移动开发系列

# 基于Swift的Apple Watch开发教程

王永超◎编著

内容全面 | 代码齐全 | 讲解清楚

全面阐述 Swift 3、watchOS 3 健康监测

与 iPhone 交互等多项手表专属内容及案例

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

移动开发系列

# 基于 Swift 的 Apple Watch 开发教程

王永超　编著


電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

## 内 容 简 介

Apple Watch 是苹果公司推出的智能手表，其系统为 watchOS，Swift 是苹果公司新推出的开发语言。本书介绍基于 Swift 3 进行 watchOS 3 应用的开发，对 Swift 3 基础、watchOS 3 基础、多媒体、游戏、传感器、GPS、与 iPhone 交互、健康监测等多项手表专属内容进行全面阐述和案例学习，可以更好地引导和帮助读者掌握新语言 Swift 和学习 Apple Watch 应用开发。

本书适合 iOS 开发、Swift 开发、Apple Watch 开发和学习，以及 Apple watch 爱好等众多读者。



**图书在版编目（CIP）数据**

基于 Swift 的 Apple Watch 开发教程 / 王永超编著. —北京：电子工业出版社，2017.9
（移动开发系列）
ISBN 978-7-121-32377-5

Ⅰ. ①基… Ⅱ. ①王… Ⅲ. ①程序语言－程序设计－教材 Ⅳ.①TP312

中国版本图书馆 CIP 数据核字（2017）第 183889 号

策划编辑：张 迪（zhangdi@phei.com.cn）
责任编辑：张 迪
印 刷：北京季蜂印刷有限公司
装 订：北京季蜂印刷有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：710×1 000 1/16 印张：9.5 字数：243 千字
版 次：2017 年 9 月第 1 版
印 次：2017 年 9 月第 1 次印刷
定 价：39.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）88254469；zhangdi@phei.com.cn。

# 前　言

Apple Watch 是苹果（Apple）公司推出的一款比较成熟的智能穿戴电子产品，具有运动追踪、健康监测、消息推送、多媒体、游戏、GPS 定位等多种功能，并支持第三方应用，如今已有较大市场占有率。Apple Watch 需要配合 iPhone 手机使用，通过经配对的 iPhone 访问应用商店进行第三方应用的下载和安装，Watch 上的应用都会包括一个手表端运行包和手机端运行包，分别运行在手表和手机上，并且两者之间可以交换数据。

Apple Watch 之所以被称为“智能手表”，是因为其上运行着智能操作系统 watchOS。watchOS 之所以被称为“智能操作系统”，笔者认为不是因为其可以连接手机并接受手机控制，而是因为其拥有应用程序的概念，为开发者提供了应用程序编程接口（API）、开发语言和开发编译工具 Xcode，并且允许开发者开发、部署和运行第三方应用程序（App）。watchOS 1 的应用运行在 iPhone 上并把运行结果发送到 watch 上进行显示。从 watchOS 2 开始，应用直接安装和运行在手表上，称为本地应用，又称原生应用（Native App），大幅提高了运行效率。

watchOS 允许开发者使用两种语言开发应用，分别是原有的 Objective-C 和新语言 Swift。Swift 是苹果公司推出的新开发语言，Swift 已经更新了多个版本，从 Swift 3 开始，Cocoa 类名去掉了 Objective-C 中的 NS 开头，直接定义为含义名称，例如 NSString 改为 String，NSTimer 改为 Timer，等等，并且类的成员也都有少量变化。Swift 也可以调用已有的 Objective-C 类进行混合编译。为了更好地推广 Swift，苹果公司已经将 Swift 开源化，允许开发者直接基于 C 语言开发 Swift 源码。Swift 具有语法简洁、易读、易写的特点，本书采用 Swift 开发 watchOS 应用，也为读者提供一个学习和使用 Swift 参考资料。

