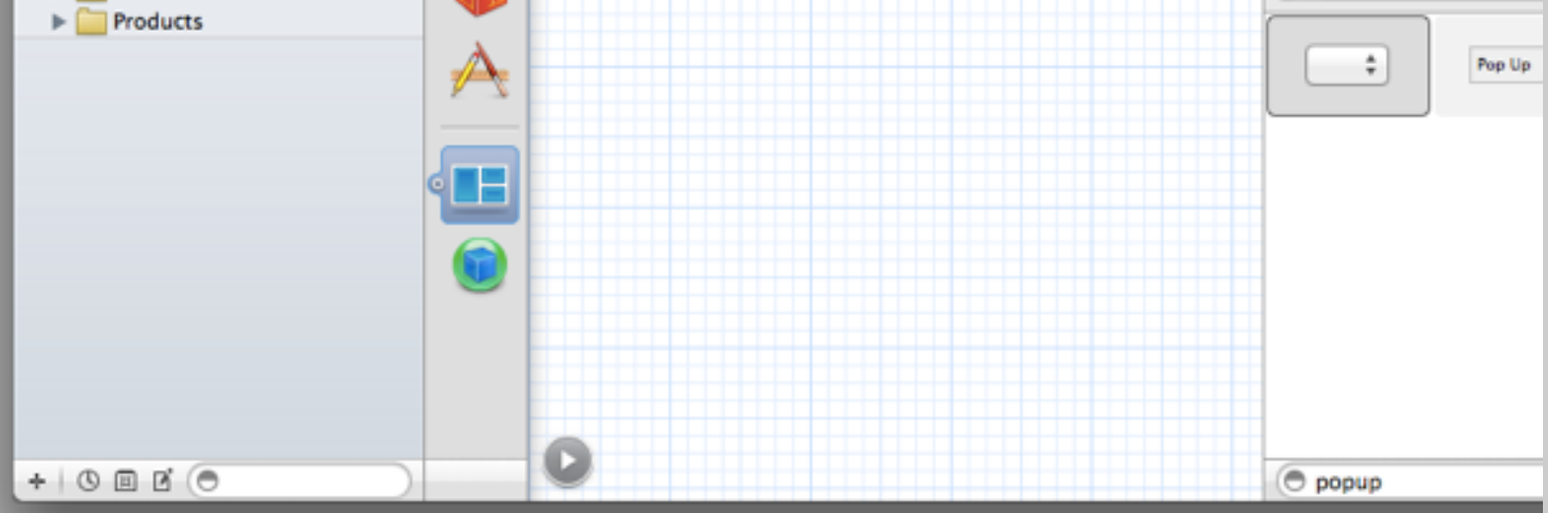
DO THIS ▶ Using the Library palette in Interface Builder (Tools > Library) find and drag a popup menu control to the view. In the Size Inspector pane of the Inspector palette, set the size parameter of the selected control to be small. Change the contents of the default

label to be "Rotate images:" and position in to the left of the control:



DO THIS ▶ Following the UI guidelines for Automator actions, position the label and control pair to the top left of the view, and resize the view window so that a 10-pixel margin is around all sides of the label/control pair:

