



ИЗУЧАЕМ MonoTouch

Создание приложений
на платформе iOS с помощью C# и .NET



Майкл Блюстайн

Изучаем MonoTouch

Создание приложений
на платформе iOS
с помощью C# и .NET

Москва, 2023

Learning MonoTouch

A Hands-On Guide to Building iOS
Applications with C# and .NET

Michael Bluestein

◆ Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

Майкл Блюстайн

Изучаем MonoTouch

Создание приложений на платформе iOS
с помощью C# и .NET

2-е издание, электронное



Москва, 2023

УДК 004.451:004.43MonoTouch

ББК 32.973.26-018.2

Б71

Блюстайн, Майкл.

Б71 Изучаем MonoTouch. Создание приложений на платформе iOS с помощью C# и .NET / М. Блюстайн ; пер. с англ. — 2-е изд., эл. — 1 файл pdf : 337 с. — Москва : ДМК Пресс, 2023. — Систем. требования: Adobe Reader XI либо Adobe Digital Editions 4.5 ; экран 10". — Текст : электронный.

ISBN 978-5-89818-504-6

Вы прекрасно владеете C# и .NET и хотите выйти на огромный рынок iPhone/iPad? Теперь вы можете создавать великолепные приложения для iOS, не изучая язык Objective-C и средства разработки от Apple. Платформа MonoTouch позволяет применить уже имеющиеся навыки к разработке приложений, которые можно продавать через Apple App Store или развернуть в своей организации.

Книга Майкла Блюстайна поможет вам быстро освоить MonoTouch и извлечь максимум пользы из заложенного в ней гигантского потенциала.

В начале автор описывает, как настроить среду разработки MonoTouch, и объясняет, как MonoTouch абстрагирует iOS SDK, позволяя использовать родные классы iOS из C#. На простых примерах вы увидите, как пишутся приложения, и это станет фундаментом для решения более интересных задач. Шаг за шагом, на практических примерах вы будете осваивать все более сложные технологии. Автор даже демонстрирует применение в iOS-приложениях таких продвинутых средств .NET, как LINQ.

УДК 004.451:004.43MonoTouch

ББК 32.973.26-018.2

Электронное издание на основе печатного издания: Изучаем MonoTouch. Создание приложений на платформе iOS с помощью C# и .NET / М. Блюстайн ; пер. с англ. — Москва : ДМК Пресс, 2012. — 336 с. — ISBN 978-5-94074-824-3. — Текст : непосредственный.

В соответствии со ст. 1299 и 1301 ГК РФ при устранении ограничений, установленных техническими средствами защиты авторских прав, правообладатель вправе требовать от нарушителя возмещения убытков или выплаты компенсации.

ISBN 978-5-89818-504-6

© 2012 Pearson Education, Inc.

All rights reserved.

© Оформление, перевод на русский язык

ДМК Пресс, 2012

Посвящается Розе, Лилли и Джошуа



ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ	10
БЛАГОДАРНОСТИ	12
ОБ АВТОРЕ	13
ВВЕДЕНИЕ.....	14
ГЛАВА 1. Здравствуй, MonoTouch.....	18
Настройка среды	18
Установка iOS SDK и Apple Developer Tools	18
Установка MonoTouch	22
Создание приложения для MonoTouch	26
Создание пользовательского интерфейса	26
Добавление выходов.....	28
Разработка на устройстве	34
Подготовка к разработке	34
Работа с отладчиком MonoTouch	40
Резюме	42
ГЛАВА 2. iOS SDK сквозь призму MonoTouch....	43
Обзор iOS SDK.....	43
Сравнение Objective-C с MonoTouch на примере	46
Создание приложения в Xcode	46
Реализация той же функциональности в MonoTouch	51
Сравнение реализаций AppDelegate	53
Реализация UIAlertController в Xcode.....	56
Реализация UIAlertController в MonoTouch.....	59
Как работает MonoTouch	63
Управление памятью	66
Резюме	68
ГЛАВА 3. Представления и контроллеры	
представлений.....	69
Структурирование приложения MonoTouch	
в соответствии с паттерном MVC	69

Работа с представлениями и контроллерами в Interface Builder	71
Расширение функциональности контроллера и его представления	79
Работа с несколькими представлениями и контроллерами	83
Реализация подкласса UIView	88
Резюме	92
ГЛАВА 4. Основные классы iOS	93
Представления и элементы управления для построения пользовательского интерфейса	93
Элемент UISegmentedControl	93
Элемент UISlider	96
Элемент UISwitch	97
Элементы UIPageControl и UIScrollView	98
Класс UIActivityIndicatorView	101
Класс UIProgressView	102
Класс UIImageView	103
Класс UIWebView	104
Класс ADBannerView	106
Возможности устройства	110
Класс MFMailComposeViewController	110
Классы MPMediaPickerController и MPMusicPlayerController ...	112
Адресная книга	115
Класс UIImagePickerController	118
Резюме	123
ГЛАВА 5. Таблицы и навигация	124
Введение в классы UITableView и UITableViewController	124
Для чего применяются таблицы?	124
Отображение данных в UITableView	126
Части UITableViewCell и стили	132
Таблицы и навигация	135
Поддержка редактирования, встроенная в UITableView	143
Дополнительная настройка UITableView	151
Создание нестандартных ячеек	152
Добавление нескольких секций	156
Проект MonoTouch.Dialog	161
Резюме	162
ГЛАВА 6. Графика и анимация	163
Подсистема Core Graphics	163
Основные принципы Core Graphics	163
Рисование изображений	170

