

SECTION III

Storing Data and Network Programming

- ▶ LESSON 22: Property Lists
- ▶ LESSON 23: Application Settings
- ▶ LESSON 24: Introduction to iCloud Storage
- ▶ LESSON 25: Introduction to CloudKit
- ▶ LESSON 26: Introduction to CoreData
- ▶ LESSON 27: Consuming RESTful JSON Web Services

23

Application Settings

Most applications that perform complex tasks will at some point need to allow users to customize the applications' operation to suit their specific needs. These customizable options are usually referred to as *application preferences* or *application settings*. iOS applications can either expose their preferences within Apple's Settings application, or provide a user interface within the application where the user can customize them appropriately.

To integrate your application's preferences with Apple's Settings application, your application must include a `Settings.bundle` file. A settings bundle file enables you to declare the preferences in your application as a property list, and the Settings application provides the user interface for editing those preferences.

Keep in mind that to access the Settings application your users will have to first exit your application if they were using it. You should always refresh settings data when the application is activated so that your application can learn about the changes made by the user via the settings app. In this lesson, you learn to create this file and use it to expose system preferences.

ADDING A SETTINGS BUNDLE

To add a `Settings.bundle` file to your application, right-click your application's group in the project navigator and select New File from the context menu. Select the Settings Bundle file type from the iOS Resource section of the dialog box (see Figure 23-1).

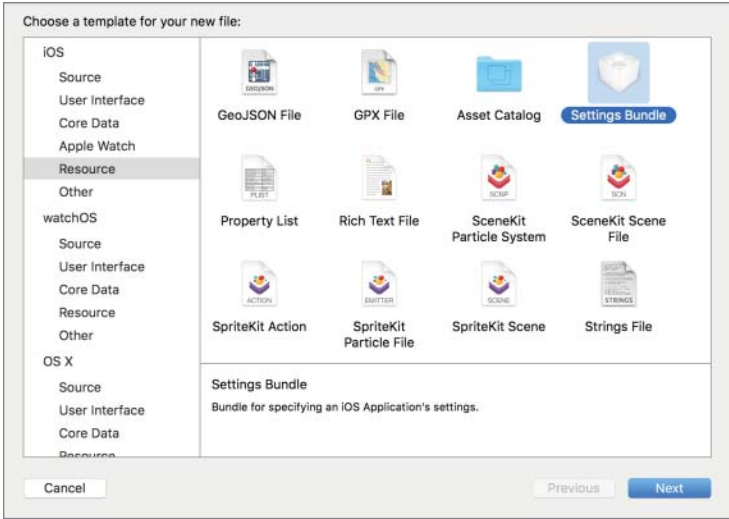


FIGURE 23-1

When the Settings application is launched on an iOS device, every third-party application is checked to see if it has a `Settings.bundle` file. For each application on the iOS device that has this file, its name and icon are added to a list on the main page of the Settings application (see Figure 23-2).



FIGURE 23-2

Tapping on the icon will take the user to the particular application’s settings page. By default, the Settings application will use an application’s standard icon file when listing it. If you want to provide

a custom icon to be used for your application in the Settings application, include the appropriate 2x and 3x images for the `AppIcon` asset in the project's asset catalog.

The Settings application can display application preferences in a series of hierarchical pages. Creating hierarchical settings pages is not covered in this lesson, but if you are interested in this topic, you should read the Preferences and Settings Programming Guide available at <https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/UserDefaults/Introduction/Introduction.html>.

A settings bundle is actually a collection of files. To see the contents of the bundle, simply click the triangle beside the `Settings.bundle` file in the project navigator (see Figure 23-3).

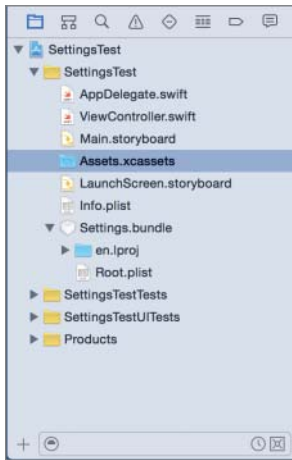


FIGURE 23-3

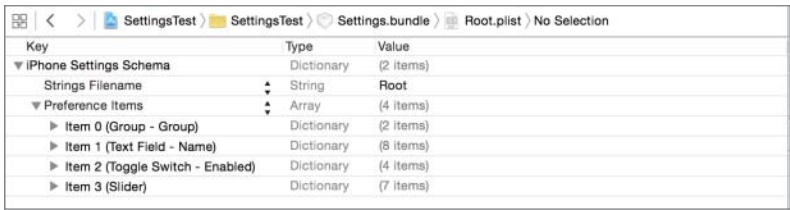
Inside the settings bundle, you will find a file named `Root.plist`. This file controls how your application's preferences will appear within the Settings application. Clicking the file opens it in the property list editor. When you do this, you will see a table with three columns—Key, Type, and Value. This file contains two properties: an array called `Preference Items` and a string called `Strings Filename` (see Figure 23-4).