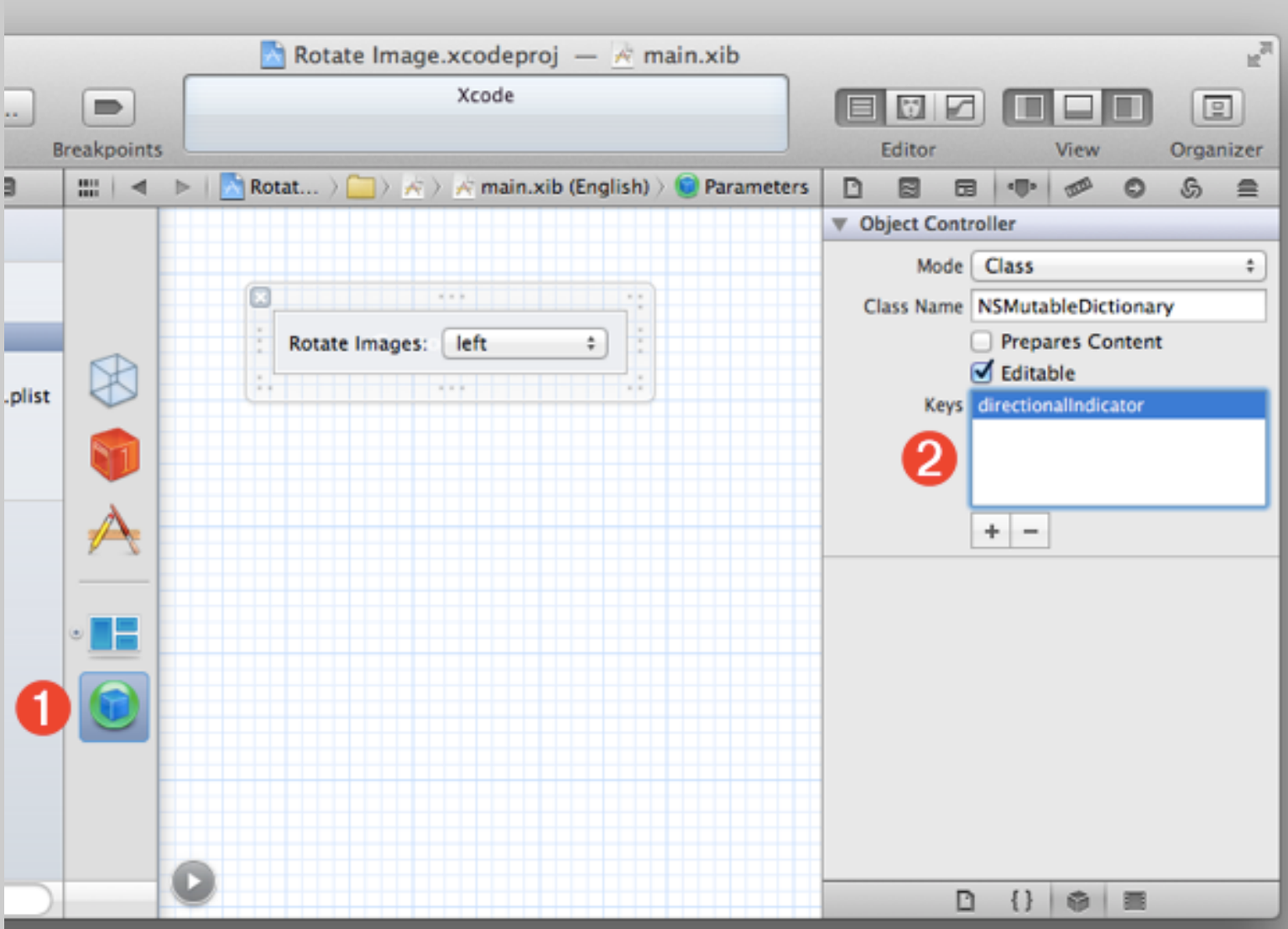




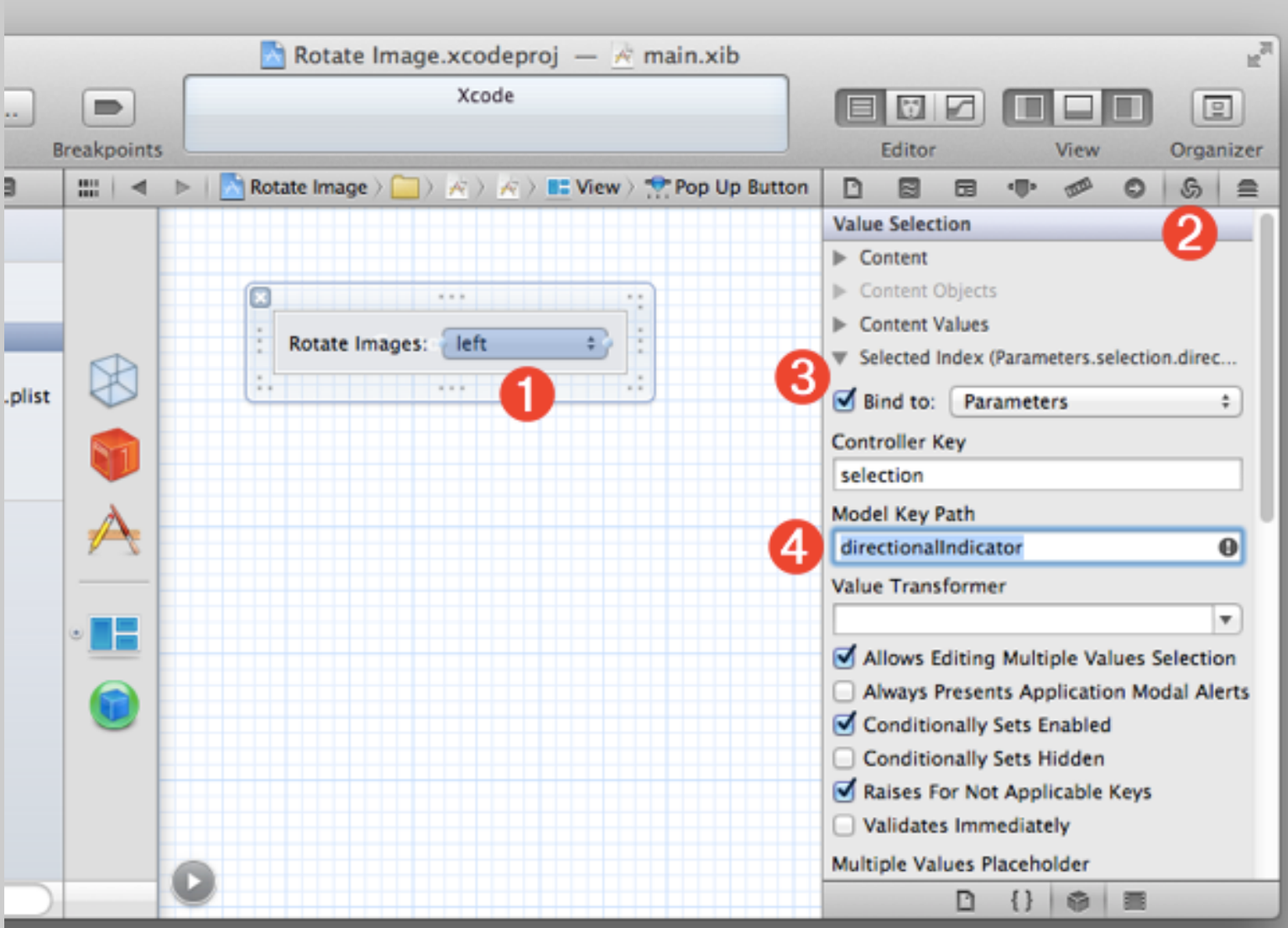
STEP 03 – Creating and Binding Parameters

