

# 豆瓣iOS自动化测试实践和经验

豆瓣iOS自动化测试实践和经验

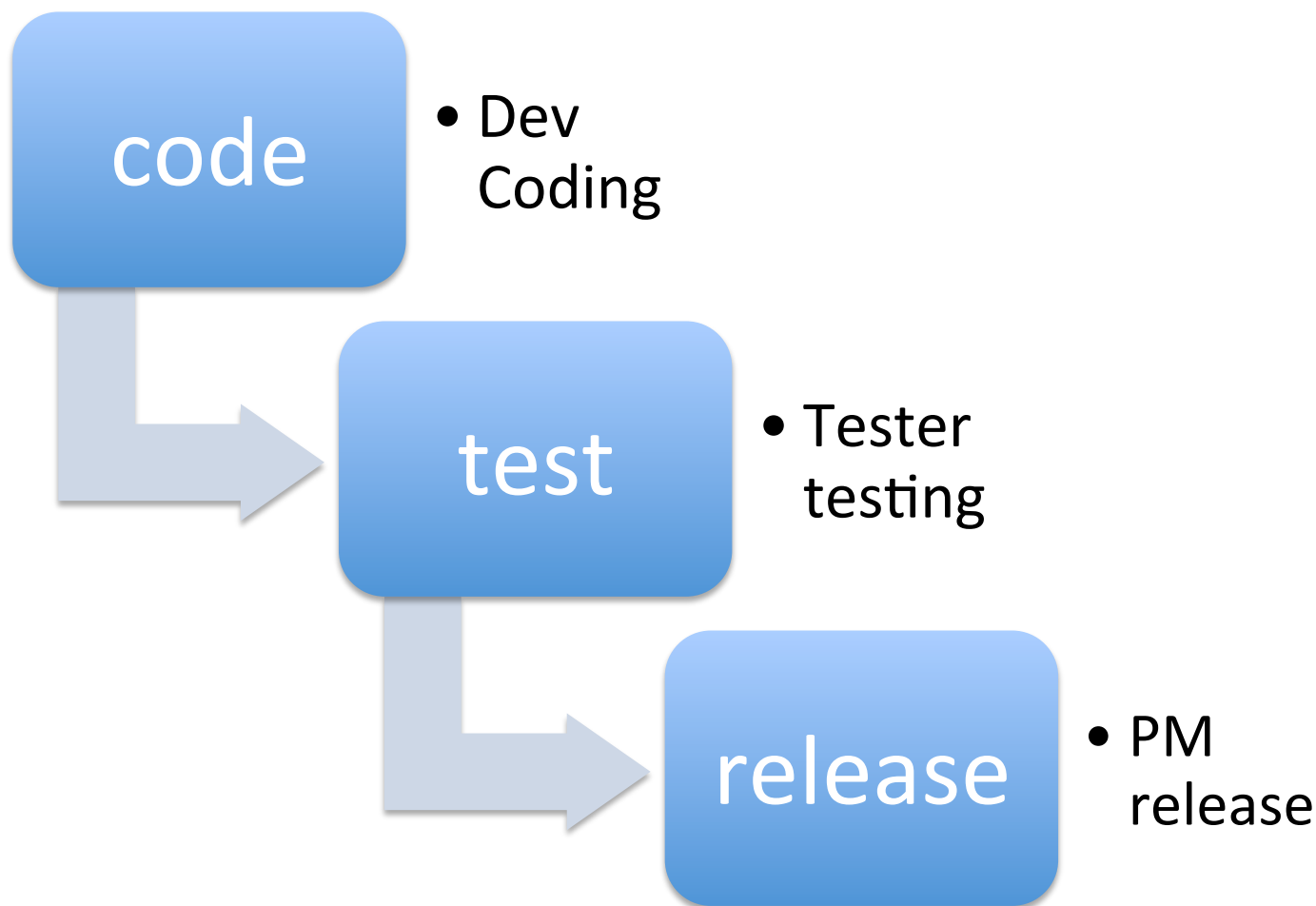
豆瓣QA： 半嵴

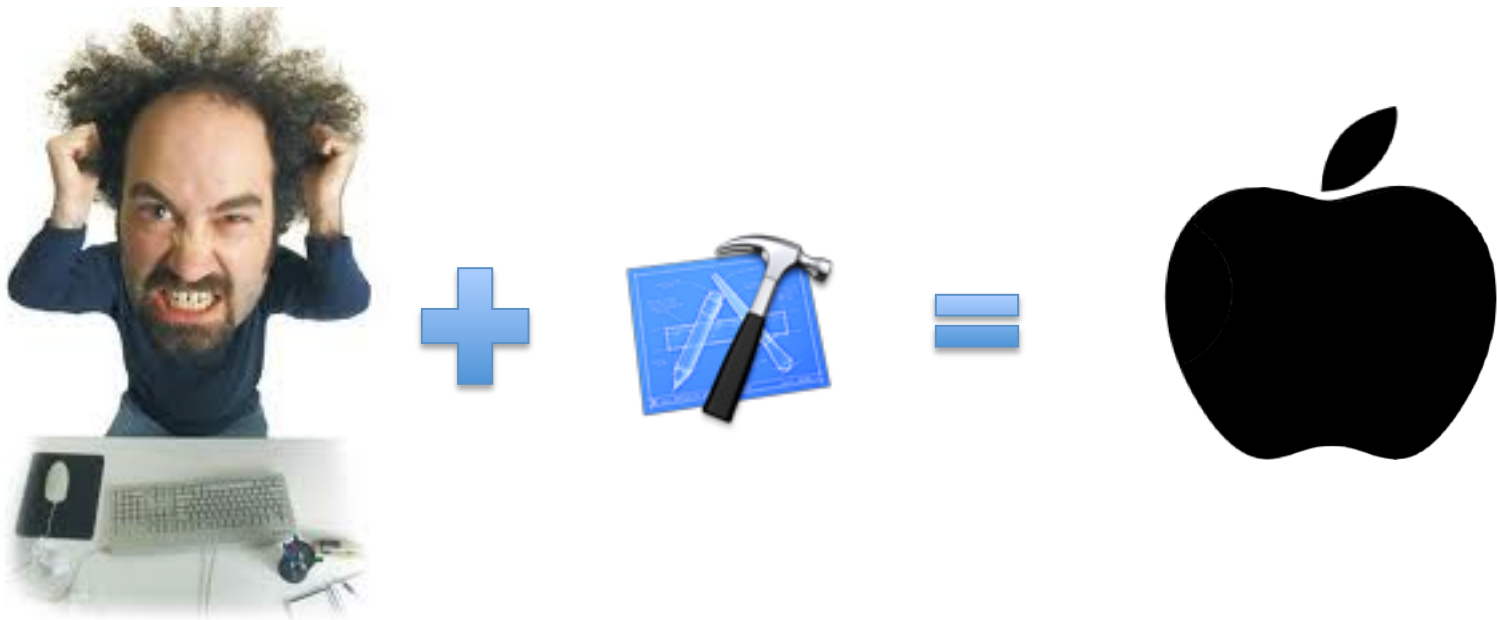
# iOS自动化测试

- 持续构建iOS应用
- 自动化测试工具评估
- Ynm3k功能简介