SettingsTest > SettingsTest > Settings.bundle > Root.plist > No Selection		
Key	Type	Value
▼ iPhone Settings Schema	Dictionary	(2 items)
Strings Filename	String	Root
▶ Preference Items	Array	(4 items)

FIGURE 23-4

Each preference that you want to expose to your users will be an entry in the `Preference Items` array. To see the contents of the `Preference Items` array, simply expand it within the property list editor. When you create a new settings bundle, this array contains four items by default (see

Figure 23-5). Each entry in the array is a dictionary of key-value pairs. Technically speaking, the `Preference Items` property is an array of dictionaries.



The screenshot shows the Xcode interface for editing a Settings bundle. The breadcrumb trail at the top reads: SettingsTest > SettingsTest > Settings.bundle > Root.plist > No Selection. The main table displays the following structure:

Key	Type	Value
▼ iPhone Settings Schema	Dictionary	(2 items)
Strings Filename	String	Root
▼ Preference Items	Array	(4 items)
▶ Item 0 (Group - Group)	Dictionary	(2 items)
▶ Item 1 (Text Field - Name)	Dictionary	(8 items)
▶ Item 2 (Toggle Switch - Enabled)	Dictionary	(4 items)
▶ Item 3 (Slider)	Dictionary	(7 items)

FIGURE 23-5

Each entry within the `Preference Items` array, given that it’s a dictionary, can have several key-value pairs, but you will always find four keys in each entry—`Title`, `Type`, `Identifier`, and `DefaultValue`.

The value of the `Title` key is used by the Settings application to label the preference when it is presented to the user. The value of the `Type` key determines what kind of preference value it is and thus what user interface component will be used by the Settings application when presenting it. The value of the `Identifier` key contains a string that you can use to read the value of the preference in your Objective-C code. The value of the `DefaultValue` key contains the default value for the preference.

The default settings bundle created by Xcode contains four entries in the `Preference Items` array:

- ▶ Group
- ▶ Text Field
- ▶ Toggle Switch
- ▶ Slider

If you were to run this app on an iOS device, and look at its settings page in the Settings application, you would see something similar to that shown in Figure 23-6.

Table 23-1 describes the element types that can be used in the settings bundle.

TABLE 23-1: Preference Types

TYPE	DESCRIPTION
Text Field	An editable text field
Toggle Switch	On/Off toggle button
Title	A read-only text string
Slider	A slider to allow the user to select from a range of values
Multi Value	A list of values

TYPE	DESCRIPTION
Group	A logical group of preferences
Child Pane	Child preferences page, used to implement hierarchical preference pages

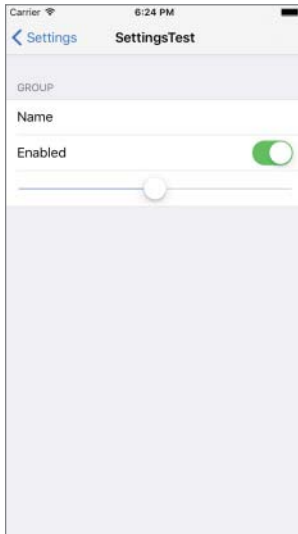


FIGURE 23-6

READING PREFERENCES WITH CODE

To read the value of a preference in a settings bundle from your code, you need to use an `NSUserDefaults` object. `NSUserDefaults` is part of the Core Foundation framework and provides a set of methods that allow you to manage application preferences. `NSUserDefaults` is a singleton class, and thus only one object should exist during the lifetime of an application. To get access to this one instance, use the following code:

```
let userDefaults = NSUserDefaults()
```

Recall that each preference within a settings bundle is represented by a dictionary of key-value pairs, and one of the four keys that each dictionary must contain is `identifier`. To retrieve the value of a preference that has the identifier `user_name`, use the following code:

```
let userName = userDefaults.valueForKey("user_name") as? String
```

This code assumes that the value being retrieved is a string. The `NSUserDefaults` class provides several methods that allow you to retrieve preference values of different data types, including:

- `boolForKey`
- `floatForKey`

- `doubleForKey`
- `integerForKey`

Although you have provided default values for the preferences in the settings bundle, these values will not be applied until the users launch the Settings application on their device after installing your application. To get around this problem, you should specify a default value for each of your preferences in code as well as the settings bundle.