When the workflow containing the action is executed, the current value of the control is passed to the Shell script in the main.command file. To accomplish this, you declare a parameter for the control and then attach that parameter to the control using Cocoa bindings.

DO THIS ▶ Select the Parameters object 1 in the project window. In the Attributes Inspector panel, click the plus-sign button 2 to add a new parameter to the parameters list. Name the parameter: directionalIndicator



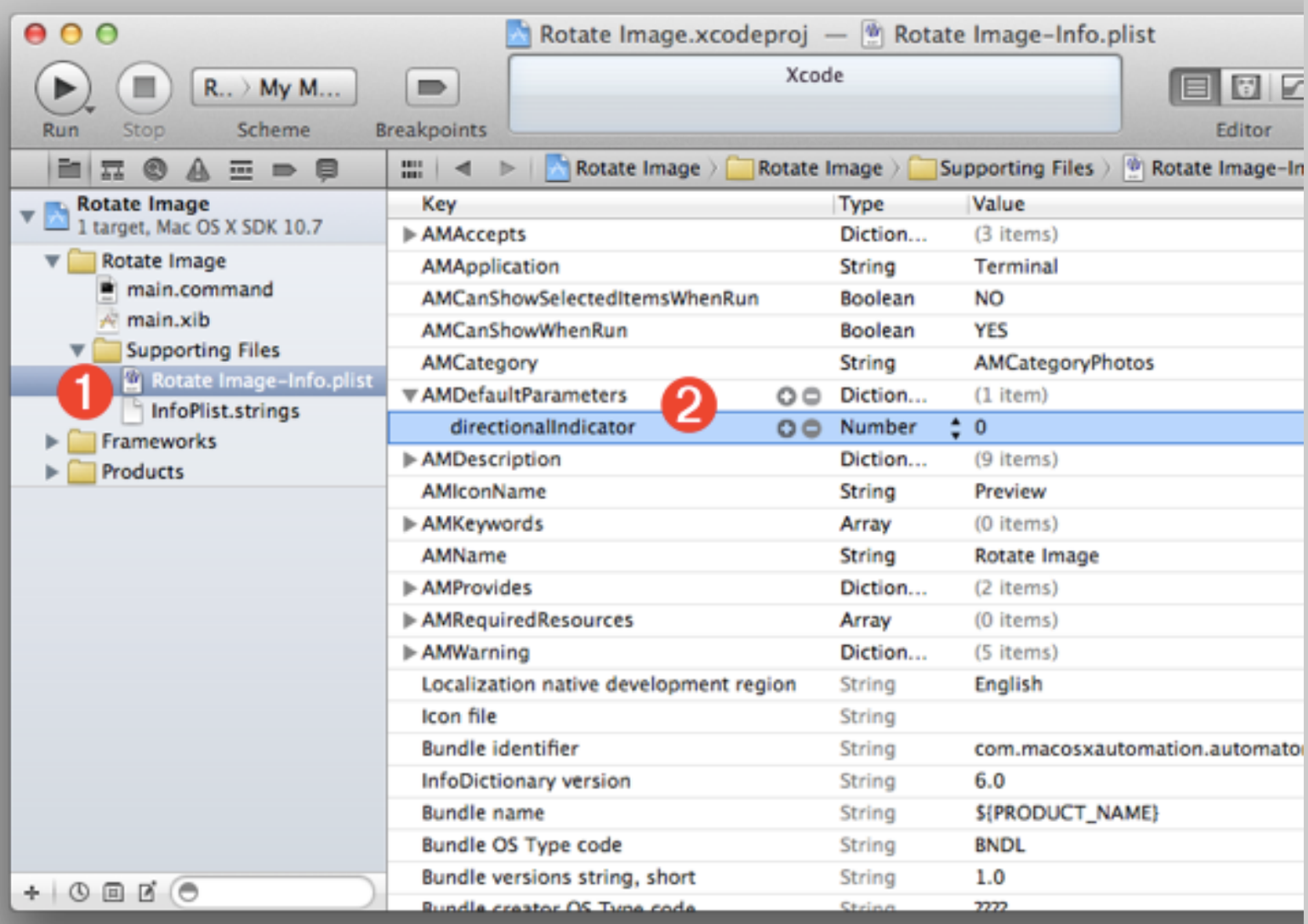
DO THIS ▶ Next, select the popup menu control 1 in the action view, and select the Bindings Inspector panel 2 of the Inspector palette. Bind the Selected Index value 3 to the Parameters object using the Model Key Path 4 that you just created: directionalIndicator.