Рисование в PDF-файлы.....	174
Анимация	185
Анимация с помощью UIView	185
Подсистема Core Animation	189
Резюме	198
ГЛАВА 7. Подсистема Core Location	199
Введение в подсистему Core Location	199
Стандартная служба местоположения	202
Получение обновлений курса	212
Значительное изменение местоположения	214
Мониторинг региона	216
Фоновое определение местоположения.....	219
Резюме	222
ГЛАВА 8. Подсистема MapKit	223
Введение в MapKit	223
Добавление аннотаций.....	228
Наложения на карту	236
Резюме	243
ГЛАВА 9. Соединение с веб-службами	244
Соединение с REST-совместимыми веб-службами.....	244
Соединение по протоколу HTTP	244
Разбор результата в формате XML	247
Разбор результата в формате JSON	250
Потребление веб-служб на основе SOAP	251
Использование прокси-клиента в стиле .NET 2.0.....	252
WCF в MonoTouch	254
Использование HTTP-классов из CocoaTouch	255
Класс NSURLConnection и связанные с ним.....	255
Резюме	257
ГЛАВА 10. Сетевые возможности	259
Сетевое программирование с помощью GameKit	259
Основные сетевые классы в GameKit.....	259
Класс GKPeerPickerController	269
Голосовой чат на базе GameKit	273
Служба Bonjour	276
Резюме	286
ГЛАВА 11. Сохранение данных приложений ..	287
Демонстрационное приложение MTNotes	287
SQLite	292

Сериализация	299
Settings.bundle и NSUserDefaults.....	301
Резюме	309
ГЛАВА 12. Разработка приложений для iPad..	311
Перенос программ на iPad.....	311
Приложения для iPhone с удвоением пикселей	311
Универсальные приложения.....	312
Проектирование для iPad	315
Класс UISplitViewController.....	315
Класс UIPopoverController.....	326
Резюме	330
ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ	331



ПРЕДИСЛОВИЕ

О том, что команда, работающая над Mono, занимается чем-то, связанным с iPhone, я впервые узнал в 2008 году из презентации Мигеля де Иказа на конференции профессиональных разработчиков Microsoft (Microsoft Professional Developer Conference). Мигель рассказывал о том, как в процессе создания игровой платформы Unity3D они переносили .NET и C# на iPhone. Я был потрясен тем, что им удалось это сделать. К тому времени я уже программировал для iPhone на языке Objective-C в среде Xcode и, как и многие другие, считал эту комбинацию отличной платформой. Но учитывая, что я много лет работал с .NET и C#, мысль о том, что эти знания удастся применить и для iPhone, меня заинтриговала. Получилось бы сочетание двух моих любимых технологий, .NET и iPhone. Однако разработка игр меня никогда особенно не интересовала, поэтому в тот момент я не стал копать дальше.

На следующий год на встречу наших местных программистов на .NET заглянул Джозеф Хилл (Joseph Hill) и провел презентацию по проекту Mono в целом. К тому времени команда Mono уже начала разрабатывать продукт, которому суждено было стать MonoTouch. Джозеф упомянул, что скоро выйдет закрытая бета-версия, и предложил всем интересующимся обращаться к нему. Я подумал, как было бы здорово, если бы они смогли предложить хотя бы малую толику того, что есть в Objective-C или в .NET, например сборку мусора, потому что это открыло бы новые возможности для программирования iPhone, и, быть может, даже для повторного использования кода.

Я и вообразить себе не мог, что они предоставят все, что я мог делать на iPhone с помощью Objective-C, да еще и большую часть .NET впридачу. К тому же, оказалось, что команда разработчиков и сообщество MonoTouch состоят из очень толковых ребят, по-настоящему преданных своему делу. Успех MonoTouch зиждется на том, что вы получаете дополнительные инструменты для решения задач в системе iOS, не принося в жертву привычный пользователям интерфейс или возможности платформы. Я наслаждался каждым мгновением работы с MonoTouch и уверен, что вы испытаете такие же чувства.

Для кого предназначена эта книга

Эта книга ориентирована преимущественно на разработчиков, имеющих несколько лет опыта работы на платформе .NET/C#, но ничего или почти ничего не знающих о разработке для iPhone или Mac. Предполагается средний уровень знаний о C#/.NET. Однако в этой книге рассказывается о многих языково-независимых аспектах ядра iOS, поэтому и пишущие на языке Objective-C тоже найдут в ней кое-что полезное. Разработчики на C#/.NET из этой книги узнают, как применить накопленные знания к iPhone и создавать с помощью MonoTouch приложения для системы iOS.

О примерах кода

Все приведенные в книге примеры можно скачать из моего раздела на сайте Github по адресу <https://github.com/mikebluestein>.



БЛАГОДАРНОСТИ

Хочу поблагодарить свою жену и детей, которым приходилось мириться с моим невниманием во время работы над книгой. Без их поддержки я не смог бы закончить книгу. Каждому понятно, что в создании книги участвует много людей. Благодарю весь коллектив издательства Pearson за оказанную мне помощь и поддержку, особенно Чака Топорека (Chuck Toporek), Шерри Кэин (Sheri Cain), Оливию Басседжио (Olivia Basegio), Барта Рида (Bart Reed) и Энн Гебель (Anne Goebel), а также всех, кто оставался за сценой.

Хочу также сказать спасибо всем участникам команд Mono и MonoTouch за то, что создали такую потрясающую платформу, а особенно Джозефу Хиллу за слова ободрения и постоянную поддержку, которую он оказывал с момента моего приобщения к сообществу MonoTouch, а также Мигелю де Иказа за ту достойную восхищения работу, которую он делает для разработчиков по всему миру. Отдельное спасибо Джеффу Нортону (Geoff Norton), который возглавил создание MonoTouch и оказывает беспрецедентную поддержку и дает консультации всем нуждающимся в них. Сколько раз я засиживался допоздна над какой-нибудь трудной проблемой, а он всегда был готов помочь мне — да и любому другому — всем, чем мог. И еще спасибо Джеффу, Крису Харди (Chris Hardy) и Роберту Козаку (Robert Kozak) за техническое рецензирование. Мне было очень приятно работать над книгой о MonoTouch и надеюсь, что вы получите от чтения такое же удовольствие, как я от написания.



