

第 3 章 iOS 开发起航

掌握了 iPhone 的开发语言 Objective-C，并且对 iOS SDK 有了一定的了解，那么就可以开始 iPhone 应用程序的学习了。本章将会从通过一个最简单的程序“Hello World”入手，貌似所有的编程语言入门程序都是这个。读者将会学习如何使用 Xcode 创建、编写、运行和调试程序，并学习用 SDK 自带的可视化编译工具 Interface Builder 设计界面。

本章主要涉及到的知识点如下。

- ❑ 开发工具使用：学习利用 Xcode 创建 iPhone 程序，熟悉 Xcode 的基本操作。
- ❑ iPhone 程序：了解一个完整的 iPhone 项目各个组成部分。
- ❑ Interface Builder：学习 Interface Builder 组件的构成、变量的关联。
- ❑ 程序完善：为程序添加图标、标题。

 **注意：**本章的程序只需要在模拟器上演示即可，不需要 iPhone、iTouch 等真机设备。如果想要把自己的程序运行在真实设备上，必须保证已经加入苹果开发者计划中，具体操作流程，可以参考苹果的相关指南。

3.1 创建项目 HelloWorld

本节首先带领读者学习如何使用 Xcode 创建 iPhone 项目，并对其中涉及到的知识点进行详细说明，还会为读者介绍 Xcode 各工作面板的基本组成部分。在创建和编写程序的过程中，具体讲解 Xcode 的使用，以及 iPhone 应用程序的基本架构和组成部分。

3.1.1 应用程序模板

Xcode 为开发者提供的几种应用程序模板，它们可以满足常见的 iPhone 开发，我们在这些预先配置好的工程上修改就可以了。

1. Master-Detail Application

如果想要在初始创建的 iPhone 程序中实现导航控制，或者想在创建的 iPad 程序里拥有切割视图，可以选择这个模板。iPad 自带的“邮件”程序就是一个典型的例子，界面如图 3.1 所示。

这是一个分割视图，左侧是导航界面，显示收件人和账户。右侧是内容界面，展现邮件的具体内容或账户信息。

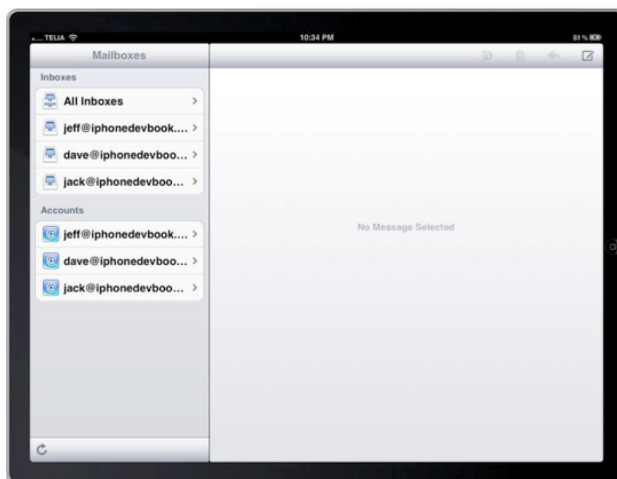


图 3.1 iPad 邮箱界面

⚠注意：开发 iPad 应用程序时，可以使用 `UISplitViewController` 类实现切割视图。

2. OpenGL Game

这个模板用于创建 3D 游戏或者图形。它会创建一个配置好的视图，专门用来显示 GL 场景，并提供了一个例子计时器可以令其演示动画，如图 3.2 所示。

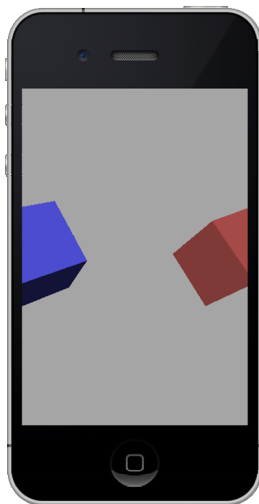


图 3.2 3D 图型效果

3. Page-Based Application

这个是 iOS 5 引入的一个新模板，选择此项目模板就可以创建一种“基于页”的应用程序，实现翻页效果，如图 3.3 所示。



图 3.3 翻页程序界面

4. Single View Application

此模板拥有一个单页面的视图界面，并提供一个视图控件来管理界面显示。较早的 Xcode 版本的 View-Based Application 模板和这个是一致的。本章节的程序就要从这个模板开始。选择此模板后，程序运行在模拟器里的外观如图 3.4 所示，看起来像是什么也没有，其实只有一个空空的视图。

 **注意：**某些版本的 Xcode 会为这唯一的视图加点背景色，看起来不至于太过单调。

5. Tabbed Application

此模板会在底部显示一行标签，提供一系列的快捷方式。iPhone 的“时钟”就采用了这样的样式，如图 3.5 所示。

“闹钟”程序包含 World Clock、Alarm、Stopwatch 和 Timer 4 个标签页，它们在层级关系上彼此平等且独立。



图 3.4 带有背景色的单视图程序



图 3.5 iPhone 自带的闹钟程序

6. Utility Application

这个模板提供了一个主视图和一个切换视图，主视图有按钮，单击后视图之间会相互切换。iPhone 的“天气”采取了这种样式，如图 3.6 所示。

主视图显示天气的详情信息，单击右下角的“i”按钮，页面会翻转到城市设置界面，如图 3.7 所示。



图 3.6 “天气”主视图界面



图 3.7 “天气”城市设置界面

7. Empty Application

如果其他模板都不适合自己的需求，可以考虑这个模板。基于此模板的应用程序只包含了一个主窗口。

3.1.2 新建 iPhone 工程

本小节将分 5 个步骤为读者详细介绍如何创建一个 iPhone 应用程序。读者将了解创建新项目的流程，同时提高使用 Xcode 的熟练度。

1. 新建工程

打开 Xcode，单击 File|New|Project 命令，如图 3.8 所示。

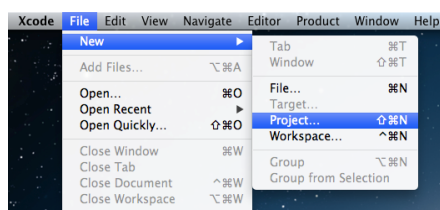


图 3.8 创建新工程

2. 选择模板

此时弹出工程模板选择界面，窗口左侧罗列了很多模板类别。选择 iOS 下的 Application，面板上就会出现当前类别的所有模板，选中某个模板后，Xcode 会展示相关的模板描述。我们选择 Single View Application，如图 3.9 所示。

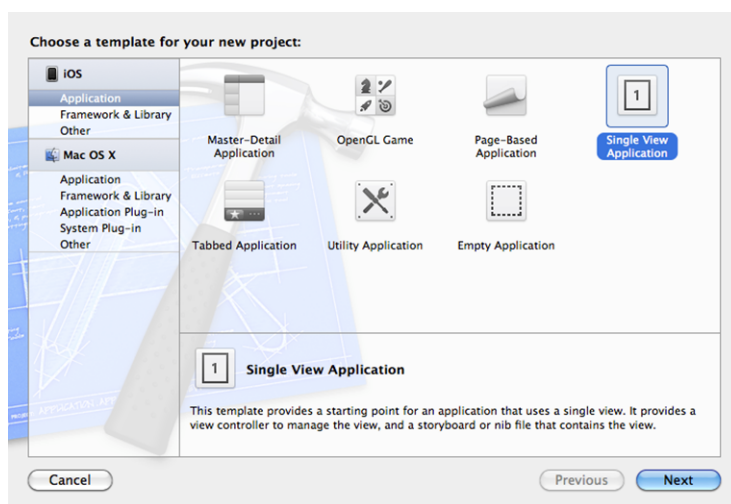


图 3.9 工程模板选择界面

前面介绍过，Single View Application 模板会带一个默认的视图，选择 Single View Application 模板，单击 Next 按钮。