When the workflow containing the action is run, the index of the current control menu item will be passed as an integer value to the Shell code in the main.command file.

STEP 04 – Declare the Parameters

DO THIS ▶ In Xcode, select the info.plist file, in the Supporting Files folder 1 on the left of the project window, to display its contents for editing. Add a child property list item to the AMDefaultParameters entry 2, and name the new list item the name of the parameter you created in Interface Builder: directionalIndicator. Assign the new list item a numeric value of: 0



NOTE: other important defaults are also declared in the info.plist file, including:

- the **application** used by the action (AMApplication),
- the **category** in which the action will be displayed in the Automator library (AMCategory),
- the default **icon** to display with the action in Automator (AMIconName),
- the action **description** strings (AMDescription),
- the **input types** (AMAccepts - com.apple.cocoa.path),
- the **output types** (AMProvides - com.apple.cocoa.string),

as well as the standard developer copyright strings and bundle identifier.

In addition, you'll want to edit the InfoPlist.strings file in the Resources group in the Groups and Files list to display appropriate description and instructional information about the action for the user.

These edits are not covered in this tutorial, but are referenced in detail in the Automator Programing Guide.

STEP 05 – Edit the Run Code

This example Shell Script action is designed to process image files passed to it in a workflow, so its input will be an array of POSIX paths to the passed image files. Input comes from stdin, and passed parameter values are stored in environment variables.

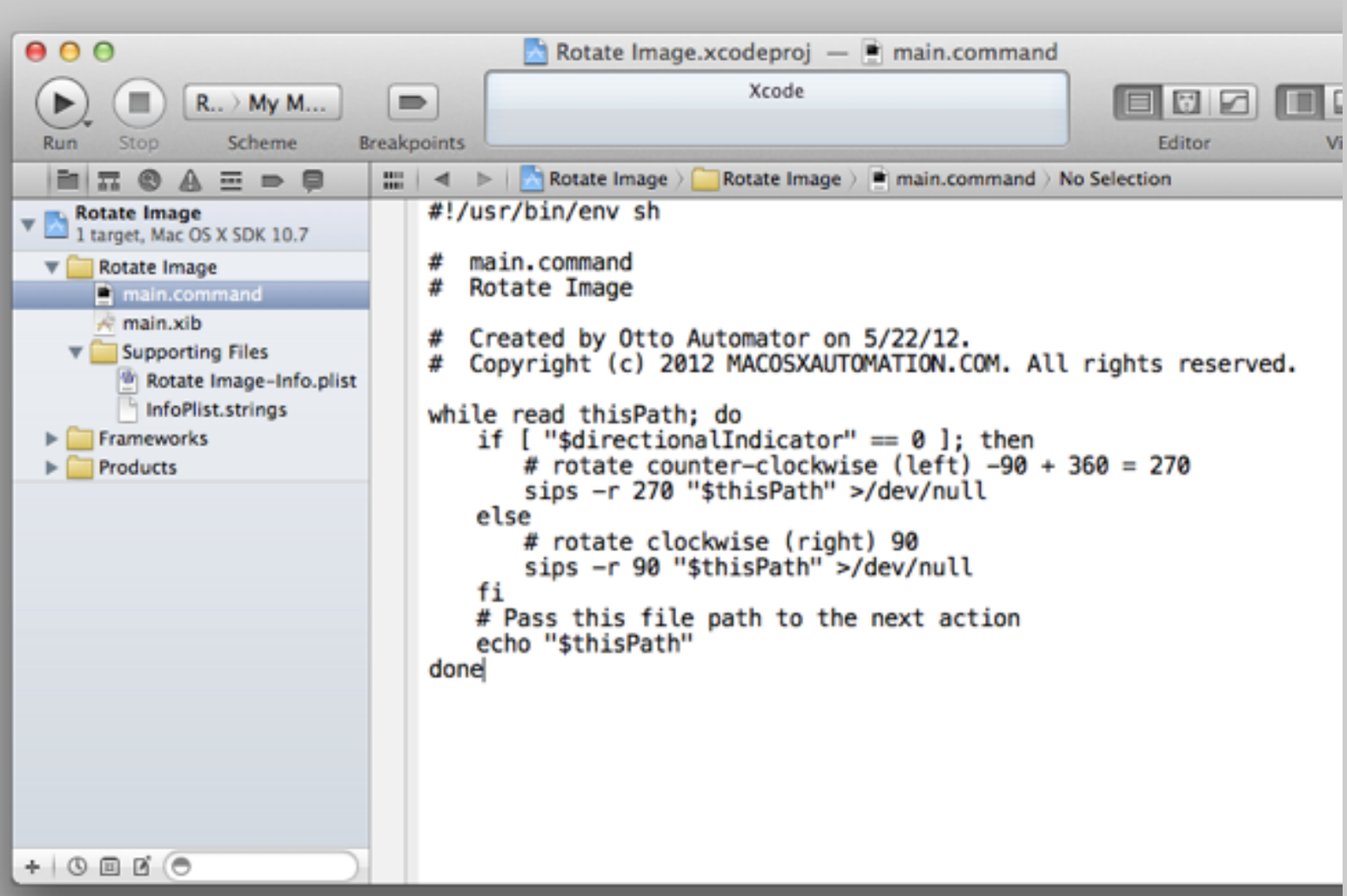
DO THIS ▶ Select the main.command file in the in the Groups and Files column on the left of the project window, to display its contents for editing. Replace the default cat command with the following script:

```
while read thisPath; do
	if [ "$directionalIndicator" == 0 ]; then
		# rotate counter-clockwise (left) -90 + 360 = 270
		sips -r 270 "$thisPath" >/dev/null
	else
		# rotate clockwise (right) 90
		sips -r 90 "$thisPath" >/dev/null
	fi
	# Pass this file path to the next action
	echo "$thisPath"
done
```

