**快速入门**

自动化测试是非常有用的，以测试您的应用程序“，而你的睡眠”。它使您能够快速跟踪回归和性能问题，同时也开发新的功能，而不必担心打破你的应用程序。

由于，苹果公司发布了iOS 4.0的一个的框架UIAutomation，可用于实现自动化测试在真实设备上的iPhone模拟器上。上的文档UIAutomation是相当小的，而且也没有大量的资源在网络上。本教程将告诉你，如何整合UIAutomation在您的工作流程。

最好的指针开始是[苹果的文档的UIAutomation](http://developer.apple.com/library/ios/" \l "documentation/DeveloperTools/Reference/UIAutomationRef/_index.html)，[苹果工具文档的](http://developer.apple.com/library/ios/" \l "documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/Built-InInstruments/Built-InInstruments.html#//apple_ref/doc/uid/TP40004652-CH6)一个很好的快速教程，当然，这些幻灯片/视频 [WWDC 2010 -会话306 - ￼自动化与仪器的用户界面测试](http://developer.apple.com/devcenter/download.action?path=/wwdc_2010/wwdc_2010_video_assets__pdfs/306__automating_user_interface_testing_with_instruments.pdf)。你需要一个免费的developper帐户来访问此资源的整合。

另一个框架要提到是OCUnit的，它包含在Xcode中，可用于单元测试添加到您的应用程序。

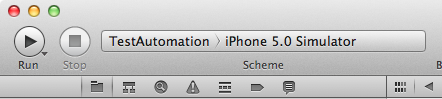
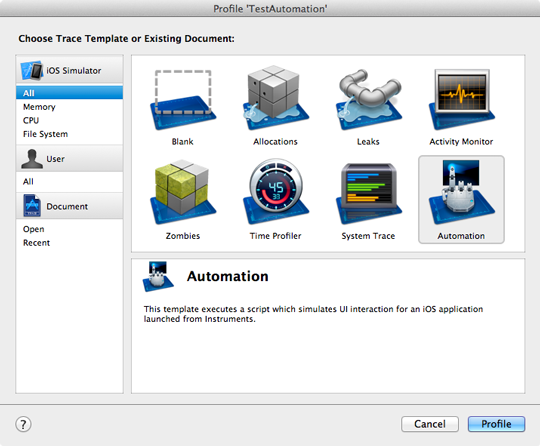
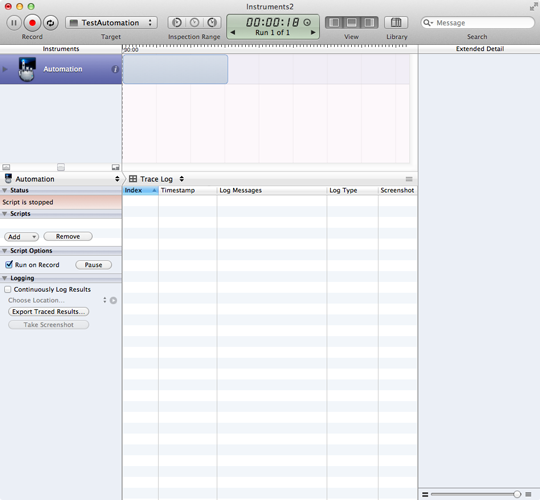
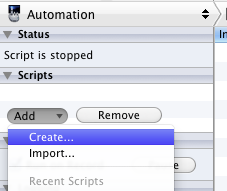
1. **[你的第一UIAutomation脚本](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "1)**
   * [使用iOS模拟器](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "1.1)
   * [使用iOS设备](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "1.2)
2. **[处理与UIAElement和辅助功能](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "2)**
   * [UIAElement层次](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "2.1)
   * [模拟用户交互](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "2.2)
3. **[简化你的生活的秘诀](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "3)**
   * [介绍调整](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "3.1)
   * [导入外部脚本](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "3.2)
   * [通过命令行的力量](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "3.3)
   * [以交互方式记录互动](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "3.4)
   * [“当事情不工作，UIATarget.delay（1）;”](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "3.5)
4. **[先进的互动](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "4)**
   * [处理意外和预期的警报](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "4.1)
   * [多工作业](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "4.2)
   * [方向](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "4.3)
5. **[最终的](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "5)**
   * [有用的链接](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "5.1)
   * [一个视频](http://blog.manbolo.com/2012/04/08/ios-automated-tests-with-uiautomation" \l "5.2)

