

单元测试与自动化

李智维

简介

- 李智维
- 21 岁
- 中学参加算法竞赛
- 大二休学在 LeanCloud 工作了一年半，联合创立了 Reviewcode.cn，和聚美副总裁创业过一段时间
- Android、iOS、前端、后端都做过一些线上产品
- 现在创业做「趣直播」，知识直播平台，上线一个月，用户5200，流水4万，主播9成分成



一个测试例子

保存一个对象到 LeanCloud

```
AVObject *todo = [[AVObject alloc] initWithClassName:@"Todo"];  
[todo setObject:@"做完PPT" forKey:@"content"];  
[todo save];
```

网站控制台

数据 

Class Name

_Conversation 1.2k

_File 0

_Followee 0

_Follower 0

_Installation 40

_Role 0

_User 0

Todo 1

LeanCloud 从 9 月 18 日开启余额检查和欠费自动停机机制：因为中秋假期，原定从 9 月 15 日起开启余额检查，服务将自动停止。详情请阅读：<https://blog.leancloud.cn/5060/>

添加行 删除行 添加列 查询 刷新 其他 ▾ 编辑单元格

<input type="checkbox"/>	objectId STRING	content STRING	createdAt DATE
<input type="checkbox"/>	581e2b6a2f301e005c182be2	做完PPT	2016-11-06 02:56:42



如何确保代码永远正确?

从 LeanCloud 获取该对象，看是否存在且一致

完整的测试代码

```
✓ 699 - (void)testSave {
700
701     AVObject *todo = [[AVObject alloc] initWithClassName:@"Todo"];
702     [todo setObject:@"做完PPT" forKey:@"content"];
703     BOOL saved = [todo save];
704     XCTAssertTrue(saved);
705
706     AVObject *fetchedTodo = [AVObject objectWithClassName:@"Todo" objectId:todo.
        objectId];
707     [fetchedTodo fetch];
708     XCTAssertEqualObjects([fetchedTodo objectForKey:@"content"], @"做完PPT");
709
710 }
```


这段测试发生了什么？

到 [dataTask resume] 发网络请求时经历了这些层级调用

▼ Thread 1 Queue: com.apple.main-thread (serial)

- 0 -[AVPaasClient performRequest:success:failure:]
- 1 -[AVPaasClient performRequest:saveResult:block:retryTimes:]
- 2 -[AVPaasClient performRequest:saveResult:block:]
- 3 -[AVPaasClient postBatchSaveObject:headerMap:eventually:bl...
- 4 -[AVObject sendBatchRequest:eventually:error:]
- 5 -[AVObject saveWithOptions:eventually:verifyBefore:error:]
- 6 -[AVObject saveWithOptions:eventually:error:]
- 7 -[AVObject saveWithOptions:error:]
- 8 -[AVObject save:]
- 9 -[AVObject save]
- 10 -[AVStatusTest testSave]

