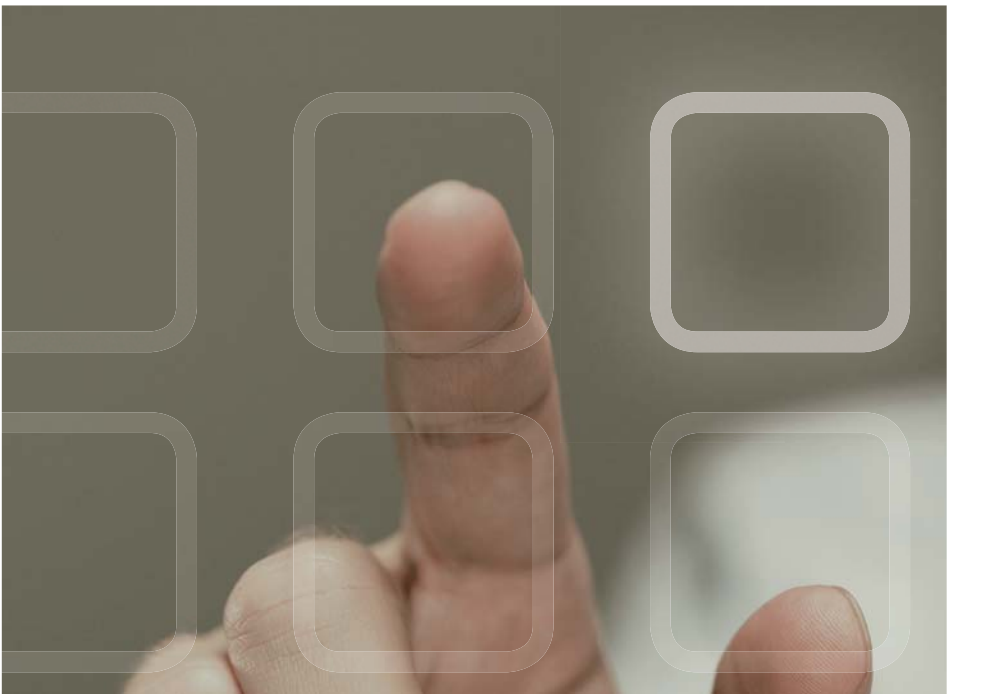
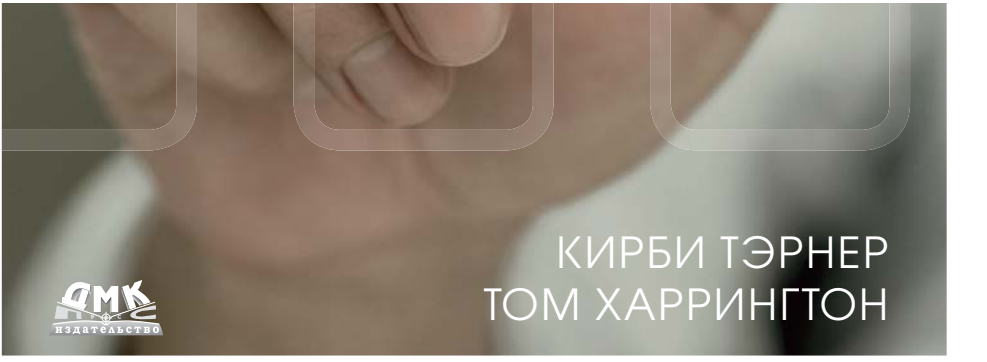


ИЗУЧАЕМ ПРОГРАММИРОВАНИЕ для iPAD

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



УДК 004.451iOS
ББК 32.973.26-018.2
Т35

Т35 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!

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

ОГЛАВЛЕНИЕ

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'а – 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.

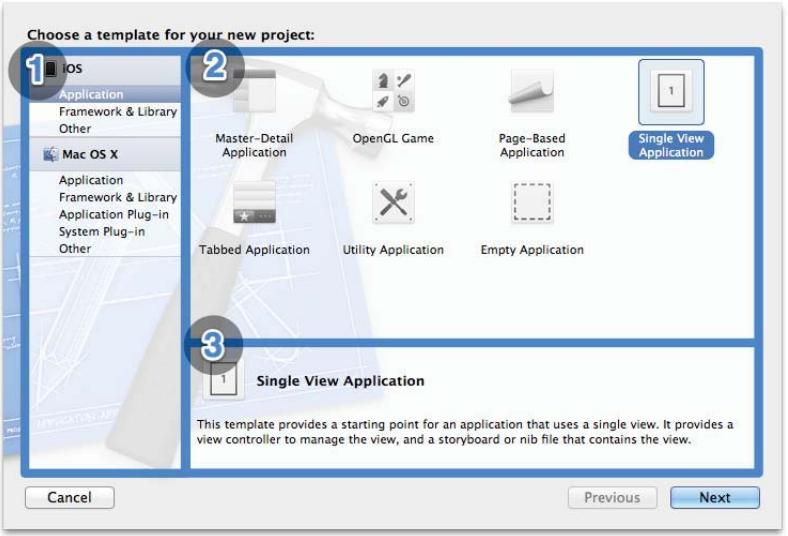


Рис. 1.3. Окно нового проекта в Xcode с помеченными секциями. Цифрой 1 обозначена секция для выбора целевой платформы, цифрой 2 – секция шаблона проекта, а цифрой 3 – секция с дополнительной информацией о шаблоне

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 программирования 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 раскадровку), 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 раскадровки, ни автономные тесты, поэтому не отмечайте эти флажки. Однако флажок Use Automatic Reference Counting все же отметьте.