# 持续构建iOS应用

# 传统的测试发布流程

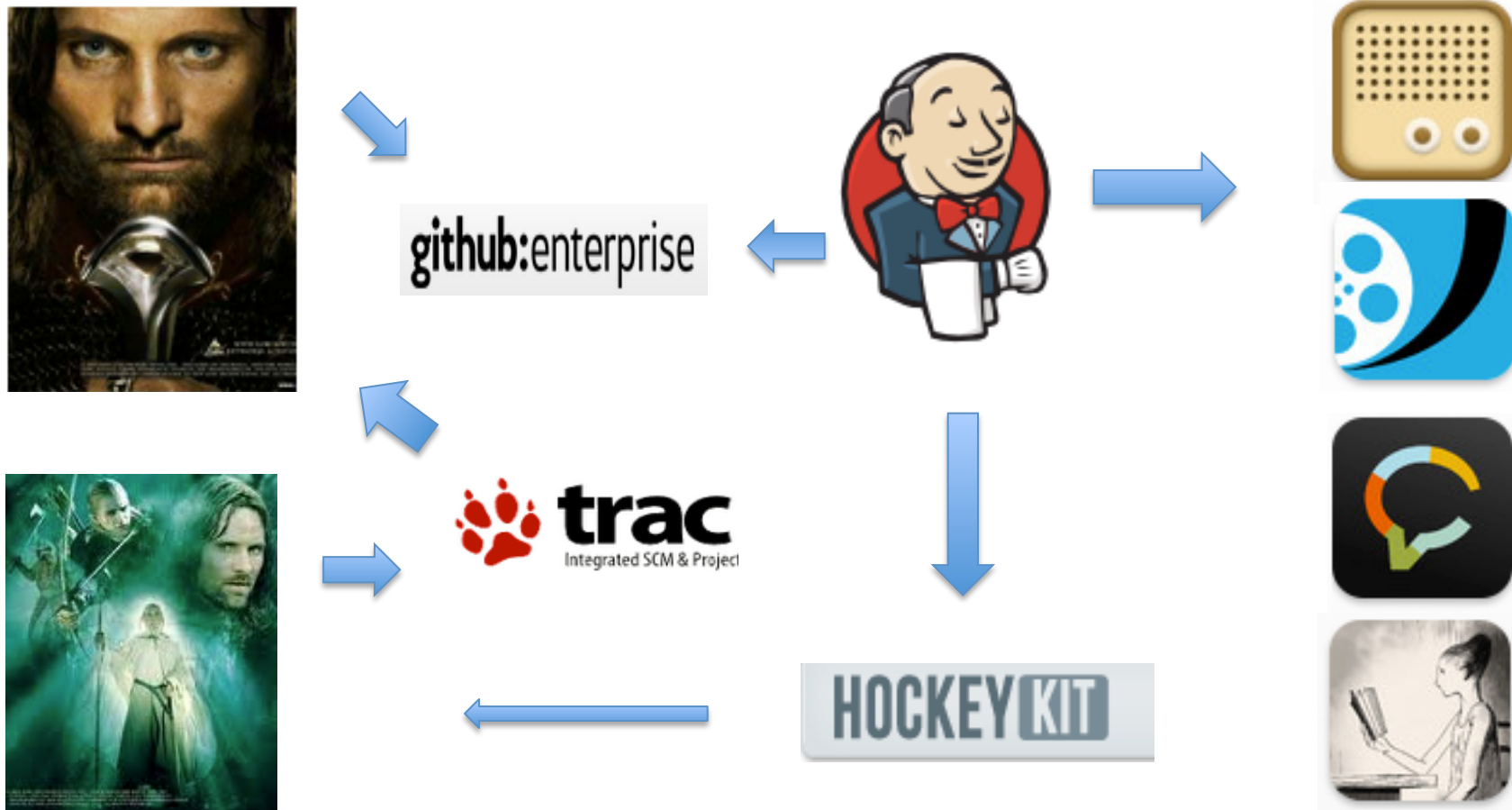




原始的让你抓狂的流程

# 发现问题

- UED 和PM无法在测试阶段无缝的介入
- 编译打包的过程没有纳入测试管理
- 测试人员很难建立被测app和版本管理工具中的版本号之间的关联关系
- 整个流程需要更加容易执行，并且需要解放dev，让dev把更多的精力放在coding上。



解决方案——工具化，自动化

# 实际操作



Wait



豆瓣电影

Then

**Version:** ad-hoc (ad-hoc-version-number)

**Size:** 1.5 MB

**Released:** 05/09/2012 07:04:39

Install Profile

Install Application



# 实现方法

- [Jenkins](#) + [hockey](#)