ОБ АВТОРЕ

Майкл Блюстайн работает с MonoTouch с момента выхода первой закрытой бета-версии и является активным членом сообщества MonoTouch. Написанные им с помощью MonoTouch приложения были одними из первых опубликованы на сайте Apple App Store. Он профессионально занимается разработкой программного обеспечения с начала 1990-х годов, ранее работал в должности главного инженера-программиста в компании Dassault Systimes Solidworks Corporation. Его блог, посвященный различным вопросам программирования, в том числе MonoTouch, размещен по адресу mikebluestein.wordpress.com.



ВВЕДЕНИЕ

Добро пожаловать в мир MonoTouch. Если вы работаете в .NET и интересуетесь созданием приложений для устройств под управлением iOS, то платформа MonoTouch – как раз то, что надо. В результате элегантного объединения языка Objective-C и каркаса CocoaTouch с C# и .NET получилась хорошо продуманная технология, работать с которой – одно удовольствие. MonoTouch можно использовать для публикации в магазине App Store и для развертывания на предприятии (при условии, что у вас есть необходимая лицензия). Существует даже бесплатная версия, включающая только эмулятор, так что приступить к изучению и экспериментам можно без дополнительных затрат. А студентам предлагается издание со скидкой.

MonoTouch позволяет создавать приложения с применением того же API, что имеется в Objective-C, но одновременно предлагает многие языковые средства и API, заимствованные из Mono, C# и .NET. В дополнение к элегантно абстрагированному механизму управления памятью в Objective-C, вы получаете сборку мусора, повторное использование кода, не относящегося к пользовательскому интерфейсу, ADO.NET-интерфейс к SQLite, веб-службы, Linq, универсальные типы – и это еще далеко не всё.

Несомненное достоинство MonoTouch заключается в том, что она базируется на технологиях Apple, добавляя к ним многочисленные новые возможности, которые помогают разрабатывать приложения. Необходимо также отметить команду разработчиков и сообщество MonoTouch. Принимая участие в форумах, списках рассылки и очень активном IRC-канале, вы можете получить помощь со стороны разработчиков и сообщества, обсудить свои идеи или просто потусоваться. Сообщество – одна из самых привлекательных сторон MonoTouch. Загляните в IRC, там можно встретить кого угодно – от новичков до создателей Mono и MonoTouch, которые прилагают все силы к тому, что разработчикам жилось лучше.

Как организована эта книга

Книга состоит из 12 глав, в которых рассказывается все, что необходимо знать об использовании MonoTouch для разработки в iOS.

Глава 1. Здравствуй, MonoTouch

В этой главе рассказывается о том, как настроить среду разработки, и описываются различные инструменты, применяемые в процессе разработки на платформе MonoTouch. Затем демонстрируется пример разработки простого приложения, сопровождаемый объяснением его внутреннего устройства. Завершается глава инструкциями по развертыванию приложения на устройстве и отладке с помощью отладчика MonoTouch.

Глава 2. iOS SDK сквозь призму MonoTouch

В этой главе объясняется, как MonoTouch абстрагирует iOS SDK, позволяя вести разработку на уровне стандартных классов C#. Сначала мы дадим обзор iOS SDK, представим простой пример, написанный на Objective-C, а затем покажем, как сделать то же самое на C#. На этом примере демонстрируется работа с переменными-выходами (outlet) и показывается, как использовать в программе на C# типичные паттерны разработки для iOS. Завершается глава обзором механизма управления памятью в Objective-C и сравнением его со сборкой мусора в MonoTouch; мы покажем, как MonoTouch берет на себя эти заботы, и объясним, в каких случаях следует принимать во внимание модель памяти, принятую в Objective-C, работая на C#.

Глава 3. Представления и контроллеры представлений

В этой главе показано, как структурировать приложение MonoTouch в соответствии с паттерном модель-представление-контроллер (MVC). Мы познакомимся с классами UIView и UIViewController и покажем, как работать с ними из программы и из конструктора Interface Builder, попутно продемонстрировав поддержку сенсорного управления и акселерометра.

Глава 4. Основные классы iOS

В этой главе объясняется, как пользоваться некоторыми наиболее важными для iOS классами. Мы представим многие входящие в состав iOS SDK представления и элементы управления, которые упрощают создание пользовательских интерфейсов, а также некоторые контроллеры, абстрагирующие такие средства, как адресная книга, доступ к камере, отправка электронной почты и воспроизведение музыки из библиотеки iPod.

Глава 5. Таблицы и навигация

Эта глава представляет собой введение в классы `UITableView` и `UITableViewController`. В ней описаны некоторые типичные случаи применения таблиц. Объяснение основного сценария использования `UITableViewController` сопровождается рассмотрением нескольких способов его модификации – визуальной и с целью повышения производительности – для более удобной работы. Здесь же мы познакомимся с классом `UINavigationController` и покажем, как использовать его в сочетании с `UITableViewController`.

Глава 6. Графика и анимация

В этой главе рассматриваются подсистемы графики и анимации – `Core Graphics` и `Core Animation` – и объясняется, как их использование в сочетании с `UIKit` образует основу большей части того, что вы видите в `iOS`.

Глава 7. Подсистема Core Location

В этой главе описывается подсистема `Core Location` и демонстрируется ее использование для получения данных о местоположении с помощью различных технологий позиционирования, в частности, триангуляции по базовым станциям, `Wi-Fi` и `GPS`. Далее мы рассматриваем некоторые более современные технологии, такие, как служба значительного изменения местоположения и мониторинг региона.

Глава 8. Подсистема MapKit

В этой главе обсуждается подсистема `MapKit`, в том числе элемент `MKMapViewControl`. Показано, как с ее помощью создавать в своем приложении интерактивные карты. Объясняется интеграция `MapKit` с `Core Location`, а также добавление к картам аннотаций и наложений.