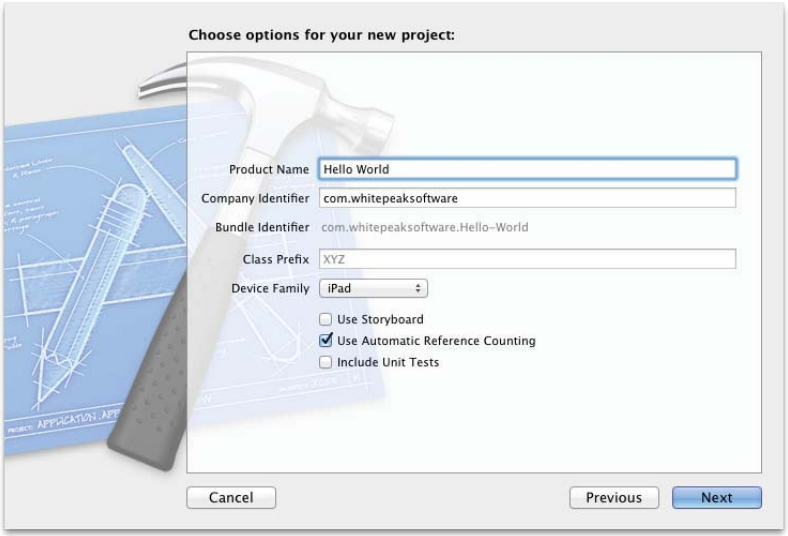


Рис. 1.4. Параметры проекта для шаблона *Single View Application*

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.

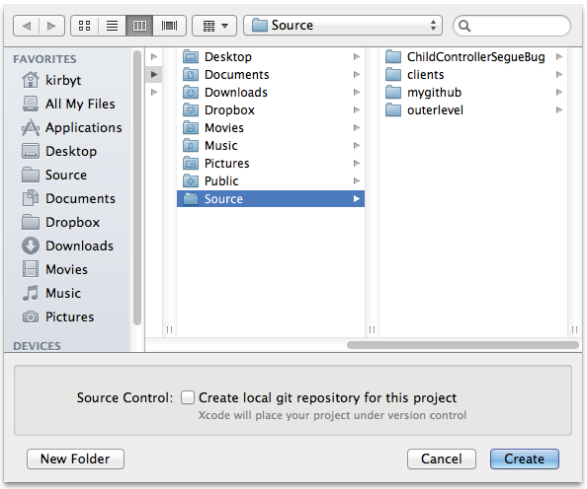


Рис. 1.5. Выбор места размещения проекта Xcode

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.

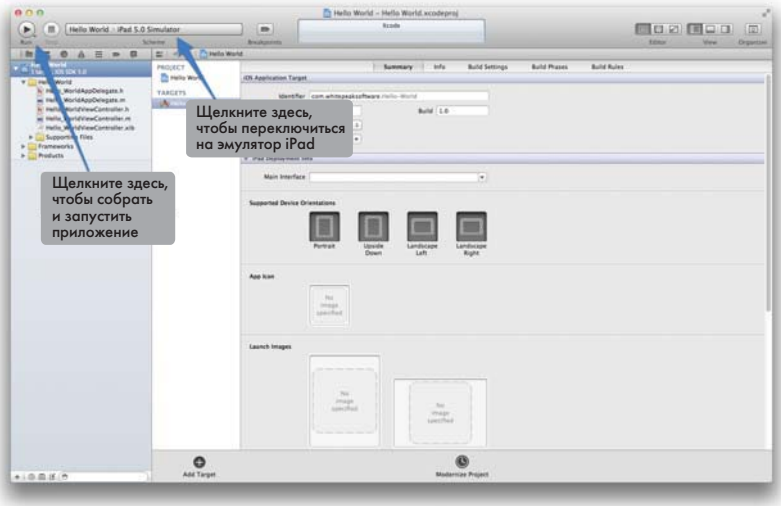


Рис. 1.6. Окно проекта Xcode для приложения «Здравствуй, мир»

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 невизуальных 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.

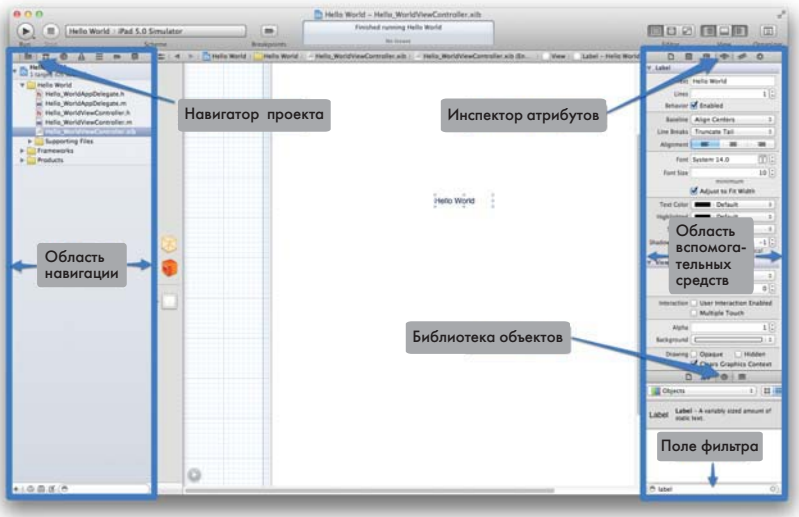


Рис. 1.8. Добавление строки «Hello World» в главное представление приложения

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 мышь 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 IB 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 к 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 Cocoa».

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.