

2

Write Your First Hello World! Application

WHAT YOU WILL LEARN IN THIS CHAPTER

- How to create a new iPhone project
- Building your first iPhone application using Xcode
- Designing the user interface (UI) of your iPhone application with Interface Builder
- How to add an icon to your iPhone application

Now that you have installed the SDK, you are ready to start developing for the iPhone! Programming books customarily start by demonstrating how to develop a “Hello World!” application. This approach enables you to use the various tools quickly without getting bogged down with the details. It also provides you with instant gratification: You see for yourself that things really work, which can be a morale booster that inspires you to learn more.

GETTING STARTED WITH XCODE

Power up Xcode and you should see the Welcome screen, shown in Figure 2-1.



NOTE The easiest way to start Xcode is to type **Xcode** in Spotlight and then press the Enter key to launch it.



FIGURE 2-1

To create a new iPhone project, choose **File** ⇨ **New Project**. Figure 2-2 shows the different types of projects you can create using Xcode. The left panel shows the two primary categories — iPhone OS and Mac OS X. The iPhone uses the iPhone OS (which has since been renamed as iOS), so click the Application item listed under iPhone OS to view the different templates available for developing your iPhone application.

Although there are quite a few types of iPhone applications you can create, for this chapter select the View-based Application template; and for the product, select iPhone (to target the iPad, select the iPad as the product). Click the Choose... button.



NOTE Subsequent chapters show you how to develop some of the other types of iPhone applications, such as Tab Bar applications and Split View-based applications.

Name the project `HelloWorld` and click **Save**. Xcode proceeds to create the project for the template you have selected. Figure 2-3 shows the various files and folders automatically created for your project.

The left panel of Xcode shows the groups in the project. You can expand each group or folder to reveal the files contained in it. The right panel of Xcode shows the files contained within the group or folder you have selected from the left panel. To edit a particular file, select it from the list, and the editor at the bottom of the right panel opens the file for editing. If you want a separate window for editing, simply double-click the file to edit it in a new window.

