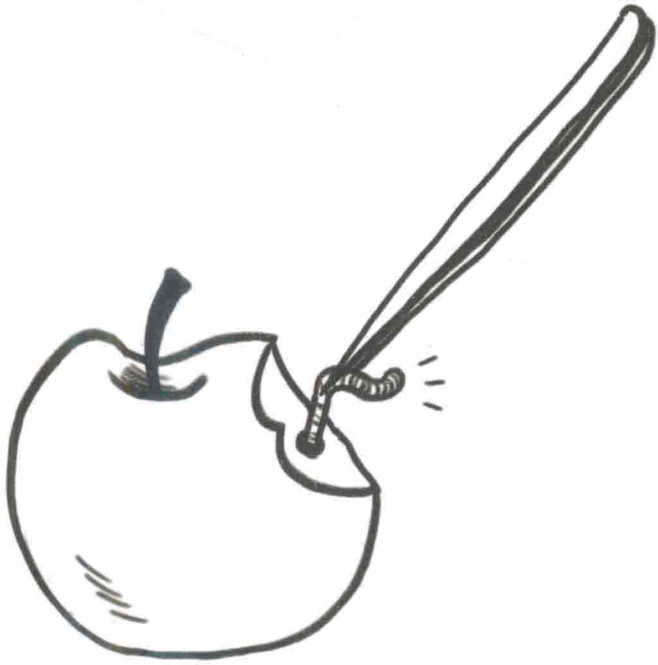
Broadview
www.broadview.com.cn

iOS Application Testing Guide

iOS测试指南

芈峮 著



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

iOS Application Testing Guide

iOS测试指南

芈峮 著

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

内 容 简 介

《iOS 测试指南》是一本专注于 iOS 测试领域的书，其中重点讲述了各个测试阶段的具体实践方法，并且通过持续集成串联了各个测试阶段的活动。本书中所有的测试实践并非纸上谈兵，而是出自于笔者实际工作中的探索和实践。在测试实例上有一定的简化，是为了脱离复杂的业务。本书的重点在于对方法的介绍。

以下几个方面的读者可能会受益：有一定技术功底的测试工程师；有一定经验的移动测试工程师；iOS 开发工程师；测试技术爱好者。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

iOS 测试指南 / 芈峮著. —北京：电子工业出版社，2014.5
ISBN 978-7-121-22758-5

Ⅰ. ①i… Ⅱ. ①芈… Ⅲ. ①移动终端－应用程序－程序设计 Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字（2014）第 059296 号

策划编辑：张春雨
责任编辑：徐津平
印　　刷：北京天来印务有限公司
装　　订：北京天来印务有限公司
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：720×1000　1/16　印张：14　字数：270 千字
印　　次：2014 年 5 月第 1 次印刷
印　　数：4000 册　定价：55.00 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。

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。

服务热线：（010）88258888。

推荐序一　你的iOS App还在裸奔么？

2014年3月的TIOBE编程语言排行榜，Objective-C排行第三位，紧随C语言和Java之后，甚至在C++前面。其他这三种语言，都是有着非常广泛的应用场景和悠久的使用历史的。Objective-C的诞生也并不晚，但是仅在近年才开始流行，而且Objective-C几乎只用于iOS/Mac平台开发。所以，这一排行榜多少能告诉我们iOS开发到底有多热。2013年7月，苹果公司CEO Tim Cook宣布仅在中国就有50万iOS开发者。

然而，这个行业的开发水平和测试水平到底怎么样呢？我在新浪微博简单地做了一个调查，34.5%的人说他们完全没有任何测试；44.8%的团队有独立测试人员；而有单元测试、UI自动化测试、持续集成的团队就屈指可数了。

那么，在这个平台下测试不重要吗？我觉得恰恰相反。

任何App要想在苹果的AppStore上架，都需要经过苹果的审核员的审核，不管你是世界五百强的大公司，还是小作坊，都会一视同仁，绝无例外。如果你的App没有经过良好的测试，被审核员发现有闪退、崩溃或者其他严重质量问题，他们会毫不犹豫地拒绝你的App。而你则需要修改App，重新提交，这往往就意味着再等7~8天的排队才有机会被审核。