3. 输入产品名称等信息

选择完模板后，Xcode 会弹出一个工程信息的配置界面，在这里必须输入产品的名称和公司标识符等信息，如图 3.10 所示。

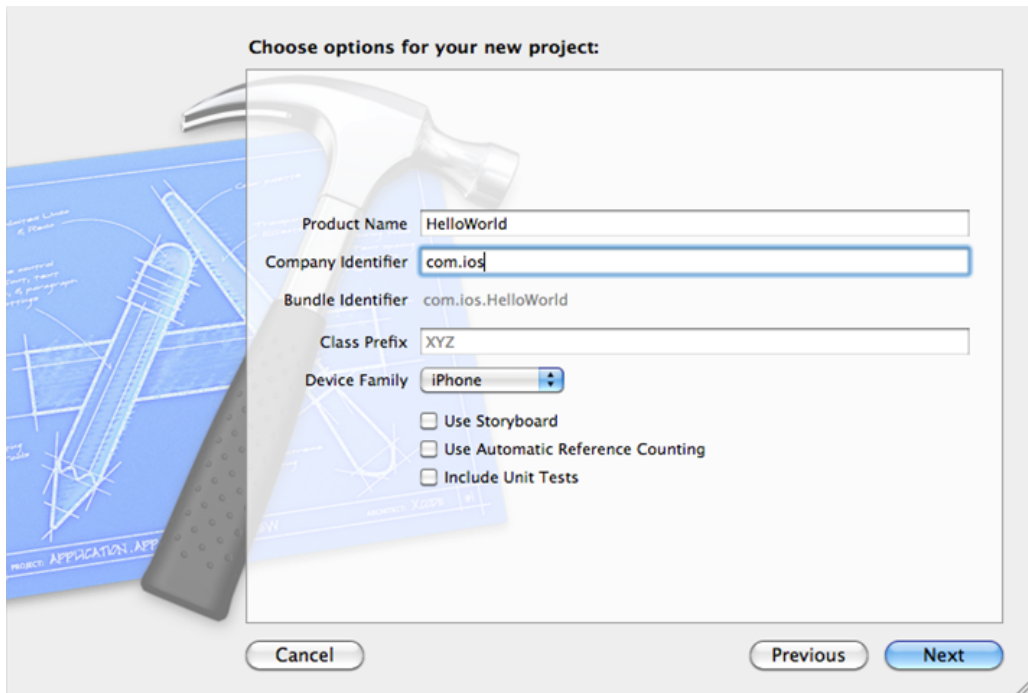


图 3.10 工程配置界面

- ❑ Product Name 为工程名称，输入“HelloWorld”。
- ❑ Company Identifier 为公司标识符，一般命名规则为“com.公司名”。
- ❑ Bundle Identifier 为包标识符，用于唯一标识应用程序，默认的会根据 Company Identifier 和 Product Name 生成。
- ❑ Class Prefix 不能输入，不用理会。Device Family 为程序支持的设备类型，分为 iPhone、iPad 和 Universal（兼容）3 种，选择“iPhone”。
- ❑ 下面有 3 个复选框，Use Storyboard 表示使用 iOS 5 新增的故事版功能来创建 UI，Use Automatic Reference Counting 表示内存的释放是否交给系统自动处理，Include Unit Tests 如果被选中，工程下多出一个单元测试代码模块。这里全部都不选。

注意：上面所有的配置项除了产品名称在工程创建完毕后，还可以再修改。所以，即使这里设置错了，也不必担心。

4. 选择存储位置

单击 Next 按钮，然后选择工程文件的保存路径，如图 3.11 所示。

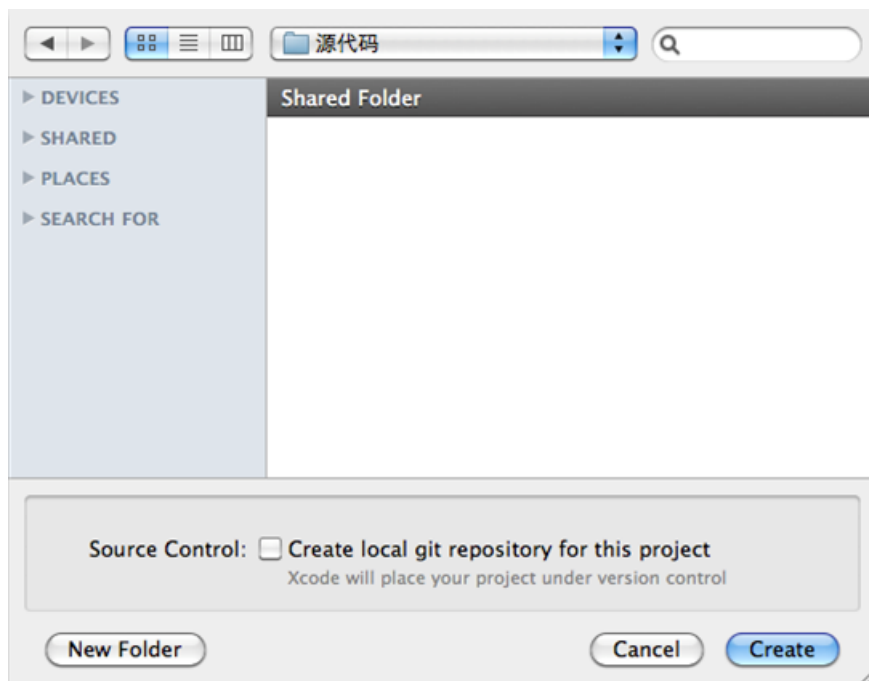


图 3.11 工程路径保存界面

 提示：这里不选择 Source Control 复选框，它主要用于代码的本地版本控制。

5. 完成创建

单击 Create 按钮，接下来弹出 Xcode 工程面板，待所有文件加载完成后，工程就算创建完毕了。工程界面如图 3.12 所示。

 注意：退出程序后，在项目文件夹中单击 HelloWorld.xcodeproj 文件能重新打开工程。

3.1.3 详解 Xcode 的各个部分

上一章大致介绍了 Xcode。接下来将会更具体地分析 Xcode，这个工具上面太多眼花缭乱的按钮和图标会让不熟悉它的人感到晕头转向，因此对它了解的越深入，越有助于开发工作。Xcode 内置 Interface Builder 设计工具，并且与 Apple LLVM 编译器进行深度整合。Instruments 分析工具也包含于工具包之中，另外还有其他很多提供各种支持的开发者工具。