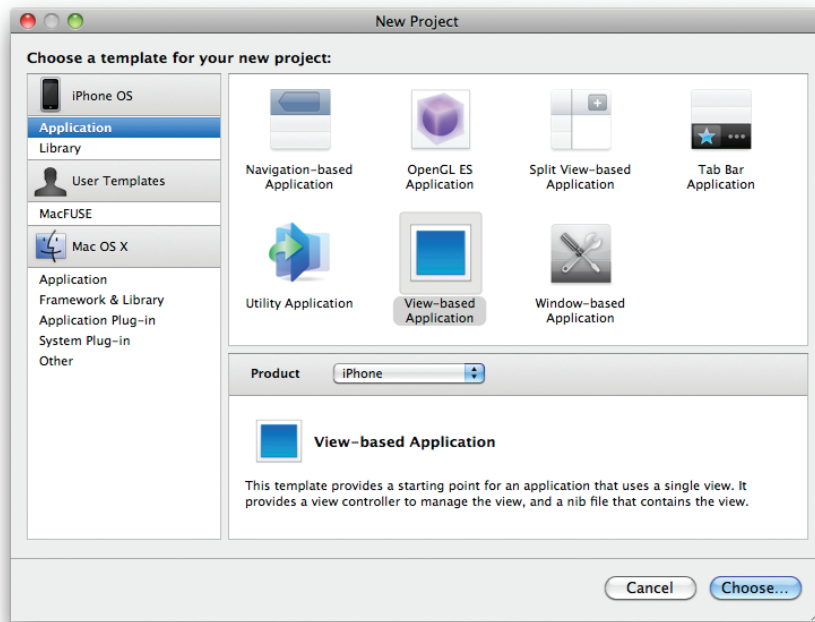


FIGURE 2-2

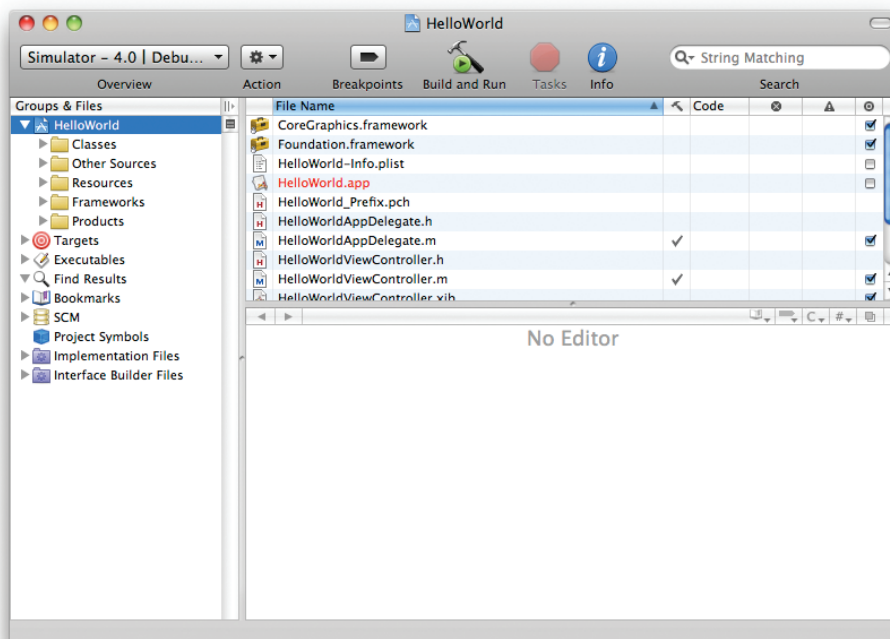


FIGURE 2-3

Using Interface Builder

At this point, the project has no UI. To prove this, simply press Command-R (or select Run ⇨ Run), and your application is deployed to the included iPhone 4 Simulator. Figure 2-4 shows the blank screen displayed on the iPhone Simulator. It's good to see this now, because as you go through the chapter you will see changes occur based on your actions.



FIGURE 2-4



NOTE If you are not seeing the Simulator as shown in Figure 2-4, that's because it is still simulating the older iPhone. To change to the iPhone 4 Simulator, select Hardware ⇨ Device ⇨ iPhone 4.

Obviously, a blank screen is not very useful. Therefore, it's time to try adding some views to your application's UI. In the list of files in your project, you'll notice two files with the `.xib` extension — `MainWindow.xib` and `HelloWorldViewController.xib`. Files with `.xib` extensions are basically XML files containing the UI definitions of an application. You can edit `.xib` files by either modifying their XML content or, more easily (and more sanely), editing them using Interface Builder.

Interface Builder, included as part of the iPhone SDK, enables you to build the UI of iPhone (and Mac) applications by using drag and drop.

Double-click the `HelloWorldViewController.xib` file to launch Interface Builder. Figure 2-5 shows Interface Builder displaying the content of `HelloWorldViewController.xib`, which contains three items: File's Owner, First Responder, and View. As you can see, the Library window shows the various views that you can add to the UI of your iPhone application. The View window shows the graphical layout of your UI. You will see the use of the other windows shortly.



NOTE Refer to Appendix C for a crash course on Interface Builder if you are not familiar with it.