本书包括 8 章。第 1 章概述，介绍 watchOS 项目的结构组成，并列举了已经提供部分和完全支持的框架。第 2 章 Swift 编程基础，Swift 更新到第 3 个版本已经基本稳定，该部分介绍了 Swift 的基本语法、函数调用、常用数据结构和类，为后续开发做好准备。第 3 章 watchOS 基础，开始正式进入 watchOS 应用的开发，

介绍了页面和常用控件。第 4 章 watchOS 高级进阶，是第 3 章 watchOS 基础开发的进一步深入，主要涉及按压交互和组件交互，具体内容包括 Force Touch 菜单、振动引擎、表盘功能栏、提醒、后台刷新任务、URL 后台下载、Dock 截图、Apple Pay、通知等多种高级功能。第 5 章多媒体和游戏引擎，多媒体包括录音、连接蓝牙耳机播放音频、视频播放和喇叭外放，游戏引擎包括 2D 引擎 SpriteKit 和 3D 引擎 SceneKit，这里还结合游戏讲解了手势交互。第 6 章运动传感器和 GPS，Apple Watch 已经明确支持加速计、陀螺仪两种主要运动传感器，同时也支持 GPS 定位。第 7 章与 iPhone 交互，Apple Watch 不是 iPhone 的附属部件，而是运行独立操作系统的独立计算机，与 iPhone 之间进行平等的数据交换，watchOS 与配对的 iOS 交互是通过 WatchConnectivity 框架实现的，包括覆盖式后台传输、队列式后台传输、文件传输、消息传输、消息数据传输、功能栏传输等多个交互类型。第 8 章健康，Apple Watch 提供强大和全面的健康监测功能，如心率、步数、活动能量消耗等，同时会将监测到的数据发送到 iPhone 上的健康库中。上述的每一个部分在详细讲解的基础上，均提供了案例代码。

以 Apple Watch 为代表的智能穿戴产品还远远不如手机那么普及，存在巨大的市场增长空间，同时 Apple Watch 的应用和功能的开发还处在挖掘和探索阶段，而本书以此为契机，为广大开发者和其他兴趣读者学习和开发 Apple Watch 应用提供详细、系统的中文学习参考资料。笔者也衷心希望广大开发者早日能创造出几款杀手级应用。

# 目　录

# 第 1 章

# 概　述

## 1.1 watchOS 项目结构

如今的 Apple Watch App 都要求是原生应用（native app），原生应用即是高于 watchOS 2 及以上的版本，并作为一个完整的应用包在 Apple Watch 上独立运行。本书介绍的手表操作系统版本是 watchOS 3，使用的语言版本是 Swift 3，如图 1-1 所示是 watchOS 3 App 的结构图。

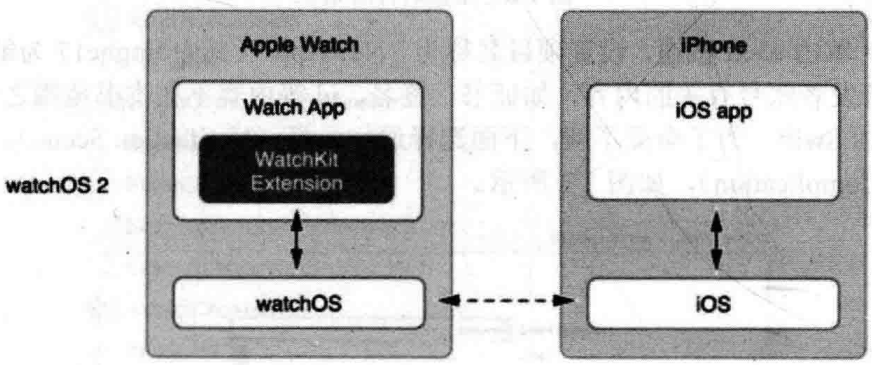


图 1-1　watchOS 3 App 结构图

从图 1-1 中可以看出，Watch App 作为一个独立应用在 watchOS 上运行，wachOS2/3 的 Xcode 项目包含 3 个部分，分别是 iOS App 包、Watch App 包和 WatchKit Extention 包。其中，iOS App 包负责 iPhone 端所有运行内容，Watch App 包包含界面编辑和手表应用整体参数，WatchKit Extention 包包括 watch 端运行的代码及资源。WatchKit Extension 包含在 Watch App 包中，而手表端 App 和手机端 App 通过操作系统（watchOS 和 iOS）进行交互。