Глава 9. Соединение с веб-службами

В этой главе показано, как в `MonoTouch` подключаться к веб-службам с помощью нескольких разных технологий. Обсуждается подключение к службам на базе протокола `SOAP`, `REST`-службам, а также форматы `JSON`, `XML`, `RSS` и `WCF`-службы. Помимо `.NET`, в этой главе рассматривается обращение к `HTTP`-стеку `CocoaTouch` из `MonoTouch`.

Глава 10. Сетевые возможности

В этой главе речь идет о предоставляемых подсистемой `GameKit` средствах для обнаружения служб и организации взаимодействия

в сети Bluetooth, в том числе о создании голосовой связи между устройствами. Затем описывается прямое использование технологии Bonjour для публикации и обнаружения служб, а также работа с хорошо знакомыми сетевыми технологиями .NET, например с классом TcpClient.

Глава 11. Сохранение данных приложений

В этой главе рассказывается о нескольких технологиях сохранения данных, предоставляемых MonoTouch, в том числе о поставщике ADO.NET для SQLite, сериализации в смысле .NET и классе NSUserDefaults. Здесь же начинается создание приложения, которое в следующей главе послужит иллюстрацией к разработке для iPad.

Глава 12. Разработка приложений для iPad

В этой главе рассматриваются некоторые классы, предназначенные специально для разработки приложений для iPad. Мы продолжим работу над приложением, начатым в предыдущей главе, и продемонстрируем, как превратить приложение для iPhone в универсальное, способное работать не только на iPhone и iPod Touch, но и на iPad.



ГЛАВА 1.

Здравствуй, MonoTouch

В этой главе мы познакомимся с основами разработки на платформе MonoTouch. Мы проделаем все шаги, необходимые для настройки среды разработки, а затем создадим первое приложение для MonoTouch и дадим обзор интегрированной среды MonoDevelop IDE и отладчика. Кроме того, мы подготовим устройство для развертывания приложения.

Настройка среды

Перед тем как приступить к разработке для MonoTouch, необходимо установить инструментарий. Сначала установим различные продукты Apple, а потом перейдем собственно к MonoTouch.

Установка iOS SDK и Apple Developer Tools

Добро пожаловать в мир разработки для iOS с использованием .NET. Уверен, вам не терпится поскорее начать кодировать, но сначала нужно подготовить среду на вашем новеньком сверкающем Mac'e. Вот и давайте настроим всё необходимое на эмуляторе и на iPhone.

Примечание. Приведенная ниже информация относится в равной мере к iPod Touch и iPad. Говоря в этой книге об «устройстве», я буду иметь в виду iPhone, iPod Touch или iPad. Если какой-то материал относится к конкретному устройству, то я специально оговорю это.

Прежде всего, потребуется зарегистрироваться в программе iOS Developer Program. Имея бесплатную учетную запись, вы получаете доступ к средствам разработки и комплекту iOS SDK для эмулятора iOS, а также к ресурсам, имеющимся в центре разработки iOS (iOS Developer Center). Чтобы иметь возможность развернуть приложение на устройстве и опубликовать его в магазине iTunes App Store,

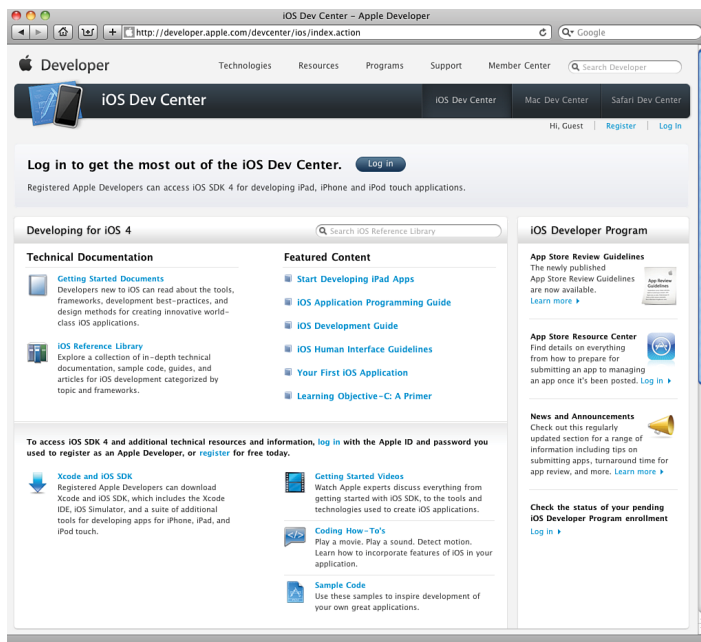


Рис. 1.1. Страница iOS Developer Center со ссылками для регистрации и входа

придется приобрести у компании Apple платную подписку. Ссылка на страницу регистрации размещена на главной странице сайта iOS Developer Center (рис. 1.1).

Зарегистрировавшись, зайдите в iOS Developer Center и скачайте iOS SDK вместе с Xcode. Это и есть полный комплект инструментов, необходимых для создания приложений для iPhone с применением только Objective-C. Да, вы не ошиблись. Платформа MonoTouch построена на базе инструментария Apple и добавляет всяческие «вкусности» от C# и .NET. MonoTouch берет все лучшее из обоих миров, не отказываясь от исходного дизайна, предложенного Apple. Поэтому вы получаете «родные» приложения, которые выглядят естественно для пользователей. Мы еще не раз будем возвращаться к этой теме, но сейчас все-таки продолжим готовить среду.