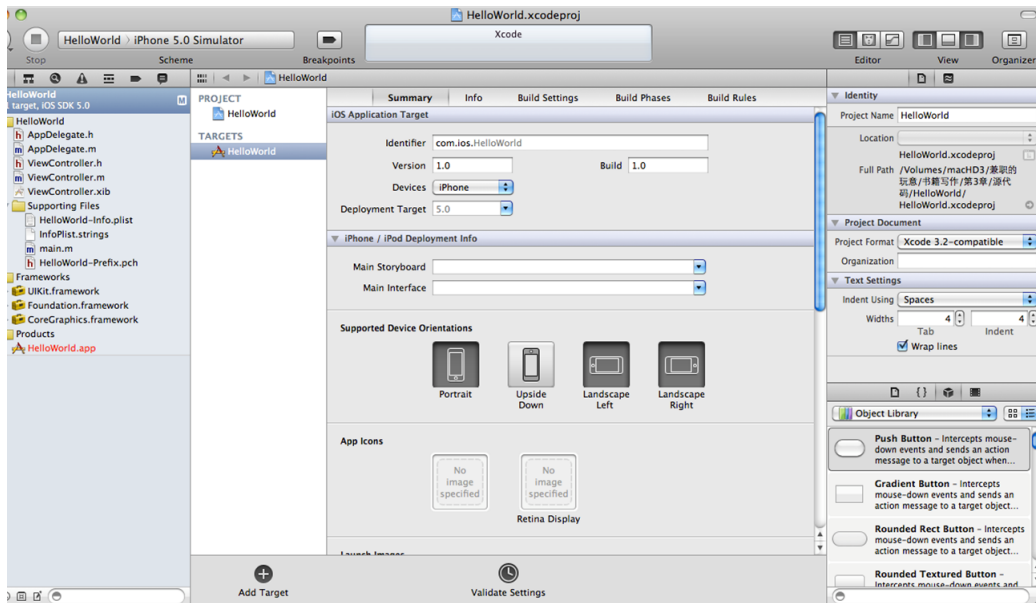


图 3.12 Xcode 工程面板

(1) 单窗口：Xcode 4 以前的版本中，开发任务的窗口是分散在多个窗口中的，如今全部合并成一个。这样即使同时打开多个工程，也不会让开发者产生工作区混乱的感觉了。

(2) 导航栏：Xcode 的左上方是导航栏，如图 3.13 所示。



图 3.13 Xcode 导航栏

导航栏包括一个工程文件列表、已排序的符号、搜索界面、正在跟踪的问题、调试数据、断点以及开发日志。通过这些功能，开发者便可对工程的内容以及搜索结果进行过滤。

(3) 跳转栏：位于 Xcode 中上方，显示正在编辑文件的相对位置，单击路径中的某一位置就可立刻跳转过去。跳转栏可有效应用于不同文件的快速跳转，如图 3.14 所示。

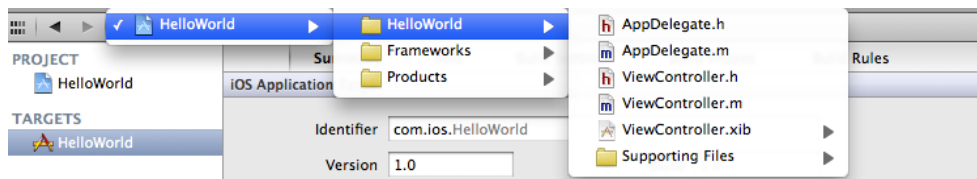


图 3.14 Xcode 跳转栏

(4) 界面编辑器：Xcode 的中间部分就是界面编辑器，如图 3.15 所示。从 Xcode 4 开始，编辑器就有了代码补全和消息泡泡等功能，消息泡泡会在代码出现警告或者错误

的时候自动弹出。

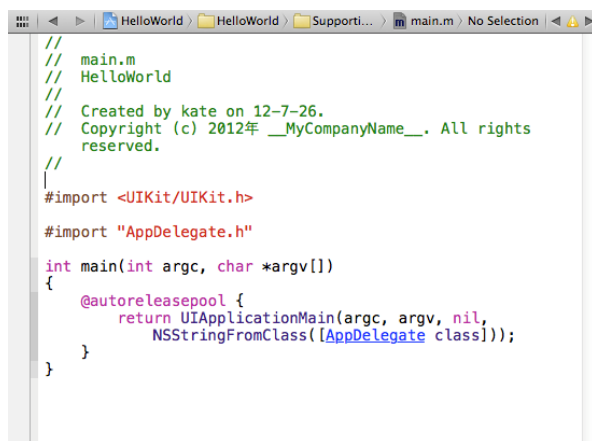


图 3.15 我们在这里编写代码，编辑区上方显示当前文件的路径

(5) Assistant Editor: 位于 Xcode 右上方，如图 3.16 所示。



图 3.16 Assistant Editor

Assistant 按键可以把编辑器一分为二，其中一个是辅助面板。它会根据当前的工作自动显示最有帮助的文件。假设当前正在编辑的某个源文件，则它会显示与之对应的头文件，如图 3.17 所示。

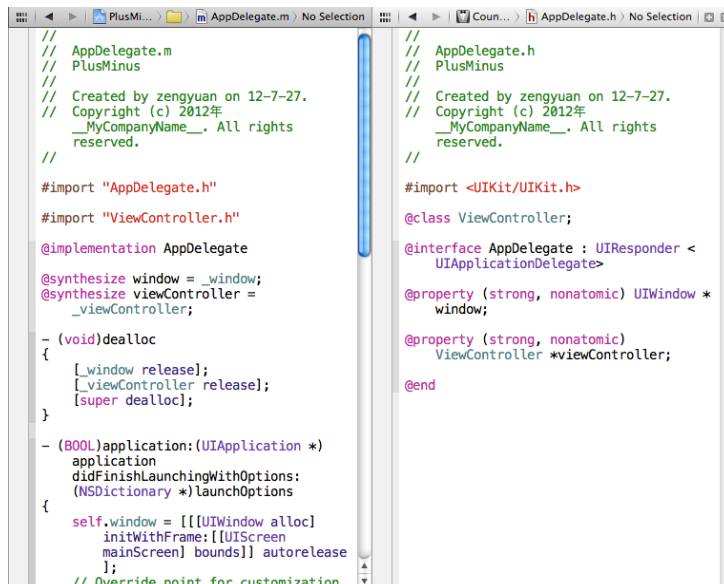


图 3.17 编辑区双屏效果

说明：编辑区左侧是实现文件，右侧是头文件。

(6) 内置 Interface Builder: Xcode 4 中, Interface Builder 整合到了 Xcode 中, 不再是独立的应用程序, 如图 3.18 所示。

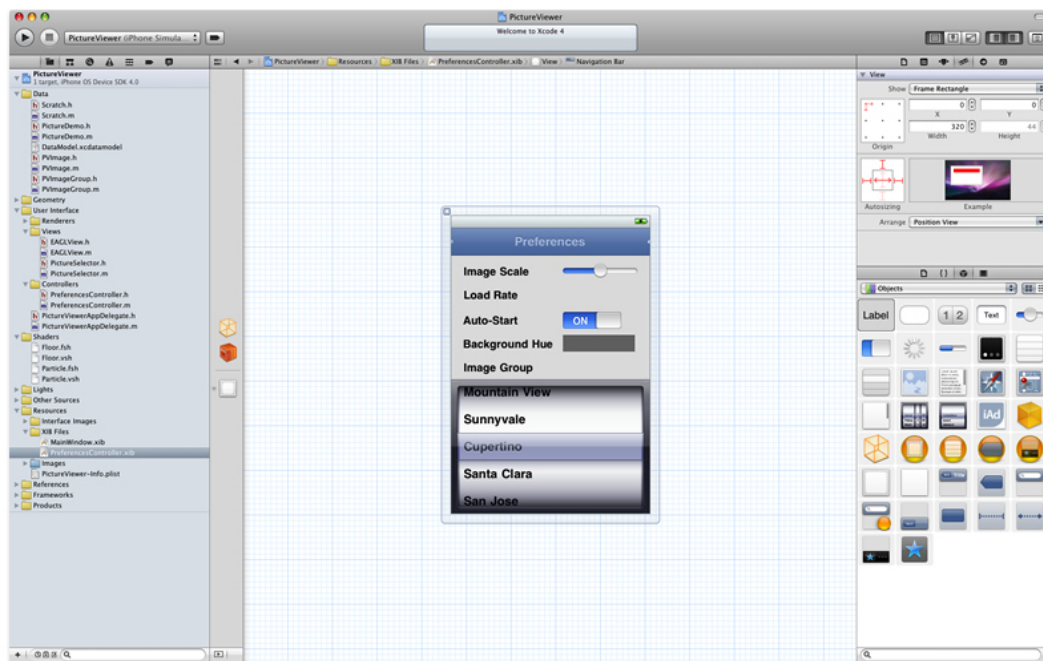


图 3.18 Interface Builder 界面

选中一个界面文件 (具有 .nib 或者 .xib 后缀的文件), 中间的编辑区就会打开 Xcode 的 Interface Builder 编辑器。打开右手边的工具区, 就可以看到控件库以及 UI 对象等。Interface Builder 使用起来很有意思, 非常直观, 稍后会使用它来设计程序界面。

