# 单元测试

1. 简述：

   1) 在 iOS 开发中，程序员通常用单元测试来保证代码的可靠性，什么是单元测试？在计算机编程语言中，单元测试又称为模块测试，是针对程序模块的最小单位进行正确性检验的测试工程。程序单元是应用的最小可测试部件。在过程化编程中，一个单元就是单个程序、函数、过程等；对于面向对象编程，最小单元就是方法，包括基类(超类)、抽象类或者派生类(子类)中的方法。有了单元测试，就没必要为了测试某个小模块去编译整个程序，运行后点到相应模块去了。一般来说，写完代码或修改完 bug 后，需要写单元测试来验证代码是否有问题。

   2) 在 xcode4 时代，集成的是 OCUnit，到了 xcode5 时代就升级为 XCTest，并且到了 xcode7 时代还有了 UI 测试的能力。新建项目的时候可以为工程选择是否带上单元测试，新建项目多两个目录：TestDemoTests、TestDemoUITests，如果没有，可通过 File->New->Targe->Test->iOS Unit Testing Bundle / Ios UI Testing Bundle

2. Tests 目录

   (1) 包含一个.m 文件和一个 plist 文件。.m 文件中包含如下方法：

      1) setup:每个 test 方法执行前调用，在这个测试用例里进行一些初始化工作

      2) teardown:每个 test 方法执行后调用

      3) testExample:测试方法用例，记得以 test 开头来命名

      4) testPerformanceExample:主要做性能测试，评估一段代码的运行时间，在回调中放置需测试效率的代码

   (2) 只要点击测试方法左边的菱形按钮，即可运行该测试方法。如果要全部运行，按 command＋u 即可，或者 Product->Test。在测试方法里主要通过断言判断，不适预期的正确结果则该测试方法失败，成功显示绿色对勾，失败显示红色叉叉。断言相关介绍如下：

      1) XCTFaile(format...)：生成一个失败的测试

      2) XCTAssertNil(a1,format...)：为空判断，a1 为空通过，否者不通过

      3) XCTAssertNotNil(a1,format...)：不为空判断，a1 不为空通过，否则通过

      4) XCTAssert(expression,format...)：当表达式为 true 时通过

5) XCTAssertTrue(expression,format...)：当表达式为 true 时通过

6) XCTAssertFalse(expression,format...)：当表达式为 false 时通过

7) XCTAssertEqualObjects(a1,a2,format...)：判断相等，[a1 isEqualTo:a2]为 true 通过

8) XCTAssertNotEqualObjects(a1,a2,format...)：判断不等，[a1 isEqualTo:a2 为 false 时通过

9) XCTAssertEqual(a1,a2,format...)：判断相等，当 a1,a2 为 C 语言标量、结构体时

10) XCTAssertNotEqual(a1,a2,format...)：判断不等，当 a1,a2 为 C 语言标量、结构体时通过

11) XCTAssertEqualWithAccuracy(a1,a2,accuracy,format...)：判断相等，当 a1,a2 位 double 或 float，误差在绝对值 accuracy 之间时通过

12) XCTAssertNotEqualWithAccuracy(a1,a2,accuracy,format...)：判断不等，当 a1,a2 为 double 或 float，误差范围在绝对值 accuracy 之间时通过

13) XCTAssertThrows(expression,format...)：异常测试，当 expression 发生异常时通过

14) XCTAssertThrowsSpecific(expression,specificException,format...)：异常测试，仅当 expression 发生 specific 异常时通过

15) XCTAssertThrowsSpecificNamed(expression,specificException,exception_name,format...)：异常测试，异常名一致时通过

16) XCTAssertNotThrows(expression,format...)：异常测试，当 expression 没发生异常时通过

17) XCTAssertNotThrowsSpecific(expression,specificException,format...)异常测试，当 expression 没发生 specificException 异常时通过

18) XCTAssertNotThrowsSpecificNamed(expression,specificException,exception_name,format...)：异常测试，当 expression 没有发生指定异常名的异常时通过

(3) 测试用例文件名约定：以类名起始，添加 Tests 结束

(4) 单元测试的目标是模型文件，用处：不基于界面 UI 的情况下，保证模型类的逻辑正确

(5)测试用例：

```
-(void)testAge{//测试年龄范围在 18-60 岁的人，失败说明程序没做好范围判断
 Person *p = [[Person alloc] init];
 //在单元测试中一定要有边界的测试
 //测试情况 1:小于 18 岁的人不能使用某些功能->person.age>=18
 p.age = 18;
 XCTAssertTrue(p.age>=18,@”年龄应该大于或等于 18 岁”);
 //测试情况 2:大于 60 岁的人不能使用某些功能->person.age<=60
 p.age = 61;
 XCTAssertTrue(p.age<=60,@”年龄应该不大于 60 岁”);
}
-(void)testName{
 Person *p = [[Person alloc] init];
 //姓名不能为空
 p.name = @”a”;
 XCTAssertFalse(p.name == nil,@”用户姓名不能为空”);
 XCTAssertTrue(p.name.length>2,@”用户姓名长度至少三个字符”);
}
-(void)testClassMethod{
 Person *p = [Person personWithName:@”zhangsan” age:18];
 XCTAssertTrue([p.name isEqualToString:@”zhangsan”,@”姓名正确赋值”]);
 XCTAssertEqual(p.age,18,@”年龄正确赋值”);
}
```

(6)测试方法性能

```
- (void)testPerformanceExample {//测试性能例子
    // This is an example of a performance test case.
    [self measureBlock:^{                                    Time: 0.021 sec (57% better, 46% STDEV)
        int num = 0;
        for (int a = 0; a < 100; a ++) {
            num = a + 10 * 100;
            NSLog(@"%d", num);
        }
        // Put the code you want to measure the time of here.
    }];
}
```

能够非常直观的看出其调用时间，可以用来对比性能的优劣

## (7) 异步单元测试

```
- (void)testExpectationExample{
    XCTestExpectation *expectation = [self expectationWithDescription:@"MYGetJobDetailRequest"];

    MYGetJobDetailRequest *request = [[MYGetJobDetailRequest alloc] init];
    request.ID = [@"5865" integerValue];

    NSInteger requestId = [[MYNetworkProxy defaultProxy] httpPostWithRequestObj:request entityClass:@"MYGetJobDetailResponed"
        withCompleteBlock:^(MYNetworkResponse *response) {
        [expectation fulfill];
        if(response.status == MYNetworkResponseStatusSuccessed){
            MYGetJobDetailResponed *responed = (MYGetJobDetailResponed *)response.content;
            XCTAssertNotNil(responed, @"请求成功，数据出错");
            return ;
        }else if(response.status == MYNetworkResponseStatusFailed){
            XCTFail(@"请求失败");
        }
    }];

    [self waitForExpectationsWithTimeout:10 handler:^(NSError * _Nullable error) {
        [[MYNetworkProxy defaultProxy] cancelRequestWithRequestId:requestId taskType:MYNetworkTaskTypeData];
    }];
}
```