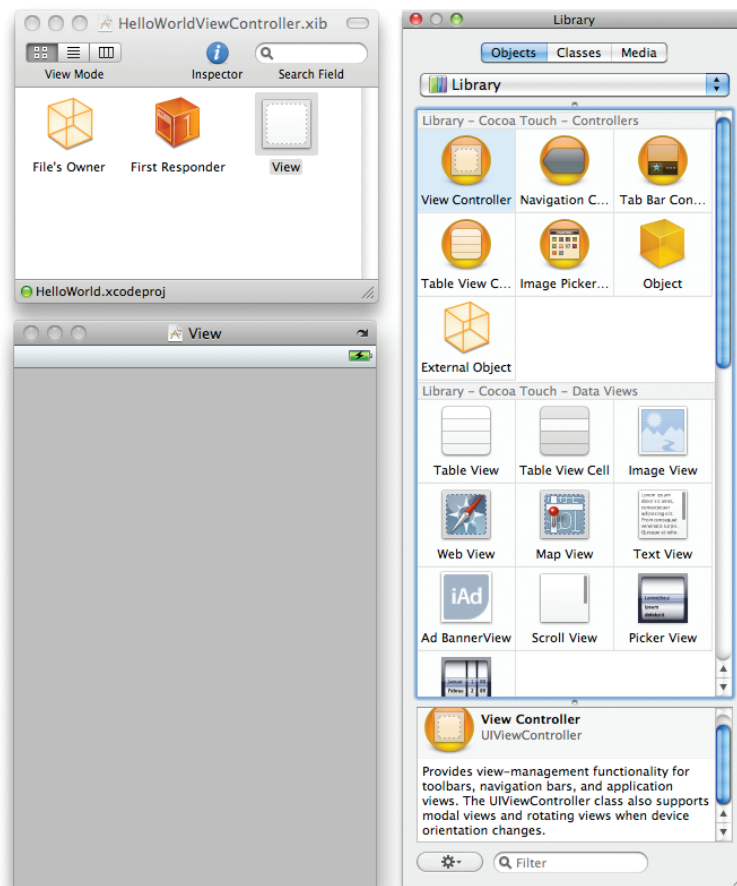


FIGURE 2-5

Scroll down to the Label view in the Library window and drag and drop a Label view onto the View window (see Figure 2-6).

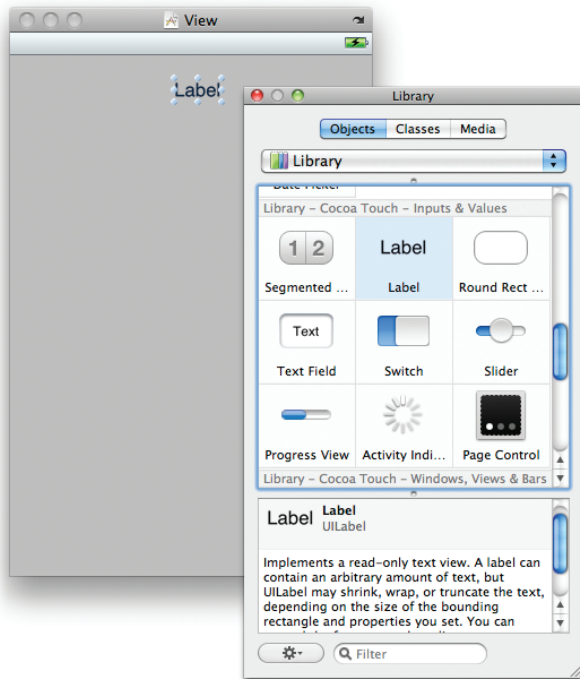


FIGURE 2-6

After the Label view is added, select it and choose Tools ⇨ Attributes Inspector. Enter **Hello World!** in the Text field (see Figure 2-7). Then, next to Layout, click the center Alignment type.

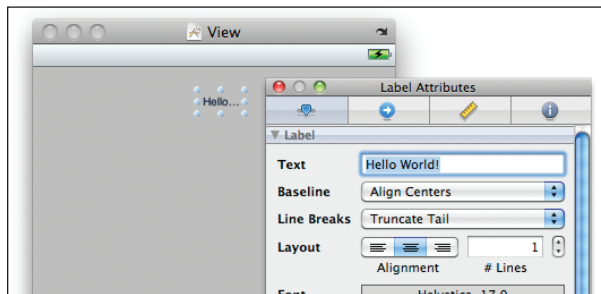
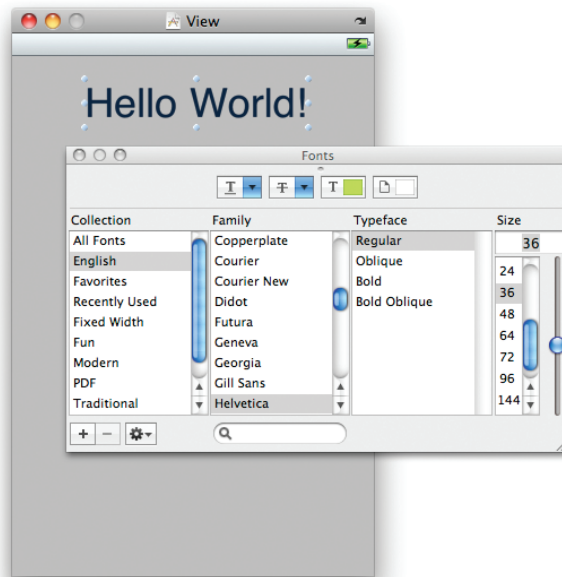
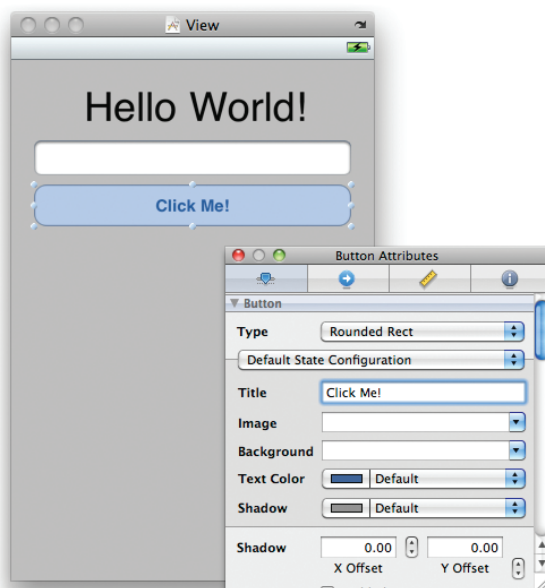