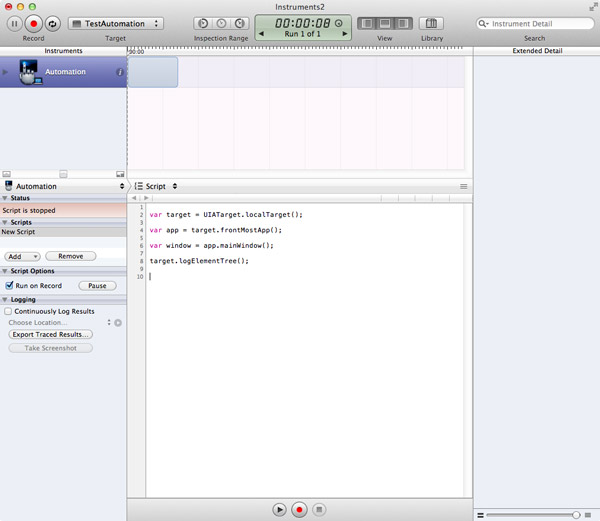
**1。你的第一UIAutomation脚本**

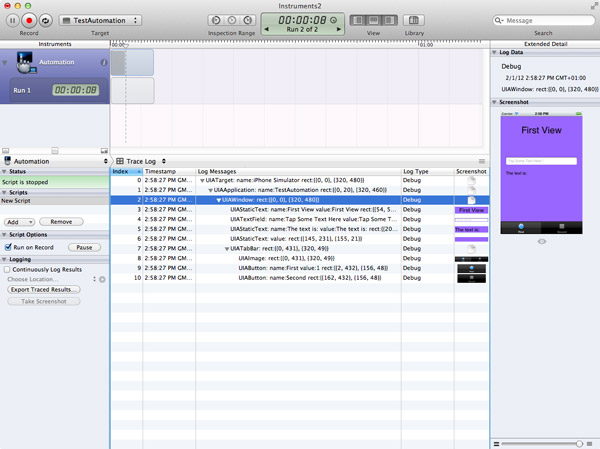
UIAutomation Javascript编写的功能测试。有一种强烈的在UIAutomation和可访问性之间的关系，所以你会使用辅助的标签和值模拟和模拟用户界面交互的检查结果。

让我们去了，写我们的第一个测试！

**使用iOS模拟器**

1. 下载的同伴的项目[TestAutomation.xcodeproj](http://blog.manbolo.com/2012/04/08/TestAutomation.zip)，并打开它。该项目是一个简单的2个选项卡的标签栏的应用程序。
2. 确保选择以下方案“TestAutomation> iPhone 5.0模拟器”（也许你已经切换到5.1，所以它可能是iPhone 5.1）
3. 启动工具（产品简介）或⌘一
4. 在iPhone模拟器，选择自动化模板，然后在“个人资料”
5. 仪器启动，并立即开始录制。停止记录（红色按钮或⌘R）。[](http://blog.manbolo.com/2012/04/08/instruments-empty.png)
6. 在“脚本”窗口中，单击“添加”>“创建”来创建一个新的脚本
7. 在“脚本”窗口中编辑器，请点击下面的代码
8. 目标= UIATarget.localTarget（）;
9. VAR的应用程序target.frontMostApp（）;
10. 变种窗口= app.mainWindow（）;
11. target.logElementTree（）;

[](http://blog.manbolo.com/2012/04/08/uiautomation-1.png)

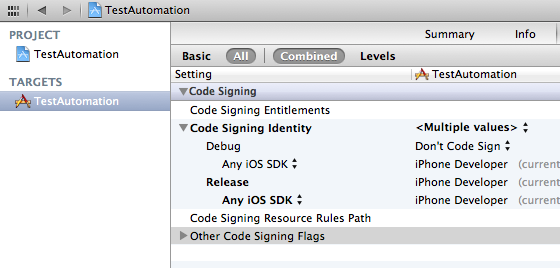
1. 重新启动脚本⌘R（你不需要保存）。脚本运行后，日志出现，你可以阻止它。[](http://blog.manbolo.com/2012/04/08/uiautomation-2.png)

瞧！您已经写了您的第一UIAutomation测试！

**使用iOS设备**

一个真正的设备，也可以运行这个测试，而不是模拟器。自动测试设备，支持多任务的iPhone 3GS，iPad的，仅适用于运行iOS 4.0。UIAutomation不幸的是，在iPhone 3G上，无论是操作系统版本。