步骤:

- 1 下载代码
- 2 使用Xcode integration生成ipa
- 3 把生成的ipa拷贝到hockey的目录下

# iOS 自动化测试工具评估

# 豆瓣需要测试工具的特点

- 框架方面： 标准的xUnit框架结构
- UI识别： 以扩展库的形式提供UI识别方法
- 启动方法： 需要有命令行启动方式
- 测试结果： xml或者html格式结果， 或者提供相关方法定制输出测试结果

# 商业工具

- [DeviceAnywhere](#)
- QTP
- .....

# Unit Test Tools

- [OCUnit](#)
- [GTM](#)
- [GHUnit](#)

# Acceptance Test Tools (open Source)

- [UISpec](#)  
[Frank](#)
- [KIF](#)

# Acceptance Test Tool

- UI Automation(官方)
- [FoneMonkey](#)

# FoneMonkey VS UI Automation

	FoneMonkey	UI Automation
框架结构	OCUnit	无
ui识别方式多样化	无	有，支持至少2种方式
命令行启动测试	有	有
输出测试报告集成CI	有	无



# UI Automation Advanced

- [Tuneup.js](#)
- [Jasmine with UI Automation](#)

# 结论

- UI Automation

官方提供，还提供了网络，内存等方面的监控。不满足框架的部分，我们可以改造，我们可以扩展。

Ynm3k来了

# 为什么是ynm3k



# Ynm3k功能列表

功能结构	解决方案	详细说明
xUnit	引入了YUITest	可以使用testSuite testcase等，并且可以使用YUITest的相关事件。
ui识别	引入了一个Finder，并且可以使用UI Automation 原生的API	原生API定位方式，写法过于复杂。并且在语义表达方式不友好。Finder模仿selenium的接口形式，对ui 识别的方法进行了封装。
命令行启动测试	原生支持，	封装了一个更友好的shell脚本来完成这个工作
测试结果xml格式输出	YUITest +performTaskWithPathArgumentsTimeout	YUITest生成测试结果。并且由performTaskWithPathArgumentsTimeout 方法写成文件
Monkey测试	UI Automation API组合使用	模仿android Monkey 写了一个做随机健壮性测试的iOSMonkey

# Ynm3k Dev



# 现在开始使用ynm3k

- 下载

-- <https://github.com/douban/ynm3k>

- 初始化环境

-- sh setup.sh

# 无需coding, 即可Monkey test

- /robot4ios/util/iosMonkey.js

设置并且运行:

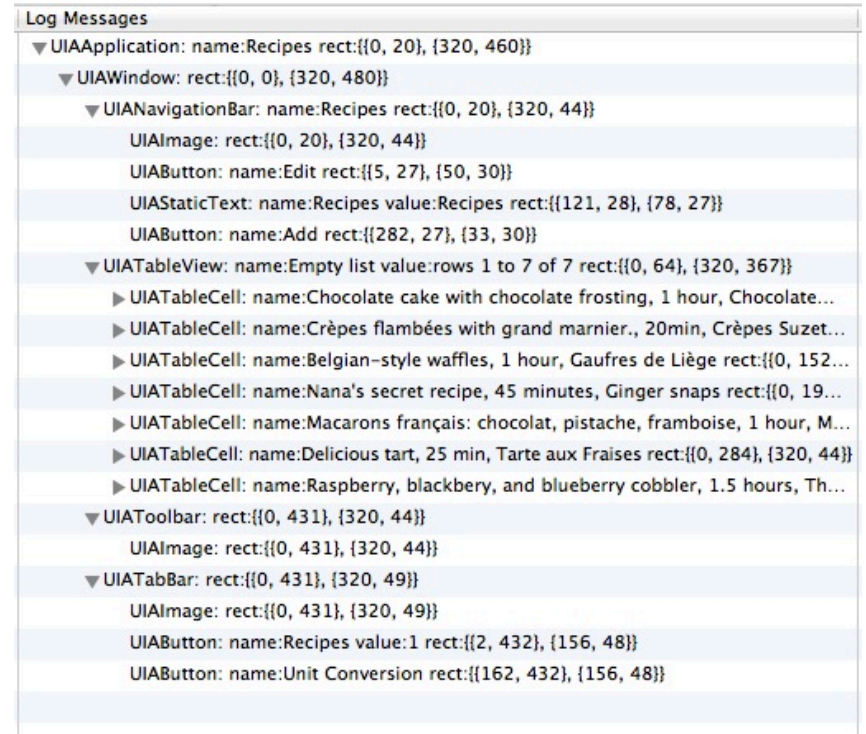
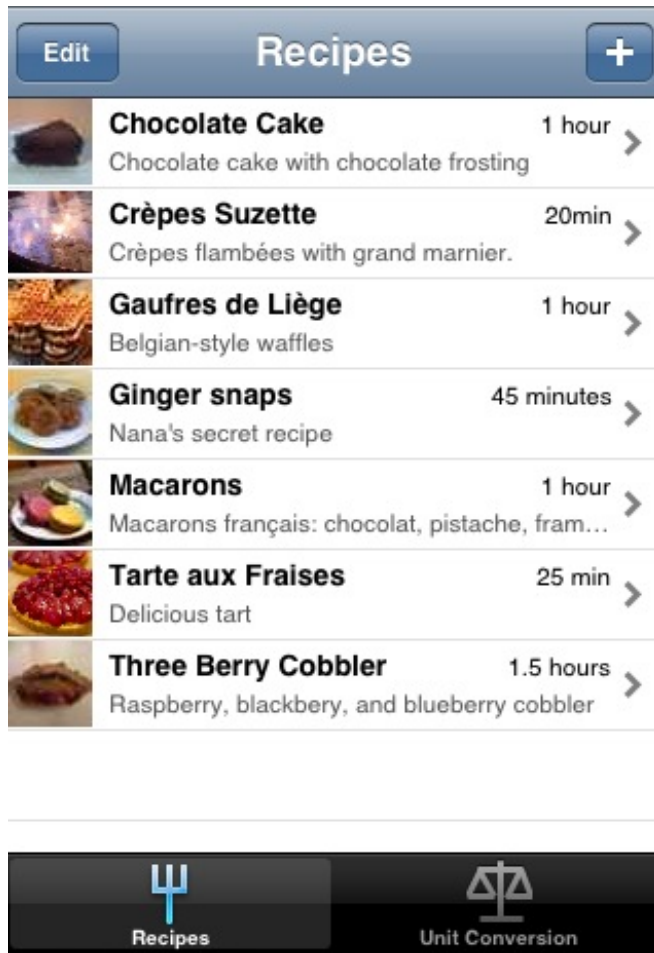
```
129 action = {$  
130   tap: false,$  
131   flick_NS: false,$  
132   flick_WE: true,$  
133   doubleTap:true,$  
134   shake:true,$  
135   volumeUD:true,$  
136   lock:true$  
137 }$  
138 IOSMonkey.run(action,30,0.5);$
```



# Hello Ynm3k

```
4 var hello_ynm3k = new DYUI.tool.TestCase({$
5   name : 'hello_ynm3k',$
6   setUp : function(){$
7     UIAlogger.logMessage("setUp");$
8   },$
9   testsomething : function(){$
10    Assert.areEqual(1,1,"must be equal");$
11  },$
12  testsomefailed : function(){$
13    Assert.isFalse(true,"some error here");$
14  },$
15  tearDown : function(){$
16    UIAlogger.logMessage("TearDown");$
17  }$
18 });$
```