Итак, скачайте iOS SDK и Xcode (рис. 1.2). Для разработки под iOS нам потребуется операционная система Mac OS X Snow Leopard. Скачав и установив Apple iOS SDK, проверьте, что все работает. Не пугайтесь, если эта процедура покажется вам непонятной; мы всего



Рис. 1.2. Страница Apple Developer Center со ссылками для скачивания

лишь хотим удостовериться, что установка прошла нормально. Откройте Xcode, интегрированную среду разработки от Apple, для чего введите в терминале команду `open -n /Developer/Applications/xcode.app`. Когда Xcode загрузится, создайте новое приложение для iPhone, выбрав из меню команду **File > New Project** (**Файл > Создать проект**). В разделе iOS слева в окне New Project (Новый проект) выберите вкладку Application (Приложение) и затем шаблон View-based Application (Приложение на базе представления) (рис. 1.3). Назовите проект **LMT1-1** и нажмите кнопку Save (Сохранить).

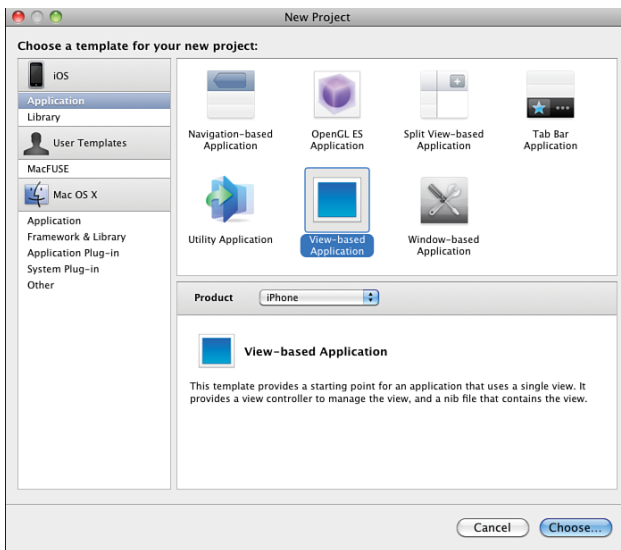


Рис. 1.3. Создание нового проекта в Xcode

Когда Xcode IDE загрузит шаблон проекта, выберите на верхней панели инструментов команду Build and Go (Построить и запустить). Если все пройдет нормально, то на экране появится эмулятор iPhone, в который будет загружено приложение, не содержащее ничего, кроме пустого экрана (рис. 1.4).

Закройте эмулятор и вернитесь в Xcode. Раскройте узел NIB Files в дереве в окне Groups & Files (Группы и файлы) и дважды щелкните по узлу LMT1_1ViewController.xib (рис. 1.5). В результате появится конструктор пользовательских интерфейсов Interface Builder, в котором отображается содержимое файла LMT1_1ViewController.xib (рис. 1.6). Если все это получилось, значит, iOS SDK и средства разработки Apple установлены правильно и можно переходить к установке MonoTouch.



Рис. 1.4. Созданное в Xcode приложение для iPhone с пустым окном

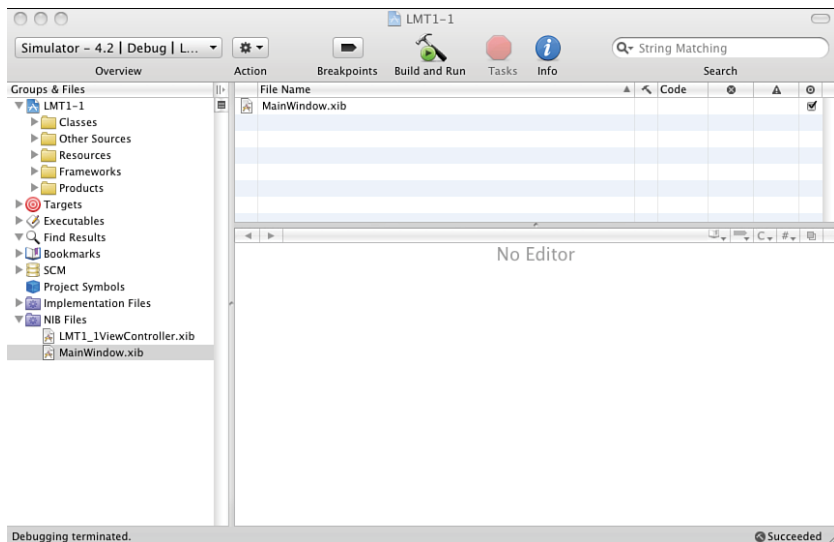


Рис. 1.5. Дерево Groups & Files с раскрытым узлом NIB Files



Рис. 1.6. Отображение MainWindow.xib в конструкторе Interface Builder

Установка MonoTouch

MonoTouch построена на базе Mono, открытой реализации .NET для Linux и Mac OS X. Прежде чем использовать MonoTouch, необходимо установить последнюю версию Mono для OS X (в настоящий момент 2.10.6), которую можно скачать со страницы www.go-mono.com/mono-downloads/download.html. В разделе Select Platform (Выбор платформы) выберите Mac OS X, а затем Intel Mac (рис 1.7)¹.

Установив Mono, можно приступить к установке MonoTouch SDK. Для разработки приложений, которые можно будет развернуть на ус-

¹ В настоящее время сайт выглядит несколько иначе, но в целом процедура осталась неизменной. *Прим. перев.*

