1 Реализация контроллеров и видов

1.0. Введение

В iOS 7 появилось множество новых пользовательских возможностей, а также масса новых API, с которыми мы, программисты, можем вволю экспериментировать. Вероятно, вы уже знаете, что в iOS 7 разительно изменился пользовательский интерфейс. Во всех предыдущих версиях он оставался практически неизменным по сравнению с первой версией iOS, и поэтому многие приложения разрабатывались так, как будто пользовательский интерфейс никогда не изменится. В настоящее время графические дизайнеры столкнулись с целым букетом проблем, так как теперь требуется создавать интерфейсы и продумывать пользовательские взаимодействия с программой так, чтобы программа хорошо смотрелась и в iOS 7, и в более ранних версиях.

Чтобы программировать приложения для iOS 7, вы должны знать основы языка Objective-C, с которым мы будем работать на протяжении всей этой книги. Как понятно из названия, язык Objective-C основан на C, но имеет определенные расширения, которые облегчают оперирование объектами. Объекты и классы имеют фундаментальное значение в объектно-ориентированном программировании (ООП). К числу объектно-ориентированных языков относятся Objective-C, Java, C++ и многие другие. В Objective-C, как и в любом объектно-ориентированном языке, вы имеете доступ не только к объектам, но и к примитивам. Например, число –20 (минус двадцать) можно выразить в виде примитива следующим образом:

```
NSInteger myNumber = -20;
```

В этой простой строке кода определяется переменная `myNumber`, относящаяся к типу данных `NSInteger`. Ее значение устанавливается в 20. Так определяются переменные в языке Objective-C. Переменная — это простое присваивание имени местоположению в памяти. В таком случае если мы задаем 20 в качестве значения переменной `myNumber`, то сообщаем машине, что собираемся выполнить фрагмент кода, который поместит указанное значение в область памяти, соответствующую переменной `myNumber`.

В сущности, все приложения iOS используют архитектуру «модель — вид — контроллер» (MVC). С архитектурной точки зрения модель, вид и контроллер — это три основные составляющие приложения iOS.

Модель — это мозг приложения. Она выполняет все вычисления и создает для себя виртуальный мир, в котором может существовать сама, без видов и контроллеров. Иными словами, вы можете считать модель виртуальной копией вашего приложения, без интерфейса.

Вид — это окно, через которое пользователь взаимодействует с вашим приложением. В большинстве случаев вид отображает содержимое модели, но, кроме того, он же воспринимает и действия пользователя. Любые контакты между пользователем и вашим приложением отправляются в вид. После этого они могут быть перехвачены контроллером вида и переданы в модель.

Контроллеры в программах iOS — это, как правило, контроллеры видов, которые я только что упомянул. Контроллер вида является, в сущности, переходным звеном между моделью и видом. Он интерпретирует события, происходящие с одной стороны, и по мере необходимости использует эту информацию для внесения изменений на другой стороне. Например, если пользователь изменяет какое-либо поле в виде, то контроллер гарантирует, что и модель изменится соответствующим образом. А если модель получит новые данные, то контроллер прикажет виду отобразить их.

В этой главе вы узнаете, как выстраивать структуру приложения iOS и использовать виды и контроллеры видов для создания интуитивно понятных приложений.

В этой главе мы будем создавать большинство компонентов пользовательского интерфейса на базе шаблона Single View Application из Xcode. Чтобы воспроизвести приведенные инструкции, следуйте рекомендациям, приведенным в подразделе «Создание и запуск вашего первого приложения для iOS» данного раздела. Убедитесь в том, что ваше приложение является универсальным, а не ориентировано только на iPhone или на iPad. Универсальное приложение может работать как на iPhone, так и на iPad.

Создание и запуск вашего первого приложения для iOS

Прежде чем подробнее познакомиться с возможностями Objective-C, вкратце рассмотрим, как создать простое приложение для iOS в среде Xcode. Xcode — это интегрированная среда разработки (IDE) для работы с Apple, позволяющая создавать, строить и запускать ваше приложение в эмуляторе iOS и даже на реальных устройствах с iOS. По ходу книги мы подробнее обсудим Xcode и ее возможности, а пока научимся создавать и запускать самое простое приложение. Я полагаю, что вы уже скачали Xcode из Mac App Store и установили ее на своем компьютере. В таком случае выполните следующие шаги.

1. Откройте Xcode, если еще не сделали этого.
2. Выберите в меню пункт File (Файл), далее — New Project (Новый проект).
3. Слева в диалоговом окне создания нового проекта выберите подкатегорию Application (Приложение) в основной категории iOS. Затем справа щелкните на варианте Single View Application (Приложение с единственным видом) и нажмите кнопку Next (Далее).