FIGURE 2-7

With the Label view still selected, press Command-T to invoke the Fonts window (see Figure 2-8). Set the font size to 36.

**FIGURE 2-8**

Next, from the Library window, drag and drop a Text Field view to the View window, followed by a Round Rect Button view. Modify the attribute of the Round Rect Button view by entering **Click Me!** in the Title field (see Figure 2-9).

**FIGURE 2-9**



NOTE Rather than specify the `Text` or `Title` property of a view to make the text display in the view (for example, the `Label` and the `Round Rect Button` views), you can simply double-click the view itself and type the text directly. After you've done this, you can rearrange the views and resize them to suit your needs. Interface Builder provides you with alignment guidelines to help you arrange your controls in a visually pleasing layout.

Save the `HelloWorldViewController.xib` file by pressing Command-S. Then, return to Xcode and run the application again by pressing Command-R. The iPhone 4 Simulator now displays the modified UI (see Figure 2-10).



FIGURE 2-10



NOTE Always remember to save all your changes in Interface Builder before you run the application in Xcode.

Click the Text Field view and watch the keyboard automatically appear (see Figure 2-11).

Click the Home button on the iPhone 4 Simulator, and you will see that your application has been installed on the Simulator. To go back to the application, simply click the HelloWorld icon (see Figure 2-12).



FIGURE 2-11

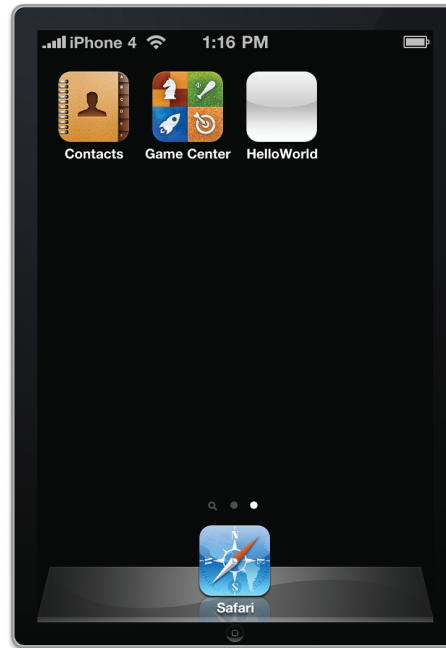


FIGURE 2-12



NOTE By default, starting with iOS 4, all applications built using the iPhone 4.0 SDK support multi-tasking. Hence, when you press the Home button on your iPhone, your application is not terminated; it is sent to the background and suspended. Tapping an application icon resumes the application. Chapter 21 contains more details about background execution of your iPhone applications.

Writing Some Code

By now you should be comfortable enough with Xcode and Interface Builder to write some code. This section will give you a taste of programming the iPhone.

In the `HelloWorldViewController.h` file, add a declaration for the `btnClicked:` action:

```
#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController {

}

- (IBAction) btnClicked:(id) sender;

@end
```

The bold statement creates an action (commonly known as an *event handler*) named `btnClicked:`. With the action declared, save the file and return to Interface Builder.