NOTE: the selected index value of the parameter you created for the rotation

direction control is passed into the script in the environment variable: $directionalIndicator.

Also note, it is important to pass the results of your action to any following action. This is accomplished with the echo command towards the end of the script.



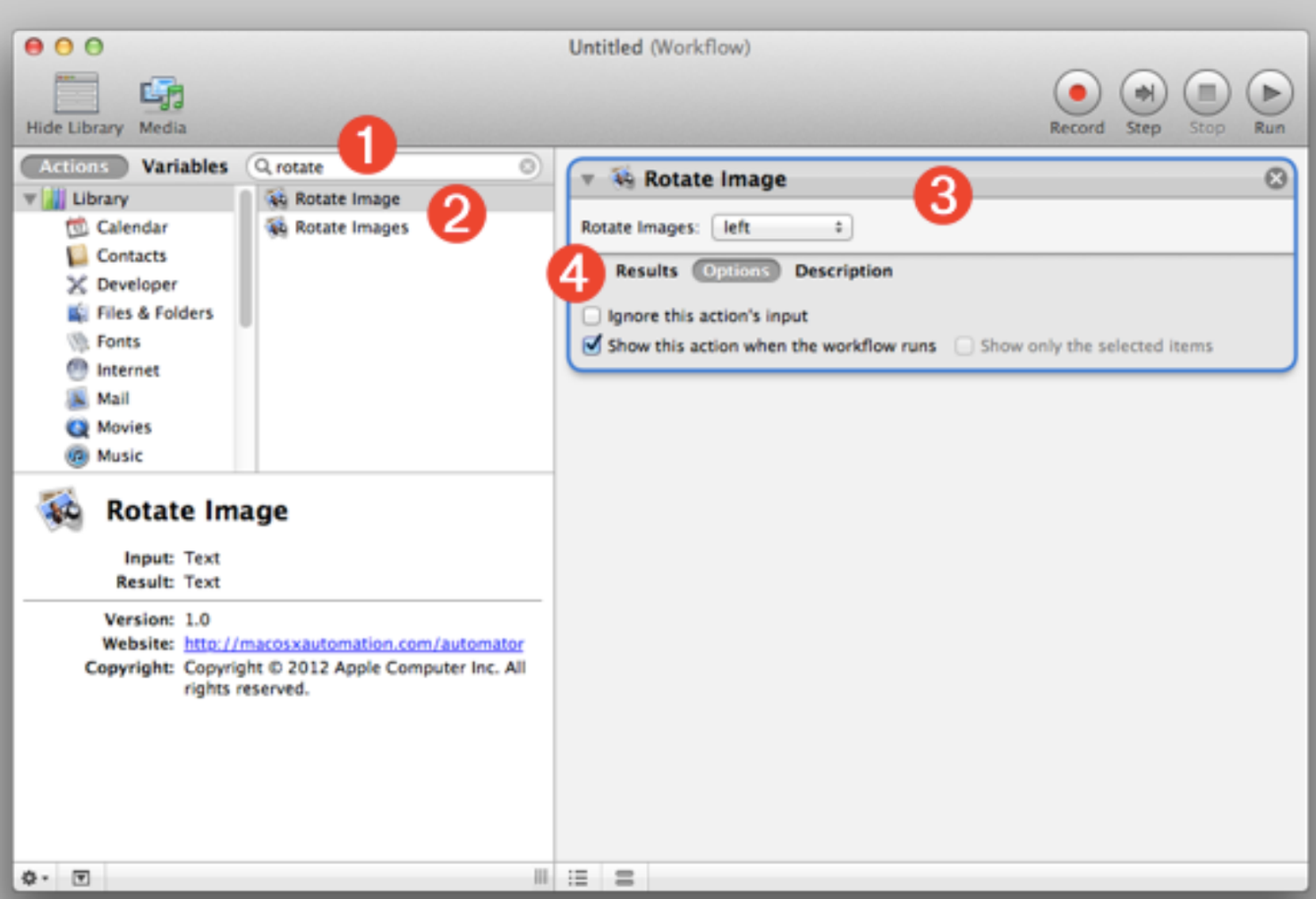
STEP 06 – Test the Project

IMPORTANT: Xcode has an issue with build settings for Automator action projects. Project build settings need to be adjusted in order for the completed project to launch Automator and be installed