



ИЗУЧАЕМ ПРОГРАММИРОВАНИЕ ДЛЯ **iPAD**

Практическое руководство по созданию приложений
для iPad с операционной системой iOS 5



КИРБИ ТЭРНЕР
ТОМ ХАРРИНГТОН

УДК 004.451iOS
ББК 32.973.26-018.2
T35

T35 Кирби Тэрнер, Том Харрингтон
Изучаем программирование для iPad. Пер. с англ. Слинкин А. А. – М.:
ДМК Пресс, 2013. – 808с.: ил.

ISBN 978-5-94074-844-1

В книге читатель пройдет весь путь создания приложения PhotoWheel, предназначенного для управления фотографиями, познакомившись при этом со всеми аспектами программирования iOS 5. PhotoWheel позволяет распределять любимые фотографии по альбомам, делиться ими с друзьями и родственниками, просматривать на экране телевизора.

В процессе разработки приложения вы изучите установку и настройку Xcode 4.2 на Mac; основы языка Objective-C и управления памятью с помощью механизма ARC; работу с Core Data и службой iCloud; использование новой функции Xcode – раскадровок – для создания функционального прототипа пользовательского интерфейса; создание жестов и интеграция с Core Animation; использование в приложении функций AirPrint, электронной почты и AirPlay; применение к изображениям фильтров и эффектов с помощью Core Image; диагностика и исправление ошибок с помощью Instruments; подготовка приложения к отправке в App Store.

Но самое важное – вы получите практический опыт разработки приложения для iPad. Если хотите освоить программирование для iPad, то эта книга – как раз то, что надо!

Original English language edition published by Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290. Copyright © 2012 Pearson Education, Inc. Russian-language edition copyright © 2012 by DMK Press. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-0-321-75040-2 (англ.)
ISBN 978-5-94074-844-1 (рус.)

© 2012 Pearson Education, Inc.
© Оформление, перевод на русский язык
ДМК Пресс, 2013



ОГЛАВЛЕНИЕ

Предисловие	16
Вступление	19
Чему я научусь?	20
В чем состоят особенности iPad?	21
Большой экран	21
Менее глубокие иерархии	22
Ориентация имеет значение	25
Мультисенсорность наступает	26
iPad ликвидирует разрыв между телефоном и компьютером	27
Здравствуй, iPad	28
Структура книги	28
На кого рассчитана эта книга	34
Получение исходного кода PhotoWheel	34
Благодарности	36
Благодарности Кирби Тэрнера	36
Благодарности Тома Харрингтона	36
Об авторах	38
ЧАСТЬ I. ПРИСТУПАЯ К РАБОТЕ	39
Глава 1. Первое приложение	40
Создание проекта «Здравствуй, мир».....	40
Вывод текста на экран	48
Поздоровайся с дядей.....	50
Резюме	55
Глава 2. Знакомство с Xcode	56
Интегрированная среда разработки	57
Рабочее пространство	57
Панель инструментов	58
Область навигации.....	59
Область редактора.....	60
Область вспомогательных средств.....	61
Область отладки.....	63
Настройки	64
Шрифты и цвета	64
Редактирование текста	65
Настройки клавиш.....	68
Автозавершение кода	70
Документация для разработчика	71

Редакторы	72
Параметры проекта	74
Схемы	76
Организатор	77
Другие инструменты Xcode	78
Резюме	79
Глава 3. Знакомство с Interface Builder	81
Interface Builder	81
Как работает IB?	82
Практическое занятие	83
Выбор и копирование объектов	87
Выравнивание объектов	87
Компоновочный прямоугольник	90
Изменение состояния	91
Связывание NIB-файла с кодом	94
Определение выхода в коде	95
Использование разделенного редактора Assistant	99
Раскадровки	101
Резюме	102
Глава 4. Знакомство с Objective-C	103
Что такое Objective-C?	103
Практикум по Objective-C	104
Пишем код	107
Объект	109
Класс	109
NSObject	112
Интерфейс	112
Переменные экземпляра	113
Объявленные свойства	114
Методы	117
Реализация	118
Директива synthesize	119
init	120
super	121
flip	121
Селекторы	122
Синтаксис с точками	124
Использование класса CoinTosser	124
Управление памятью	126
Автоматический подсчет ссылок	127
Резюме	128
Глава 5. Знакомство с Cocoa	129
Стек Cocoa	129
Подсистема Foundation	131
Типы данных	132
Классы коллекций	138

Служебные классы и функции	140
Другие анализаторы XML	143
Подсистема UIKit.....	144
Паттерны проектирования в Сосоа	153
Модель-Представление-Контроллер	154
Цель-действие	154
Резюме	155
Глава 6. Подготовка iPad	156
О портале подготовки для iOS	156
Процедура подготовки: краткий обзор	158
Что такое идентификатор устройства?	159
Что такое идентификатор приложения?	159
Что такое профиль подготовки к разработке?.....	161
Настройка компьютера, используемого для разработки.....	162
Запрос сертификата разработки.....	163
Отправка CSR на рассмотрение	166
Скачивание и установка сертификата.....	168
Настройка устройства	170
Использование устройства для разработки.....	170
Использование портала подготовки для iOS	173
Добавление идентификатора устройства	173
Добавление идентификатора приложения	175
Создание профиля подготовки к разработке	177
Скачивание профиля подготовки к разработке.....	179
Установка профиля подготовки к разработке	179
Резюме	181
Глава 7. Проектирование приложений	182
Определение приложения	182
Название приложения	183
Краткое описание приложения.....	184
Список функций	184
Целевая аудитория	186
Конкурирующие продукты.....	187
Пример хартии приложения	188
О проектировании пользовательского интерфейса	190
Прочитайте рекомендации по разработке человеко-машинного интерфейса.....	190
Учитывайте наличие сенсорного интерфейса.....	190
Проектируйте интерфейс для конкретного устройства.....	191
Пользователи работают с устройствами на базе iOS не так, как с настольными ПК или веб-сайтами.....	191
Преобразитесь в промышленного дизайнера.....	192
Метафоры	193
Звуковые эффекты.....	194
Настройка имеющихся элементов управления	195
Наймите дизайнера.....	196
Имитации	197