涉及到的逻辑

- 更改了哪些字段，生成请求内容的逻辑
- 缓存逻辑
- 批量请求的逻辑
- **POST** 请求逻辑
- 每层的错误处理

几行测试代码确保了内部的
成百上千行代码运行基本正
确

**10秒的时间就能验证原
先需要3分钟验证的代码**

测试来代替手工验证

- 我改了 `save` 里的代码，到底有没有改错呢？现在还能正确执行吗？还能保存到服务器吗？
- 跑一遍测试

对于复杂的情况，测试可以代替繁琐的手工验证

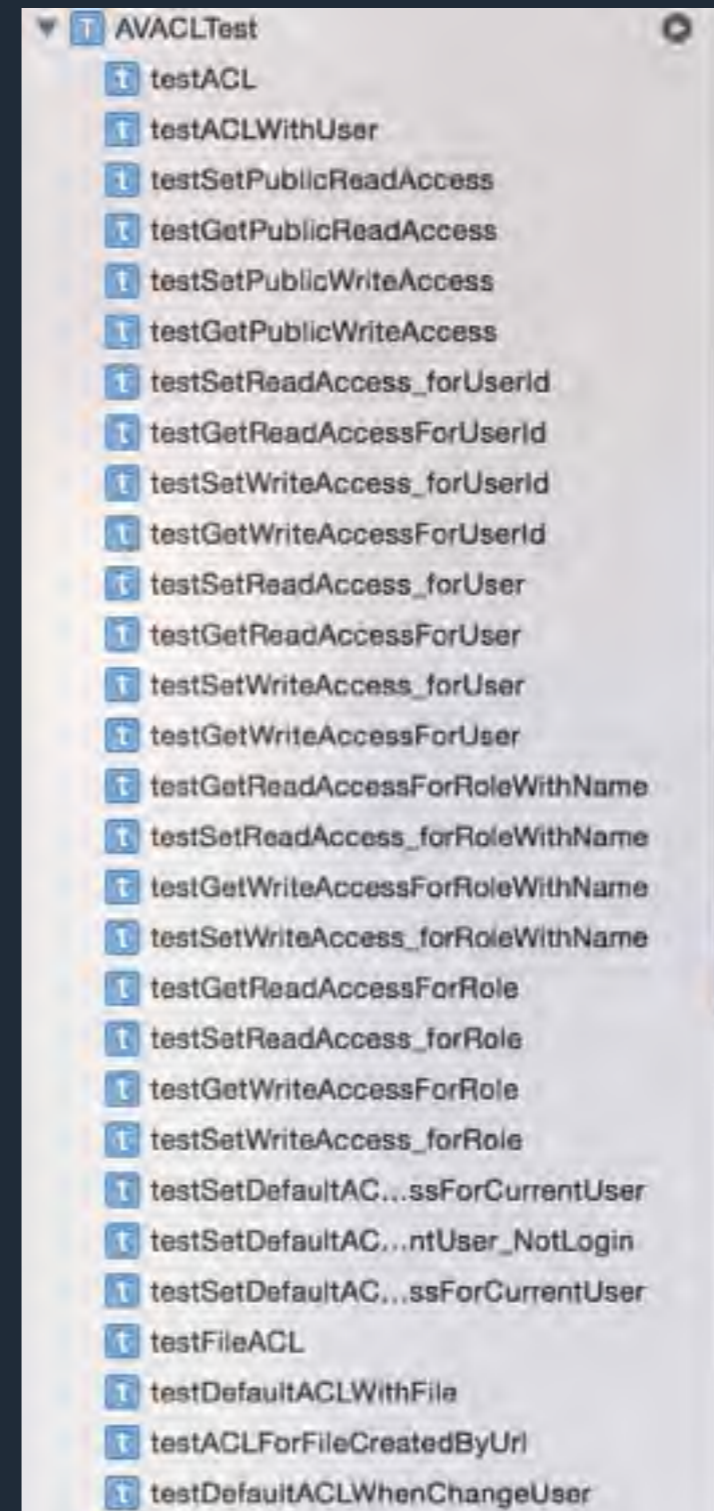
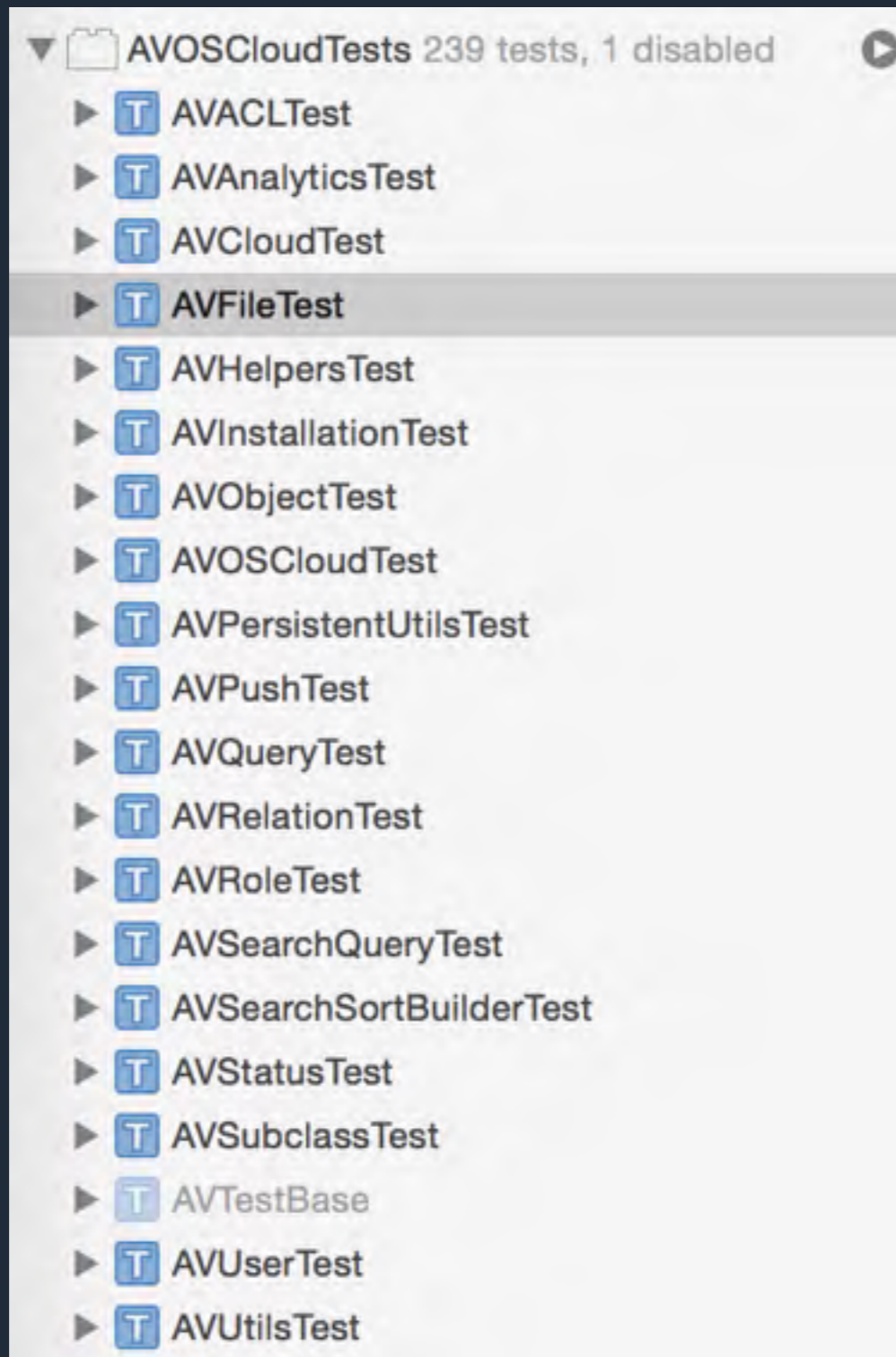
测试的本质

- 让强大的计算机来帮助你低效的手工验证

从此可以放心改 `save` 里的 代码

- 如果改错了，比如没有保存到服务器，那么会获取对象失败
- 改完内部逻辑之后，跑一遍单元测试，能放心当前的改动没有影响旧的测试
- 如果在其他地方不小心

LeanCloud iOS SDK



LeanCloud 的测试情况


(一)