in its default action library for testing. Step-by-step instructions for making this simple adjustment are here.

Now you're ready to test your new action. Build and run the Xcode project. Automator will launch with your action automatically installed as part of its default library of actions. In the template sheet in the new Automator workflow window, choose the option for creating a basic workflow document.

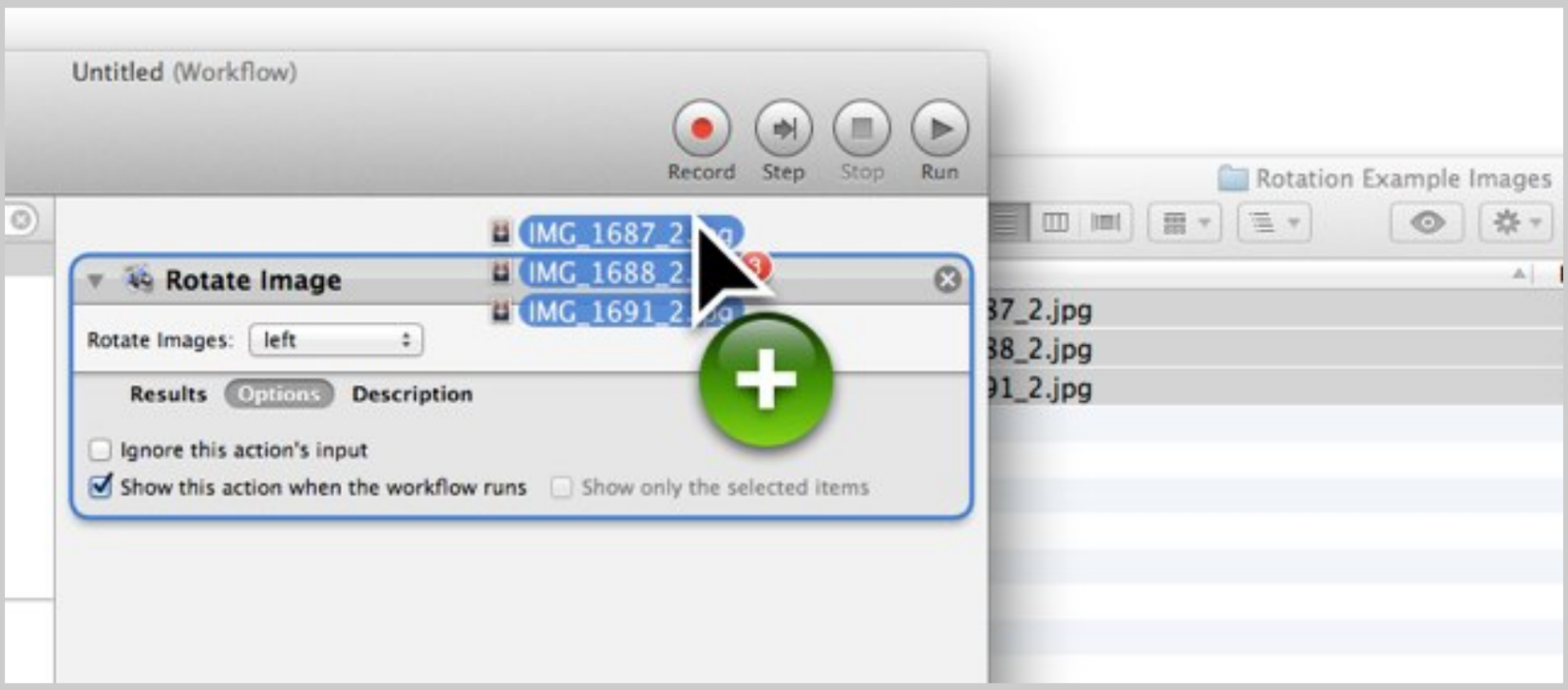
DO THIS ▶ Locate your action by entering the term "rotate" in the library search field 1. All actions related to rotating will appear below

