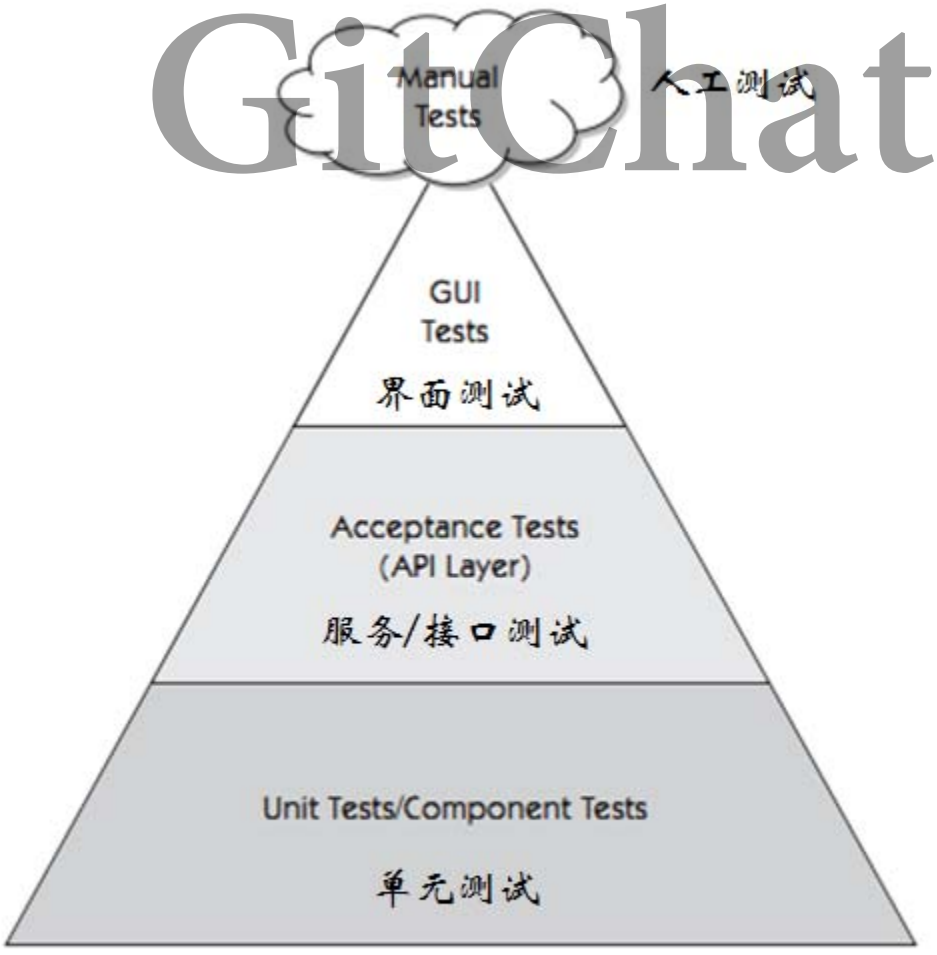
互联网服务端接口自动化测试

在飞速发展的互联网时代，服务端测试已经成为一个重要的产品保障手段。如何在互联网公司提供稳定的服务端接口测试，各个公司实施的方法和技术也不同，今天我们就此展开讨论学习。

互联网服务端接口自动化是各个公司都需要的一部分业务，如何快速高效地完成接口测试呢？以帮助大家实现高效的接口测试为出发点，本场 Chat 包含了我在互联网接口测试领域的一些方法和心得，希望大家一起讨论和分享，内容包括但不仅限于：