Earlier in this chapter, you saw a window labeled `HelloWorldViewController.xib`. Within this window are three components: File's Owner, First Responder, and View. Control-click the Round Rect Button view in the View window and drag it to the File's Owner item in the `HelloWorldViewController.xib` window (see Figure 2-13). A small popup containing the `btnClicked:` action appears. Select the `btnClicked:` action. Basically, what you are doing here is linking the Round Rect Button view with the action (`btnClicked:`) so that when the user clicks the button, the action is invoked.

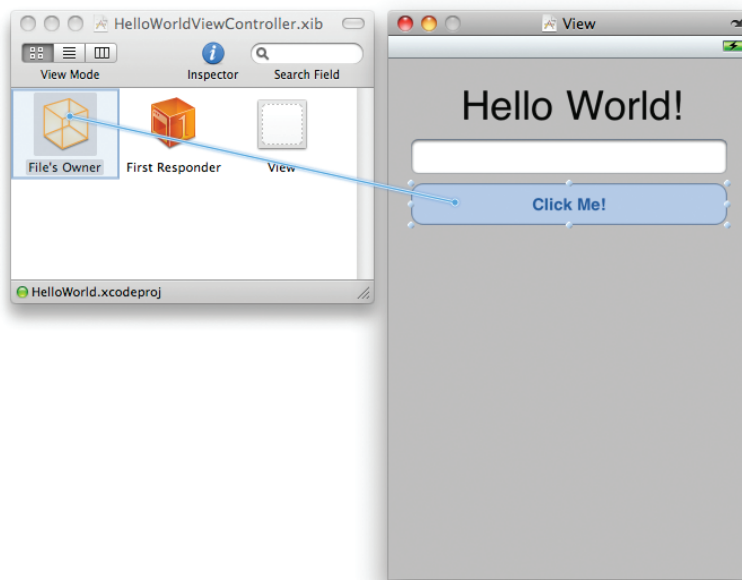


FIGURE 2-13

In the `HelloWorldViewController.m` file, add the code that provides the implementation for the `btnClicked: action`:

```
#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

-(IBAction) btnClicked:(id) sender {
    //--display an alert view--
    UIAlertView *alert =
        [[UIAlertView alloc] initWithTitle:@"Hello World!"
                                         message:@"iPhone, here I come!"
                                         delegate:self
                                         cancelButtonTitle:@"OK"
                                         otherButtonTitles:nil];

    [alert show];
    [alert release];
}
```

The preceding code displays an alert containing the sentence “iPhone, here I come!”

That’s it! Go back to Xcode and run the application again. This time, when you click the Button view, an Alert view displays (see Figure 2-14).



FIGURE 2-14

CUSTOMIZING YOUR APPLICATION ICON

As you saw earlier, the application installed on your iPhone Simulator uses a default white image as an icon. It is possible, however, to customize this icon. When designing icons for your iPhone and iPad applications, bear the following in mind:

- Design your icon to be 57×57 pixels (for iPhone), 114×114 pixels (for iPhone high-resolution), or 72×72 pixels (for iPad). Larger size is acceptable because the iOS automatically sizes it for you. For distribution through the App Store, you also need to prepare a 512×512 pixel image.
- Use square corners for your icon image because iPhone automatically rounds them. It also adds a glossy surface (you can turn off this feature, though).



NOTE Apple has published a technical Q&A on the various icon files that you can use in your iPhone application. This document is located at <http://developer.apple.com/iphone/library/qa/qa2010/qa1686.html>.

HOW TO TURN OFF THE GLOSSY SURFACE ON YOUR ICON

To turn off the glossy effect applied on your icon, you need to add the `UIPrerenderedIcon` key to the `HelloWorld-Info.plist` file in your Xcode project and then set it to `YES`. For more details on the various keys that you can set in your `HelloWorld-Info.plist` file, refer to Apple's documentation at: <http://developer.apple.com/iphone/library/documentation/General/Reference/InfoPlistKeyReference/Articles/iPhoneOSKeys.html>

The following Try It Out demonstrates how to add an icon to your application so that the iPhone will use it instead of the default white image.