如果你的运气好，Bug没有被审核员发现，或者说，在审核员审核的环境下，你的App表现良好，你的App就成功上架了。但是如果它在用户的iPhone/iPad上面发生闪退、崩溃，等等，其实你会更倒霉。因为愤怒的用户会迅速让你收获大量的1星，即使你好不容易做了一年的好评度，也会一下子跌落谷底。如果你熟悉AppStore的话，就知道这往往意味着你的下载量将一落千丈，你的App也有可能从此无人问津。

App这个形式和网站类应用最大的区别就是，如果网站的程序员发现某个页面有一个小Bug，修改后，经过合理的内部审核流程，它的一个部署脚本就可以升级代码，用户在刷这个页面和那个页面之间的空隙，这个升级就完成了。而iOS App出现了问题以后，不管你修改得多快，都需要被苹果的审核员审核，这往往需要7~14天，然后，你往往需要在用户方便的时候，通过重新下载整个App的方式得到更新（虽然iOS 7.0后，更新普遍可以自动在后台完成，但是时间周期仍旧是这么长）。

所以，对iOS开发者强调测试的重要性，我觉得说100遍、1万遍都不嫌多，都有其现实意义。

但是为什么还有那么多团队和个人开发者没有进行完善的测试呢?

懒、侥幸心理、怕麻烦一定是少不了的。

还有，我觉得就是一般的入门书、教程，甚至包括苹果的官方文档，讲到的测试部分都太简单，缺乏可操作性。

所以，当我得知本书的作者芈峮在写这样一本专注于iOS平台测试工具和方法的书的时候，我很高兴，而他邀请我做序的时候，我感到十分荣幸。

最早知道芈峮时，他还在豆瓣的测试团队工作，他做了一个开源的测试工具ynm3k（要你命3000）。仅仅是这个充满幽默感的名字就征服了我。

后来我了解了一下这个工具，由此我才知道原来在iOS下也是可以进行UI自动化测试的。在此之前，我只是经常跟人们在一起人云亦云地说："嗯，单元测试是好，但是iOS开发主要都是UI逻辑，这可怎么测试呢？"

不经过完善的测试，我们的App其实就是在裸奔，会不会出问题，会出什么样的问题，完全看运气。在以前，我们可以有这样或那样的借口，甚至可以直接说，我就是找不到资料嘛，学不会那还能怎么办呢?

现在，有了这本书，对不起，没有借口了，请把这本书带回家，仔细阅读，按照这本书改善你的开发测试流程，别再让你的App裸奔了……

郝培强

OurCoders.com创始人

2014年3月11日于上海

试读结束：需要全本请在线购买： www.ertongbook.com

推荐序二

前几天毕崌告诉我，他完成了一本与iOS测试相关的书，希望我能写个推荐序。毕崌在豆瓣的测试团队时，就一直专注于移动App和移动Web应用的测试。经常能够见到他鼓捣各种工具，用充满创意的方式将小工具连接起来解决实际问题，当然，最让人称道的，是他对移动测试的热情。

读完了毕崌这本不算厚的书，不觉眼前一亮。这本书并没有刻意地为了拔高而选择一些生涩的主题，而是系统性地介绍了移动测试的方方面面：对于移动App的测试，本书介绍了如何搭建移动App测试的环境，如何进行移动App的单元测试，如何选择和使用UI自动化测试方案；对于移动Web应用的测试，本书介绍了如何使用Selenium和Appium轻松地完成这类应用的测试；此外，毕崌还格外体贴地在书中介绍了如何使用持续集成（CI）更好地搭建持续测试的环境——在Web开发中引入持续集成并不少见，但在移动应用开发中发挥持续集成的威力，这可是真有些挑战的！

随着移动互联网渐入佳境，越来越多的组织和个人开始进入移动互联网领域。在测试行业内，越来越多的测试者开始关心移动应用的测试。移动应用明显有不同于Web应用和桌面应用的特点，移动开发平台（iOS、Android，也许还可以算上WP）自身的特性，设备的兼容性（即使是iOS的开发者，现在也不得不考虑兼容性的问题了），移动设备本身的某些特性（网络连接，交互特性等）都给移动应用的测试带来了新的挑战。明显能够看到，移动应用的测试已经成为测试领域的热点，近期在各个渠道也颇能见到一些与移动应用测试相关的介绍文章。然而，大部分我读到的介绍性的文章都是对某工具的介绍，或是某个具体的测试手段，虽然热闹非凡，但对那些希望系统性地一窥移动应用测试门径的测试者来说，实在是不堪大用。从这一点上来说，毕崌的这本《iOS测试指南》算是恰逢其时。本书这些扎扎实实来自一线实践的内容，一定能够让对这方面有兴趣的工程师系统性地了解移动应用测试。