You can then use methods in the `NSUserDefaults` class to ensure that the default values are applied only once regardless of whether your user launches the Settings application or your application first. To do this, you need to create a dictionary with the default values of each preference and use the `registerDefaults` and `synchronize` methods of the `NSUserDefaults` object as follows:

```
let registrationDictionary:[String: String] = ["user_name":"Paul Woods",
    "user_age":"28"]

NSUserDefaults.registerDefaults(registrationDictionary)
NSUserDefaults.synchronize()
```

TRY IT

In this Try It, you create a simple iPhone application based on the Single View Application template called `SettingsTest` that allows the user to specify a name and age value within the Settings application. Your application, when launched, will display this name and age.

Lesson Requirements

- Launch Xcode.
- Create a new iPhone project based on the Single View Application template.
- Add a settings bundle to the application.
- Add user interface elements to the default scene of the storyboard.
- In the `viewDidLoad` method, read the preference values and display them in the labels.

REFERENCE *The code for this Try It is available at www.wrox.com/go/swiftios.*

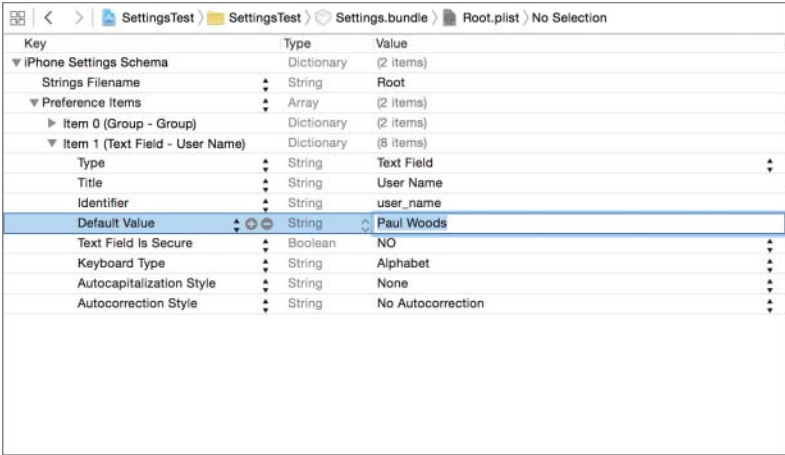
Hints

- To display your application's preferences in the Settings application, you must include a `Settings.bundle` file.
- To access the preference values specified by the user in the settings page from within your code, each preference must have a unique string identifier.

- When creating a new project, you can use your website's domain name as the Company Identifier in the Project Options dialog box.
- To show the Object library, select View ⇨ Utilities ⇨ Show Object Library.
- To show the Assistant editor, select View ⇨ Assistant Editor ⇨ Show Assistant Editor.

Step-by-Step

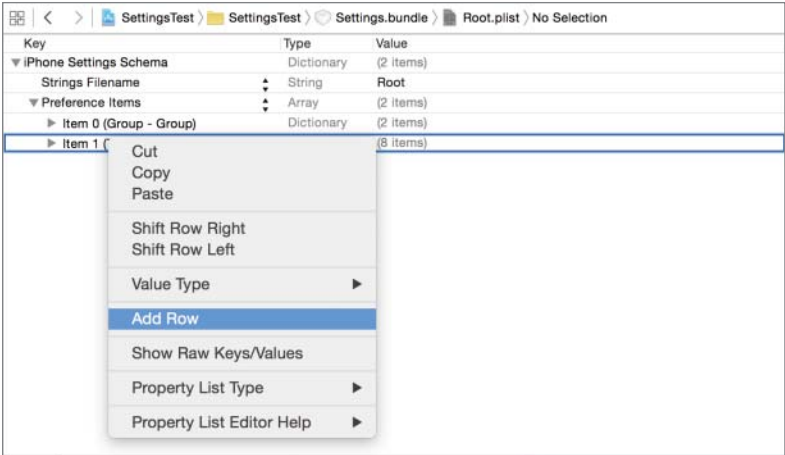
- Create a Single View Application in Xcode called `SettingsTest`.
 1. Launch Xcode and create a new application by selecting File ⇨ New ⇨ Project menu item.
 2. Select the Single View Application template from the list of iOS project templates.
 3. In the project options screen, use the following values:
 - **Product Name:** `SettingsTest`
 - **Organization Name:** your company
 - **Organization Identifier:** `com.yourcompany`
 - **Language:** Swift
 - **Devices:** iPhone
 - **Use Core Data:** Unchecked
 - **Include UI Tests:** Unchecked
 4. Save the project onto your hard disk.
- Add a settings bundle to the project.
 1. Ensure the project navigator is visible.
 2. Right-click the `Settings Test` group and select New File from the context menu.
 3. Select the Settings Bundle template from the iOS Resources section. Save the file as `Settings.bundle`.
- Edit the `Settings.bundle` file.
 1. Expand the `Settings.bundle` file in the project navigator and click the `Root.plist` file to edit it with the property editor.
 2. Expand the `Preference Items` property.
 3. Delete items 2 and 3. These are the Toggle Switch and Slider items, respectively. To delete an item, select it and hit the backspace key.
 4. Edit the Text Field preference.
 - Expand the `Item 1 (Text Field - Name)` dictionary.
 - Set the `Title` to **User Name**, `Identifier` to `user_name`, and `Default Value` to **Paul Woods** (see Figure 23-7).