TRY IT OUT Adding an Icon to the Application

1. To add an icon to your application, drag and drop an image onto the Resources folder of your project (see Figure 2-15). You will be asked if you want to make a copy of the image you are dropping. Check this option so that a copy of the image will be stored in your project folder.
2. Select the `HelloWorld-Info.plist` item (also located under the Resources folder; the `HelloWorld-Info.plist` file is commonly referred to as the `info.plist` file). Select the `Icon` file item and set its value to the name of the icon, `app-icon.png` (see Figure 2-16). This specifies the name of the image to be used as the application icon.

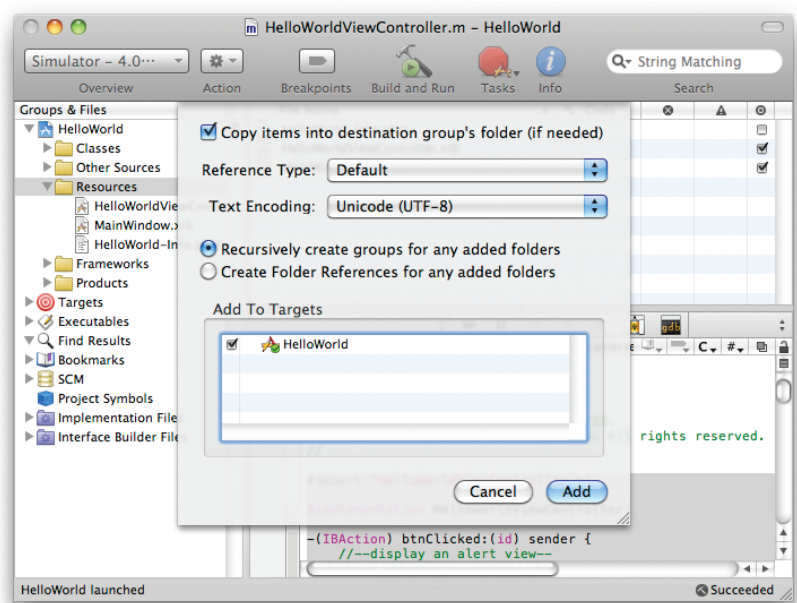


FIGURE 2-15

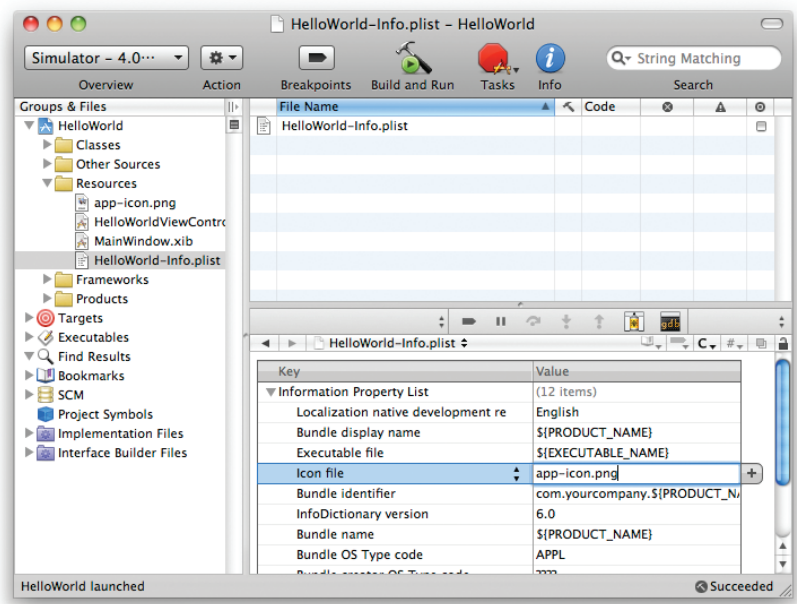


FIGURE 2-16

3. Press Command-R to run the application and test it on the iPhone 4 Simulator. Click the Home button to return to the main screen of the iPhone. You should see the newly added icon (see Figure 2-17).