(7) 文档查看器: 单击 Xcode 工具栏的 “Help” 菜单, 打开文档查看器, 如图 3.19 所示。

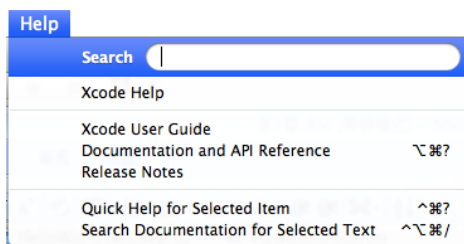


图 3.19 文档查看器

文档查看器帮助开发者搜索某个框架、类或变量的说明使用, 还可以下载官方源

代码。

(8) 调试区：位于 Xcode 的最下方，代码运行的信息显示在调试区。程序调试时，可以把某个类成员或者临时变量的值打印在控制台上。

3.1.4 详解项目的各个部分

现在，我们最关心的是 Xcode 会为项目生成哪些文件。导航栏的第一个标签页下罗列着项目的 3 个文件目录。

1. HelloWorld

展开工程目录下的 HelloWorld 文件夹，可以看到程序包含了几个初始文件。

- ❑ AppDelegate：执行了程序的入口函数等一系列方法。
- ❑ ViewController：初始加载的视图控制类，对 ViewController 界面的修改都会在程序运行的时候显现出来。
- ❑ Supporting Files 子文件夹：里面包含了 HelloWorld-Info.plist、InfoPlist.strings、main.m 和 HelloWorld-Prefix.pch 四个文件。HelloWorld-Info.plist 包含了应用程序一些属性信息，InfoPlist.strings 用于应用程序本地化，main.m 包含应用程序的 main() 方法。
- ❑ HelloWorld-Prefix.pch：Xcode 预编译包含在此文件中的头文件，这会减少使用程序运行所需的时间，通常不需要编辑或修改此文件。

2. Frameworks

在此文件夹中添加的任何框架或库都将链接到应用程序中，并且代码将能够使用包含在该框架或库中的对象、函数和资源。项目中默认链接了最常用的框架和库。

Xcode 4 添加框架或者库相较于以前的版本还是差别很大的，先选中导航栏下的工程图标，再选中“TARGETS”，找到“Linked Frameworks and Libraries”，如图 3.20 所示。

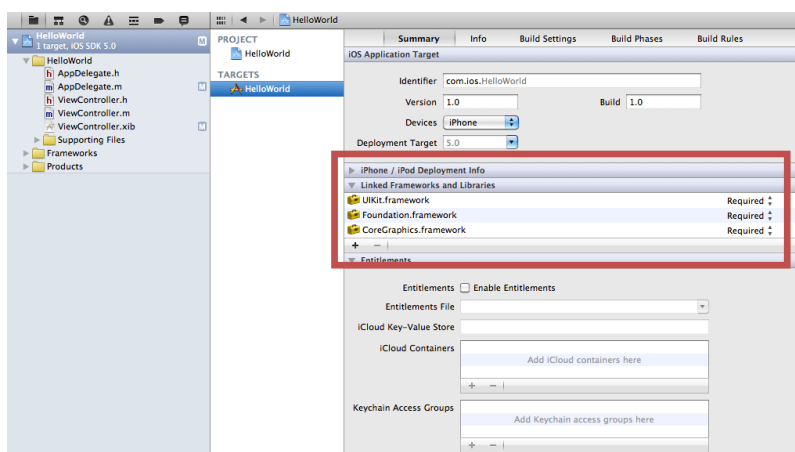


图 3.20 如何为项目添加框架和库

单击“+”按钮，弹出“框架和库列表”对话框，如图 3.21 所示。

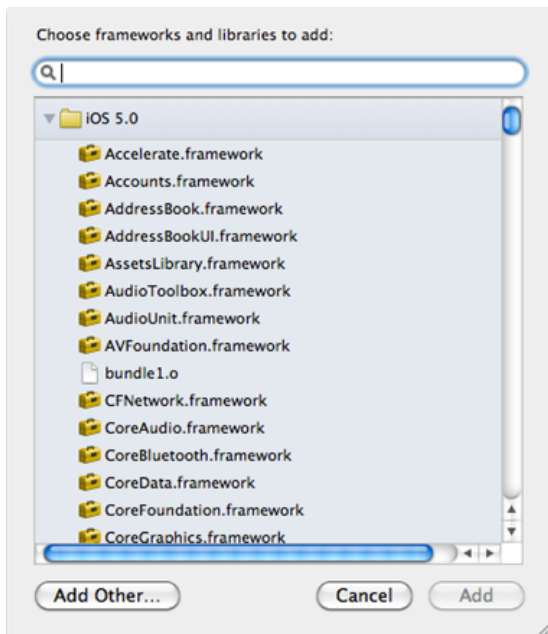


图 3.21 框架和库列表

选中想要添加的框架或者库后，单击 Add 按钮就可以了。单击 Add Other 按钮，可以手动添加框架。

注意：选中 Linked Frameworks and Libraries 栏下的某个框架，单击“-”按钮，就可以将其从工程中移除。

3. Products

展开 Products，可以看到一个名称为 Hello World.app 的项。Hello World.app 是项目的唯一产品。编译成功后，可以在 Finder 中查看这个具体的文件。

3.2 解密 Interface Builder

Interface Builder (IB) 是 Mac OS X 平台下用于设计和测试用户界面 (GUI) 的应用程序，开发者也可以选择完全用代码的方式来创建界面。但熟悉了 Interface Builder 的使用后就会发现，只需要简单地拖曳和布局就可以设计出复杂的界面。

在 Xcode 4 以前的版本，Interface Builder 是独立于 Xcode 存在的。Xcode 4 以后，Interface Builder 都集成在 Xcode 中。

3.2.1 掀开 Nib 神秘的面纱

学习 Interface Builder 之前必须要了解下 Nib 和 Xib 内容。

Nib 文件是基于磁盘的资源文件。在 Xcode 4 以前, iPhone 应用程序的主 nib 文件通常包含一个窗口对象 (MainWindow) 和一个应用程序委托对象 (AppDelegate), 还可能包含一个或多个管理窗口的其他对象。

那 Nib 究竟是什么呢? 它是一种数据文件, 大多数情况下, 应用程序使用 Nib 文件来存储构成用户界面的窗口和视图。当 Nib 文件载入应用程序时, Nib 装载代码会将文件中的内容转化为应用程序可以操作的真正对象。通过这个机制, Nib 文件省去了用代码创建那些对象的工作。

总之, Nib 文件是 Interface Builder 文档, 它被 Interface Builder 用来设计窗口或视图, 即程序的界面。从 Interface Builder 3 之后, 引入了新的文档格式 Xib, 从此以后, Xcode 在创建项目时, 都使用 Xib 格式的文档来设计界面。

从本质上来讲, Xib 和 Nib 没有任何区别, 只是在苹果的官方文档里习惯性地使用“nib”作为统一术语。

3.2.2 详解 Interface Builder 的组成部分

在工程目录下, 有一个 ViewController.xib 文件。选中 ViewController.xib, Xcode 的编辑区域就会自动展现 Xib 文件的详细信息, 如图 3.22 所示。