## 1.2 建立 watchOS App 实例

（1）打开 xcode，新建项目，进入 watchOS 下的 Application，选择“iOS App with WatchKit App”，如图 1-2 所示。

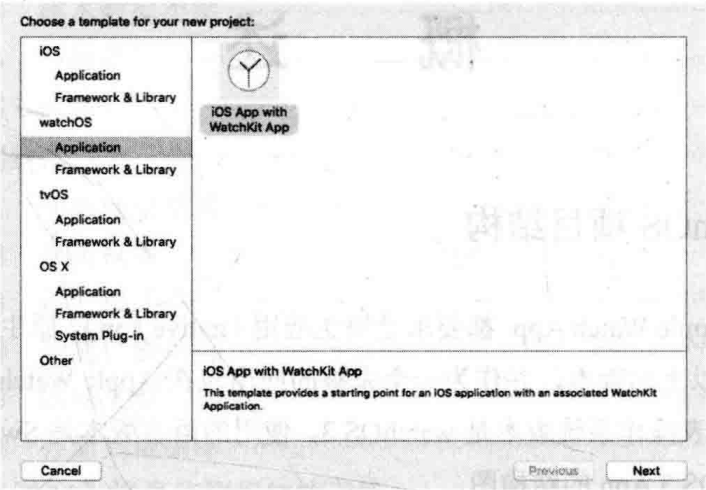


图 1-2　工程选择对话框

（2）单击 Next 按钮，设置项目名称为“NewApp”（其中 tinghe17 为笔者常用 id，与开发者账号有关的内容，如证书、签名、id 等内容不在本书范围之内），选择语言为 Swift，为了全面了解，下面选择通知场景（Notifaction Scene）、表盘功能栏（Complication），如图 1-3 所示。

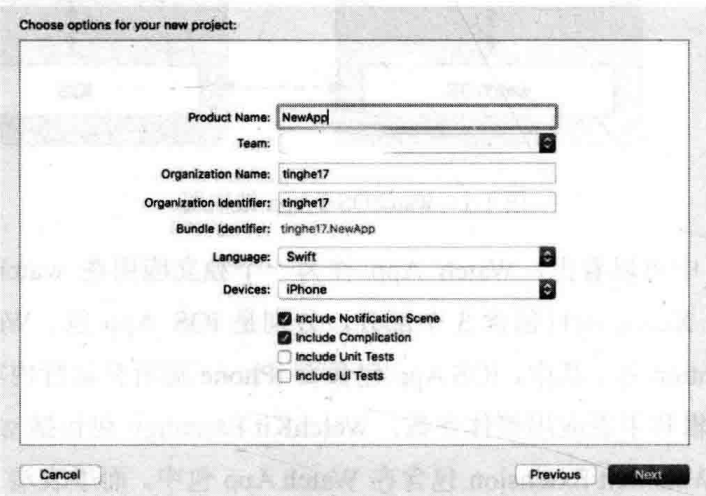


图 1-3　项目参数对话框

（3）单击 Next 按钮，选择保存路径（英文），项目建立成功。iOS App 包的名称为 NewApp，手表端包会自动起名字，即添加了后缀“WatchKit App”和“WatchKit Extension”。整个 NewApp 项目包含 3 个相应的 TARGETS。3 个 TARGETS 的 Bundle Identifier 存在前后的从属关系，依次为 tinghe17.NewApp、tinghe17.NewApp.watchkitapp 和 tinghe17.NewApp.watchkitapp.watchkitextension，如图 1-4 所示。