- 存储框架，数据层，适合写单元测试
- 项目早期就有单元测试
- 提交一个 PR，Jenkins 上跑单元测试
- 覆盖率约 80%


LeanCloud 的测试情况(二)


- 1) 直接写测试，来检验正在开发的新接口。2) 直接写测试，来重现 Bug。
- 积累的测试越多越强大。
- 两种场景：改完一些代码后，本地跑相关测试，发现某些不通过；Jenkins 自动化跑测试的时候，发现最新改动引起其它问题，某些测试不通过
- 放心地发布与重构，省下手动验证的时间


提交 PR 跑测试


 6 Open ✓ 286 Closed


 **fix bug:still online when enter background** ✗
#517 opened a day ago by ChenYilong

 **Add code signature for dynamic framework** ✗
#511 opened 4 days ago by tang3w

 **Improve Test** ✗
#510 opened 4 days ago by lzwjava

 **fix main thread blocked problem** ✓
#507 opened 5 days ago by lzwjava

 **Fix duplicated product name** ✓
#504 opened 5 days ago by tang3w

 **Fix duplicated bundle identifier** ✓
#503 opened 5 days ago by tang3w

输入暗号触发测试

add auto test script #428

Closed lzwjava wants to merge 1 commit into `leancloud:master` from `lzwjava:master`

Conversation 6 Commits 1 Files changed 1

lzwjava commented on 9 Oct

No description provided.

add auto test script ✗ 88d2100

lzwjava commented on 9 Oct

!build-me

Use github hooks for build triggering

Trigger phrase

Only use trigger phrase for build triggering

Close failed pull request automatically?

Test Flow 1

```
1184 - (void)testSaveDescriptionKey {
1185     NSString * className = NSStringFromClass([self class]);
1186     NSError * error = nil;
1187     AVObject * object = [AVObject objectWithClassName:className];
1188     [object setObject:@"test" forKey:@"description"];
1189     [object save:&error];
1190     XCTAssertNil(error, @"%@", error);
1191     AVQuery* query = [AVQuery queryWithClassName:className];
1192     [query getObjectInBackgroundWithId:object.objectId block:^(AVObject *model, NSError *error) {
1193         XCTAssertNil(error, @"%@", error);
1194         XCTAssertEqualObjects([model objectForKey:@"description"], @"test");
1195         NOTIFY
1196     }];
1197     WAIT
1198 }
```

用户反应说保存有 description 为 key 的对象时会报错，我写了测试来验证，接着找到问题，修复后，保留下了测试。

Test Flow 2

```
1166 - (void)testSetNull {  
1167     AVObject *object = [self objectForAVObjectTest];  
1168     [object setObject:[NSNull null] forKey:@"name"];  
1169     NSError *error;  
1170     [object save:&error];  
1171  
1172     [AVObject setConvertingNullToNil:NO];  
1173     AVObject *fetchedObject = [self fetchObjectById:object.objectId];  
1174     NSString *name = [fetchedObject objectForKey:@"name"];  
1175     XCTAssertNotNil(name);  
1176     XCTAssertTrue([name isEqual:[NSNull null]]);  
1177  
1178     [AVObject setConvertingNullToNil:YES];  
1179     NSString *name1 = [fetchedObject objectForKey:@"name"];  
1180     XCTAssertNil(name1);  
1181     XCTAssertFalse([name1 isEqual:[NSNull null]]);  
1182 }
```

开发新接口时，写了相应的代码后，我加入了测试，看看刚写的代码是否工作正常。

Test Flow 3

- 改了 AVObject.m 的代码后，我运行 AVObjectTest.m 测试，看看改动是否造成测试不通过

Test Flow 4

- 提交 PR, Jenkins 自动跑测试

总结：为什么要写单元测试

- 当发现自己改了代码后，经常需要手动验证是否正常时
- 发现改了代码后，经常引起其它地方的 Bug 时
- 多人合作的项目、可能交给其他人的项目
- 高质量、流行的开源项目一般都有丰富的单元测试

如何写单元测试

- 模块化代码，数据层和 UI 层分离
- 最少的测试代码达到最高的覆盖率
- 异步处理
- 框架选择
- 覆盖率

框架评析

- **expecta** `expect(error).not.beNil()`
- **specta** `describe("") it("")`
- **Kiwi** `describe("") it("")`
- **TDD 和 BDD 框架的坏处**: 和 Xcode 结合不好, 比如没有测试按钮, 左边栏没有列全单元测试。

异步处理

```
130 - (void)waitNotification:(const void *)notification {
131     NSString *name = [NSString stringWithFormat:@"%p", notification];
132     [self expectationForNotification:name object:nil handler:nil];
133     [self waitForExpectationsWithTimeout:AV_YEAR_SECONDS handler:nil];
134 }
135
136 - (void)postNotification:(const void *)notification {
137     NSString *name = [NSString stringWithFormat:@"%p", notification];
138     [[NSNotificationCenter defaultCenter] postNotificationName:name object:nil];
139 }
```