当然，移动应用的测试热才刚刚在测试领域内兴起，这本《iOS测试指南》为这个方向开了个好头。我期望能够有越来越多的测试者（包括开发者）愿意深入到这个方向中，研究如何能够不断提高移动应用开发的效率和质量，帮助组织以最小的成本实现目标。我也更期望有越来越多的人愿意用图书的方式记录和分享自己的心得，为这个值得重视的领域添砖加瓦。

段念

豆瓣工程副总裁

2014年3月9日于北京

致谢

本书的成稿离不开许多人的帮助和支持。首先是家人的支持。对于一个1岁多孩子的父亲，利用业余时间完成一本书的编写是非常困难的。我美丽的妻子不但承担了日常所有的家务，还要独立养育孩子，甚是辛苦。如果没有她的理解和支持，本书与读者见面的时候可能iPhone 9都发布了。同样，我还要感谢我的父母、岳父、岳母和朋友，没有他们的鼓励我同样不会完成本书。

当然，还要感谢侠少和任晓露编辑。侠少负责扮黑脸，认真严格地对初稿进行修改，修改意见一度多于书稿字数。虽然那些改进意见都是正确的，但是由于数量过多，有时无法接受。这时晓露编辑就出现了，并且给了我无限的鼓励。就是在这样的不断重复中，我完成了书稿。在此，真心感谢博文视点的每一位编辑。

对本书中的iOS测试方法，我在豆瓣的工作期间几乎都实践过。在此，我还要感谢豆瓣的每一位移动开发工程师。因为没有你们高质量的代码，我不可能玩出这么多花哨的小技巧。当然，最应该感谢的是解彦博和耿新跃两位老师。如果不是当初两位老师给了我去豆瓣工作的机会，本书的成稿更是无从谈起。

最后，感谢本书的每一位读者，感谢你们对本书的关注和支持。

前言

为什么要写这本书

随着iOS应用开发的持续火热，iOS测试越来越受到重视。但是，由于其生态系统的封闭性，导致iOS测试方面的资料非常少并且难以搜索。在一次技术交流会上认识了博文视点的任晓露老师，她鼓励我应该写一本iOS测试方面的书。这种约稿一般都会在第1次时被拒绝。我同样拒绝了任老师的约稿，不为别的，只是因为自己水平有限。

我在平时的工作中还是不断地搜索着那些零星的资料，并且发现在iOS测试方面没有任何书籍，国内没有，国外貌似也没有。又是一次技术交流会，又见到了任晓露老师，当然，又谈到了出书的事情。这次互换了联系方式，并且在之后认真考虑了出书的事情。

之后在老婆大人的鼓励下，决定写一本iOS测试方面的书。心想只要动作快就是国内的第1本iOS测试书籍，全当是抛砖引玉了，谁让我在豆瓣的ID是厚脸皮呢，就厚着脸皮写了出版吧。

本书的内容

在测试领域内，分歧不断，争论不断。在如何做测试、测试的目的是什么等问题上都会有很大的争议。而测试活动本身受业务需求和团队能力等因素的影响，也会有很大的不同。本书抛开争论和不同，只谈技术相关的问题，通过简单的实践介绍了通过某些工具或者框架来对应某一些测试类型。

第1章

简短地介绍了测试和iOS测试，并且对本书涉及的内容范围进行了介绍。

第2章

介绍了iOS开发和测试使用的基本工具。

第3章

本章首先介绍单元测试的工具，之后通过实践，详细介绍了基于MVC模式的单元测试的使用方法，其中包括针对Model、Controller和View的基本的测试方法。在实践中使用到了一些高级的断言工具和Mock工具。最后再次针对这些工具进行了详细介绍。

第4章

提到UI自动化测试，第一入手点必须是官方工具。本章通过实践详细介绍了iOS官方的自动化测试工具——UI Automation，不但有实践的应对和基本API的讲解，还加入了笔者对UI自动化的总结和第三方工具的简单介绍。希望能做到深入浅出。