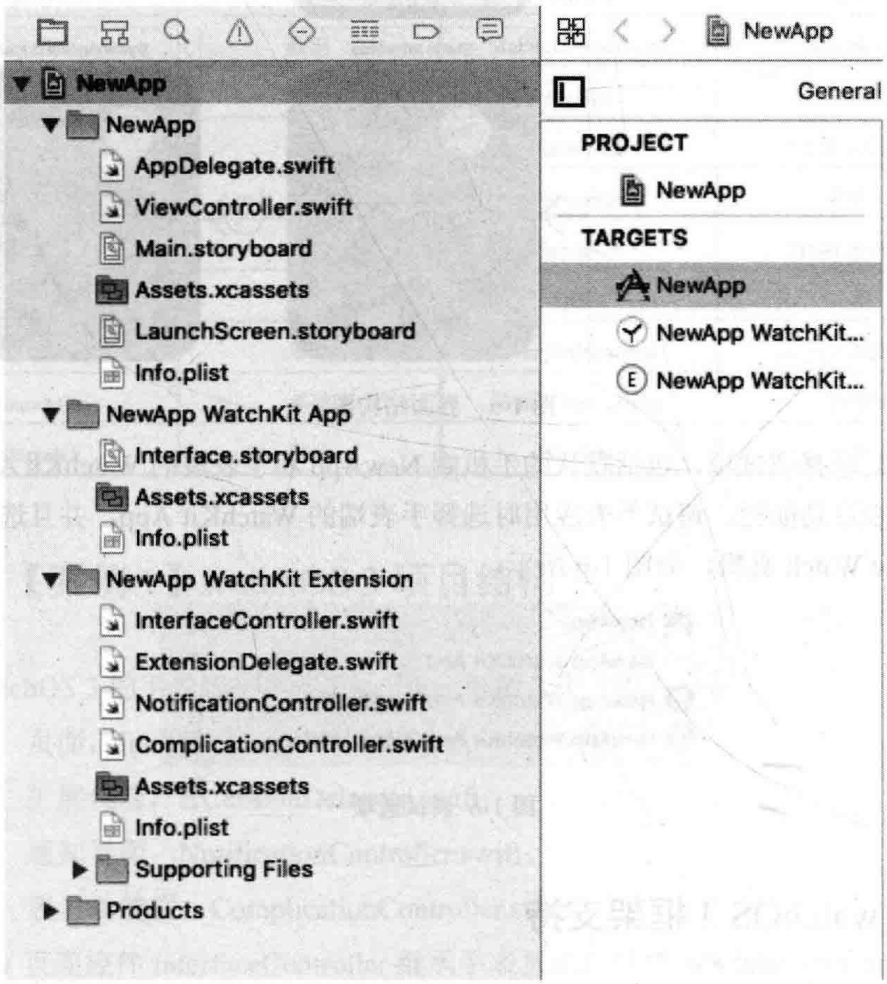


图 1-4 项目结构图

（4）在项目列表中选择 NewApp WatchKit App 中的 interface.storyboard，即可从主视图看到手表界面详细结构，包括一个应用运行时的界面（Interface Countroller）、一个静态通知界面（Static Interface）和一个由前者变化而显示的动态通知界面（Dynamic Interface），如图 1-5 所示。

