**Appecker**

**宏录制模式  
使用说明**

# 概述：

宏录制模式分为两个主要功能：

1. 辅助控件定位；
2. 辅助代码生成；

# 激活方法：

1. 按照《集成说明》中的步骤将Appecker集成到待测工程；
2. 在main.mm中增加如下内容：



**附注：**enableMacroRecMode会开启宏录制模式，在此模式中Appecker不会运行测试用例；而是监听用户的点击操作，并辅助用户定位或者生成代码；

1. 在**模拟器**中启动app；

如果步骤无误，此时app将正常启动，但是app窗口中出现一个灰色的悬浮窗口，此窗口表示宏录制模式已经成功激活；

## 辅助控件定位：

基于如下原因Appecker制作了辅助定位功能：

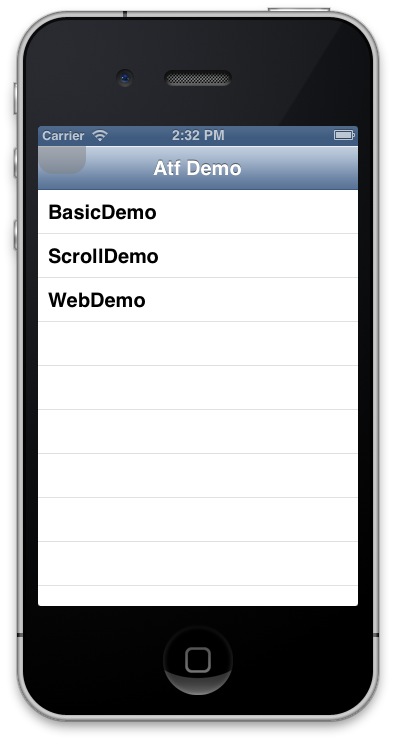
1. 有时复杂界面的app在painter（参见painter工具的说明）中绘制结果不容易读

2）有些控件不在主窗口中，为定位带来一定不便

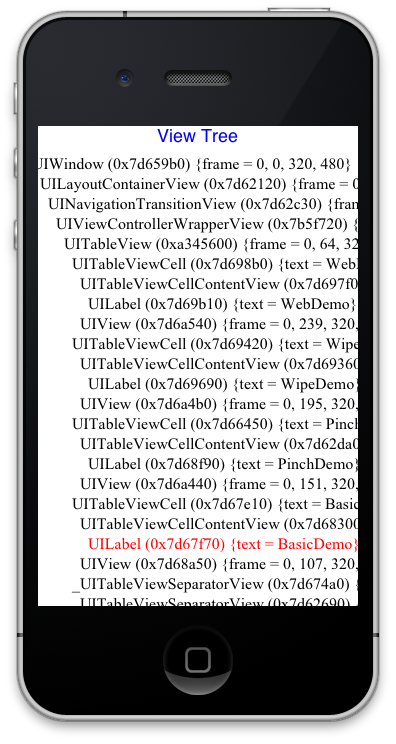
## 使用说明：

一旦开启宏录制模式，用户的任何长按动作都会被Appecker认为是辅助定位请求，Appecker会立即做出相应动作；

比如，在下图中尝试点击Basic Demo：



点击后出现如下界面：



窗口中显示的是当前window的printTree结果，而图中红色的部分为Appecker探测到的被点击控件；

双击标题”View Tree”可以将此界面关闭，返回SUT继续操作

注意：

Appecker并没有拦截用户操作，而只是做了监听；即用户点击操作在产生定位帮助的同时也会在SUT中生效；

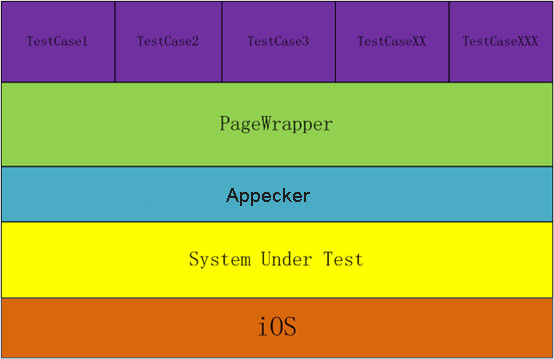
## 辅助代码生成：

基于如下目的，Appecker制作了辅助代码生成功能:

1. 为入门级使用者提供测试用例编写的基本示例;
2. 简化针对简单繁琐控件操作的case编写

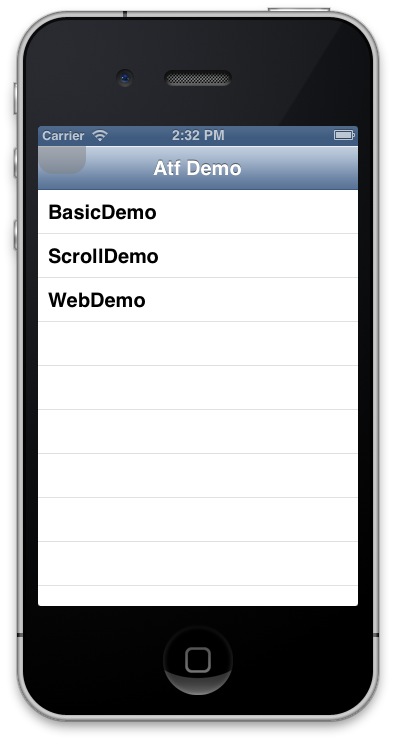
## 适用范围说明:

本功能可以生成一些简单的case，但是过于依赖本功能会导致case代码违背良好的封装原则，从而导致后期维护成本增加；因此，如果时间充裕，仍然建议按照下图所示来组织测试工程中的代码：



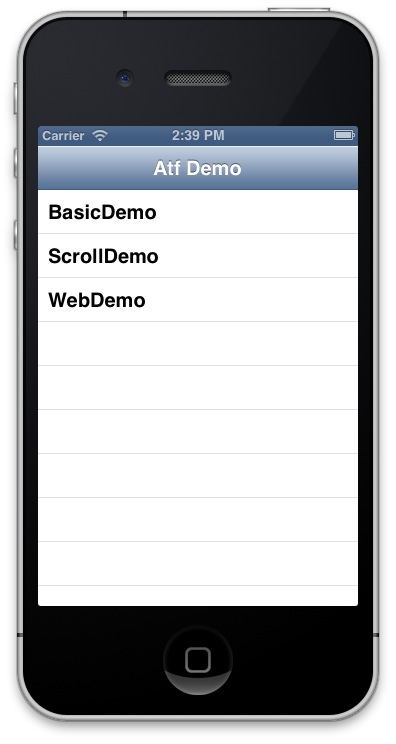
## 使用方法：

app启动之后，如图：



图中的灰色窗口是宏录制的开关，为了避免影响对SUT本身的操作，此**窗口允许随意拖动，并且会自动吸附边缘；**

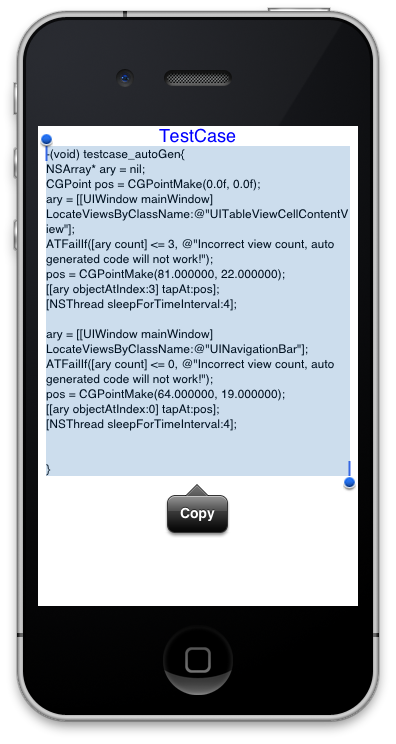
现在双击灰色窗口，灰色窗口消失(此时辅助定位模式自动关闭)，如图：



此时已经开始宏录制，可以对SUT进行点击操作，点击操作的超时时间为3s，即3s内如果没有任何操作 Appecker会自动生成相关操作代码；操作超时从第一次操作之后开始计算；

进行相关操作之后停留3s，此时会自动出现代码窗口；

如图：



此时，先点击copy，再按下cmd+c键可以将生成的代码复制到mac的剪切板；

可以在xcode中直接粘贴，也可以截取其中的一部分单独使用；

如果需要再次录制，可以双击标题栏中的TestCase，此时代码界面会消失，回到SUT；

如果需要再次录制，再次重复上述步骤即可；

# FAQ：

Q：从生成的testcase本身的流程来看，如果我没有从首页开始录制，那么岂不是录制出来的代码是无法运行的？

A：宏录制是一个比较机械化的过程，不太可能做的足够灵活而实现对SUT状态的动态检测；这种情况下，在生成case中截取部分使用集成到现有case中就可以了；

Q：宏录制支持拖动、长按、pinch、spread等手势么？

A：不支持，因为Appecker并不以宏录制为主要使用方式，过多地设计这些特性会导致生成大量不易维护的测试用例；如果你需要这些高级特性，建议在遵循软件工程一般准则的前提下，手动设计测试用例从而充分利用代码封装和重用已达到case的易读性和易维护性；

Q：那个悬浮窗会影响录制代码对控件的定位么？

A：不会，悬浮窗口是独立与SUT中的任何窗口的，所以不会影响SUT控件的访问路径；