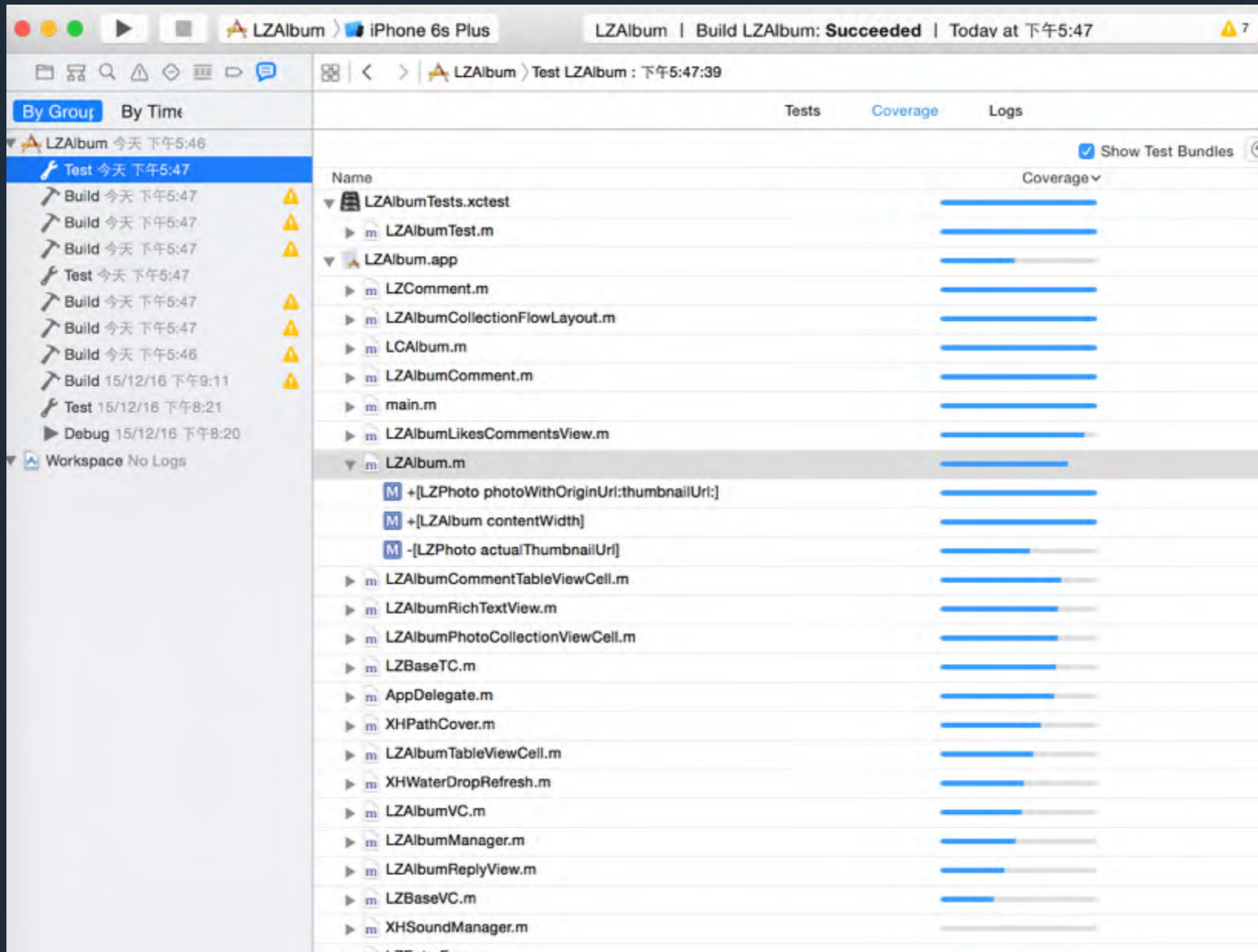
```
1 #define WAIT
2 do {
3     [self expectationForNotification:@"LCUnitTest" object:nil handler:nil]; \
4     [self waitForExpectationsWithTimeout:60 handler:nil]; \
5 } while(0);
6
7 #define NOTIFY
8 do {
9     [[NSNotificationCenter defaultCenter] postNotificationName:@"LCUnitTest" object:nil];
10 } while(0);
11
```

最终版本

```
11 #define assertTrue(expr)          XCTAssertTrue((expr), @"")
12 #define assertFalse(expr)       XCTAssertFalse((expr), @"")
13 #define assertNil(a1)            XCTAssertNil((a1), @"")
14 #define assertNotNil(a1)        XCTAssertNotNil((a1), @"")
15 #define assertEquals(a1, a2)     XCTAssertEqual((a1), (a2), @"")
16 #define assertEqualsObjects(a1, a2) XCTAssertEqualObjects((a1), (a2), @"")
17 #define assertNotEqual(a1, a2)   XCTAssertNotEqual((a1), (a2), @"")
18 #define assertNotEqualObjects(a1, a2) XCTAssertNotEqualObjects((a1), (a2), @"")
19 #define assertAccuracy(a1, a2, acc) XCTAssertEqualWithAccuracy((a1), (a2), (acc))
20
21 #define WAIT
22 do {
23     [self expectationForNotification:@"LCUnitTest" object:nil handler:nil]; \
24     [self waitForExpectationsWithTimeout:60 handler:nil]; \
25 } while(0);
26
27 #define NOTIFY
28 do {
29     [[NSNotificationCenter defaultCenter] postNotificationName:@"LCUnitTest" object:nil];
30 } while(0);
```

```
26 - (void)testFindAlbums {
27     [[LZAlbumManager manager] findAlbumWithBlock:^(NSArray *objects, NSError *error)
28     {
29         assertNil(error);
30         assertTrue(objects.count > 0);
31         NOTIFY
32     }];
33     WAIT
34 }
```

覆盖率



覆盖率报告

- Xcode 7 开始，自带覆盖率报告功能
- 设置要点：1) Scheme 中开启 Gather Coverage Data 2) 针对于 App Target 来测试，而非 Test Target
- <https://www.bignerdranch.com/blog/weve-got-you-covered/>

Gather Coverage Data

The screenshot shows the Visual Studio Test Explorer interface. On the left, a sidebar contains several options: Build (2 targets), Run (Debug), Test (Debug, highlighted in blue), Profile (Release), Analyze (Debug), and Archive (Release). The main area is divided into three tabs: Info, Arguments, and Diagnostics. The Info tab is active, showing the following settings:

- Build Configuration: Debug
- Code Coverage: Gather coverage data (highlighted with a pink box)
- Debugger: Debug executable
- Debug Process As: Me (markd), root

Below the settings is a table with the following columns: Tests, Test Application Data, Test Location, and Test. The table contains one entry:

Tests	Test Application Data	Test Location	Test
▶ MusicJot Tests	None	None	<input checked="" type="checkbox"/>

At the bottom of the interface, there are expand/collapse icons (+ and -) and a search box.