第5章

iOS程序不只有Native应用，还有Web应用。本章结合笔者的工作经验和总结，介绍了iOS Web自动化测试的最佳实践，并且从组成结构上剖析了当下最流行的Appium和WebDriver。

第6章

持续集成是现代软件开发的一种体现。没有持续集成的自动化测试都是半自动化测试。本章不但介绍了通用的持续集成工具，还基于之前章节的实践成果，进行了iOS持续集成方面的介绍。

第7章

除了功能测试之外，iOS程序还需要很多的专项测试，例如兼容性测试等。本章主要介绍了几种通用的专项测试类型和方法。

第8章

iOS自动化测试有很多第三方的开源工具。本章从工具本身的技术特点和实现原理上对工具进行了分类，并且对每一类工具选出了佼佼者进行实践介绍。当然，读者可以根据本章的内容写出自己喜欢的自动化工具。

第9章

在2013年的第4季度，Apple公司大爆发似地发布了开发工具Xcode 5、手机操作系统iOS 7和Mac操作系统OS X 10.9。这一系列工具的发布，也带来了测试方面的一些新特性的引入。本章结合之前的内容，针对这些新特性进行了补充介绍。

目录

试读结束：需要全本请在线购买： www.ertongbook.com

第1章

软件测试与iOS测试

iOS测试是软件测试的一个分支，有着软件测试的一些共性。同时，由于iOS平台及其生态系统具有特殊性，iOS测试也有很多自己的特性。本章将从共性与特性两个方面来总结一下iOS测试。

1.1 什么是软件测试

业界对软件测试及其目的、手段和方法是什么，一直存在很大争议。作为一家之言，本书尝试对软件测试作出如下描述：软件测试的目标应该服从于软件项目的目标，虽然它不直接产生有价值的成果，但是可以通过建议使用更高效的方法和工具，提升软件开发效率及软件开发质量；还可以通过一些手段，更早、更快、更多地发现缺陷，从而降低这些缺陷可能带来的风险。

基于这一观点，书中会介绍一些可能不包括在传统测试范围内的内容，例如iOS平台的持续集成、iOS相关的打包和发布等。以下将围绕软件测试概念中较有争议的3个方面来阐述本书的观点。

1.1.1 测试活动何时展开

在软件产品开发中，测试越早，发现问题后解决问题的成本就越小。如果开发过程中的各个阶段所写的代码都能够正确且稳定地运行，那么在后期将它们集成起来或加入新部件时，相比于项目结束后再一次性执行所有测试，所产生的错误自然要少。尽早并且持续地反馈问题，对于QA来说是一项很重要的职责，对于一个iOS应用也同样重要。具体而言，开发工程师如能在提交一次代码后立刻发现新版本iOS程序中导致应用程序崩溃的问题，则这时修复问题的成本最低。首先，由于发现及时，问题可能涉及的代码非常有限，开发工程师可以很轻松地定位问题；其次，开发工程师对代码的熟悉程度很高，能快速并且完善地解决问题。试想一下，3～5天之后代码库中已经积累了很多次的代码提交，这时再去定位问题并且修改已经是一件比较困难的事情。情况再糟一些，如果两个月以后才发现问题，则该问题可能已经给团队带来了很多的风险，修复这个问题的成本也会很高。

以上只是从时间这个维度来说明问题，除此之外，在实际工作中随着时间的推移，产品的功能和代码量都是在不断增加的。随着代码量的增加，修正软件缺陷的难度也在增加。

1.1.2 软件测试与软件缺陷

软件缺陷被测试工程师和开发工程师们称作Bug。软件缺陷会导致软件不能正常运行，它的存在会在一定程度上导致软件不能满足用户的需求，甚至有可能破坏或者泄露用户的重要数据。

不管开发人员的水平有多高，不管开发人员多么小心地开发软件，软件缺

陷都会悄无声息地存在于软件系统之中。虽然软件测试是为了消灭软件缺陷而生的，但是迄今为止，还没有一种有效的软件缺陷检测机制。软件运行的环境多种多样，其本身逻辑关系的高复杂度和多种多样的数据结构等因素都决定着软件测试活动不可能通过遍历所有的功能和使用场景来发现软件系统中所有的缺陷。