4. На следующем экране вы увидите поле Product Name (Название продукта). Здесь укажите название, которое будет понятно вам, например `My First iOS App`. В разделе Organization name (Название организации) введите название вашей компании или, если работаете самостоятельно, любое другое осмысленное название. Название организации — довольно важная информация, которую, как правило, придется здесь указывать, но пока она нас не особенно волнует. В поле Company Identifier (Идентификатор компании) запишите `com.mycompany`. Если вы действительно владеете собственной компанией или пишете приложение для фирмы, являющейся вашим работодателем, то замените `mycompany` настоящим названием. Если просто экспериментируете, придумайте какое-нибудь название. В разделе Devices (Устройства) выберите вариант Universal (Универсальное).
5. Как только зададите все эти значения, просто нажмите кнопку Next (Далее).
6. Система предложит сохранить новый проект на диске. Выберите желаемое местоположение проекта и нажмите кнопку Create (Создать).
7. Перед запуском проекта убедитесь, что к компьютеру не подключено ни одного устройства iPhone или iPad/iPod. Это необходимо, поскольку, если к вашему Mac подключено такое устройство, то Xcode попытается запускать приложения именно на устройстве, а не на эмуляторе. В таком случае могут возникнуть некоторые проблемы с профилями инициализации (о них мы поговорим позже). Итак, отключите от компьютера все устройства с системой iOS, а затем нажмите большую кнопку Run (Запуск) в левом верхнем углу Xcode. Если не можете найти кнопку Run, перейдите в меню Product (Продукт) и выберите в меню элемент Run (Запуск).

Ура! Вот и готово простое приложение, работающее в эмуляторе iOS. Может быть, оно и не кажется особенно впечатляющим: в эмуляторе мы видим просто белый экран. Но это лишь первый шаг к освоению огромного iOS SDK. Давайте же отправимся в это непростое путешествие!

Определение переменных и понятие о них

Во всех современных языках программирования, в том числе в Objective-C, существуют переменные. Переменные — это просто псевдонимы, обозначающие участки (местоположения) в памяти. Каждая переменная может иметь следующие свойства:

- тип данных, представляющий собой либо примитив (например, целое число), либо объект;
- имя;
- значение.

Задавать значение для переменной приходится не всегда, но вы обязаны указывать ее имя и тип. Вот несколько типов данных, которые необходимо знать для написания типичного приложения iOS.

Если тип данных является изменяемым, то вы можете изменить такие данные уже после инициализации. Например, вы можете откорректировать одно из значений в изменяемом массиве, добавлять в него новые значения или удалять их оттуда. Напротив, при работе с неизменяемым типом вы должны предоставлять все значения для него уже на этапе инициализации. Позже нельзя будет пополнить набор этих значений, удалить какие-либо значения или изменить их. Неизменяемые типы полезны в силу своей сравнительно более высокой эффективности. Кроме того, они помогают избежать ошибок, если все значения должны оставаться неизменными на протяжении всего жизненного цикла данных.

- `NSInteger` и `NSUInteger`. Переменные этого типа могут содержать целочисленные значения, например `10`, `20` и т. д. Тип `NSInteger` может содержать как положительные, так и отрицательные значения, но тип `NSUInteger` является беззнаковым, на что указывает буква U в его названии. Не забывайте, что слово «беззнаковый» в терминологии языков программирования означает, что число ни при каких условиях не может быть отрицательным. Отрицательные значения могут содержаться только в числовом типе со знаком.
- `CGFloat`. Содержит числа с плавающей точкой, имеющие десятичные знаки, например `1.31` или `2.40`.
- `NSString`. Позволяет сохранять символьные строки. Такие примеры мы рассмотрим далее.
- `NSNumber`. Позволяет сохранять числа как объекты.
- `id`. Переменные типа `id` могут указывать на объект любого типа. Такие объекты называются *нетипизированными*. Если вы хотите передать объект из одного места в другое, но по какой-то причине не хотите при этом указывать их тип, то вам подойдет именно такой тип данных.
- `NSDictionary` и `NSMutableDictionary`. Это соответственно неизменяемый и изменяемый варианты хеш-таблиц. В хеш-таблице вы можете хранить ключ и ассоциировать этот ключ со значением. Например, ключ `phone_num` может иметь значение `0 5552 4 87700`. Для считывания значений достаточно ссылаться на ассоциированные с ними ключи.
- `NSArray` и `NSMutableArray`. Неизменяемые и изменяемые массивы объектов. Массив — это упорядоченная коллекция элементов. Например, у вас может быть 10 строковых объектов, которые вы хотите сохранить в памяти. Для этого хорошо подойдет массив.
- `NSSet`, `NSMutableSet`, `NSOrderedSet`, `NSMutableOrderedSet`. Это типы множеств. Множества напоминают массивы тем, что могут содержать в себе наборы объектов, но в отличие от массива множество может включать в себя только уникальные объекты. Массив может содержать несколько экземпляров одного и того же объекта, а в множестве каждый объект может присутствовать только в одном экземпляре. Рекомендую вам четко усвоить разницу между массивами и множествами и использовать их правильно.
- `NSData` и `NSMutableData`. Неизменяемые и изменяемые контейнеры для любых данных. Такие типы данных очень вам пригодятся, если вы, например, хотите выполнить считывание содержимого файла в память.