远程自动化测试

- 搭建 Jenkins
- 本地空闲的 Mac 电脑或将 Mac 电脑拿到数据中心
- 测试脚本，设置项目
- Github PR 构建插件，<https://github.com/janinko/ghprb>
- 配置 Jenkins Job
- 本地测试能发挥出测试的 70% 好处，远程自动化则 100%

配置 Jenkins

- 要让 Jenkins 能读取你的代码
- 配置 Hook，当提交 GitHub PR 时，发送相应的事件到 Jenkins 上
- Jenkins 要知道是哪一个 PR，来拉取最新代码
- 设置 Jenkins 测试脚本
- 当测试失败之后，要让 Jenkins 通知 GitHub 页面
- 测试失败后，可发送通知到 Slack 或邮件

Jenkins 如何知道测试失败或成功

- 根据进程返回值
- 如果是 0，则 Jenkins 认为测试成功，否则失败
- Xcode 7 之前，尽管 `xcodebuild test` 失败了，依然返回 0；Xcode 7 之后，失败了则返回 65

```

test-all.sh
1  #!/bin/bash -v
2
3  xcodebuild clean \
4      -workspace AVOS.xcworkspace \
5      -scheme AVOSCloudIMTests \
6
7  script -q -t 0 test.log \
8      xcodebuild \
9      -workspace AVOS.xcworkspace \
10     -scheme AVOSCloudIMTests \
11     -configuration Release \
12     -sdk iphonesimulator \
13     test
14
15  grep "\*\* TEST FAILED \*\*" test.log > /dev/null
16  if [[ $? -eq 0 ]]; then
17      rm test.log
18      exit 1
19  fi
20
21  xcodebuild clean \
22      -workspace AVOS.xcworkspace \
23      -scheme AVOSCloudTests \
24
25  script -q -t 0 test.log \
26      xcodebuild \
27      -workspace AVOS.xcworkspace \
28      -scheme AVOSCloudTests \
29      -configuration Release \
30      -sdk iphonesimulator \
31      test
32
33  grep "\*\* TEST FAILED \*\*" test.log > /dev/null
34  if [[ $? -eq 0 ]]; then
35      rm test.log
36      exit 1
37  else
38      rm test.log
39      exit 0
40  fi
41

```

Xcode 7之前的测试脚本

1. 检测是否有 TEST FAILED 字符串

Xcode 7 之后

```
test-all.sh *
1  #!/bin/bash -v
2
3  xcodebuild \
4      -workspace AVOS.xcworkspace \
5      -scheme AVOSCloudIMTests \
6      -sdk iphonesimulator \
7      clean \
8      test &&
9  xcodebuild \
10     -workspace AVOS.xcworkspace \
11     -scheme AVOSCloudTests \
12     -sdk iphonesimulator \
13     clean \
14     test
15
```

Travis CI

- 设置 Test Scheme 为 shared
- Travis 上连接 GitHub 账号，读取项目
- .travis.yml
- Travis 免费版可以给开源项目测试、收费版给私有项目测试
- Travis 测试比较慢
- 如果会用 Jenkins 自己搭自动化测试框架，Travis 眨眼间就会

远程打包发布

- 发布脚本
- Jenkins 上配置 Job 运行
- LeanCloud 上是发布框架，远程打包 SDK 推送到 GitHub 仓库，同步网站更新
- 针对于 App，似乎没必要远程打包发布

实现思路

- 让 Jenkins 读取你的代码
- 读取发布版本
- **unlock keychain**, 构建一个项目会弹出输入密码框, 来读取签名的 **keychain**

unlock-keychain

```
# Unlock the build certificate  
execute_command('security unlock-keychain -p "xxx" ~/Library/Keychains/login.keychain')
```

在命令行中解 **Keychain**。一般会弹框输入密码

LeanCloud iOS SDK

- <https://github.com/leancloud/objc-sdk>
- 自动化测试
- 自动化打包 SDK
- 自动化生成、推送 podspec
- 自动化发布

脚本:命令行build

```
def build_target(target_name, sdk, build_dir, archs, misc='')
  command = <<-EOC.strip.gsub(/^[\t]+/, '')
  xcodebuild \
  -project #{project_path} \
  -target #{target_name} \
  -configuration #{configuration}
  EOC

  command += " -sdk #{sdk}" unless empty_string?(sdk)
  command += " CONFIGURATION_BUILD_DIR=\"#{build_dir}\"" unless empty_string?(build_dir)
  command += " ARCHS=\"#{archs}\"" unless empty_string?(archs)
  command += " #{misc}" unless empty_string?(misc)

  execcmd command
end
```