Что такое имитация?	197
Что имитировать	199
Инструментарий	200
Создание прототипа	205
Что такое прототип?	206
Как создается прототип	207
Резюме	208

ЧАСТЬ II. РАЗРАБОТКА ПРИЛОЖЕНИЯ

PhotoWheel..... 209

Глава 8. Создание приложения с интерфейсом

«основной-подробности» 210

Создание прототипа	210
Что такое контроллер разделенного представления?	212
Создание проекта	212
Работа с эмулятором	215
Познакомимся поближе	216
Структура проекта	217
Делегат приложения	218
Параметры запуска	223
Другие методы в классе UIApplicationDelegate	225
Знакомство с классом UISplitViewController	226
Инициализация делегата контроллера разделенного представления	229
Контроллер подробного представления	230
Контроллер основного представления	230
Резюме	232
Упражнения	232

Глава 9. Табличные представления 233

Начнем с начала	233
Присмотримся поближе	237
Класс UITableView	238
Класс UITableViewCell	239
Класс UITableViewDelegate	239
Класс UITableViewDataSource	239
Класс UITableViewController	239
Работа с табличным представлением	240
Простая модель	240
Отображение данных	242
Добавление данных	249
Редактирование данных	266
Удаление данных	271
Переупорядочение данных	273
Выбор данных	274
Резюме	277
Упражнения	277

Глава 10. Представления	278
Пользовательские представления	278
Контроллер или представление?	279
Круговое представление	280
Карусельное представление	288
Ячейка для кругового представления фотографий	295
Использование PhotoWheelViewCell	298
Резюме	300
Упражнения	301
Глава 11. Жесты	302
Что такое жесты?	302
Предопределенные жесты касания	303
Типы жестов	304
Как пользоваться распознавателями жестов	304
Нестандартные жесты касания	308
Создание распознавателя жеста раскручивания	309
Использование распознавателя жеста раскручивания	312
Резюме	317
Упражнения	317
Глава 12. Добавление фотографий	318
Два подхода	318
Подсистема Assets Library	318
Контроллер выбора изображения	320
Работа с контроллером выбора изображения	321
Списки действий	324
Использование UIImagePickerController	328
Сохранение в каталоге отснятых фотографий	333
Резюме	334
Упражнения	334
Глава 13. Сохранение данных	335
Модель данных	335
Фотографии	335
Фотоальбомы	336
Предусмотрительность полезна	336
Построение модели на основе списков свойств	336
Что такое список свойств?	337
Подготовка модели данных	337
Чтение и сохранение фотоальбомов	339
Добавление в альбом новых фотографий	344
Отображение фотографий в альбоме	349
Построение модели на основе Core Data	351
Что такое Core Data?	351
Управляемые объекты и описания сущностей	352
Контекст управляемого объекта	354
Постоянное хранилище и координатор постоянного хранилища	355

Добавление Core Data в PhotoWheelPrototype	355
Добавление средств Core Data	355
Конфигурирование стека Core Data	357
Использование Core Data в приложении PhotoWheel	361
Редактор модели Core Data	361
Добавление сущностей	362
Создание подклассов NSObject	365
Добавление собственного кода в модельные объекты	370
Чтение и сохранение фотоальбомов с помощью Core Data	375
Добавление в альбом новых фотографий с помощью Core Data	378
Отображений фотографий в альбоме с помощью Core Data	381
Работа с SQLite напрямую	382
Резюме	383
Упражнения	383
Глава 14. Раскадровки в Xcode	384
Что такое раскадровка?	384
Использование раскадровки	385
Сцены	386
Переходы	387
Включение раскадровки в PhotoWheel	388
Рабочее пространство	388
Добавление главной раскадровки	391
Задание UIStoryboardFile	393
Внесение изменений в файл AppDelegate	394
Добавление изображений	395
Значок приложения	395
Начальный контроллер представления	396
Еще одна сцена	399
Создание перехода	402
Резюме	404
Упражнения	404
Глава 15. Контроллеры представлений – добавим код	405
Реализация контроллера представления	405
Переход	409
Создание нестандартного перехода	410
Готовим сцену	410
Реализация нестандартного перехода	413
Перед компиляцией	417
Настройка перехода с выталкиванием	419
Контейнерный контроллер представления	422
Создание контейнерного контроллера представления	424
Добавление дочерних сцен	424
Добавление дочерних контроллеров представлений	427
Исправление перехода с заталкиванием	431
Резюме	432