# UI LogTree



# UI Auto Test

## UI Automation API

```
1 var target = UIATarget.localTarget();$
2 var app = target.frontMostApp();$
3 var windows = app.mainWindow();$
4 windows.tabBar().buttons()["Unit Conversion"].tap();$
5 $
6 target.delay(2);$
7 $
8 var ele_title =-$
9     windows.navigationBars()["Unit Conversion"]$
10     .staticTexts()["Unit Conversion"];$
11 $
12 if(ele_title.value()=="Unit Conversion"){$
13     UIALogger.logPass("test passed");$
14 }else{$
15     UIALogger.logFail("test failed");$
16 }$
```

## Ynm3k API

```
1 #import "../robot4ios/importAll.js";$
2 var sample = new DYUI.tool.TestCase({$
3     name: "sample", $
4     test_title_should_be_equals : function(){$
5         Finder.findElement_By_name("Unit Conversion").tap();$
6         Waiter.wait(Waiter.MIN_TIME);$
7     }$
8     var title = Finder.findElement_By_value("Unit Conversion")$
9     Assert.element_is_not_null(title);$
10 });$
11 DYUI.tool.TestRunner.add(sample);$
12 DYUI.tool.TestRunner.run();$
```

# 豆瓣电影实践



```
3 Finder.findElement_By_name("影片评论").tap();!
```

# 豆瓣FM实践



# 豆瓣阅读实践

春风沉醉的晚上

## 春风沉醉的晚上

郁达夫

在沪上闲居了半年，因为失业的结果，我的寓所迁移了三处。最初我住在静安寺路南的一间同鸟笼似的永也没有太阳晒着的自由的监房里。这些自由的监房的住民，除了几个同强盗小窃一样的凶恶裁缝之外，都是些可怜的无名文士，我当时所以送了那地方一个 Yellow Grab Street\* 的称号。在这 Grub Street 里住了一个月，房租忽涨了价，我就不得不拖了几本破书，搬上跑马厅附近一家相识的栈房里去。后来在这栈房里又受了种种逼迫，不得不搬了，我便在外白渡桥北岸的邓脱路中间，日新里对面的贫民窟里，寻了一间小小的房间，迁移了过去。

邓脱路的这几排房子，从地上量到屋顶，只有一丈几尺高。我住的楼

春风沉醉的晚上

把灰黑的屋顶穿通的。从前面的街里踱进了那房子的门，便是房主的住房。在破布，洋铁罐，玻璃瓶，旧铁器堆满的中间，侧着身子走进两步，就有一张中间有几根横档跌落的梯子靠墙摆在那里。用了这张梯子往上面的黑黝黝的一个二尺宽的洞里一接，即能走上楼去。黑沉沉的这层楼上，本来只有猫额那样大，房主人却把它隔成了两间小房，外面一间是一个 N 烟公司的女工住在那里，我所租的是梯子口头的那间小房，因为外间的住者要从我的房里出入，所以我的每月的房租要比外间的便宜几角小洋。

我的房主，是一个五十来岁的弯腰老人。他的脸上的青黄色里，映射着一层暗黑的油光。两只眼睛是一只大一只小，颧骨很高，额上颊上的几条皱纹里满砌着煤灰，好像每天早晨洗也洗不掉的样子。他每日于八九点钟的时候起来，咳嗽一阵，便挑了一双竹篮出去，到午后的三四点钟总仍旧是挑了一双空篮回来的，有时挑了满担回来的时候，他的竹篮里便是那些破布，破铁器，玻璃瓶之类。像这样的晚上，他必要去买些酒来喝喝，一个人坐在床沿上瞎骂出许多不可捉摸的话来。

我与隔壁的同寓者的第一次相遇，是在搬来的那天午后。春天的急景已经快晚了的五点钟的时候，我点了一枝蜡烛，在那里安放几本刚从栈房里搬过来的破书。先把它们叠成了两方堆，一堆小些，一堆大些，然后把两个二尺长的装画的画架覆在大一点的那堆书上。因为我的器具都卖完了，这一堆书和画架白天要当写字台，晚上可当床睡的。摆好了画架的

```
19 screenShoot("test1");$
20 Finder.findElement_By_name("reader").tapWithOptions({tapOffset:{x:0.9, y:0.5}})
21 Waiter.wait(Waiter.MIN_TIME);$
22 screenShoot("test2");$
23 Assert.IsFalse(diffPNG("test1","test2"))$
```