脚本：命令行build

```
def build_ios_sdk(target_name)
  clean_target target_name

  prefix = File.join(slice_build_path, target_name)

  build_target target_name, 'iphonesimulator', File.join(prefix, 'iphonesimulator'), 'i386 x86_64'
  build_target target_name, 'iphoneos', File.join(prefix, 'iphoneos'), 'armv7 armv7s'

  merge_frameworks(prefix, target_name)
end
```

脚本：合并framework

```
def merge_frameworks(prefix, target_name)
  product_name = product_name(target_name)

  frameworks = Dir.glob(File.join(prefix, '**', "#{product_name}.framework"))
  first_framework = frameworks.first

  return if first_framework.nil?

  output = File.join(build_path, "#{target_name}.framework")
  execmd "cp -RLp #{first_framework} #{output}"

  return if frameworks.count == 1

  executables = frameworks.map { |framework| File.join(framework, product_name) }

  lipo File.join(output, product_name), *executables
end
```

脚本：拉取远程代码

```
remote_url = 'git@github.com:leancloud/objc-sdk.git'

tags = execute_command "git ls-remote --tags #{remote_url}"
abort 'Git tag not found on remote repository. You can push one.' unless tags.include?

commit_sha = tags[/([0-9a-f]+\srefs\/tags\/#{version})/, 1]

temp_remote = "_origin-temp-remote-for-deployment"
temp_branch = "_branch-temp-branch-for-deployment"

execute_command "git remote remove #{temp_remote} >/dev/null 2>&1", false
execute_command "git remote add #{temp_remote} #{remote_url} >/dev/null 2>&1", false
execute_command "git fetch #{temp_remote} --tags >/dev/null 2>&1"
execute_command "git checkout -b #{temp_branch} #{commit_sha} >/dev/null 2>&1"
```


脚本：生成 Podspec

```
template = File.read('Podspec/AVOSCloud.podspec.mustache')

podspec = Mustache.render template, {
  'version'           => version,
  'source_files'      => "'AVOS/AVOSCloud/**/*.{h,m}'",
  'resources'         => "'AVOS/AVOSCloud/AVOSCloud_Art.inc'",
  'public_header_files' => file_list_string(public_header_files, 4),
  'osx_exclude_files' => file_list_string(osx_exclude_files, 4),
  'watchos_exclude_files' => file_list_string(watchos_exclude_files, 4),
  'xcconfig'          => "'{'OTHER_LDFLAGS' => '-ObjC'}'"
}

write 'AVOSCloud.podspec', podspec
```

脚本：自动化推送 Podspec

```
def push_podspec_in_path(path)
  iterate_r(path, 'podspec') do |file|
    podspec_name = File.basename(file, ".podspec")
    podspec_version = version[1..-1]
    exists = is_podspec_exists(podspec_name, podspec_version)
    if exists
      log("#{podspec_name} #{podspec_version} exists!")
      next
    else
      log("#{podspec_name} #{podspec_version} not exists, now try to push it")
    end

    ok = false
    for i in 0..10
      ok = system("pod trunk push #{file} --verbose --allow-warnings")
      if ok
        log("succeed to push #{file}")
        break
      else
        log("failed to push #{file}")
      end
    end

    if !ok
      exit_with_info('fail to push podspec, please check.')
    end
  end
end
```

重复推送 Podspec

```
def is_podspec_exists(name, version)
  podspec_url = "https://github.com/CocoaPods/Specs/blob/master/Specs/#{name}/#{version}/#{name}"
  url = URI.parse(podspec_url)
  req = Net::HTTP::Get.new(url.to_s)
  http = Net::HTTP.new(url.host, url.port)
  http.use_ssl = true
  res = http.request(req)
  res.code == '200'
end
```

lipo

- 查看 **framework** 的编译架构 **lipo -info**
- **lipo** 合并不同架构的 **framework**

代码质量

代码质量

- 一致性
- 不写重复代码
- 少用全局变量
- 逻辑分层
- 每一行代码都应有它存在的意义

零碎的知识

- **lipo** 使用
- 清空所有的生成文件，**Clean Build Folder**
- **Xcode** 快捷键，根据当前文件展开左侧导航、**Open Quickly**、查看 **Macro** 预编译、**.h** 与 **.m** 文件跳转
- **Pod** 高级用法
- 如何制作 **Framework**
- **Xcode Configurations**
- **Reveal In GitHub** 插件
- **Instrument** 工具使用，定位代码

Xcode 快捷键

- **Shift + Command+ J**
- **Shift + Command + Up/Down**
- **Shift + Command + O**
- **Emacs Way**
- **根据语法来删除, Delete expression backward**
- **preferences Xcode 设置中, 搜索快捷键的 keyword**