Упражнения	432
Глава 16. Конструирование главного экрана.....	433
Повторное использование кода прототипа	434
Копирование файлов	435
Модель Core Data	436
Модификация класса WheelView	441
Отображение фотоальбомов	454
Реализация контроллера представления фотоальбомов	456
Настройка контекста управляемого объекта	463
Добавление фотоальбомов	465
Управление фотоальбомами	466
Выбор фотоальбома	466
Задание названия фотоальбома	471
Исправление внешнего вида панели инструментов	478
Удаление фотоальбома	479
Улучшаем миниатюру фотоальбома.....	482
Добавление фотографий	486
Отображение фотографий.....	492
Использование класса GridView.....	503
Создание класса ячейки сеточного представления	508
Резюме	512
Упражнения	512
Глава 17. Создание обозревателя фотографий....	513
Использование прокручиваемого представления.....	513
Пользовательский интерфейс обозревателя фотографий.....	523
Запуск обозревателя фотографий.....	523
Улучшение заталкивания и выталкивания.....	526
Добавление эффектов, связанных с обрамлением	534
Масштабирование.....	539
Удаление фотографии	547
Резюме	555
Упражнение.....	555
Глава 18. Поддержка поворота устройства.....	556
Как поддерживаются повороты	556
Поддерживаемые ориентации	557
Авторазмер	558
Специальная обработка поворотов	559
Поворот сцены с фотоальбомами	564
Поворот сцены с фотоальбомом	565
Шлифовка класса WheelView	567
Поворот представления About.....	568
Поворот обозревателя фотографий	568
Устранение оставшихся проблем	569
Исправление обозревателя фотографий.....	569
Исправление главного экрана	575
Стартовые изображения.....	578

Резюме	581
Упражнения	581
Глава 19. Печать с помощью AirPrint	582
Как работает механизм печати	582
Центр печати.....	583
Требования к печати.....	583
API печати	583
Эмулятор принтера	587
Резюме	588
Упражнения	589
Глава 20. Отправка электронной почты.....	590
Как это работает.....	590
Класс MFMailComposeViewController	592
Класс SendEmailController	593
Обзор класса SendEmailController	594
Использование SendEmailController	597
Резюме	603
Упражнения	603
Глава 21. Веб-службы.....	604
Основы.....	604
REST-службы в Cocoa.....	606
Flickr	607
Добавление работы с Flickr в PhotoWheel.....	609
Изменение сцены контроллера представления Flickr	610
Отображение сцены Flickr	612
Обертывание Flickr API	615
Асинхронная загрузка фотографий	622
Реализация класса FlickrViewController	629
И еще одно	639
Чего не хватает?	641
Резюме	641
Упражнения	642
Глава 22. Синхронизация с iCloud	643
Синхронизация – это просто	643
Основные понятия iCloud.....	644
Файловые координаторы и презентаторы	645
Классы UIDocument и UIManagedDocument	645
Глобально доступные постоянные хранилища	646
И снова о подготовке устройства	647
Конфигурирование идентификатора приложения	647
Подготовка для работы с iCloud.....	649
Настройка прав для работы с iCloud	650
Адаптация PhotoWheel к работе с iCloud	653
Не синхронизируйте больше, чем необходимо.....	653
Транзитные атрибуты в Core Data	654

Модификация PhotoWheel для работы с iCloud.....	655
Синхронизация фотографий с iCloud.....	660
Модификация координатора постоянного хранилища для работы с глобально доступными данными.....	661
Получение изменений из iCloud	666
Резюме	671
Упражнения.....	671
Глава 23. Создание слайд-шоу с помощью AirPlay... 672	
Варианты подключения внешнего дисплея	672
Требования к внешним дисплеям со стороны приложения	673
API для работы с внешним дисплеем	674
Добавление в PhotoWheel функции слайд-шоу.....	675
Изменение раскладки	676
Добавление показа слайд-шоу.....	677
Управление внешними дисплеями.....	681
Переход к следующей фотографии	685
Добавление элементов управления слайд-шоу	687
Изменение обозревателя фотографий	690
Замечание о тестировании и отладке	691
Добавление поддержки AirPlay	692
Использование AirPlay	694
Резюме	695
Упражнения.....	696
Глава 24. Визуальные эффекты с помощью Core Image..... 697	
Основные понятия Core Image	697
Введение в CIFilter	699
Типы фильтров	700
Использование CIFilter	701
Анализ изображений	702
Автоматическое улучшение качества изображения	703
Распознавание лиц	704
Добавление эффектов Core Image в PhotoWheel	705
Новые методы делегатов	706
Переменные экземпляра для управления фильтрами.....	707
Модификация пользовательского интерфейса.....	708
Создание эффектов CIFilter	715
Применение фильтров	720
Реализация автоматического улучшения качества	721
Реализация панорамирования лиц.....	721
Прочие необходимые методы	723
Резюме	725
Упражнения.....	725
ЧАСТЬ III. ЗАВЕРШАЮЩИЕ ШТРИХИ 727	
Глава 25. Отладка 728	

Понять, в чем проблема	728
Что не так?	728
Воспроизведение ошибки	729
Основные понятия отладки	729
Точки прерывания	730
Отладка в Xcode	731
Установка и управление точками прерывания	731
Настройка точек прерывания	732
Останов в точке прерывания	734
Просмотр значений переменных	736
Пример отладки: код вывода на внешний дисплей	739
Когда без NSLog не обойтись	743
Профилирование кода с помощью инструментальных средств	746
Пример профилирования: обновление элементов управления в пользовательском интерфейсе слайд-шоу	749
Резюме	753
Глава 26. Распространение приложения	754
Способы распространения	754
Сборка для разового распространения	755
Подготовка к разовому распространению	755
Подготовка сборки для разового распространения	756
Сборка для распространения через App Store	759
Подготовка для App Store	759
Подготовка сборки для распространения через App Store	760
Последующие шаги	762
Процедура размещения в App Store	763
Что делать, если Apple отклоняет заявку?	764
Информация о приложении для App Store	765
Ресурсы App Store	768
Использование iTunes Connect	769
Роли пользователей	769
Управление приложениями	770
Подача заявки	770
Что дальше?	773
Резюме	774
Глава 27. Послесловие	775
Что дальше?	776
Приложение А. Установка средств разработки	777
У участников программы есть привилегии	777
Присоединение к программе iOS Developer Program	778
Какую программу выбрать?	779
Что нужно для регистрации	781
Скачивание Xcode	783
Установка Xcode	783
Предметный указатель	785



ГЛАВА 1.