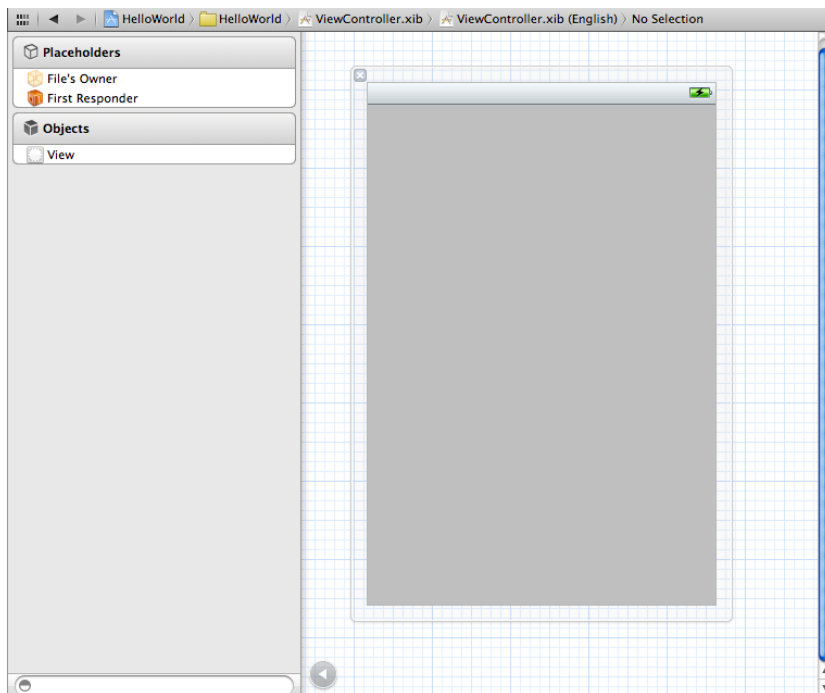


图 3.22 Xib 详细信息显示在 Interface Builder 编辑区内

下面介绍下 Xib 文件的构成。

1. 场景对象

整个 Interface Builder 编辑器左侧的区域是场景对象。

- ❑ **Files's Owner:** 顾名思义, Files's Owner 是拥有此 nib 文件的对象, 当前 nib 文件的拥有者就是 **ViewController** 这个类。File's Owner 负责管理 Interface Builder 中的控件与实例 **IBOutlet** 变量的关联。
- ❑ **First Responder:** 用户当前正在与之交互的对象。例如, 如果正在一个登录框输入密码, 那么这个登录框就是当前的 **First Responder**。First Responder 将随着用户与界面的交互而变化。
- ❑ **View:** Files's Owner 和 First Responder 放在了 **Placeholders** 图标下, View 单独放在了“Objects”图标下, View 用于图形化设计用户界面。

2. 编辑区域

中间的部分是 **View**, 它是一个可视化的操作区域。这里所说的 **View** 都是 **UIView** 实例, 它负责视图控制器显示在设备屏幕上时的布局和界面。在本章所创建的程序里, 只包含一个视图, 所有和界面有关的操作都要发生在这个 **View** 上。更多视图、更复杂的程序在以后的章节里会讲到。

3. 属性区域

Xcode 有一个带有 3 个按键的工具栏, 位于 Xcode 面部的右上角, 如图 3.23 所示。单击最右边的按键, 此时位于 Xcode 最右侧的属性区就会被打开, 属性区上面部分可以用来设置类和变量的各个属性, 如图 3.24 所示。



图 3.23 Xcode 工具栏

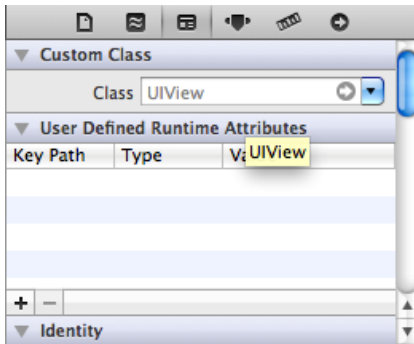


图 3.24 属性设置面板

🔔 **注意:** 这部分也被称之为属性查看器 (File Inspector)。

下面的部分是资源库 (Library), 如图 3.25 所示。资源库包含 IB 支持的所有的 Cocoa Touch 对象, 如“Label”表示一个标签, “Round RectButton”表示一个带有圆角的矩形按钮。

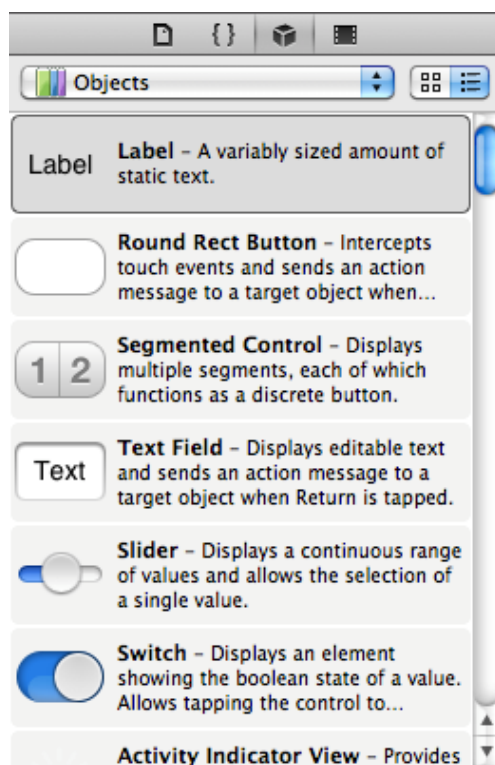


图 3.25 资源库

如果需要特定类型的控件，可以从“Objects”下拉列表里选择控件类型，下拉列表如图 3.26 所示。

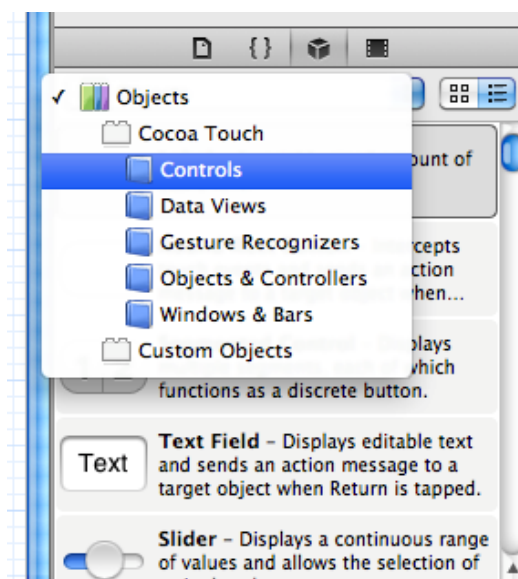


图 3.26 控件分类列表

- ❑ Controls: 基本的 UI 控件, 如标签、按钮。
- ❑ Data Views: 有数据交互的控件, 如选择器、地图。
- ❑ Gesture Recognizers: 手势操作, 如滑动、放大、缩小。
- ❑ Objects & Controllers: 控制器, 如标签控制器和导航控制器。
- ❑ Windows & Bars: 窗口和功能条, 如工具条和导航条。