要运行测试的设备上：

1. 将iPhone连接到USB
2. 选择计划“TestAutomation> iOS装置”
3. 检查发布配置与一个Developper配置文件（而不是一个Ad-Hoc分布情况）。默认情况下，分析是在释放（没有任何理由来分析一个应用程序在调试！）
4. 个人应用程序（⌘我）
5. 比以前在模拟器上按照相同的步骤。

**2。处理与UIAElement和辅助功能**

**UIAElement层次**

有很强的关系之间无障碍UIAutomation：如果配备无障碍访问控制，您将能够设置/获取值，产生动作等的控制是不是“看得见”的辅助工具，将无法访问通过自动化。

您可以允许访问/自动化控制是否使用Interface Builder，或通过编程方式设置属性isAccessibilityElement。你必须支付一定的关注时，**设置辅助容器视图**（即包含其他UIKit元素）。启用无障碍的整个视图可以“隐藏”其子视图访问/自动化。比如，在项目中，控制器的视图出口应不能够访问，否则的子控制将不访问。如果您有任何问题，logElementTree是您的朋友：列出所有当前可见的元素可以访问。

每个UIKit的控制，可以访问可以表示为一个JavaScript对象，UIAElement。UIAElement有几个属性，名称，价值，元素，父。你的主窗口包含了很多的控制，它定义了一个UIKit的层次结构。这UIKit的层次结构，对应的UIAElement的层次结构。例如，通过调用logElementTree在前面的测试中，我们有以下三种：

+ - UIATarget：名称：iPhone模拟器正确的：{{0,0}，{320480}}

| + - UIAApplication：名称：TestAutomation的正确：{{0,20}，{320460}}

| | - UIAWindow：正确的：{{0,0}，{320480}}

| | | + - UIAStaticText的：姓名：一是该价值：首先查看正确的：{{54,52}，{212,43}}

| | | + - UIATextField的名称：用户的文本值：点击一些文字在这里！正确的：{{20179}，{280,31}}

| | | + - UIAStaticText的：姓名：文字是：值：文字是：正确的：{{20231}，{112,21}}

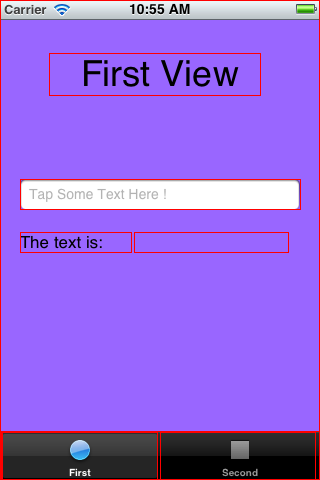
| | | + - UIAStaticText：值：正确的：{{145231}，{155,21}}

| | | + - UIATabBar：正确的：{{0431}，{320,49}}

| | | | + - UIAImage：正确的：{{0431}，{320,49}}

| | | | + - UIAButton：名称：第一个值：1 RECT：{{2432}，{156,48}}

| | | | + - UIAButton：名称：二正确的：{{162432}，{156,48}}



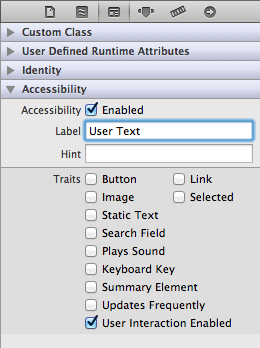
要访问该文本字段，你可以这样写：

。mainWindow的变种textField的。frontMostApp = UIATarget.localTarget（）（）（）。文本字段（）[0];

您可以选择基于0的索引或元素的名称来访问元素。例如，以前的文本字段也可以被这样的访问：

VAR textField的。frontMostApp = UIATarget.localTarget（）（）。主窗口（）。文本框（）[“用户文本”];

较新版本更清晰，更应该是首选。在Interface Builder中，您可以设置的UIAElement名：



或计划性：

myTextField.accessibilityEnabled = YES;

myTextField.accessibilityLabel = @“用户文本”;

你可以看到现在所使用的辅助功能属性UIAutomation针对不同的控件。这是非常聪明的，因为：1）只有一个框架，以学习; 2）编写自动化测试，你也将确保您的应用程序可以访问！所以，通过调用下面的每个UIAElement可以访问它的孩子 要访问的第一个选项卡的标签栏，你可以这样写：