Первое приложение

Лучший способ чему-то научиться – практика, поэтому сразу же займемся написанием простого приложения для iPad. Это будет программа «Здравствуй, мир». Конечно, это заезженная тема, однако не переживайте – далее мы будем заниматься куда более сложными приложениями. Но ведь надо же на каком-то примере познакомиться с инструментарием и посмотреть, как пишется код.

Цель этой главы – получить самое первое понятие об инструментах, используемых при создании приложений для iPad. Если вы уже знакомы со средой Xcode, то можете сразу перейти к главе 4 «Знакомство с Objective-C» или к главе 6 «Подготовка iPad». Если же Xcode для вас внове, пожалуйста, продолжайте читать.

В этой главе мы опишем все шаги создания вашего первого приложения для iPad. Детальное описание Xcode будет приведено в двух последующих главах: «Знакомство с Xcode» и «Знакомство с Interface Builder».

Примечание. До начала работы вы должны установить на свой Mac Xcode и iOS SDK. Если они еще не установлены, обратитесь к приложению А «Установка средств разработки» и выполните приведенные в нем инструкции по подготовке Mac к программированию для iPad. Да, кстати, наличие Mac'a – обязательное условие.

Создание проекта «Здравствуй, мир»

Для начала запустим приложение Xcode. Если вы устанавливали его из App Store для Mac, то значок уже находится в Launchpad, как показано на рис. 1.1. В противном случае значок находится в области Dock. Щелкните по значку Xcode.

Примечание. Если Xcode нет ни в Launchpad, ни в Dock, его нужно добавить. В предположении, что при установке были оставлены параметры по умолчанию, Xcode находится в каталоге /Developer/Applications/ на жестком диске. Чтобы добавить Xcode в Dock, запустите его. Когда программа запустится, щелкните правой кнопкой мыши (или левой кнопкой, одновременно нажимая клавишу **Control**) по значку Xcode в Dock и выберите из контекстного меню пункт **Options > Keep in Dock** (Параметры > Оставить в Dock). В результате значок Xcode останется в Dock и после выхода из программы, так что в следующий раз запустить ее будет проще.



Рис. 1.1. Значок Xcode в Launchpad. Если Xcode скачана из App Store для Mac, то установщик помещает программу в группу Developer

Сразу после запуска на экране появляется окно Welcome to Xcode (рис. 1.2). В нем можно выполнить несколько действий: создать новый проект, подключиться к репозиторию исходного кода, перейти к руководству пользователя по работе с Xcode (*Xcode 4 User Guide*) или зайти на сайт разработчиков для Apple (developer.apple.com). Если раньше вы уже создавали или открывали какие-то проекты Xcode, то справа будет показан список недавних проектов. Чтобы открыть недавний проект, нужно выбрать его в списке и нажать кнопку **Open**.

В левом нижнем углу находится кнопка **Open Other...** (Открыть другой). Она позволяет открыть любой имеющийся на диске проект Xcode. Рядом с этой кнопкой расположен флажок, определяющий, нужно ли открывать окно Welcome to Xcode при запуске Xcode.

Примечание. Если вы раньше не работали с Xcode, то имеет смысл прочитать руководство пользователя. В нем описан весь набор инструментов, входящих в состав Xcode. Из этой книги вы многое узнаете о Xcode, но изучение официальной документации, поставляемой Apple, никогда не бывает лишним.

Мы собираемся создать новое приложение для iPad, поэтому щелкните по ссылке **Create a new Xcode project** (Создать новый проект Xcode). В результате откроется окно, показанное на рис. 1.3. Давайте познакомимся с ним, прежде чем двигаться дальше. В этом окне мы видим три секции:



Рис. 1.2. Окно Welcome to Xcode

В секции 1 задается целевая платформа: iOS или Mac OS X. Приложения для iPad работают на платформе iOS, так что про вариант Mac OS X можете пока забыть. Для платформы iOS можно выбрать один из двух типов проектов: Application (Приложение) или Framework & Library (Каркас и библиотека). Тип Application соответствует приложениям для iPhone и iPad. Тип Library соответствует статическим библиотекам, пока можете на него не обращать внимания.

Понятно, что для создания приложения «Здравствуй, мир» нужно выбрать тип **Application**. Сделайте это. После выбора типа содержимое секции 2 изменяется; теперь в ней отображается перечень доступных шаблонов для проекта выбранного типа. Шаблон служит для генерации начальных файлов, являющихся частью проекта Xcode.