Одни из рассмотренных нами типов данных являются примитивами, другие — классами. Вам придется просто запомнить, какие из них относятся к каждой из категорий. Например, тип данных NSInteger является примитивом, а NSString — классом. Поэтому из NSString можно создавать объекты. В языке Objective-C, как и в C и C++, существуют указатели. Указатель — это тип данных, в котором сохраняется адрес в памяти. По этому адресу уже хранятся фактические данные. Вы уже, наверное, знаете, что указатели на классы обозначаются символом астериска (*):

```
NSString *myString = @"Objective-C is great!";
```

Следовательно, если вы хотите присвоить строку переменной типа NSString на языке Objective-C, то вам понадобится просто сохранить данные в указатель типа NSString *. Но если вы собираетесь сохранить в переменной значение, представляющее собой число с плавающей точкой, то не сможете использовать указатель, так как тип данных, к которому относится эта переменная, не является классом:

```
/* Присваиваем переменной myFloat значение PI */
CGFloat myFloat = M_PI;
```

Если вам нужен указатель на эту переменную, соответствующую числу с плавающей точкой, то вы можете поступить так:

```
/* Присваиваем переменной myFloat значение PI */
CGFloat myFloat = M_PI;

/* Создаем переменную указателя, которая направлена на переменную myFloat */
CGFloat *pointerFloat = &myFloat;
```

Мы получаем данные от исходного числа с плавающей точкой путем простого разыменования (myFloat). Если получение значения происходит с применением указателя, то требуется использовать астериск (*pointerFloat). В некоторых ситуациях указатели могут быть полезны — например, при вызове функции, которая задает в качестве аргумента значение с плавающей точкой, а вы хотите получить новое значение после возврата функции.

Но вернемся к теме классов. Пожалуй, следует разобраться с ними немного подробнее, пока мы окончательно не запутались. Итак, приступим.

Как создавать классы и правильно пользоваться ими

Класс — это структура данных, у которой могут быть методы, переменные экземпляра и свойства, а также многие другие черты. Но пока мы не будем углубляться в подробности и поговорим об основах работы с классами. Каждый класс должен следовать таким правилам.

- Класс должен наследовать от суперкласса. Из этого правила есть немногочисленные исключения. В частности, классы NSObject и NSProxy являются корневыми. У корневых классов не бывает суперкласса.
- Класс должен иметь имя, соответствующее Соглашению об именованиях методов в Cocoa.

- У класса должен быть файл интерфейса, в котором определяется интерфейс этого класса.
- У класса должна быть реализация, в которой вы прописываете все возможности, которые вы «обещали» предоставить согласно интерфейсу класса.

`NSObject` — это корневой класс, от которого наследуют практически все другие классы. В этом примере мы собираемся добавить класс под названием `Person` в проект, который был создан в подразделе «Создание и запуск вашего первого приложения для iOS» данного раздела. Далее мы добавим к этому классу два свойства, `firstName` и `lastName`, которые относятся к типу `NSString`. Выполните следующие шаги, чтобы создать класс `Person` и добавить его в ваш проект.

1. Откройте проект в Xcode и в меню File (Файл) выберите New ▸ File (Новый ▸ Файл).
2. Убедитесь, что слева, в разделе iOS, вы выбрали категорию Cocoa Touch. После этого выберите элемент Objective-C Class (Класс для Objective-C) и нажмите Next (Далее).
3. В разделе Class (Класс) введите имя `Person`.
4. В разделе Subclass of (Подкласс от) введите `NSObject`.

Когда справитесь с этим, нажмите кнопку Next (Далее). На данном этапе Xcode предложит вам сохранить этот файл. Просто сохраните новый класс в том каталоге, где находятся ваш проект и все его файлы. Это место выбирается по умолчанию. Затем нажмите кнопку Create (Создать) — и дело сделано.

После этого в ваш проект будут добавлены два новых файла: `Person.h` и `Person.m`. Первый файл — это интерфейс вашего класса `Person`, а второй — файл реализации этого класса. В Objective-C `.h`-файлы являются заголовочными. В таких файлах вы определяете интерфейс каждого файла. В `.m`-файле пишется сама реализация класса.

Теперь рассмотрим заголовочный файл нашего класса `Person` и определим для этого класса два свойства, имеющие тип `NSString`:

```
@interface Person : NSObject

@property (nonatomic, copy) NSString *firstName;
@property (nonatomic, copy) NSString *lastName;

@end
```

Как и переменные, свойства определяются в особом формате в следующем порядке.

1. Определение свойства должно начинаться с ключевого слова `@property`.
2. Затем следует указать квалификаторы свойства. Неатомарные (`nonatomic`) свойства не являются потокобезопасными. О безопасности потоков мы поговорим в главе 14. Вы можете указать и другие квалификаторы свойств: `assign`, `copy`, `weak`, `strong` или `unsafe_unretained`. Чуть позже мы подробнее поговорим и о них.
3. Затем укажите тип данных для свойства, например `NSInteger` или `NSString`.
4. Наконец, не забудьте задать имя для свойства. Имена свойств должны соответствовать рекомендациям Apple.