变种的tabbar = UIATarget.localTarget（）。frontMostApp（）。TabBar中（）;

VAR tabButton tabBar.buttons（）[“第一”];

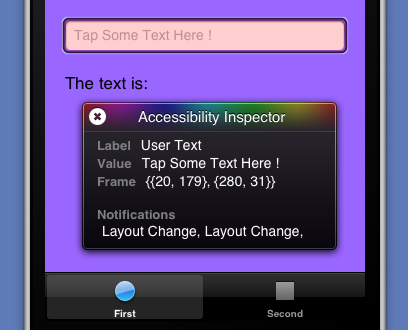
UIAElement层次是非常重要的，你要处理不断。记住，你可以转储的层次结构，每次在你的脚本由调用logElementTree上UIAApplication：

。logElementTree。frontMostApp UIATarget.localTarget（）（）（）;

在模拟器上，您还可以激活的可访问性检查。启动模拟器，前往“设置”>“通用”>“辅助功能”>“辅助功能”督察“，并将其设置为”开“。



这个小彩虹中的可访问性检查。折叠时，辅助功能是关闭的，并扩展了无障碍时。要激活/ desactivate辅助功能，你只需要点击箭头按钮。现在，到我们的测试应用程序，启动它，并启动督察。



然后，点击文本字段和检查的名称和值属性的相关的UIAElement（也NSObject的accessibilityLabel和性能相当accessibilityValue）。检查员将帮助您调试和编写脚本。

**模拟用户交互**

让我们更进一步，模拟用户交互。轻按一键，您只需拨打自来水（）这个元素：

变种的tabbar = UIATarget.localTarget（）。frontMostApp（）。TabBar中（）;

VAR tabButton tabBar.buttons（）[“第一”];

/ /点击标签栏！

tabButton.tap（）;

您也可以拨打doubleTap（）中，twoFingerTap（） UIAButtons。如果你不想要的元素为目标，但只在指定的坐标画面在屏幕上的互动，你可以使用：

* 水龙头：
* UIATarget.localTarget（）。点选（X：100，Y：200}）;
* UIATarget.localTarget（）。doubleTap（{：100，Y：200}）;
* UIATarget.localTarget（）。twoFingerTap（{：100，Y：200}）;
* 捏：
* UIATarget.localTarget（）。pinchOpenFromToForDuration（{：20，Y：200}，{×：300，Y：200}，2）;
* UIATarget.localTarget（）。pinchCloseFromToForDuration（{：20，Y：200}，{×：300，Y：200}，2）;
* 拖动和弗里克：
* UIATarget.localTarget（）。dragFromToForDuration（{：160，Y：200}，{160，Y：400}，1）;
* UIATarget.localTarget（）。flickFromTo（{：160，Y：200}，{160，Y：400}）;

在您指定的时间，只在一定范围内接受即：拖时间，值必须大于或等于0.5秒或不到60岁。

现在，让我们把在实践中：

1. 停止（⌘R）仪器
2. 在“脚本”窗口中，删除当前的脚本
3. 点击“添加”>“导入”，然后选择TestAutomation/TestUI/Test-1.js
4. 点击记录（⌘R）和观看的发生。

该脚本是：

测试名=“测试”;

目标= UIATarget.localTarget（）;

VAR的应用程序target.frontMostApp（）;

变种窗口= app.mainWindow（）;

UIALogger.logStart（测试名）;

app.logElementTree（）;

/ / - 选择元素

UIALogger.logMessage（“选择第一个选项卡”）;

VAR标签栏app.tabBar（）;

selectedTabName = tabBar.selectedButton（）。名称（）;

（selectedTabName！“第一”）{

tabBar.buttons（）[“第一”。点击（）;

}

/ / - 点选文本框就会自动

UIALogger.logMessage（“塔上的文本字段”）;

recipeName =“相当长的配方名称”;

window.textFields（）[0] =。的setValue（recipeName）的;

target.delay（2）;

/ / - 点选文本框就会自动

UIALogger.logMessage（“关闭键盘”）;

app.logElementTree（）;

app.keyboard（）按钮（）“回归”。点击（）;

VAR textValue window.staticTexts（）“RecipeName”。值（）;

如果（textValue === recipeName）

UIALogger.logPass（测试名）;