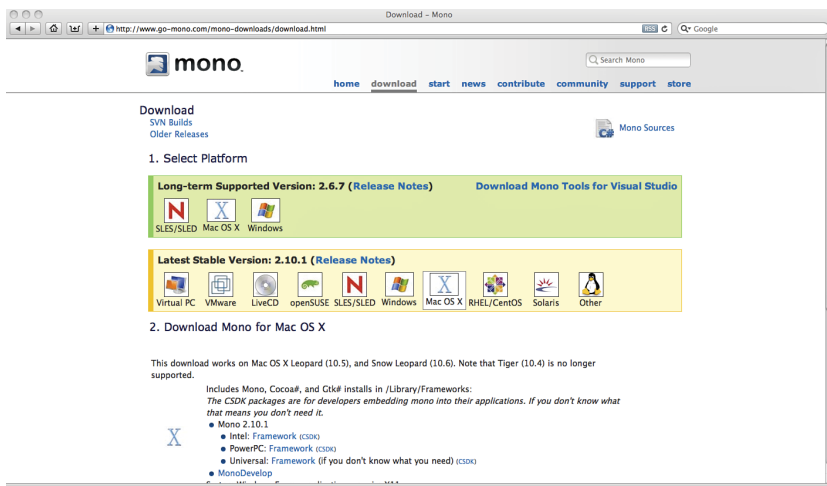


Рис. 1.7. Страница загрузки Mono для OS X

тройстве и опубликовать в iTunes App Store, необходимо сначала заплатить Apple за учетную запись разработчика для iOS (iOS Developer Account), а затем приобрести коммерческую лицензию на MonoTouch у Novell. Если вы собираетесь разрабатывать только для эмулятора, то можете скачать бесплатную пробную версию MonoTouch со страницы <http://monotouch.net/DownloadTrial>. Если вы уже приобрели у Apple подписку iOS Developer Subscription и хотите купить лицензию на MonoTouch, то зайдите в магазин MonoTouch по адресу <http://monotouch.net/Store>, где сможете сравнить условия различных лицензий и выбрать подходящую. После оплаты в браузере будет показана квитанция с кодом активации.

Примечание. Сохраните квитанцию, потому что код активации вам еще понадобится.

От компании Novell вы получите по электронной почте письмо со ссылкой для скачивания лицензированной версии MonoTouch SDK. Скачайте установщик пакета и установите MonoTouch. После выполнения всех инструкций, введите в диалоговом окне активации MonoTouch код активации, полученный ранее в квитанции. Чтобы убедиться, что установка MonoTouch прошла успешно, откройте окно терминала и введите команду `/Developer/MonoTouch/usr/bin/mtouch -version`, которая напечатает номер приобретенной вами версии MonoTouch. Теперь можно устанавливать MonoDevelop.

Примечание. Последующие версии продуктов на основе Mono для iOS разрабатываются под маркой компании Xamarin, основанной командой, которая создала Mono и MonoTouch. Они сохраняют обратную совместимость с MonoTouch. Дополнительные сведения можно найти на сайте xamarin.com.

MonoDevelop – это интегрированная среда, предназначенная в числе прочего и для разработки программ для MonoTouch. Она хорошо интегрирована с инструментами Apple и в полном объеме поддерживает отладку, о чем мы еще поговорим ниже. Для скачивания версии MonoDevelop, включающей средства для разработки на платформе MonoTouch, зайдите на страницу http://monodevelop.com/Download/Mac_MonoTouch, щелкните по соответствующей ссылке и запустите установщик. Когда установка MonoDevelop завершится, откройте программу Finder, перейдите в каталог Applications и дважды щелкните по значку MonoDevelop, чтобы запустить приложение (рис. 1.8).

Примечание. Можно одновременно запустить несколько экземпляров MonoDevelop, для чего достаточно ввести в терминале команду `open -n /Applications/Monodevelop.app`.

Запустив MonoDevelop, проверим, все ли нормально, – примерно так же, как выше проделали для Xcode. Выполните следующие действия:

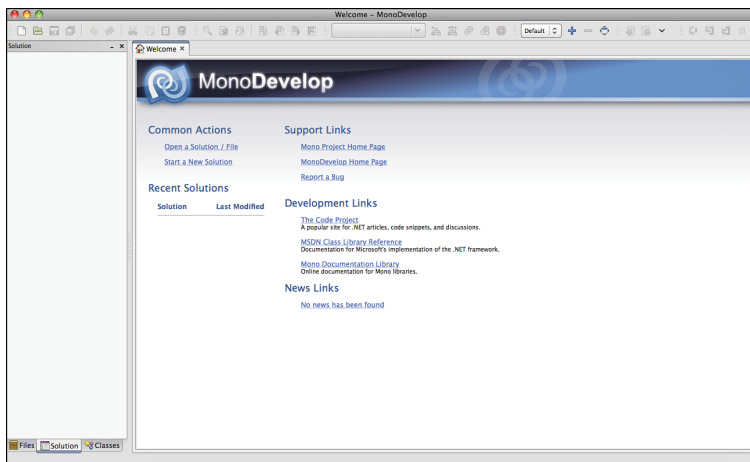


Рис. 1.8. MonoDevelop IDE

1. Создайте новое решение, выбрав из меню команду **File > New > Solution** (Файл > Создать > Решение).