🔔注意: 不仅当前的窗口是一个视图, 资源库中的 UI 控件都属于视图, 单视图程序是说当前和用户交互的主视图只有一个。

3.2.3 在视图添加第一个 UI 控件

把控件对象添加到视图中的方法非常简单, 选中一个 Label (标签) 控件, 直接拖曳到 View 视图中, 如图 3.27 所示。

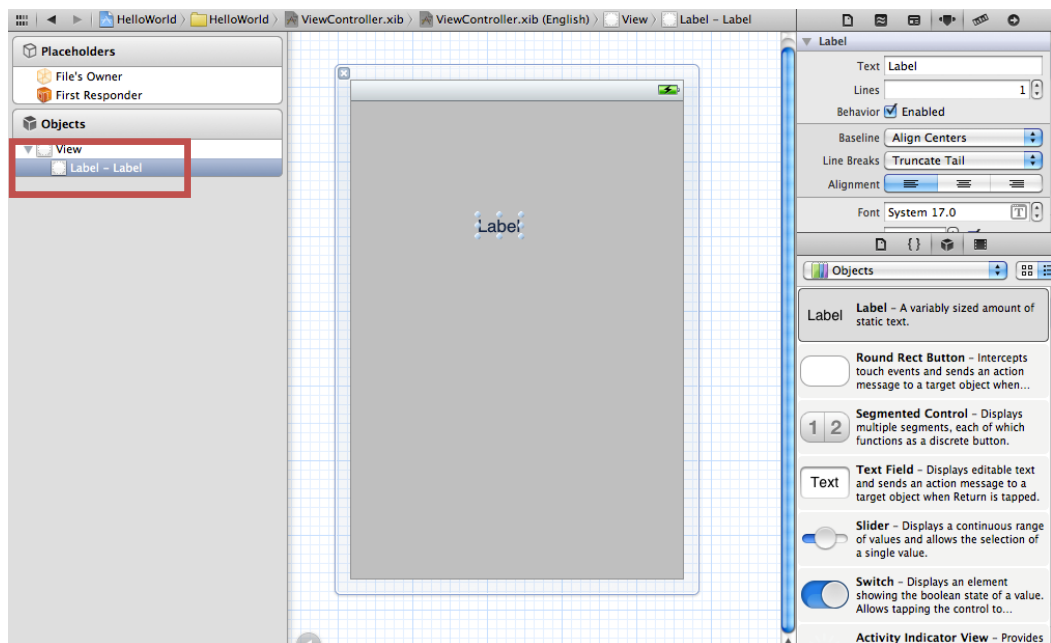


图 3.27 Interface Builder 界面

可以看到左侧的 View 下多了一个 Label-Label, 界面对象之间是具有层次结构的, Label 是显示在 View 之上的。

3.2.4 设置 UI 控件的属性

我们已经将一个 Label 控件添加到界面视图上了, 接下来需要修改它的相关属性。选中 Label 后, 单击 Xcode 右侧的属性面板上像一个小茶杯一样的图标, 如图 3.28 所示。



图 3.28 属性面板工具条

选中以后，属性面板就会把 Label 的基本属性都展现出来，如图 3.29 所示。Label 是使用频率最高的控件之一，比较常用的属性有这么几个。Text 表示 Label 的标题；Lines 表示 Label 的文本行数；Alignment 表示文字对齐方式，默认是左对齐；Font 是 Label 的字体；Text Color 表示 Label 的文本颜色，默认是黑色。

下面分 4 步设置 Label 的属性。

(1) 修改 Label 的标题为“Hello World”，设置 Label 的颜色为红色。

(2) 下面让 Label 的显示位置居中，选中工具条上的倒数第 2 个图标，从“Arrange”下拉框中选择“Center Horizontally In Container”，如图 3.30 所示。

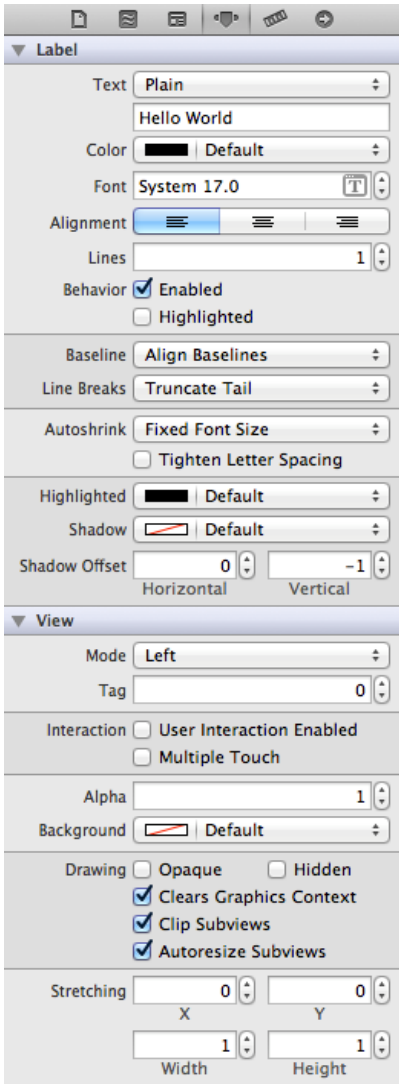


图 3.29 属性设置界面

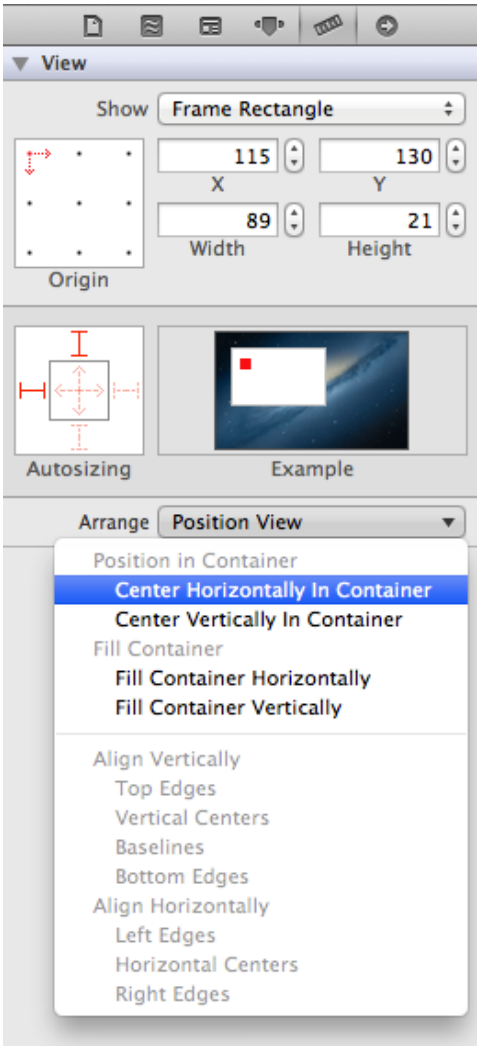


图 3.30 让 Label 控件在视图上居中显示

也可以根据 X、Y 坐标点设置 Label 的位置。这里只需要把 Label 居中就可以了。设置完毕，不要忘记按住快捷键 `command+S` 保存当前的修改。

(3) 因为程序运行在 iPhone 环境下，所以要确定选择的是 Simulator，如图 3.31 所示。除了 Simulator，还可以选择 device，不过只有在真机调试或者打包发布时才会用到。



图 3.31 选择目标环境

(4) 单击 Xcode 左上方的 Run 按钮，运行程序，模拟器就会自动启动运行，可以看到程序运行效果如图 3.32 所示。



图 3.32 模拟器运行效果

3.2.5 变量的关联

在视图层添加了一个 Label 控件，但还无法通过代码控制它，所以下一步需要在控制层利用一个变量与其关联起来。具体操作前，先介绍几个概念。

1. 输出口 (IBOutlet)

控制器使用 IBOutlet 来引用 nib 中的对象。本程序中，如果需要在代码对文本标签 (Hello World) 进行修改操作，这时就需要声明一个输入口，并将其指向该标签，这样

就可以通过输入口来操作该标签了。

IBOutlet 的定义如下：

```
#ifndef IBOutlet
#define IBOutlet           //宏定义
#endif
```