}

其他{

UIALogger.logFail（测试名）;

}

这个脚本启动应用程序，选择第一个选项卡，如果它不被选中的文本字段的值设置为“相当长的配方名称，并驳回键盘。注意到一些新的功能：延迟（数字时间间隔内） UIATarget允许引入一些延迟之间的相互作用，logMessage（字符串消息）上UIALogger可以用来记录的测试输出和logPass（弦乐信息）上UIALogger的消息表明，脚本已成功完成。  
您还可以看到一个访问不同的键盘上的按钮，点一下就可以了app.keyboard（）按钮（）[“”]。点击（）;

**3。简化你的生活的秘诀**

**介绍调整**

现在，你如何，你可以写一些测试的一个基本理念。很快，你会发现有很多的冗余和胶水代码在您的测试中，你会经常重新编写这样的代码：

目标= UIATarget.localTarget（）;

VAR的应用程序target.frontMostApp（）;

变种窗口= app.mainWindow（）;

这就是为什么我们要使用一个小的JavaScript库，它简化了编写UIAutomation测试。转到<https://github.com/alexvollmer/tuneup_js>，让图书馆和复制的热身测试“文件夹中的文件夹一边。现在，我们可以重写Test1.js利用可调谐

＃导入“的热身/ tuneup.js”

测试（“测试1”，目标，功能（应用程序）

变种窗口= app.mainWindow（）;

app.logElementTree（）;

/ / - 选择元素

UIALogger.logMessage（“选择第一个选项卡”）;

VAR标签栏app.tabBar（）;

selectedTabName = tabBar.selectedButton（）。名称（）;

（selectedTabName！“第一”）{

tabBar.buttons（）[“第一”。点击（）;

}

/ / - 点选文本框就会自动

UIALogger.logMessage（“塔上的文本字段”）;

recipeName =“相当长的配方名称”;

window.textFields（）[0] =。的setValue（recipeName）的;

target.delay（2）;

/ / - 点选文本框就会自动

UIALogger.logMessage（“关闭键盘”）;

app.logElementTree（）;

app.keyboard（）按钮（）“回归”。点击（）;

VAR textValue window.staticTexts（）“RecipeName”。值（）;

的assertEquals（recipeName，textValue）;

}）;

Tune-Up的避免了你写的相同的样板代码，再加上给你一些额外的，例如不同的断言assertTrue（表达式，消息），assertMatch（正则表达式，表达式，消息）的assertEquals（预计，接收消息），assertFalse（表达，消息），assertNull（thingie，消息），assertNotNull（thingie，消息），您可以扩展库非常容易：例如，您可以添加一个logDevice方法上UIATarget对象，通过添加此功能在uiautomation ext.js：

扩展（UIATarget.prototype，{

logDevice：函数（）{

UIALogger.logMessage（“转储设备：”）;

UIALogger.logMessage（“型号：”+ UIATarget.localTarget（）。模型（））;

UIALogger.logMessage（“矩形（矩形：”+ JSON.stringify（UIATarget.localTarget（）。）））;

UIALogger.logMessage（“名称：”+ UIATarget.localTarget（）。（））;

UIALogger.logMessage的SystemName：“+ UIATarget.localTarget（）（”的SystemName（））;

UIALogger.logMessage（“systemVersion：”+ UIATarget.localTarget（）。systemVersion（））;

}

}）;

然后，调用target.logDevice（），你应该看到：

转储设备：

型号：iPhone模拟器

矩形：{“原点”：{“×”：0，“y”的：0}，“大小”：{“宽度”：320，“高度”：480}}

产品名称：iPhone模拟器

**导入外部脚本**

你还可以看到如何引用一个脚本，使用＃import的指令。因此，创建倍数测试和链接，他们可以通过在一个单一的文件，并呼吁将其导入：

＃导入“Test1.js”

＃导入“Test2.js”

＃导入“Test3.js”

＃导入“Test4.js”

＃导入“Test5.js”