# 自动化测试的约定

- TestCase 和 TestSuite 约定:

```
#import "../robot4ios/importAll.js";
var simpleTest = new DYUI.tool.TestCase({
  name : 'simpleTest',
  testSomething : function() {
    UIALogger.logMessage("SimpleTest");
    Assert.fail("Fail Thanks");
  },
});
```

simpletest.js

```
#import "../robot4ios/importAll.js";
var simpleTest = new DYUI.tool.TestCase({
  name : 'simpleTest',
  testSomething : function() {
    UIALogger.logMessage("SimpleTest");
    Assert.fail("Fail Thanks");
  },
});
var simpleTest2 = new DYUI.tool.TestCase({
  name : 'simpleTest2',
  testSomething : function() {
    UIALogger.logMessage("SimpleTest");
    Assert.fail("Fail Thanks");
  },
});
var demotest = new DYUI.tool.TestSuite("demotest");
demotest.add(simpletest);
demotest.add(simpletest2)
```

demotest.js

# 自动化测试的约定

- name属性
- value属性（只有在控件为text的时候才有效，并且只要表示text属性就好）
- Name属性的命名规则：
  - 1 不重复
  - 2 不要变化
  - 3 只需要在被操作到的控件上添加就好，



# 加入持续集成

## Execute shell

Command

```
cd tests
sh setup.sh
./runTests.sh ./movie-testcases/alltests.js
| "/Users/komejun/jenkins/workspace/mijun_test/build/Debug-iphonesimulator/DoubanMovie.app/"
```

See [the list of available environment variables](#)

## 构建后操作

### Publish JUnit test result report

Test report XMLs

[Fileset 'includes'](#) setting that specifies the generated raw XML report file is [the workspace root](#).

☐ 保留长的标准输出/错误

# 持续集成的问题

- 开机以后第一次，命令行启动Instrument会要求输入密码。

----解决方案[Tao测试](#) 的这边文章：

[关于Instrument远程执行自动化脚本模拟器页面卡住问题](#)

# 持续集成后

## Test Result : (root)

1 failures

24 tests

Took 0 毫秒.

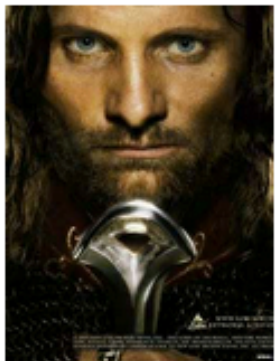
 添加说明

## All Failed Tests

Test Name	Duration	Age
>>> <a href="#">未登录状态下的更重操作.test_FM_1_04</a>	0 毫秒	<a href="#">1</a>

## All Tests

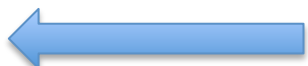
Class	Duration	Fail	(diff)	Skip	(diff)	Total	(diff)
<a href="#">未登录状态下的各种登录测试</a>	0 毫秒	0		0		4	+4
<a href="#">未登录状态下的更重操作</a>	0 毫秒	1	+1	0		9	+9
<a href="#">登录状态下的更重操作</a>	0 毫秒	0		0		11	+11



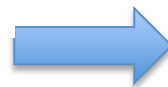
github:enterprise



ynm3k



gitcommit



HOCKEY KIT



trac  
Integrated SCM & Project



# 加入豆瓣

- [移动设备开发工程师](#)
- [QA工程师（高级移动测试工程师）](#)
- [更多豆瓣职位](#)
- [team@douban.com](mailto:team@douban.com)

# Q & A

您也可以通过以下方式找到我：

豆瓣主页：<http://www.douban.com/people/mijun/>

Email：[mijun@douban.com](mailto:mijun@douban.com)

Sina微博： @半岬

Thanks