Delete expression backward

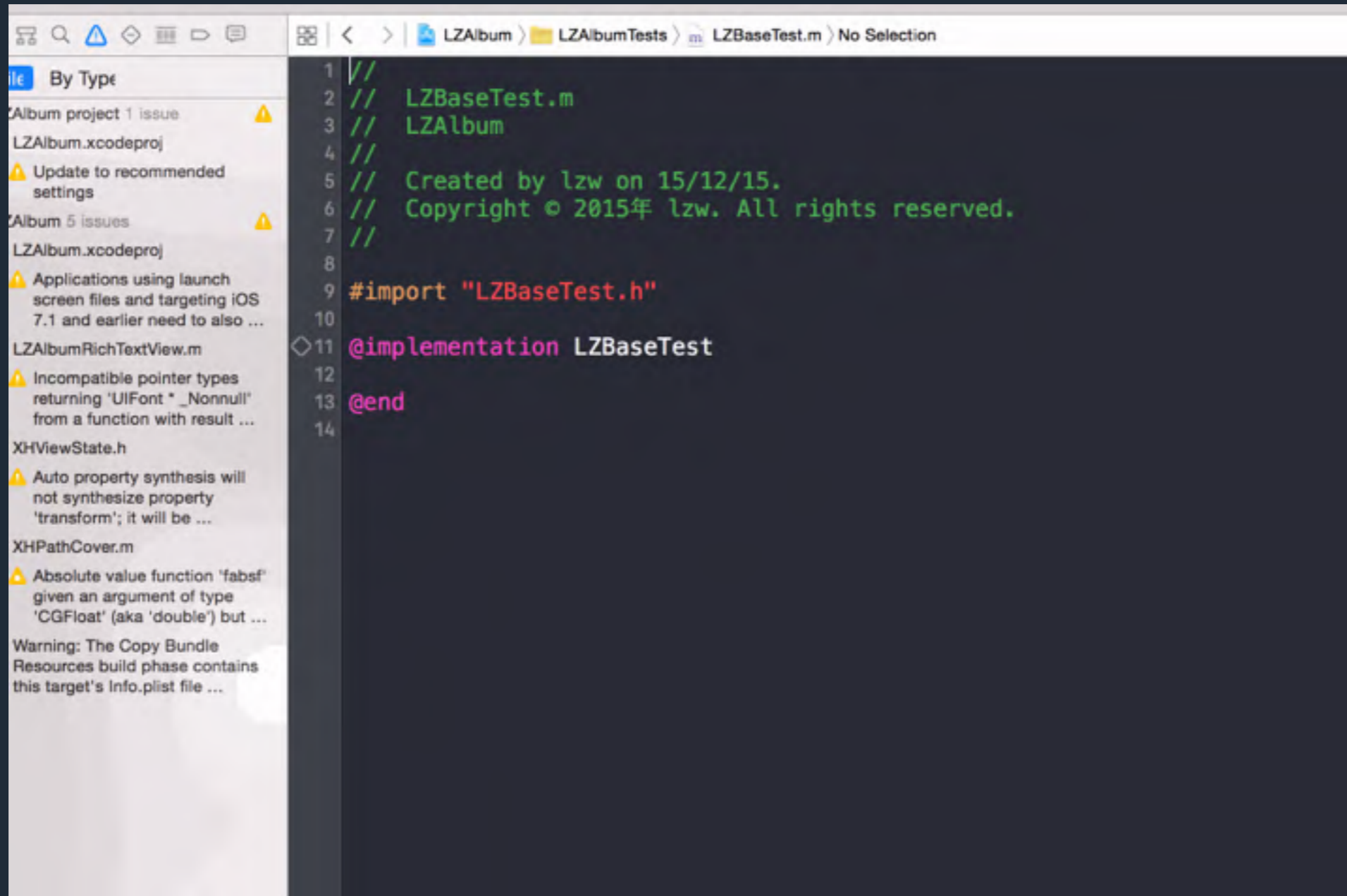
```
62 - (void)testSaveDescriptionKey {
63     NSString * className = NSStringFromClassF
64     NSError * error = nil;
65     AVObject * object = [AVObject ob
66     [object setObject:@"test" forKey
67     [object save:&error];
68     XCTAssertNil(error, @"%@", error
69     AVQuery* query = [AVQuery queryW
70     [query getObjectInBackgroundWith
71         error) {
72         XCTAssertNil(error, @"%@", e
73         XCTAssertEqualObjects([model
74         NOTIFY
75     }];
76     WAIT
77 }
```

```
1
2 - (void)testSaveDescriptionKey
3
```

Pod

- 本地引用 pod, 好处
- `—no-repo-update` `—verbose`
- 发布 pod、podfile 配置

Clean Build Folder



The screenshot shows the Xcode IDE interface. On the left, the 'Issues' pane displays several warnings, including 'Update to recommended settings' and 'Applications using launch screen files and targeting iOS 7.1 and earlier need to also ...'. The main editor area shows the code for 'LZBaseTest.m', which includes a copyright notice and a simple implementation of the 'LZBaseTest' protocol.

```
1 //
2 // LZBaseTest.m
3 // LZAlbum
4 //
5 // Created by lzw on 15/12/15.
6 // Copyright © 2015年 lzw. All rights reserved.
7 //
8
9 #import "LZBaseTest.h"
10
11 @implementation LZBaseTest
12
13 @end
14
```

- 1) Alt 键触发
- 2) 两小时捣鼓不如 Clean Build Folder 一下
- 3) Duplicate Symbol 等 Build 的问题时很好用

如何制作 Framework

- 动态库和静态库的区别
- 静态库如何用动态库伪装，包含 `.a` 和 `headers`
- 打包模拟器和真机都可以使用的 Framework

Instruments

The screenshot displays the Instruments application interface. At the top, the title bar reads "Instruments2". Below it, the target device is identified as "iPhone 6s Plus (9.2)" and the application being profiled is "LZAlbum". The status bar shows "Run 1 of 1" and a duration of "00:01:02".

The main area is divided into two panes. The top pane, titled "CPU Usage", shows a graph of CPU usage over time, with a blue line indicating the profiled process. The bottom pane, titled "Call Tree", shows a hierarchical view of the application's execution flow. The "Call Tree" pane is currently selected, and it shows a list of symbols with their running times, self times, and symbol names. The "Heaviest Stack Trace" pane on the right shows the stack trace for the selected symbol, listing the symbols and their running times in descending order.