Key	Type	Value
iPhone Settings Schema	Dictionary	(2 items)
Strings Filename	String	Root
Preference Items	Array	(2 items)
Item 0 (Group - Group)	Dictionary	(2 items)
Item 1 (Text Field - User Name)	Dictionary	(8 items)
Type	String	Text Field
Title	String	User Name
Identifier	String	user_name
Default Value	String	Paul Woods
Text Field Is Secure	Boolean	NO
Keyboard Type	String	Alphabet
Autocapitalization Style	String	None
Autocorrection Style	String	No Autocorrection

FIGURE 23-7

5. Add a new Text Field preference.
- Ensure the Item 1 (Text Field - User Name) dictionary is collapsed.
 - Right-click the row corresponding to the Item 1 (Text Field - User Name) dictionary and select Add Row from the context menu (see Figure 23-8).



Key	Type	Value
iPhone Settings Schema	Dictionary	(2 items)
Strings Filename	String	Root
Preference Items	Array	(2 items)
Item 0 (Group - Group)	Dictionary	(2 items)
Item 1 (Text Field - User Name)	Dictionary	(8 items)

- Cut
- Copy
- Paste
- Shift Row Right
- Shift Row Left
- Value Type
- Add Row**
- Show Raw Keys/Values
- Property List Type
- Property List Editor Help

FIGURE 23-8

- Expand the newly added preference dictionary.
- Ensure the Type key is set to **Text Field**, Title is set to **Age**, and Identifier is set to **user_age**.

- Add a new key to the dictionary by right-clicking the last key (`Identifier`) and selecting `Add Row` from the context menu.
- Ensure the name of the new key is `Default Value` and the value of the key is `28` (see Figure 23-9).

The screenshot shows the Xcode interface for editing a Settings.bundle file. The breadcrumb at the top indicates the path: SettingsTest > SettingsTest > Settings.bundle > Root.plist > No Selection. The main table has three columns: Key, Type, and Value. The table structure is as follows:

Key	Type	Value
▼ iPhone Settings Schema	Dictionary	(2 items)
Strings Filename	String	Root
▼ Preference Items	Array	(3 items)
▶ Item 0 (Group - Group)	Dictionary	(2 items)
▶ Item 1 (Text Field - User Name)	Dictionary	(8 items)
▼ Item 2 (Text Field - Age)	Dictionary	(4 items)
Type	String	Text Field
Title	String	Age
Identifier	String	user_age
Default Value	String	28

FIGURE 23-9

- Add two `UILabel` instances to the default scene.
 1. Open the `Main.storyboard` file in the Interface Editor.
 2. From the Object library, drag and drop two `Label` objects onto the scene and place them one below the other.
 3. Create layout constraints for each of elements on the storyboard scene using the information in Table 23-2. When creating layout constraints using the pin constraints dialog box, ensure the `Constrain to margins` option is unchecked and the value of the `Update Frames` combo box is set to `Items of New Constraints`.

TABLE 23-2: Layout Constraints

ELEMENT	LEFT	TOP	RIGHT	HEIGHT
Label 1	20	20	20	21
Label 2	20	20	20	21

4. Use the assistant editor to create outlets for each of the labels in the view controller class. Name the outlets `nameLabel` and `ageLabel`.
- Read and display the preference values provided by the user in the Settings application.
 1. Open the `ViewController.swift` file in the project explorer.
 2. Replace the implementation of the `viewDidLoad` method to resemble the following:


```
override func viewDidLoad() {
    super.viewDidLoad()

    let userDefaults = UserDefaults()
    let registrationDictionary:[String: String] =
```

```
        ["user_name": "Paul Woods", "user_age": "28"]

        userDefaults.registerDefaults(registrationDictionary)
        userDefaults.synchronize()

        nameLabel.text = userDefaults.valueForKey("user_name") as? String
        ageLabel.text = userDefaults.valueForKey("user_age") as? String
    }
}
```

- Test your app in the iOS Simulator.
 1. Click the Run button in the Xcode toolbar. Alternatively, you can use the Project ⇨ Run menu item.
 2. After changing preferences in the Settings application, ensure your application is not running in the background before launching it again.

REFERENCE *To see some of the examples from this lesson, watch the Lesson 23 video online at www.wrox.com/go/swiftiosvid.*