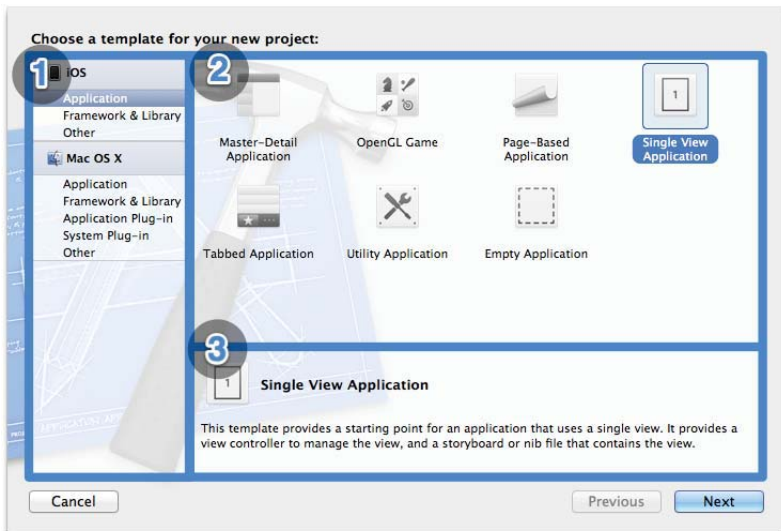


Рис. 1.3. Окно нового проекта в Xcode с помеченными секциями. Цифрой 1 обозначена секция для выбора целевой платформы, цифрой 2 – секция шаблона проекта, а цифрой 3 – секция с дополнительной информацией о шаблоне

Если вы развлекались со своим «айпадом», то, наверное, заметили, что существует несколько стандартных способов оформления приложения, или стилей. Перечисленные в секции 2 шаблоны позволяют выбрать конкретный стиль и тем самым ускорить процедуру создания приложения. Например, если вы хотите, чтобы ваше приложение было похоже на стандартное приложение Mail (Почта) для iPad, то выберите шаблон **Master-Detail Application**.

Шаблоны приложений

Если в качестве целевой платформы выбрана iOS, то Xcode предлагает следующие шаблоны.

- **Master-Detail Application:** выбирайте, если хотите, чтобы у приложения был интерфейс типа «основной-подробности», отображаемый с помощью контроллера разделенных представлений.
- **OpenGL Game:** выбирайте, если собираетесь разрабатывать игру с применением OpenGL ES. Шаблон подготавливает сцену OpenGL и таймер для анимации представления.
- **Page-Based Application:** выбирайте, если хотите создать приложение с книжным или журнальным интерфейсом, в котором используется контроллер страничного представления.

- **Single View Application:** выбирайте для приложений с единственным представлением.
- **Tabbed Application:** выбирайте для приложений с вкладками. Шаблон предоставляет контроллер панели вкладок и контроллер представления для первой вкладки.
- **Utility Application:** выбирайте для приложений с главным и альтернативным представлением.
- **Empty Application:** этот шаблон может служить отправной точкой для приложения любого вида. Выбирайте, если хотите начать программирование с нуля.

Приложение «Здравствуй, мир» будет содержать всего одно представление, поэтому выберите шаблон Single View Application. При этом изменится содержимое секции с дополнительной информацией о шаблоне. В ней приводится краткое описание выбранного шаблона.

Нажав кнопку **Next**, вы перейдете в окно параметров проекта (рис. 1.4). Параметры зависят от выбранного шаблона. Для любого шаблона нужно заполнить поля Product Name (Название продукта), Company Identifier (Идентификатор компании), Bundler Identifier (Идентификатор пакета) (это поле заполняется автоматически, исходя из идентификатора компании) и Device Family (Семейство устройств). Кроме них, для шаблона приложения можно задать флажки Use Storyboard (Использовать раскладовку), Use Automatic Reference Counting (Использовать автоматический подсчет ссылок), Use Core Data (Использовать Core Data) и Include Unit Tests (Включить автономные тесты).

Для приложения «Здравствуй, мир» введите в поле Product Name строку «Hello World». В поле Company Identifier введите свое имя или название компании в формате обращенного доменного имени. (Например, мое собственное имя записывается в виде com.kirbyturner, а название моей компании – в виде com.whitepeaksoftware). В главе 6 «Подготовка iPad» мы объясним, как связаны между собой идентификаторы компании и пакета и как из них образуется идентификатор приложения.

В списке Device Family выберите iPad. Всего iOS поддерживает три семейства устройств: iPad, iPhone и Universal. Выбирая семейство iPad, вы говорите, что приложение должно работать только на iPad. Семейство iPhone подразумевает приложение, ориентированное на iPhone, а семейство Universal означает, что приложение спроектировано для работы как на iPad, так и на iPhone.

В приложении «Здравствуй, мир» нам не понадобятся ни раскладовки, ни автономные тесты, поэтому не отмечайте эти флажки. Однако флажок Use Automatic Reference Counting все же отметьте.

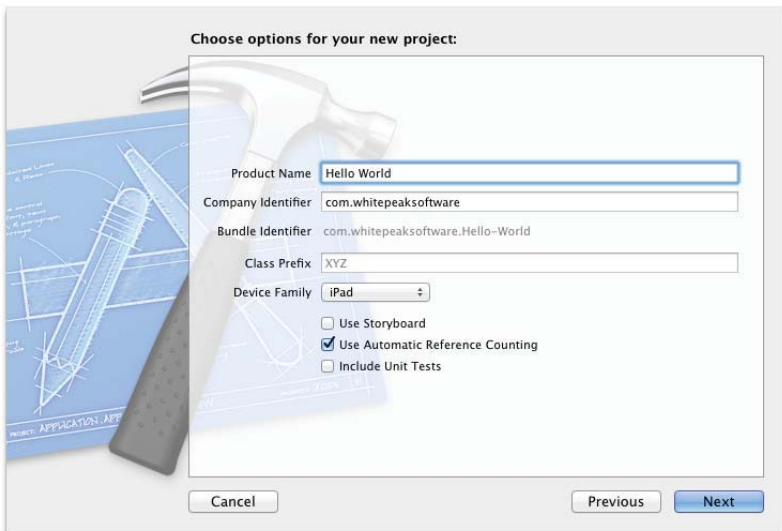


Рис. 1.4. Параметры проекта для шаблона *Single View Application*

Что означает этот параметр, мы объясним в разделе об управлении памятью главы 4 «Знакомство с Objective-C». Нажмите кнопку **Next**, выберите каталог для размещения проекта Xcode и нажмите кнопку **Create** (рис. 1.5).