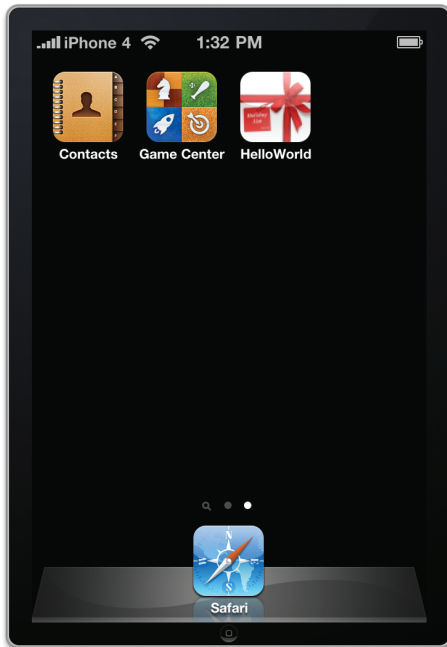


FIGURE 2-17

How It Works

Setting an icon for your application is very straight-forward — simply specify the icon filename in the Icon file item and it will appear in your iPhone when you run the application again. For more information about all the icons used in a typical iPhone application, check out Apple’s documentation at: <http://developer.apple.com/iphone/library/documentation/userexperience/conceptual/mobilehig/IconsImages/IconsImages.html>.

DISPLAYING A SPLASH SCREEN

Most iPhone applications display a splash screen whenever they are loaded. The splash screen serves as both a good way to display the logo of the company/application and a distraction to keep the user entertained while the application is busily loading itself into memory.

Displaying a splash screen is very easy using Xcode. You just need to include an image named `Default.png` in your application bundle (e.g., the `Resources` folder). This image needs to have a

resolution of 480×320 pixels (or 960×640 for iPhone high resolution). When your application is loaded, the system will automatically display this image and hide it when the first View window of your application is ready to be shown.

You can create the `Default.png` image from scratch using a photo-editing application, or easily capture one using the Organizer tool that is part of Xcode. All you need to do is view and capture the image you want to use as the splash screen on your iPhone. The following Try It Out describes how to add a splash screen using the Organizer.

TRY IT OUT Adding a Splash Screen to the Application

1. With the iPhone connected to your Mac, launch Xcode and select Window ⇨ Organizer.
2. You should now be able to see the name of the device attached to your Mac. Click the Use for Development button and then click the Screenshots tab.
3. View the desired image on your iPhone (e.g. the photos in the Photo Library) and then click the Capture button located under the image shown on the right in Organizer (see Figure 2-18).

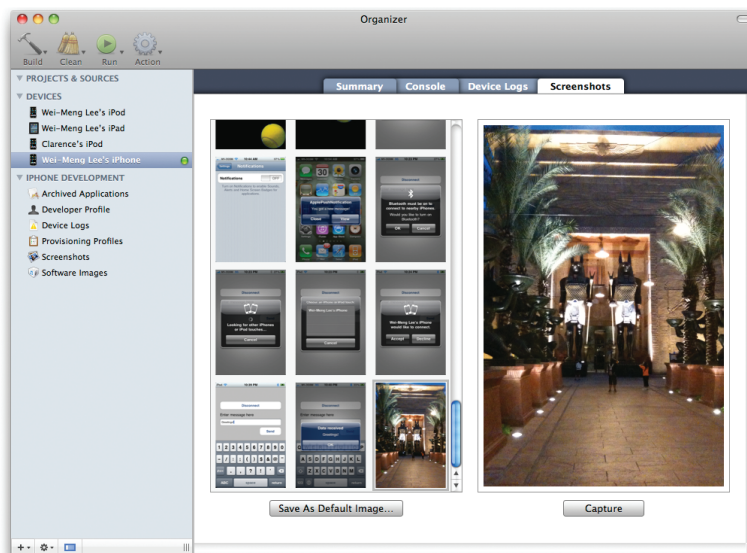


FIGURE 2-18

4. All the captured images are shown on the left of the Organizer window. Select the image that you want to use and click the Save As Default Image... button.
5. You will be prompted to select the project that you want to use for the default image (see Figure 2-19). Select the project name (`HelloWorld`) and click OK.