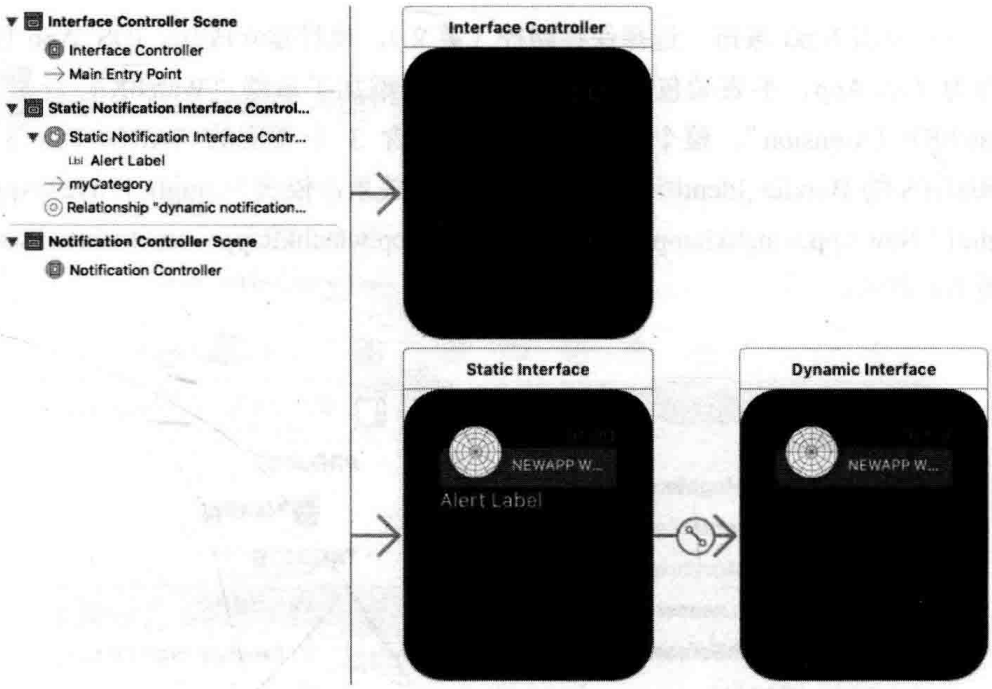


图 1-5　界面结构图

（5）选择调试项，包括默认的手机端 NewApp 和手表端的 WatchKit App、通知、与表盘功能栏。调试手表应用时选择手表端的 WatchKit App，并且选择适配的 Apple Watch 真机，如图 1-6 所示。

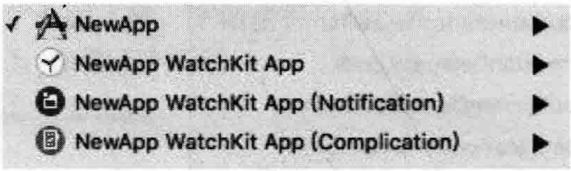


图 1-6 调试选项

## 1.3 watchOS 3 框架支持

watchOS 3 支持框架 framework 可以分为两类：一类是手表系统专属框架；另一类是 iOS 系统原有框架。其中，手表对于某些 iOS 系统原有框架是部分支持，同时某些原有框架加入了手表特有部分。本书重点介绍手表有关的框架及其内容。watchOS 3 支持的框架如表 1-1 所示。

表 1-1 watchOS 3 的框架支持

| 框架名 | 中文 | 框架名 | 中文 |
|---|---|---|---|
| AVFoundation | 音频视频 | HealthKit | 健康 |
| ClockKit | 表盘 | HomeKit | 家 |
| Contacts | 联系 | ImageIO | 图形输入输出 |
| CoreAudio | 音频 | MapKit | 地图 |
| CoreData | 数据 | MobileCoreServices | 移动服务 |
| CoreFoundation | 基础 | PassKit | 同行 |
| CoreGraphics | 图形 | SceneKit | 3D 游戏引擎 |
| CoreLocation | 定位 | Security | 安全 |
| CoreMotion | 运动 | SpriteKit | 2D 游戏引擎 |
| CoreText | 文本 | UIKit | 界面 |
| EventKit | 事件 | UserNotifications | 用户通知 |
| Foundation | 基础 | WatchConnectivity | 手表连接 |
| GameKit | 游戏 | WatchKit | 手表 |