对编译器来说，IBOutlet 并未执行任何操作，它仅仅是告诉 Interface Builder，被 IBOutlet 修饰的实例变量将被连接到 nib 中的对象。

2. 操作 (IBAction)

IBAction 是用来修饰控制器类中的函数，该关键字告诉 Interface Builder，此方法是一个操作，并且可以被某个控件触发。

本章主要学习如何使用 IBOutlet 来关联变量，暂时用不到 IBAction 的操作。

(1) 添加变量

打开 ViewController.h，添加代码如下：

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

@interface ViewController : UIViewController
{
    IBOutlet UILabel *mylabel;           //定义一个变量
}
@end
```

首先定义一个变量，类型为 UILabel，在变量前添加 IBOutlet。按住组合键 command+S 保存。

(2) 查看 Files's Owner 内容

打开 ViewController.xib，右击 Files's Owner，在弹出的内容列表中可以看到 Outlets 下有了 myLabel，这就是刚创建的实体变量，如图 3.33 所示。

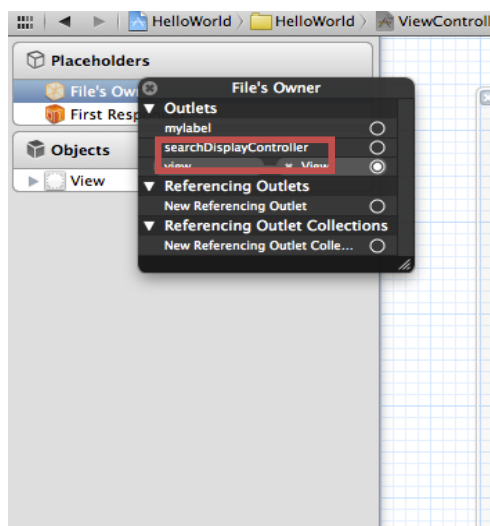


图 3.33 Files's Owner 内容信息

(3) 关联对象

鼠标左键按住 myLabel 右侧的小圆圈，拖曳到 Hello World 这个 Label 上。拖曳的时候，会出现一条蓝色的指引线。在 Files's Owner 下看到 myLabel 右侧多出了一个 Label，那说明关联成功了，如图 3.34 所示。

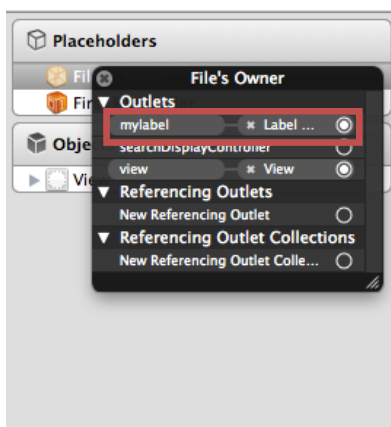


图 3.34 标签已经关联成功

3.2.6 通过代码设置属性

此时 IBOutlet 和控件已经成功关联了，相信读者已经掌握了通过 Interface Builder 设置 Label 属性的方式。接下来换个方式，利用代码来设置 Label 的属性。打开 ViewController.m，添加代码如下：

```
#import "ViewController.h"

@implementation ViewController

- (void) didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];           //内存警告函数
}

#pragma mark - View lifecycle

- (void) viewDidLoad
{
    [super viewDidLoad];

    mylabel.text=@"Say Hello to World";    //设置文本
    mylabel.font=[UIFont systemFontOfSize:20]; //设置字体
    [mylabel sizeToFit];                    //自适应大小
    mylabel.center=self.view.center;        //居中
}
```