search field 1. All actions related to rotating will appear below the search term. Select and drag your action from the list 2 to the workflow area to the right and release to add the action to the workflow 3.

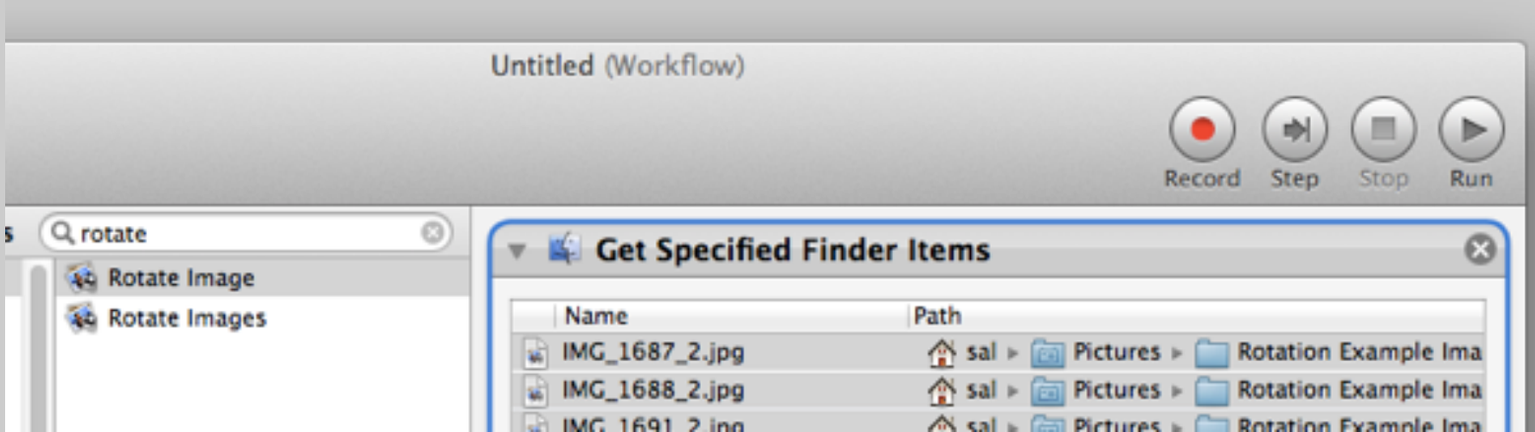


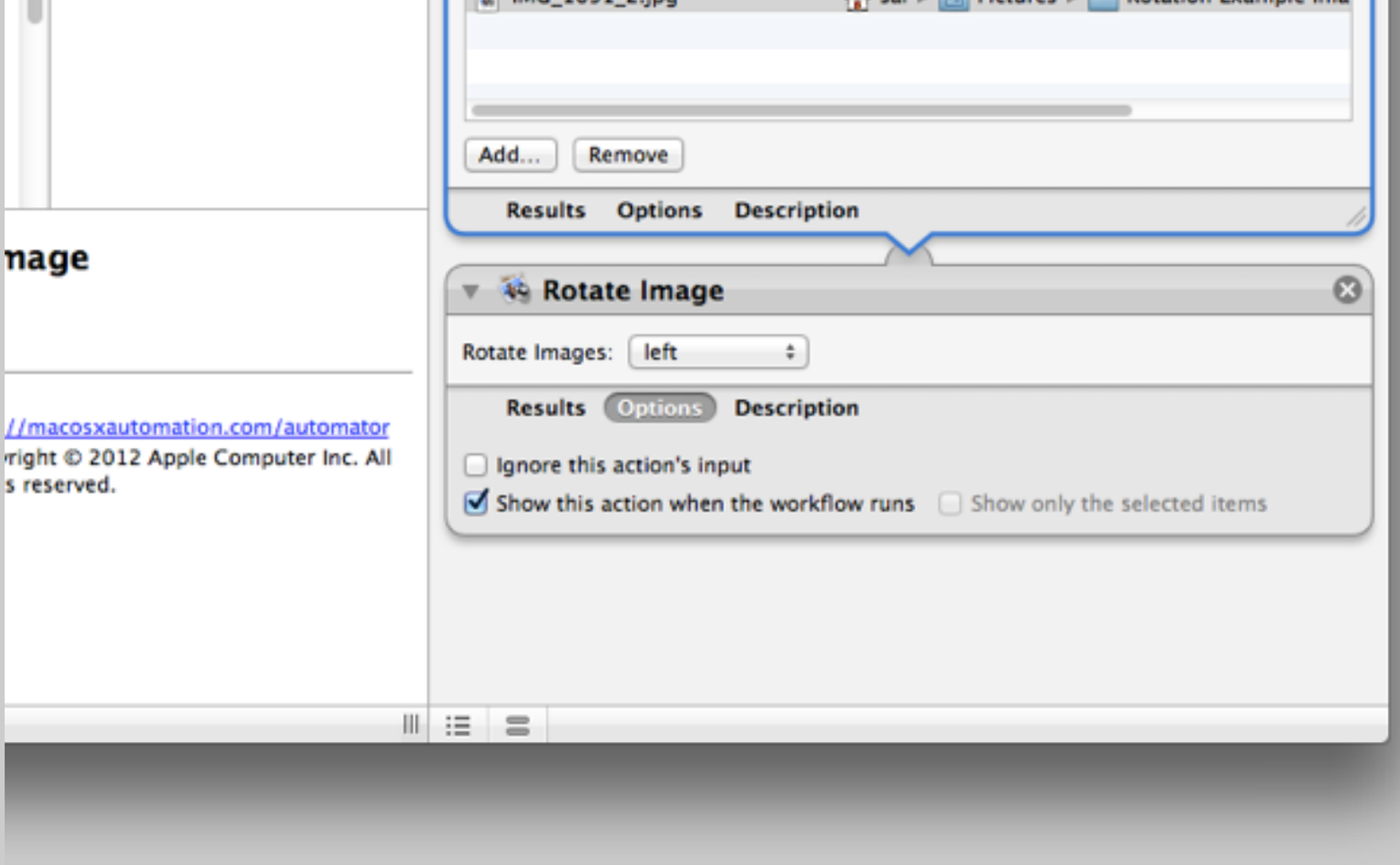
DO THIS ▶ Click the Options button at the bottom of the action view 4 and select the checkbox titled: Show this action when the workflow is run

DO THIS ▶ Next, drag some images to rotate from the Finder to just before the action view. Hover the mouse just before the action view and the view will move allowing you to drop the images into the workflow.

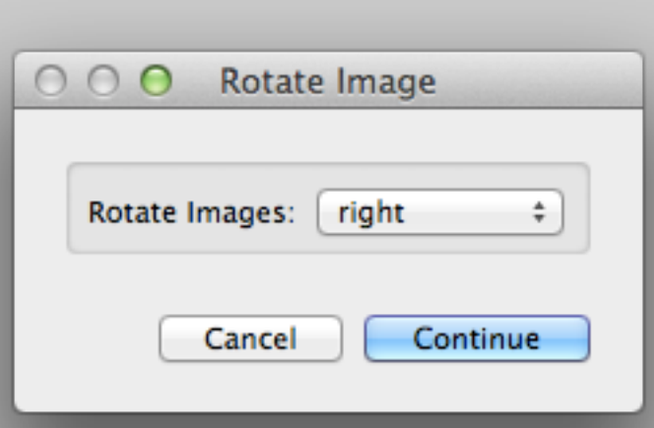


When you release the mouse, a new Get Specified Finder Items action listing the dragged images, will be added to the start of the workflow. These images will become the input to the Rotate Image action:



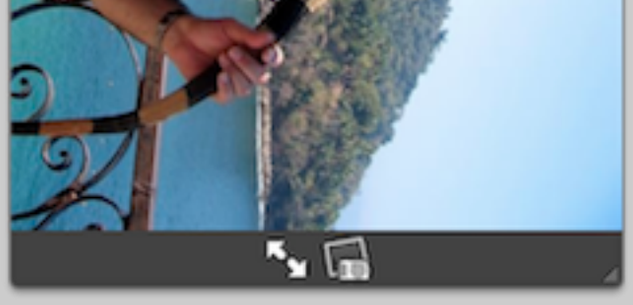


DO THIS ▶ Click the **Run** button in the Automator window and the workflow will execute. The paths to the image files will be passed to the Rotate Image action, which will display its action view for you to choose a rotation direction:



DO THIS ▶ Choose a rotation direction from the popup menu in the action view and click the **Continue** button to process the passed images.

Check you images and you'll find them rotated!



What's Next?

Now that you've discovered how easy it is to create interfaces for your favorite shell scripts, you can create actions to help you or your staff be more productive.

TIP: To learn how to create Mac OS X contextual services with your actions, visit the Services section of this website.

TOP