## 1.4 【案例 1】watchOS 3 项目结构

watchOS 3 的手表端扩展包 Extension 包括 3 个文件：

- 页面，InterfaceController.swift；
- 扩展代理，ExtensionDelegate.swift；
- 通知界面，NotificationController.swift；
- 表盘功能栏，ComplicationController.swift。

（1）页面控件 InterfaceController 继承手表页面控件类 WKInterfaceController，默认生成三个函数，分别在不同的时机由系统调用：初始化时调用 awake（withContext context: Any?），页面显示时调用 willActivate()，不显示时调用 didDeactivate()。

```
override func awake(withContext context: Any?) {
        super.awake(withContext: context
    }
override func willActivate() {
        super.willActivate()
    }
override func didDeactivate() {
        super.didDeactivate()
    }
```

（2）扩展代理 ExtensionDelegate 继承手表扩展代理 WKExtensionDelegate，之所以叫“扩展代理”而不是“代理”，是因为该代理是 WatchKit Extention 的代理，而不是 WatchKit App 的代理。默认包含涉及手表应用声明周期：应用启动完成后初始化时调用 applicationDidFinishLaunching()，应用激活前台时调用 applicationDidBecomeActive()，应用失活时调用 applicationWillResignActive()，应用被系统后台启动时调用 handle（_ backgroundTasks: Set<WKRefreshBackgroundTask>）。

```
func applicationDidFinishLaunching() {}
func applicationDidBecomeActive() {}
func applicationWillResignActive() {}
func handle(
        _ backgroundTasks:Set<WKRefreshBackgroundTask>) {
            backgroundTasks.forEach { (task) in
                //处理代码
            task.setTaskCompleted()
        }
}
```

（3）通知页面控件 NotificationController 是在手表接到通知时单击通知按钮显示的页面。通知页面的声明周期与普通页面相似（同样包括初始化、激活、失活），只是多了一个接到通知的处理函数 didReceive()，默认该函数是标注不用状态，当使要显示通知页面动态显示内容时才须要调用该函数。

```
override func didReceive(_ notification: UNNotification,
                withCompletion completionHandler: @escaping
            (WKUserNotificationInterfaceType) -> Swift.Void) {
        completionHandler(.custom)
    }
```

（4）表盘功能栏控件 ComplicationController 负责表盘功能栏的设置，包括在 iPhone Watch 应用里表盘图库里的示例显示。ComplicationController 包含了多个函数，分别对应显示功能栏的不同时机，其中最常用的直接设置表盘功能栏显示普通状态的函数是 getCurrentTimelineEntry()，设置 iPhone Watch 里的功能栏示例的函数是 getLocalizableSampleTemplate()。

```
func getCurrentTimelineEntry(for complication: CLKComplication,
                    withHandler handler: @escaping
                (CLKComplicationTimelineEntry?) -> Void) {
        handler(nil)
        }
func getLocalizableSampleTemplate(for complication: CLKComplication,
                    withHandler handler: @escaping
                    (CLKComplicationTemplate?) -> Void) {
        handler(nil)
}
```

# 第 2 章

# Swift 3 编程基础

Swift 作为一种新的编程语言，简洁易用，受到开发者的广泛喜爱。迄今为止 Apple 公司已经发布了 3 个 Swift 版本。本书在讲解手表开发之前，先介绍一下 Swift 3 语言的基本语法及其新特点。

## 2.1 基本语法

### 2.1.1 变量和常量

**1. 普通变量**

普通变量使用 var 声明，都有一定的初始值，例如：

```
//声明变量 a，初始值为 5，类型为整型 Int
var a=5
//声明变量 str，初始值为 abcd，类型为字符串 String
var str="abcd"
//声明对象变量 obj，类为 myClass
var obj=myClass()
```

**2. 可空变量**

可空变量是 Swift 特有的变量类型，其值可以为空（nil），而使用上节只用 var 声明的普通变量必须有一定的非空值。

```
//声明可空变量使用?
var a:Int?
//赋值跟普通变量相同
```