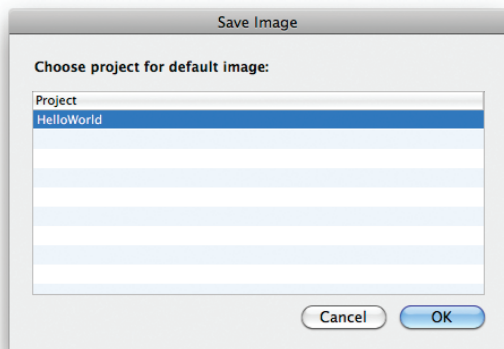


FIGURE 2-19

6. The name will be copied to the Resources folder of the HelloWorld Xcode project (see Figure 2-20).

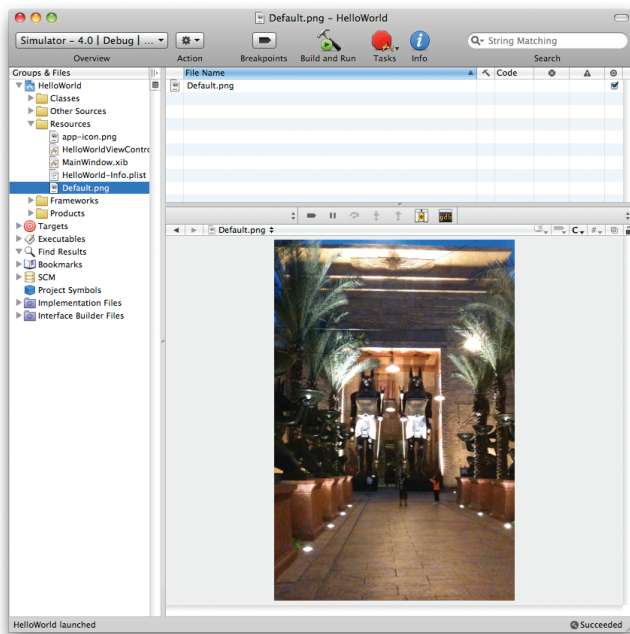


FIGURE 2-20

7. Press Command-R to test the application on the iPhone 4 Simulator. The splash screen will appear momentarily, followed by the HelloWorldViewController View window in your project.

8. If you want the splash screen to appear for a few seconds before it goes away, insert the following bold code into the `HelloWorldAppDelegate.m` file:

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    //--insert a delay of 5 seconds before the splash screen disappears--
    [NSThread sleepForTimeInterval:5.0];

    // Override point for customization after application launch.

    // Add the view controller's view to the window and display.
    [window addSubview:viewController.view];
    [window makeKeyAndVisible];

    return YES;
}
```

9. Press Command-R to test the application on the iPhone 4 Simulator again. This time, you will see the splash screen for about five seconds before it goes away.

How It Works

When you include an image named `Default.png` in your project, it will be displayed when your application is being loaded. This is a good chance to display your company's logo, or some information to keep the user occupied when your application is being loaded. Keep in mind the dimension of the image; it will not be displayed during loading if you have an image of the wrong size.

SUMMARY

This chapter provided a brief introduction to developing your first iPhone application. Although you likely still have many questions, the aim of this chapter was to get you started. The next few chapters dive deeper into the details of iPhone programming, and the secret of how all those components that seem so mysterious now work together is gradually revealed.

EXERCISES

1. You want to add an icon to an iPhone project in Xcode. What is the size of the image that you should provide?
2. What is the easiest way to add a splash screen to an iPhone application?
3. When adding an image to the Resources folder in your Xcode project, why do you need to check the "Copy items into destination group's folder (if needed)" option?

Answers to the Exercises can be found in Appendix E, on Wrox.com.

► WHAT YOU LEARNED IN THIS CHAPTER

TOPIC	KEY CONCEPTS
Xcode	Create your iPhone Application project and write code that manipulates your application.
Interface Builder	Build your iPhone UI using the various views located in the Library.
Adding an application icon	Add an image to the project and then specify the image name in the <code>Icon</code> file property of the <code>info.plist</code> file.
Adding a splash screen	Add an image named <code>Default.png</code> to the <code>Resources</code> folder of your project.
Creating icons for your iPhone applications	Icon size is 57×57 pixels and 114×114 pixels (high resolution). For App Store hosting, size is 512×512 pixels.