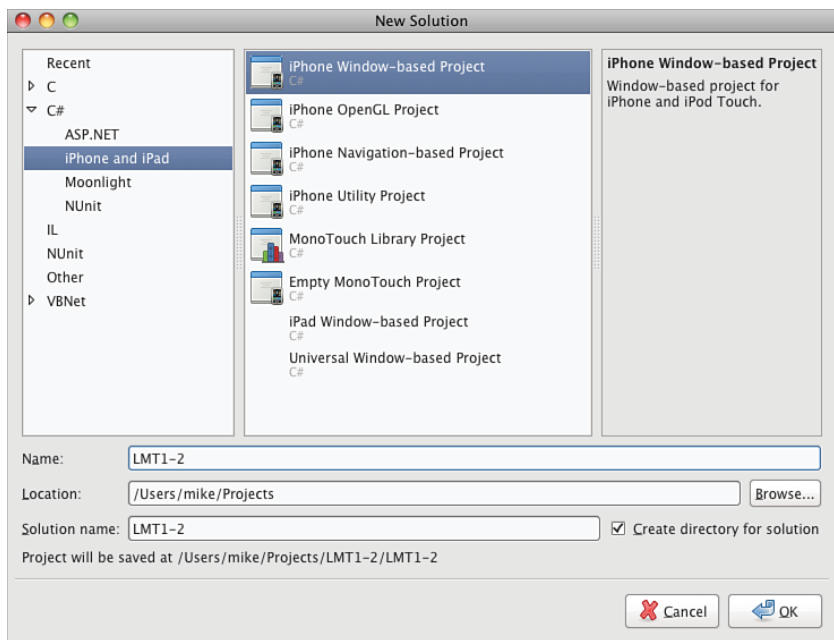


Рис. 1.9. Диалоговое окно New Solution в MonoDevelop

2. В диалоговом окне New Solution (рис. 1.9) выберите в дереве категорий узел **C# > iPhone and iPad**, а затем шаблон iPhone Window-based Project.
3. Назовите решение **LMT1-2**, нажмите кнопку Forward (Далее), а затем ОК (флажки параметров проекта (Project Features) можно не отмечать, потому что к MonoTouch они неприменимы).
4. Выберите из меню команду **Run > Run**, которая заставляет MonoDevelop построить приложение, установить его в эмулятор и запустить эмулятор, открыв в нем данное приложение, которое не содержит ничего, кроме пустого экрана.
5. Закройте эмулятор и вернитесь в MonoDevelop; далее мы проверим интеграцию с Interface Builder.

Примечание. Если после первого запуска MonoDevelop вы увидите сообщение о наличии обновлений, установите их.

MonoDevelop прекрасно интегрируется с Interface Builder и позволяет создавать приложения, которые не только по внешнему об-

лику неотличимы от разработанных с помощью Xcode и Objective-C, но в действительности даже пользуются теми же самыми элементами управления. Ниже в этой главе мы еще вернемся к этой теме, а пока проверим лишь, что все работает правильно. На вкладке решения раскройте решение и проект. Среди прочего, в проекте должен быть файл `MainWindow.xib`. Двойной щелчок по нему должен привести к запуску Interface Builder с пустым окном.

Если все прошло нормально, значит, MonoTouch на вашем компьютере работает. Нам еще предстоит кое-что сделать для развертывания приложения на устройстве, но этим мы займемся после того, как напишем что-нибудь более полезное, чем пустое окно. Прямо сейчас и напишем.

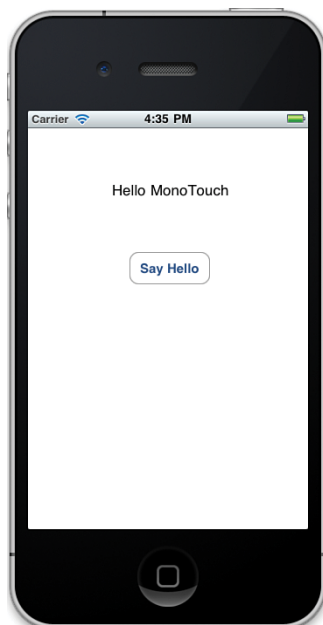


Рис. 1.10. Готовое приложение Hello MonoTouch

Создание приложения для MonoTouch

В этом разделе мы напишем простое приложение, содержащее метку и кнопку, причем касание кнопки будет приводить к изменению текста метки. Готовое приложение показано на рис. 1.10.

Создание пользовательского интерфейса

Начнем с создания пользовательского интерфейса. Вернитесь в только что созданное приложение с пустым окном и дважды щелкните по файлу `MainWindow.xib`, чтобы открыть Interface Builder, если вы успели его закрыть. Мы уже говорили, что Interface Builder – это предлагаемый Apple конструктор пользовательского интерфейса приложения. Если раньше вам доводилось писать для Windows или ASP.NET, то скажем, что это аналог средств проектирования, имеющихся в Visual Studio, хотя функционально между ними имеются различия. Interface Build-

der манипулирует **xib-файлом**, который представляет собой XML-файл, инкапсулирующий практически все объекты, необходимые для описания интерфейса. На этапе построения приложения **xib-файл** преобразуется в двоичный файл с расширением **.nib**. По существу, **xib-файл** – это сериализованный граф объектов, из которых состоит пользовательский интерфейс. Для его создания выполните следующие действия:

1. В Interface Builder (IB) выберите вкладку **Objects** (Объекты) окна **Library** (Библиотека) (рис. 1.11).
2. Прокрутите окно примерно до середины, пока не увидите элемент управления **Label**, которому соответствует тип **UILabel**. Перетащите метку на поверхность конструктора и расположите ее в верхней части окна.
3. Теперь добавим кнопку, которая будет изменять текст метки. Найдите в окне **Library** объект **Round Rect Button** и перетащите его на поверхность конструктора, расположив под меткой.
4. Дважды щелкнув по добавленной кнопке, переведите ее в режим редактирования, где сможете изменить текст на «Say Hello».
5. Сохраните **xib-файл**, вернитесь в **MonoDeveloper** и запустите приложение.

Когда приложение запустится в эмуляторе, вы увидите метку и кнопку. Касание кнопки (точнее, нажатие, поскольку вы работаете с эмулятором) изменяет цвет фона на синий, но больше ничего не происходит. Необходимо написать код, который будет выполняться при касании кнопки и изменять текст метки. Пристегните ремни – взлетаем!