**通过命令行的力量**

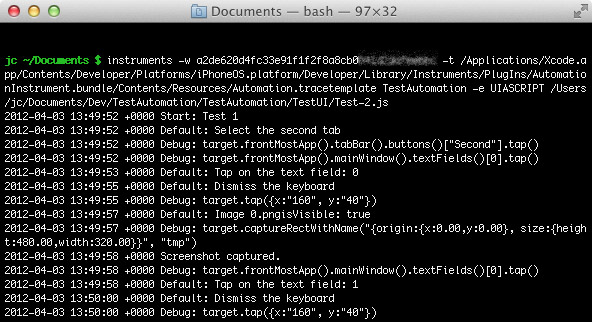
如果你想你的脚本自动化，你可以在命令行启动。事实上，我建议使用此选项，而不是使用工具的图形用户界面。仪器的用户界面是缓慢的，和测试，保持运行，即使他们已经走到了尽头。在命令行启动UIAutomation测试速度快，你的脚本将停止在测试结束。

要启动一个脚本，你将需要你的UDID和类型终端：

仪器-W your\_ios\_udid-T name\_of\_your\_app-E UIASCRIPT absolute\_path\_to\_the\_test\_file

例如，在我的情况下，该行是这样的：

仪器-W a2de620d4fc33e91f1f2f8a8cb0841d2xxxxxxxx-T TestAutomation-的ËUIASCRIPT / Users/jc/Documents/Dev/TestAutomation/TestAutomation/TestUI/Test-2.js



如果您使用的是版本的Xcode不如4.3，你需要键入：

仪器-W your\_ios\_device\_udid-T TestAutomation-的ËUIASCRIPT / Users/jc/Documents/Dev/TestAutomation/TestAutomation/TestUI/Test-2.js

一个小的缺点，不要忘记关闭通的代码在您的设备上，否则，你会看到这个跟踪：，遇到远程异常：“设备锁定，无法启动过程包标识符”com.manbolo.testautomation“ 。是的，UIAutomation不知道你的密码！

命令行也与模拟器。您需要知道您的应用程序在模拟器文件系统的绝对路径。模拟器的模拟设备的文件系统在下面的文件夹〜/资源库/应用程序支持/ iPhone Simulator/5.1 /。在这个目录下，你会发现你的应用程序目录，其中包含在模拟器中安装的应用程序的沙箱。只要确定资料库的TestAutomation在模拟器上的应用程序和类型：

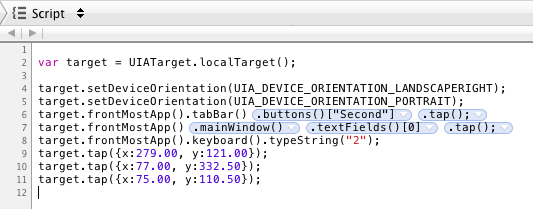
仪器-T “/用户/ JC /资源库/应用程序支持/ iPhone Simulator/5.1/Applications/C28DDC1B-810E-43BD-A0E7-C16A680D8E15/TestAutomation.app”-E UIASCRIPT /用户/ JC /文档/开发/ TestAutomation / TestAutomation / TestUI /测试2.js

在命令行上的最后一个字。如果你没有精确的输出文件，日志的结果将出现在您输入的命令的文件夹中。您可以使用-E UIARESULTSPATH results\_path的脚本的输出重定向。

我还没有成功推出多个脚本的命令行平行。使用整个晚上，链，并启动你的脚本，所以你真的会测试您的应用程序“，而你的睡眠”。

**以交互方式记录互动**

输入你的脚本，你可以录制的互动，直接在设备上或在模拟器中，稍后重播。做这样的：

1. 启动仪器（⌘我）
2. 创建一个新的脚本
3. 选择“脚本编辑器
4. 在脚本编辑器的底部，看到红色的按钮，记录的相互作用按它！
5. 现在，您可以发挥您的应用程序，你会出现在脚本窗口（甚至旋转事件）看到所拍摄的相互作用。按下方按钮停止记录。

**“当事情不工作，UIATarget.delay（1）;”**

当您编写脚本，你会玩的时间，动画等。UIAutomation具有多种功能元素和等待他们，即使他们不显示的，但最好的建议是[从这个额外的演示文稿](http://www.juddsolutions.com/downloads/UnitAndFunctionalTestingForiOSPlatform.pdf)：

当事情不工作，UIATarget.delay（1）;！

**4。先进的互动**

**处理意外和预期的警报**

处理警报一直很难在自动化测试：你精心编写的脚本，睡前推出测试套件，并且，在早上，你会发现，所有的测试已经毁了，因为你的iPhone已经收到了意想不到的文本消息已经封锁了测试。好了，UIAutomation帮助您处理。

在你的脚本中添加此代码，

UIATarget.onAlert =：功能onAlert（警报）{

标题= alert.name（）;