```
- (void)viewDidUnload
{
    [super viewDidUnload];
    mylabel=nil;                                //取消关联
}
```

当 View 从 nib 文件初始化时会调用 `viewDidLoad`。这里对已经添加到 View 上的 `mylabel` 做出进一步修改。修改 `mylabel` 的文本内容为“Say Hello to World”，修改文本字体为 20 号。`sizeToFit` 的作用让视图自适应大小，`mylabel` 的长宽就会随着文本内容的多少自动改变。`center` 是一个坐标点，这里让 `mylabel` 位于整个窗口的中心。

 **注意：**通过某些其他的属性，还可以让 Label 的宽度保持不变，字体和行数随着文本内容的多少发生变化。

运行程序，显示效果如图 3.35 所示。



图 3.35 当前程序的运行效果

3.3 为应用添加图标

程序已经在模拟器上运行起来了，单击模拟器 Home 键，可以看到 Hello World 程序图标是空白的，如图 3.36 所示。



图 3.36 空白图标

程序需要一个图标，iPhone 将自动把图标的边缘圆角化并让它具有玻璃质感。iPhone 和 iPad 的图标根据不同的功能显示都有一定的尺寸要求。

注意：图标尽量选择标准 png 格式的图片。Xcode 会自动优化 png 图像，效率相比其他格式要高。

3.3.1 iPhone 图标介绍

iPhone 4 屏幕是高分辨率，所以采用的图片的大小也会和 3GS 等不同。iPhone 程序包需要以下尺寸的图片。

- ❑ 57×57 像素：用于 App Store 和在 iPhone 中应用程序图标显示，这个是必须要有的。
- ❑ 29×29 像素：用于 Spotlight 搜索，如图 3.37 所示。如果程序有设置功能，图片还会用于设置界面，如图 3.38 所示。



图 3.37 Spotlight 搜索界面



图 3.38 iPhone 系统设置界面

- ❑ 114×114 像素：用于 iPhone 4 的应用程序图标显示，像 iPhone 4、4S 等设备都采用了 Retina 高清屏幕，图标尽量选择这个尺寸。
- ❑ 58×58 像素：用于 iPhone 4 的 Spotlight 搜索和设置界面。

3.3.2 iPad 图标介绍

iPad 程序需要以下尺寸的图片。

- ❑ 72×72 像素：用于在 iPad 桌面应用程序图标显示，这是必须要有的。
- ❑ 50×50 像素：用于 iPad 中的 Spotlight 搜索。
- ❑ 29×29 像素：显示在 Setting 应用程序中。

3.3.3 图标的命名和设置

一个良好的应用程序应该考虑到图标的多样性。例如，一个在 iPhone 3GS 上显示正常的图标在 iPhone 4 这样的高分率设备上就会显得模糊且粗糙。所以，应该尽量为自己的程序准备各种尺寸的图标文件，图标文件的命名最好遵守苹果的规范，图标列表介绍如下。

- ❑ Icon.png：57×57 像素，iPhone 应用图标。
- ❑ Icon@2x.png：114×114 像素，iPhone Retina 显示屏应用图标。

- ❑ Icon-72.png: 72×72 像素, iPad 应用图标。
- ❑ Icon-72@2x.png: 144×144 像素, iPad Retina 显示屏应用图标。
- ❑ Icon-Small.png: 29×29 像素, iPhone 系统设置和搜索结果图标。
- ❑ Icon-Small@2x.png: 58×58 像素, iPhone Retina 显示屏系统设置和搜索结果图标。
- ❑ Icon-Small-50.png: 50×50 像素, iPad 系统设置和搜索结果图标。
- ❑ Icon-Small-50@2x.png: 100×100 像素, iPad Retina 显示屏系统设置和搜索结果图标。

图片文件的设置是在 info.plist 文件中, 就本程序来说, 需要在 Supporting Files 中的 HelloWorld-Info.plist 中进行修改。

下面分两步讲解下如何添加图标。读者会发现设置图标的操作几乎没有任何难度。

1. 添加图标文件

准备符合尺寸要求的图片, 把图片直接拖曳到工程文件夹下, 文件引用方式全部采取复制, 勾选 Copy items into destination group's folder(if needed)复选框, 如图 3.39 所示。

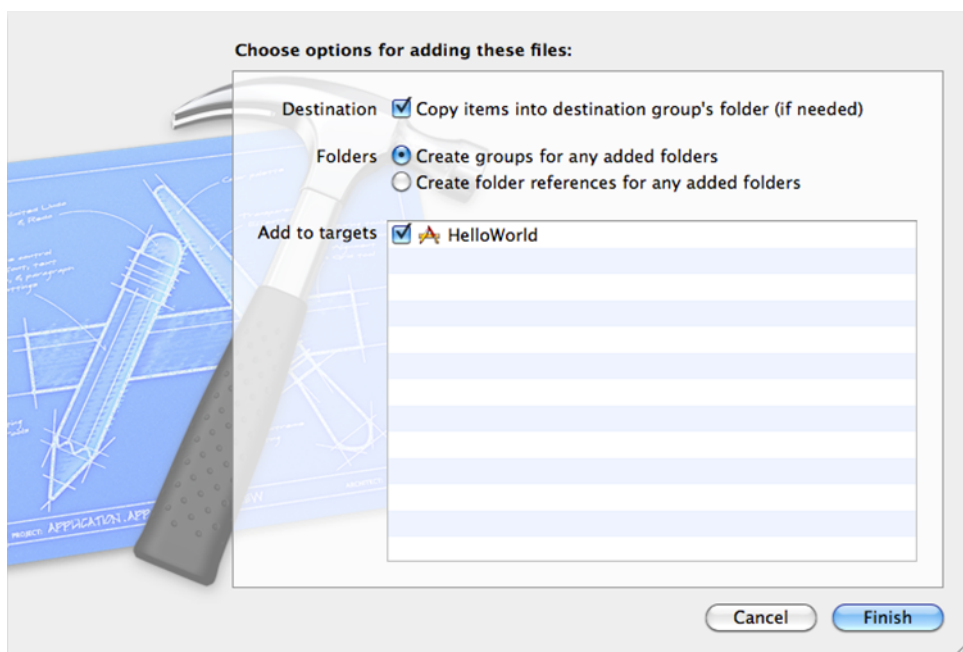


图 3.39 文件引用方式提示框

另外, Folders、Add to targets 选项保持默认。单击 Finish 按钮, 图片文件就被复制到工程中了。

2. 设置Icon files

打开 HelloWorld-Info.plist 文件, 右击任意一行, 添加 “Icon files” 文件字段, 如图

3.40 所示。

Key	Type	Value
Localization native development region	String	en
Bundle display name	String	\$(PRODUCT_NAME)
Executable file	String	\$(EXECUTABLE_NAME)
Icon files	Array	(2 items)
Bundle identifier	String	com.ios.{\$(PRODUCT_NAME:rfc1034identifier)}
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1.0
Application requires iPhone environment	Boolean	YES
Required device capabilities	Array	(1 item)
Supported interface orientations	Array	(3 items)

图 3.40 HelloWorld-Info.plist

选中 Icon files 字段，单击浮现出来的加号，添加子字段，如图 3.41 所示。

Executable file	String	\$(EXECUTABLE_NAME)
▼ Icon files	Array	(0 items)
Bundle identifier	String	com.ios.{\$(PRODUCT_NAME:rfc1034identifier)}
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)

图 3.41 Icon files 添加子字段

然后在 Icon files 里添加 6 个子字段，使其同时适应 iPhone 和 iPad 的图标要求，如图 3.42 所示。

Executable file	String	\$(EXECUTABLE_NAME)
▼ Icon files	Array	(6 items)
Item 0	String	Icon.png
Item 1	String	Icon@2x.png
Item 2	String	Icon-72.png
Item 3	String	Icon-Small-50.png
Item 4	String	Icon-Small.png
Item 5	String	Icon-Small@2x.png
Bundle identifier	String	com.ios.{\$(PRODUCT_NAME:rfc1034identifier)}
InfoDictionary version	String	6.0
Bundle name	String	PRODUCT_NAME

图 3.42 增加可能会用到的图标字段

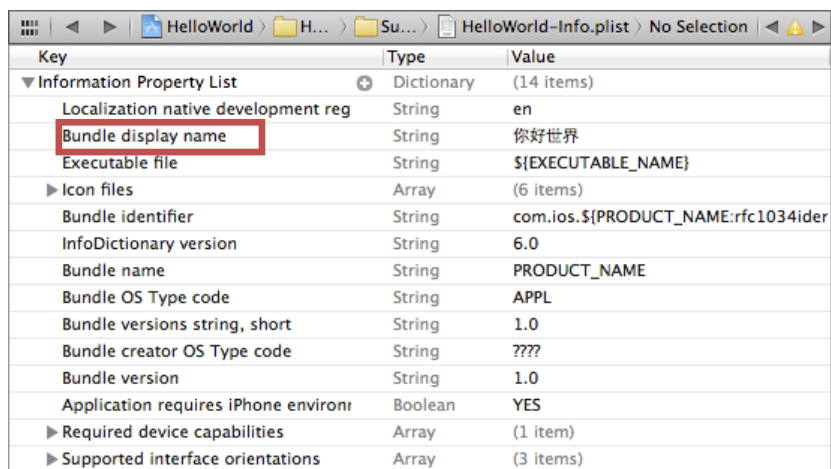
设置完成后，保存文件。以上的文件除了 57×57 和 72×72 像素的两个图标，其他的都不是必须的。如果程序中没有相关的图标，系统会自动把已有的图片进行拉伸以适应要求。

3.3.4 程序名称的设置

当前应用程序在 iPhone 模拟器中显示的名称是“HelloWorld”，默认的采用工程名

称作为程序名称。如果要自定义程序名称,需要在 HelloWorld-Info.plist 文件里修改设置。

打开 HelloWorld-Info.plist 文件,修改 Bundle display name 字段,我们设置程序显示名称为“你好世界”,如图 3.43 所示。



Key	Type	Value
▼ Information Property List	Dictionary	(14 items)
Localization native development region	String	en
Bundle display name	String	你好世界
Executable file	String	\$(EXECUTABLE_NAME)
▶ Icon files	Array	(6 items)
Bundle identifier	String	com.ios.\$(PRODUCT_NAME:rfc1034identifier)
InfoDictionary version	String	6.0
Bundle name	String	PRODUCT_NAME
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1.0
Application requires iPhone environment	Boolean	YES
▶ Required device capabilities	Array	(1 item)
▶ Supported interface orientations	Array	(3 items)

图 3.43 修改 Bundle display name 字段

保存设置,运行程序,效果如图 3.44 所示。



图 3.44 程序已经有了名称

3.4 小 结

本章带领读者初步了解了 iPhone 开发的工作原理。通过一个最简单的代码示例演示了开发工具 Xcode 的基本使用以及 iPhone 开发代码的编程方式。读者需要熟悉输入输出（IBOutlet）的使用方法，至于操作（IBAction）的内容将在下一章学到。

熟练使用 Interface Builder，可以帮助读者在未来的工作中提高 UI 界面的操作，把更多的精力放在业务逻辑上，所以读者可以在本章的程序基础上加强这方面的操作。本章的程序只设置了几个简单的属性，建议读者尝试改变其他的属性，或者添加代码，查看不同的效果。

3.5 习 题

【简答题】

1. 如何更改 Label 控件的字体大小？
2. 输出口可以关联任意的控件？
3. iPhone 4 的标准图标像素是多少？
4. 应用程序图标建议采用___格式的图标。

【实践题】

1. 创建一个基于 Single View Application 模板的应用程序，在视图上放置多个 Label 控件，尝试更改字体样式和行数。
2. 为程序准备不同大小的图标，然后根据本章介绍的方法显示图标。