Универсальные приложения

Приложения для iPhone могут работать на iPad, но запускаются при этом в эмуляторе iPhone. Они не могут воспользоваться преимуществами большого экрана, поэтому пользователю работать с ними неудобно. С другой стороны, универсальное приложение проектируется так, чтобы использовать всю доступную площадь экрана, будь то iPhone или iPad. Если запустить такое приложение на iPhone, то оно будет выглядеть так, будто писалось для iPhone, а если запустить его же на iPad, то оно будет вести себя, как приложение для iPad.

Универсальное приложение сочетает лучшее из обоих миров, поскольку замечательно выглядит на обоих устройствах. Но для этого требуются дополнительные усилия со стороны разработчика. Во многих отношениях создание универсального приложения сводится к разработке двух независимых приложений – одно для iPad, другое для iPhone – упакованных в единый двоичный файл.

Универсальные приложения рассчитаны и на iPad, и на iPhone. Но в этой книге мы будем говорить только о разработке приложений для iPad, чтобы не разбрасываться и не отвлекаться на дополнительные сложности, связанные с написанием универсальных приложений. Для начинающего разработчика на платформе iOS это было бы слишком много.

Примечание. Механизм раскадровок, появившийся в iOS 5, предназначен для визуального конструирования интерфейсов. Мы будем рассматривать его в главе 14.

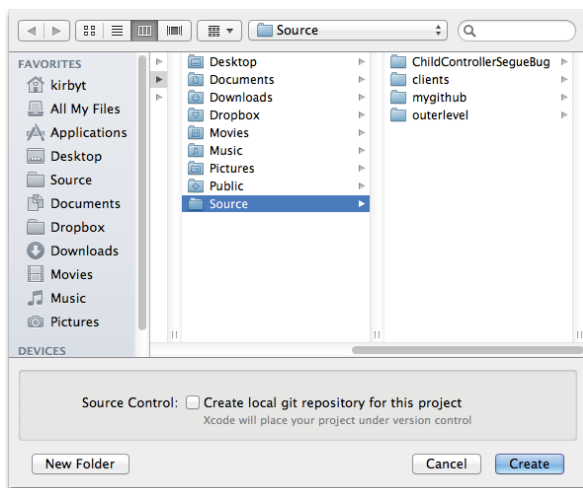


Рис. 1.5. Выбор места размещения проекта Xcode

Примечание. Я предпочитаю хранить весь исходный код в одном месте, поэтому создал в своем домашнем каталоге папку Source, куда помещаю все проекты Xcode. Так мне проще находить их впоследствии.

Примите поздравления! Вы только что создали свое первое приложение для iPad. Не верите? Тогда нажмите кнопку **Run** (рис. 1.6) или клавиши **⌘-R**. Проверьте, что в качестве активной схемы указана iPad Simulator (Эмулятор iPad). Если это не так, щелкните по схеме и измените ее.

После нажатия **Run** Xcode компилирует проект, собирает пакет приложения, устанавливает его в эмулятор iPad и запускает приложение в эмуляторе. На рис. 1.7 показано, что пока выводится лишь белый экран. Но знаете что? Ваше первое приложение для iPad вполне может выполнять функции фонарика!

Примечание. Иногда между запуском эмулятора и появлением в нем приложения проходит некоторое время. В течение этого времени в эмуляторе отображается черный экран. Это нормально и обычно происходит только при первом запуске приложения в эмуляторе.

Можете отправить свое приложение-фонарик в Apple для анализа. Впрочем, Apple, скорее всего, отвергнет его по причине недостаточной

функциональности. К тому же, приложение еще не закончено. Мы-то собирались написать программу «Здравствуй, мир», однако слов «Hello World» пока не наблюдается. Поэтому продолжим работу.

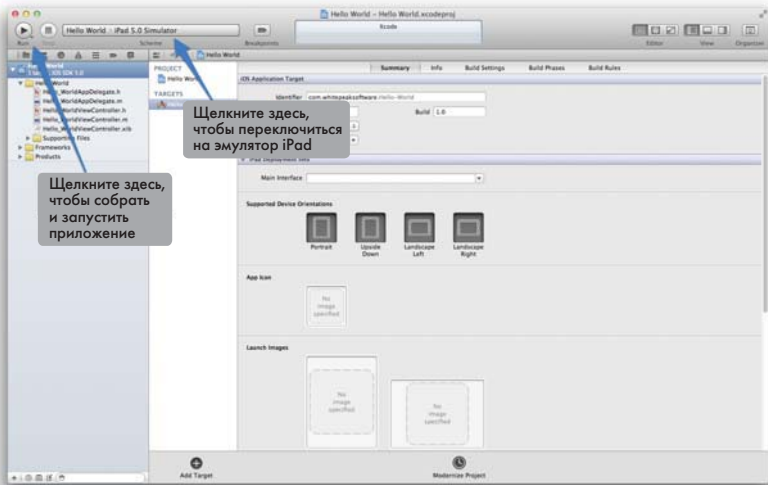


Рис. 1.6. Окно проекта Xcode для приложения «Здравствуй, мир»



Рис. 1.7. «Пустое» приложение с единственным представлением в эмуляторе iPad

Прежде всего, остановим работающее в эмуляторе приложение. Для этого нажмите кнопку **Stop** в левом верхнем углу Xcode или комбинацию клавиш **⌘-.**. Теперь можно перейти к модификации приложения.