在软件开发的每一个环节中都可能把软件缺陷引入系统中，通过测试只能发现部分缺陷，并不能检测到系统中的所有缺陷。

1.1.3 软件测试与软件质量

软件系统的质量应该更多取决于这个软件系统的生产者——开发工程师。如果开发工程师技术高超并且使用了正确的软件开发方法，软件系统的质量会更可靠。套用一句话：“一个高质量的软件系统是设计和开发出来的，并不是测试出来的”[1]。

那么，软件测试不重要吗？当然不是。一方面，测试可以做到对缺陷的预防。测试工程师要去总结一些项目产品开发活动中非常好的软件工程实践，并且推广到项目组中，建议开发工程师使用更加正确的软件开发方法，通过在开发阶段采用一些科学的管理手段和正确有效的开发方式，来降低软件缺陷的引入数量。另一方面，测试需要对缺陷的检测。工程师可以尝试通过一些持续集成的手段，尽早地开展测试活动，还可以加入自动化测试技术，通过不断、反复地测试来发现更多的缺陷。这两个方面都是有效提高软件质量的重要手段。

1.2 软件测试的类型

软件测试按照测试类型，可以划分为：单元测试（Unit Tests）、集成测试（Integration Tests）和系统测试（System Tests）。下面将分别详细阐述。

1.2.1 单元测试

单元测试是指对软件系统中最小可测试单元进行的检查和验证。对于“单元测试”中“单元”的解释，要根据实际情况去判定，一般来说是指功能不可再分割的模块或者函数。

单元测试在软件开发流程中占有一席之地。在过去的十几年中，单元测试框架的开发者们不断地完善测试技术，并且建立了一些新手段，用以将单元测试活

1 这句话在很多软件测试方面的教材中都有引用，已经完全找不到最早的出处。

动集成于软件开发的流程之中。最早的单元测试框架出现在Smalltalk语言中，单元测试的框架为SUnit，由Kent Beck发明。他也是JUnit测试框架的创始人，之后JUnit取得了巨大的成功，单元测试方法也开始流行起来，慢慢地在各个语言版本中都有了单元测试的身影，逐渐形成了xUnit体系。在Objective-C语言中最早的xUnit测试框架是OCUnit。本书将在后面的章节中简要介绍如何使用OCUnit来写单元测试，并会介绍几款类似于OCUnit的单元测试工具。

1.2.2 集成测试

集成测试是单元测试的逻辑扩展，最简单的形式是把两个已经测试过的单元组合成一个组件，并测试它们之间的接口。从这层意义上讲，组件是指多个单元的集成聚合。在现实方案中，许多单元组合成组件，而这些组件又聚合成程序的更大部分。方法是测试片段的组合，并最终扩展成进程，将模块与其他组的模块一起测试。最后，将构成进程的所有模块一起测试。

在iOS软件开发中，集成测试主要被简单地分为API接口测试和iOS功能集成测试。API接口测试是指，若一个iOS程序以网络请求的方式使用到了后台服务的功能，测试时需要验证网络的请求以及响应是否符合预期。iOS功能集成测试其实就是功能测试。在一个iOS程序中，有很多功能是在UI界面上体现的，所以在功能测试中测试的重点将会是UI的测试。本书将分别讲述在iOS系统中如何自动化测试iOS的本地应用程序（Native Application）和iOS的Web应用程序（Web Application）。

1.2.3 系统测试

系统测试（System Test）是将已经确认的软件、计算机硬件、外设、网络等其他元素结合在一起，进行信息系统组装测试和确认测试。系统测试是针对整个产品系统进行的测试，目的是验证系统是否满足需求规格的定义，找出与需求规格不符或相矛盾的地方，从而提出更加完善的方案。系统测试发现问题之后要经过调试找出错误的原因和位置，然后进行改正。

在iOS平台上，系统测试阶段需要从硬件环境和系统平台两个角度分别进行系统确认测试。在硬件环境方面主要考虑网络环境、所支持设备中性能较差的设备及屏幕分辨率等硬件环境因素。在系统平台方面，主要考虑的是在不同系统平台上的表现是否相同。

试读结束：需要全本请在线购买： www.ertongbook.com