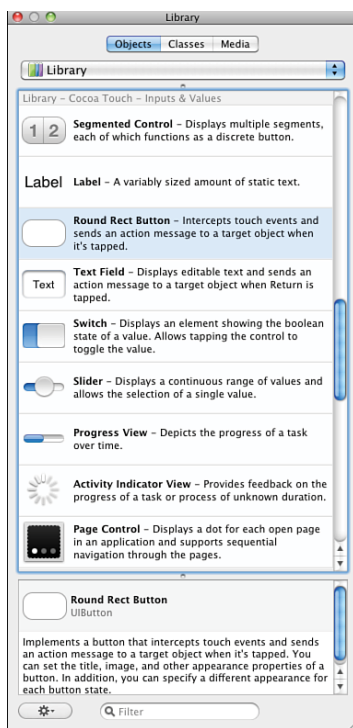


Рис. 1.11. Окно **Library** в **Interface Builder**

Добавление выходов

IB не генерирует код класса реализации, как инструменты Microsoft. Для организации взаимодействия между представлениями, определенными в IB и кодом служат **переменные-выходы** (outlet), а по сути дела просто ссылки из кода на объекты, определенные в xib-файле. Для их создания нужно проделать кое-какие действия вручную, но к этому быстро привыкаешь. К тому же, использование возможностей языка C# (точнее, частичных классов) и общий дизайн MonoTouch позволяют сократить объем ручной настройки по сравнению с Objective-C и Xcode. Но для начала вернитесь в IB.

В IB нам предстоит создать два выхода: один для кнопки (чтобы можно было подписаться на ее события) и другой для метки (чтобы можно было программно изменять ее текст). Начнем с метки. Перейдите в окне Library на вкладку Classes и выберите в списке пункт AppDelegate. В нижней части окна Library, где теперь отображается слово AppDelegate, выберите в списке пункт Outlets (рис. 1.12).

AppDelegate – это класс, объекты которого получают уведомления о том, что в объекте UIApplication что-то произошло, например, закончился этап загрузки приложения (можете считать это обратными вызовами, хотя аналогия не полная). Термин «делегат» здесь обозначает не те делегаты, к которым вы привыкли в .NET, а средство языка Objective-C, которое сплошь и рядом используется в iOS SDK. Помимо «родных» делегатов C#, MonoTouch поддерживает делегаты Objective-C, абстрагируя их так, что они становятся доступны из C#. Если сейчас вам не всё понятно, не переживайте, – мы подробно обсудим этот вопрос в следующей главе. А пока просто помните, что AppDelegate – это класс, в котором будут реализованы программные об-

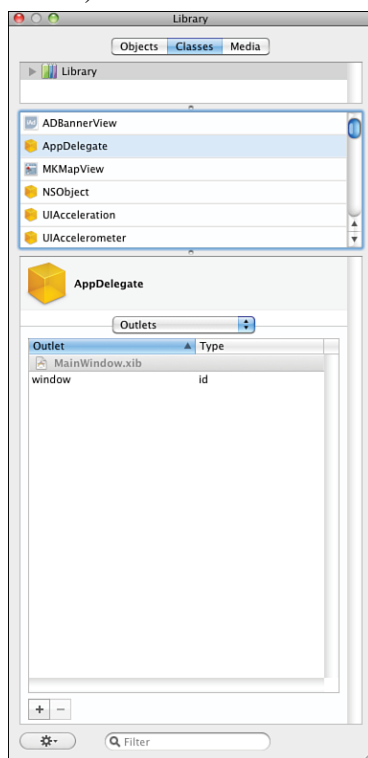


Рис. 1.12. Окно Library с переменными-выходами в классе AppDelegate

ращения к кнопке и метке, поэтому в нем необходимо создать выходы для этих объектов.

Чтобы добавить в AppDelegate выход для UILabel, нажмите кнопку + в левом нижнем углу окна Library. Измените предложенное по умолчанию имя myOutlet1 на HelloLabel. Тип (столбец Type) по умолчанию равен id, то есть объект может принадлежать любому классу, производному от NSObject. Это аналог типа object в C#. Измените тип на UILabel, тогда при связывании переменной (чуть ниже) вы не сможете подключить ее ни к чему, кроме объекта типа UILabel. После этого окно Library должно выглядеть, как показано на рис. 1.13.

Еще раз повторим, чего мы хотим добиться, поскольку для человека, имеющего опыт работы с продуктами Microsoft, все это выглядит очень непривычно. У нас есть xib-файл, в котором хранятся сериализованные объекты, описывающие пользовательский интерфейс. У нас есть также код, из которого мы хотим программно обращаться к объектам интерфейса. Чтобы подключить объект, описанный в xib, к коду, мы добавили выход для UILabel в класс AppDelegate в IB. Осталось лишь выполнить фактическое соединение UILabel с только что определенным выходом, в результате чего в коде на C# будет создано свойство, указывающее на UILabel.

Чтобы создать соединение между меткой UILabel в окне и выходом типа UILabel в AppDelegate, выберите AppDelegate в окне MainWindow.xib и откройте инспектор соединений (команда **Tools > Connections Inspector** в главном меню). В разделе Outlets вы увидите только что созданный выход HelloLabel. В окне Connections Inspector щелкните по кружочку слева от выхода и, не отпуская кнопку мыши (левую, если у вас мышь с несколькими кнопками), буксируйте его

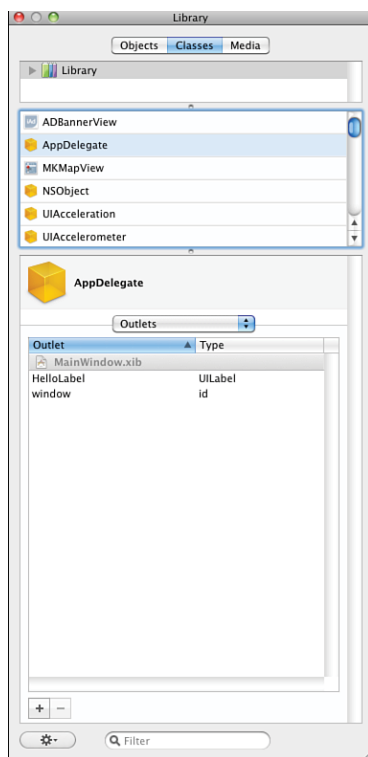


Рис. 1.13. Окно Library, где в класс AppDelegate добавлен выход, указывающий на UILabel