Примечание. Если вы пользуетесь шаблоном проекта, то Xcode генерирует работоспособное приложение для iPad, хотя вы и не написали ни одной строчки кода. У меня всегда возникает теплое чувство, когда приложение запускается в первый раз, – быть может, это дань воспоминаниям о тех временах тридцатилетней давности, когда я, будучи подростком, только начинал писать программы. Как бы то ни было, после создания нового проекта в Xcode я первым делом всегда собираю и запускаю его. И балдею, когда вижу, как оно запускается.

Вывод текста на экран

Раз уж это приложение «Здравствуй, мир», то давайте выведем на экран слова «Hello World». Для этого можно было бы написать код, но проще воспользоваться программой Interface Builder, или просто IB. Это визуальный конструктор пользовательского интерфейса, встроенный в Xcode. Подробнее о IB мы будем говорить в главе 3, а сейчас просто опишем, как превратить пустое приложение в приложение «Здравствуй, мир» – тоже не особенно полезное.

Чтобы поместить на экран текст «Hello World», нужно отредактировать файл `ViewController.xib`. Файл с расширением `.xib` (произносится «зиб») – это XML-представление NIB-файла. NIB-файл (с расширением `.nib`) – предшественник XIB-файла. У XIB-файлов есть важное преимущество по сравнению с NIB-файлами: они текстовые, поэтому лучше приспособлены для работы с системами управления версиями. Однако в процессе сборки приложения XIB-файлы по-прежнему компилируются в формат NIB.

Зачем нужен NIB-файл? Interface Builder хранит в нем информацию об объектах интерфейса и связях между ними. Иначе говоря, NIB-файл – это представление объектов, отображаемых на экране. NIB-файлы создаются и редактируются в IB, а на этапе выполнения приложение читает их и строит пользовательский интерфейс.

Примечание. Разработчики для iOS часто называют XIB-файл NIB’ом, потому что он является ни чем иным, как текстовым представлением NIB-файла.

Историческая справка. Буква N в аббревиатуре NIB – наследие «эпохи NeXTSTEP», когда она обозначала файл со списком свойств в духе NeXT. А буквы IB говорят, что файл создан Interface Builder.

Начнем с открытия файла ViewController.xib, присутствующего в окне навигатора проекта. В результате содержимое в области редактора изменится. Теперь там отображается NIB-файл в конструкторе IB (рис. 1.8).

Примечание. В главе 3 «Знакомство с Interface Builder» рассматриваются все входящие в состав IB вспомогательные средства.

В составе IB имеется ряд вспомогательных средств для работы с NIB-файлами. Нажмите **Control-Option-⌘-3**, чтобы открыть библиотеку объектов (Object library). Эта библиотека содержит набор визуальных и не визуальных компонентов, применяемых для конструирования пользовательского интерфейса. В самом низу находится поле фильтра. Введите в него слово «Label» без кавычек – в списке останутся только объекты типа метки.

Перетащите объект Label на поверхность конструирования. В результате будет создан новый экземпляр класса UILabel, представляющего метку. Затем откройте инспектор атрибутов (Option-⌘-4). В его верхней части имеется свойство Text. Введите вместо значения по умолчанию (Label) строку «Hello World». Теперь окно Xcode должно выглядеть, как на рис. 1.8.

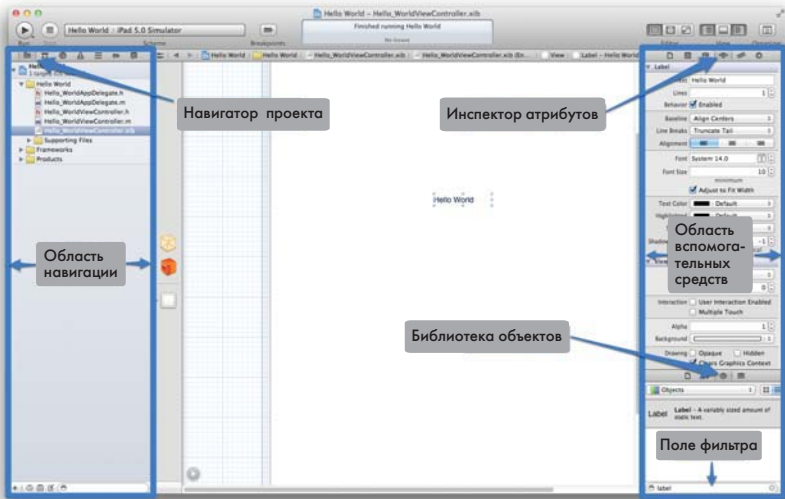


Рис. 1.8. Добавление строки «Hello World» в главное представление приложения

Примечание. Чтобы увидеть строку «Hello World», возможно, придется изменить размер метки. Для этого подведите курсор мыши к правому краю объекта Label. Курсор примет форму индикатора изменения размера. Нажмите кнопку мыши и тяните мышью вправо – ширина метки будет увеличиваться.

Соберите и запустите приложение в эмуляторе iPad. Мои поздравления! Вы написали свое первое приложение для iPad.

Примечание. Если вы не поняли, что за всем этим стоит, не расстраивайтесь. Ведь цель этой главы – всего лишь понять, как выглядит процедура программирования iPad. В последующих главах всё будет объяснено подробно, и вы сами не заметите, как порядок создания приложений для iPad войдет в вашу плоть и кровь.