UIALogger.logWarning（标题“+标题+”遇到“警报！”）;

/ /使用默认的处理程序，返回false;

}

并返回错误，你问UIAutomation自动关闭任何UIAlertView，因此不会干涉你的测试警报。您的脚本将运行好像从来没有出现过任何警报。不过，警报可以是您的应用程序和测试工作流程，在某些情况下，你不111,111自动关闭它。要做到这一点，你可以测试对标题的警报，点击一些按钮，并返回true。通过返回TRUE，则表明UIAutomation警报，这必须被视为作为测试的一部分，处理accordantly。

例如，如果你要测试“上的”添加“按钮，添加东西的警报视图用胶带，你可以写：

UIATarget.onAlert =：功能onAlert（警报）{

标题= alert.name（）;

UIALogger.logWarning（标题“+标题+”遇到“警报！”）;

如果（标题==“添加东西”）{

alert.buttons（）[“添加”。点击（）;

/ /旁路默认的处理程序返回true;

}

/ /使用默认的处理程序，返回false;

}

方便宝宝！

**多工作业**

在您的应用程序的多任务测试也是非常简单：让我们说你要测试，疯狂的背景过程中你发动每次应用程序恢复从背景和进入 - （无效）applicationWillEnterForeground：（UIApplication的\*）应用程序选择器，你可以发送的应用程序在后台，等待10秒，并恢复它的调用：

UIATarget.localTarget（）。deactivateAppForDuration（10）;

（持续时间）deactivateAppForDuration将暂停脚本，模拟用户点击home键，（和发送的应用程序在后台），等待，恢复应用程序，恢复你的测试脚本，*在一行代码*。

**方向**

最后，你可以模拟旋转你的iPhone。同样，非常简单，而且很容易：

目标= UIATarget.localTarget（）;

VAR的应用程序target.frontMostApp（）;

/ /设置景观离开

target.setDeviceOrientation（UIA\_DEVICE\_ORIENTATION\_LANDSCAPELEFT）;

UIALogger.logMessage（“目前的定位是”+ app.interfaceOrientation（））;

/ /肖像

target.setDeviceOrientation（UIA\_DEVICE\_ORIENTATION\_PORTRAIT）;

UIALogger.logMessage（“目前的定位是”+ app.interfaceOrientation（））;

**5。最终的**

**有用的链接**

这是一个相当长的职位，但我希望，你看到的力量UIAutomation和您的应用程序能够获得的潜在爆发的质量。是不是有很多文件上UIAutomation，但我列出了一大堆的链接，可以帮助您。

* 非常好的系列UIAutomation
* <http://mobilecoder.wordpress.com/2010/11/08/iphoneipodipad-automation-basics>：优秀的教程的UIAutomation，值得阅读！
* [的http://www.juddsolutions.com/downloads/UnitAndFunctionalTestingForiOSPlatform.pdf](http://www.juddsolutions.com/downloads/UnitAndFunctionalTestingForiOSPlatform.pdf)：精湛，deepful的介绍单元测试和UIAutomation的。您将欣赏幻灯片N°70“当事情不工作，UIATarget.delay（1）;”！
* <http://guerratopia.com/en/introduction-to-automating-ui-testing-in-ios>：很好的教程，很不错的介绍
* 了很多的约单元测试和UIAutomation的链接

而且，当然

* [苹果上UIAutomation文件](http://developer.apple.com/library/ios/" \l "documentation/DeveloperTools/Reference/UIAutomationRef/_index.html)
* [UIAutomation在苹果工具文档](http://developer.apple.com/library/ios/" \l "documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/Built-InInstruments/Built-InInstruments.html#//apple_ref/doc/uid/TP40004652-CH6)
* [WWDC 2010 - 306 - ￼用户界面自动化测试与仪器](http://developer.apple.com/devcenter/download.action?path=/wwdc_2010/wwdc_2010_video_assets__pdfs/306__automating_user_interface_testing_with_instruments.pdf)

你需要一个免费的developper帐户来访问此资源的整合。

**一个视频**

结束此行UIAutomation，我不抗拒，向你展示如何，我们使用UIAutomation与[米恩](http://www.manbolo.com/meon)在一个小的视频。我们使用各种测试，在这段视频中，我们测试播放器可以播放从0级到120级。*救命我，我的iPhone是活着的！*