Running Time	Self Time	Symbol Name
1814.0ms	74...	0.0 Main Thread 0xb9cf51
1718.0ms	70...	0.0 start libdyld.dylib
1718.0ms	70...	0.0 main LZAlbum
1718.0ms	70...	0.0 UIApplicationMain UIKit
1611.0ms	66...	0.0 GSEventRunModal GraphicsServices
1611.0ms	66...	0.0 CFRunLoopRunSpecific CoreFoundation
1611.0ms	66...	1.0 __CFRunLoopRun CoreFoundation
554.0ms	22.8%	0.0 __CFRUNLOOP_IS_SERVICING_THE_MAIN_DISPATCH_QUEUE__ CoreFoundation
554.0ms	22.8%	0.0 _dispatch_main_queue_callback_4CF libdispatch.dylib
554.0ms	22.8%	0.0 _dispatch_client_callout libdispatch.dylib
526.0ms	21.6%	0.0 _dispatch_call_block_and_release libdispatch.dylib
425.0ms	17.5%	0.0 _52-[AVQuery findObjectsWithBlock:waitUntilDone:error:]_block_invoke LZAlbum
418.0ms	17.2%	0.0 _40-[LZAlbumManager cacheUsersByIds:block:]_block_invoke LZAlbum
418.0ms	17.2%	0.0 _37-[LZAlbumManager findAlbumWithBlock:]_block_invoke72 LZAlbum
296.0ms	12.2%	0.0 -[LZAlbumManager iterateUsersInAlbums:block:] LZAlbum
275.0ms	11.3%	0.0 -[LZAlbumManager fillPointerUser:] LZAlbum
153.0ms	6.3%	0.0 +[AVObjectUtils copyDictionary:toObject:] LZAlbum
153.0ms	6.3%	0.0 -[_NSDictionaryM enumerateKeysAndObjectsWithOptions:usingBlock:] CoreFoundation
122.0ms	5.0%	0.0 +[AVObjectUtils objectSnapshot:] LZAlbum
20.0ms	0.8%	0.0 getter LZAlbum
1.0ms	0.0%	1.0 objc_retain libobjc.A.dylib
122.0ms	5.0%	0.0 _20-[LZAlbumVC refresh]_block_invoke LZAlbum
7.0ms	0.2%	0.0 _37-[LZAlbumManager findAlbumWithBlock:]_block_invoke LZAlbum
96.0ms	3.9%	0.0 _56-[AVPaasClient performCurl:saveResult:block:retryTimes:]_block_invoke_2 LZAlbum
4.0ms	0.1%	0.0 _55+[AVUser loginWithUsernameInBackground:password:block:]_block_invoke LZAlbum
1.0ms	0.0%	0.0 _69-[SDWebImageManager downloadImageWithURL:options:progress:completed:]_block_invoke66 LZAlbum
24.0ms	0.9%	0.0 CA::Layer::run_animation_callbacks(void*) QuartzCore
3.0ms	0.1%	0.0 CABackdropDidChange(void*) QuartzCore
1.0ms	0.0%	0.0 Block_release libsystem_blocks.dylib
458.0ms	18.8%	1.0 __CFRunLoopDoSources0 CoreFoundation
331.0ms	13.6%	2.0 __CFRunLoopDoObservers CoreFoundation
228.0ms	9.3%	0.0 __CFRunLoopDoTimer CoreFoundation

Running Time	Symbol Name
1814.0	Main Thread 0xb9cf51
1718.0	start
1718.0	main
1718.0	UIApplicationMain
1611.0	GSEventRunModal
1611.0	CFRunLoopRunSpecific
1611.0	__CFRunLoopRun
554.0	__CFRUNLOOP_IS_SERVICING_THE_MAIN_DISPATCH_QUEUE__
554.0	_dispatch_main_queue_callback_4CF
554.0	_dispatch_client_callout
526.0	_dispatch_call_block_and_release
425.0	_52-[AVQuery findObjectsWithBlock:waitUntilDone:error:]_block_invoke
418.0	_40-[LZAlbumManager cacheUsersByIds:block:]_block_invoke
418.0	_37-[LZAlbumManager findAlbumWithBlock:]_block_invoke72
296.0	-[LZAlbumManager iterateUsersInAlbums:block:]
275.0	-[LZAlbumManager fillPointerUser:]
153.0	+[AVObjectUtils copyDictionary:toObject:]
153.0	-[_NSDictionaryM enumerateKeysAndObjectsWithOptions:usingBlock:]
122.0	+[AVObjectUtils objectSnapshot:]
20.0	getter
1.0	objc_retain
122.0	_20-[LZAlbumVC refresh]_block_invoke
7.0	_37-[LZAlbumManager findAlbumWithBlock:]_block_invoke
96.0	_56-[AVPaasClient performCurl:saveResult:block:retryTimes:]_block_invoke_2
4.0	_55+[AVUser loginWithUsernameInBackground:password:block:]_block_invoke
1.0	_69-[SDWebImageManager downloadImageWithURL:options:progress:completed:]_block_invoke66
24.0	CA::Layer::run_animation_callbacks(void*)
3.0	CABackdropDidChange(void*)
1.0	Block_release
458.0	__CFRunLoopDoSources0
331.0	__CFRunLoopDoObservers
228.0	__CFRunLoopDoTimer

好用工具介绍

- **Reveal + 越狱 iPhone**, 分析任意 App 的 UI 界面
- **Flex + 越狱 iPhone**, 分析任意 App 的网络请求、UI界面、本地文件、NSUserDefaults、Log 等等

Flex 1




Flex 2



Thank you!

Thanks to 彦祖、
ufosky、tang3w、sunng87、iOS 程序猿、冉神
Thanks to LeanCloud CTO 允许我讲内部测试流程



李智维 

中国



扫一扫上面的二维码图案，加我微信

微信
lzwjava