Поздоровайся с дядей

Итак, возбуждение, охватившее вас после создания первого приложения для iPad, поутихло, и теперь можно немного расширить его функциональность. Мы сделаем так, что приложение будет не просто выводить строку «Hello World», а сначала спросит, как вас зовут, а потом выведет строку «Hello» и введенное имя. Это упражнение чуть сложнее и нам придется написать кое-какой код на языке Objective-C. Не пугайтесь, если раньше вам не доводилось работать с Objective-C. Сейчас я скажу, что нужно вводить, а в главе 4 мы познакомимся с Objective-C более подробно.

В жизни часто бывает, что задачу можно решить разными способами. Так обстоит дело и в случае программирования iPad – и это прекрасно. Именно из-за гибкости многие программисты предпочитают Xcode другим средствам разработки. Но чтобы изучить все ходы и выходы, нужно время, поэтому человек, только начинающий знакомиться с Xcode, может испытать разочарование.

Одна из целей этой книги – продемонстрировать различные подходы к решению задачи. Вооружившись знаниями, вы сможете сами решить, какой подход вам больше нравится. Например, можно заставить ИВ сгенерировать код на Objective-C, в котором будут объявляться объекты и действия, определенные в XIB-файле. Но о том, как это сделать, мы поговорим в последующих главах. А сейчас напишем код необходимый код самостоятельно.

Нам необходимо два визуальных элемента: один для ввода имени, другой – для вывода приветствия. Нужен еще и третий элемент – кнопка, которая говорит приложению, что пора вывести сообщение.

В NIB-файле определены объекты, составляющие пользовательский интерфейс, но не связи между этими объектами и исходным кодом. Такие связи вы должны установить сами.

Откройте файл *ViewController.h*, он присутствует в навигаторе проекта. После щелчка по нему в области редактора появится содержимое файла. Сотрите его и введите код, приведенный в листинге 1.1.

Листинг 1.1. Измененный файл *ViewController.h*

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

@interface ViewController : UIViewController

@property (nonatomic, strong) IBOutlet UILabel *helloLabel;
@property (nonatomic, strong) IBOutlet UITextField *nameField;

- (IBAction)displayHelloName:(id) sender;

@end
```

Теперь откройте файл *ViewController.m*. Замените сгенерированный код показанным в листинге 1.2.

Листинг 1.2. Измененный файл *ViewController.m*

```
#import "ViewController.h"

@implementation ViewController

@synthesize helloLabel;
@synthesize nameField;

- (IBAction)displayHelloName:(id) sender
{
    NSString *hello = [NSString stringWithFormat:@"Hello %@", [nameField text]];
    [helloLabel setText:hello];
}

@end
```

В листинге 1.1 мы сначала добавляем два свойства в класс *ViewController*. Оба помечены признаком *IBOutlet*, который говорит IB, что класс содержит ссылку на объект. Затем объявляется метод *displayHelloName:*. Помечающий его признак *IBAction* сообщает IB, что в определении класса имеется действие. Таким образом, мы определили интерфейс класса *ViewController*.

Что такое IBOutlet и IBAction?

IBOutlet и IBAction – это специальные признаки для Interface Builder, отсюда и префикс IB. Interface Builder использует их, чтобы связать объекты и действия с элементами пользовательского интерфейса.

IBOutlet служит для связывания ссылки на объект, определенный в коде на Objective-C, с экземпляром объекта в Interface Builder. Например, выше в этой главе мы поместили в представление метку. Эта метка на самом деле является объектом типа UILabel. (UILabel – имя класса метки.) Чтобы обратиться к этой метке из программы, нам нужна ссылка на объект. Далее в этой главе вы увидите, как связать объявленную в коде ссылку с отображаемым в IB экземпляром.

IBAction служит для связывания события, посылаемого объекту, с определенным в коде методом. Например, кнопка генерирует событие, когда пользователь касается ее пальцем. Это событие можно связать с действием IBAction, определенным в коде классе Objective-C.

В листинге 1.2 приведена реализация класса `ViewController`. В начале идет синтез свойств, объявленных в интерфейсе класса: `helloLabel` и `nameField`. Синтез свойств – это особенность компилятора Objective-C, который умеет самостоятельно генерировать методы доступа к свойствам. Подробнее об этом см. в главе 4 «Знакомство с Objective-C».

Далее следует реализация метода `displayHelloName:.` Это действие, которое вызывается, когда пользователь взаимодействует с приложением, а точнее касается кнопки. Метод создает локальную строковую переменную, содержащую строку «Hello », за которой следует введенное пользователем имя. Затем созданная строка отображается на экране в виде значения метки `helloLabel`.

Если вы сейчас запустите приложение, то не увидите никаких отличий от предыдущей версии. Код-то мы изменили, но интерфейс остался прежним, и связи между переменными-выходами (outlets) и действия пока не установлены.

Примечание. Такое отделение исходного кода (в данном случае контроллера) от пользовательского интерфейса (или представления) – характерная особенность паттерна проектирования Модель-Представление-Контроллер, который обсуждается в главе 5 «Знакомство с Сосоа».

Чтобы завершить приложение, мы должны изменить пользовательский интерфейс и соединить визуальные объекты со свойствами, определенными в классе контроллера. Снова откройте файл `ViewController.xib`. Дважды щелкните мышью по метке «Hello World» и измените ее текст на «What is your name?» (Как вас зовут?). Увеличьте размер метки, чтобы весь текст был виден на экране.