- 互联网服务端接口测试介绍；
- 接口测试常用的工具、平台、框架；
- 接口测试的一些问题和实践。

下图是经典的测试金字塔，服务接口测试就在中间位置，并占据着承上启下的作用，从这个图可以看出，服务服务端接口测试的重要性。

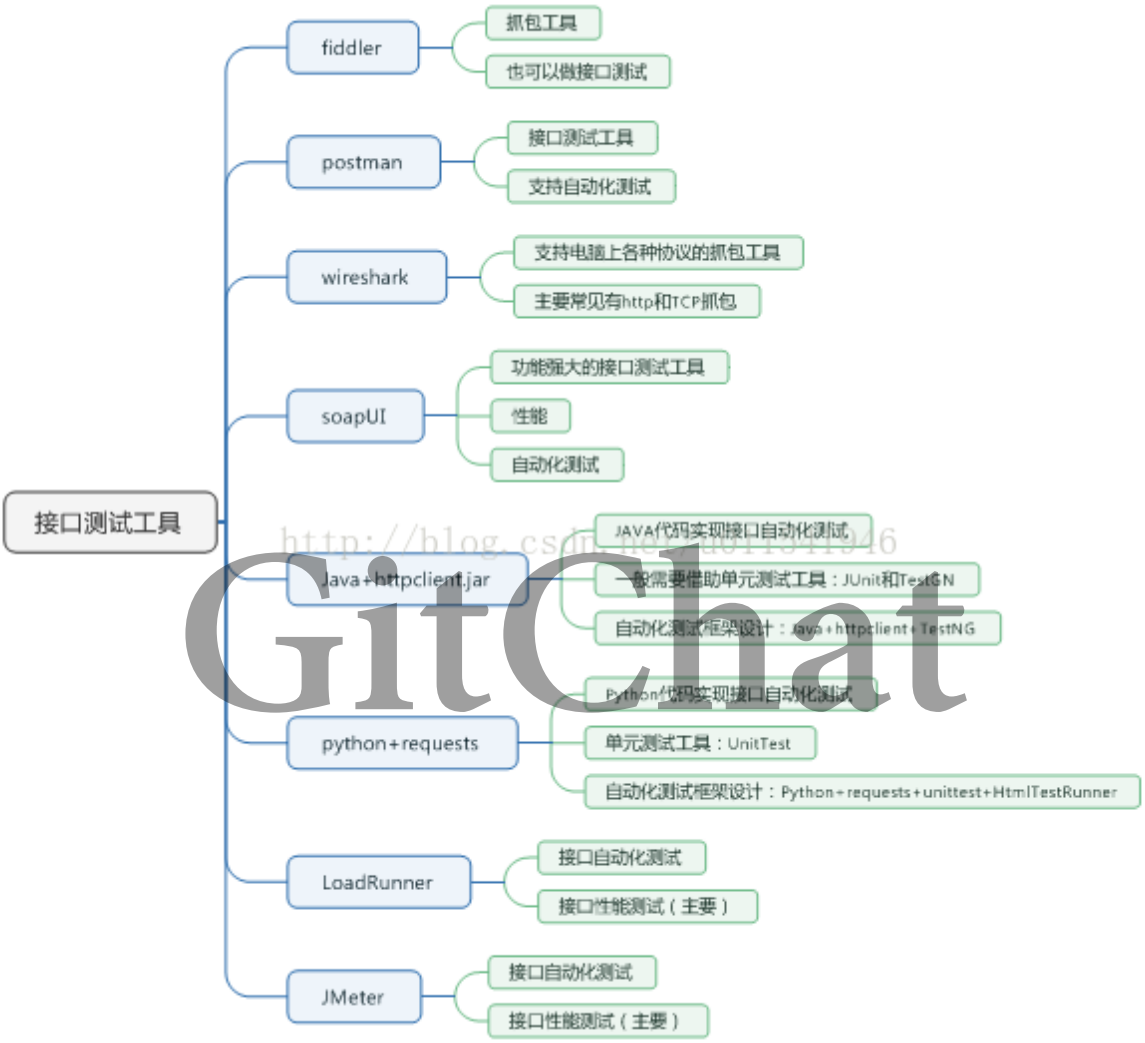


Test Automation Pyramid - Lisa Crispin

自动化测试金字塔 - 丽莎·克里斯潘

互联网服务端接口测试介绍

目前市场上有很多支持接口测试的工具。利用工具进行接口测试，能够提高测试效率。例如让你一天完成100个接口测试任务，你觉得你加班能否完成吗？你可能会说有工具可以帮忙呀，但不是所有工具都能支持你完成这个任务。下图是我挑选的几款工具，接下来对它们简单介绍一下。



1.fiddler

首先，这是一个 HTTP 协议调试代理工具，说白了就是一个抓 HTTP 包的工具。Web 测试和手机测试都能用到这个工具。既然是 HTTP 协议，这个工具也能支持接口测试。稍后文章，我们会专门介绍 fiddler 这个工具。

2.postman

这是一款 Google 工程师开发的一个插件，可以安装到 Chrome 浏览器上。支持不同接口测试请求，能够管理测试套件和自动化运行，弱点在于，自动化断言功能不强大。不能和 Jenkins、代码管理库进行持续集成测试。但是，它绝对是一个很好的半手工，半自动化测试工具。我一般在写自动化接口测试用例，会打开 postman 进行辅助测试和 Debug。这个工具稍后也会在文章中介绍。

3.wireshak

这是一款抓包工具，支持抓各种包，对 TCP、UDP、HTTP 都支持。如果做底层网络数据测试，一般都需要用到它。作为接口测试，这个软件有点不友好。因为刷新数据太快，不好定位每个操作对应的接口。所以，在这里不准备过多介绍该工具。

4.soupUI

这是一款提供有开源免费版和企业收费版的软件。在国外的接口测试中，使用较多。该工具能够支持接口自动化测试和接口性能测试，也支持和 Jenkins 做持续集成测试。了解一下就可以，可以下载一个社区免费版，做一个 Demo 试试。

5.Java 代码做接口测试

为什么要用代码做接口自动化测试呢？一些工具功能是有限制，很多公司需要一些特定的功能，工具不支持，只好用代码进行开发。一般用 Java 做自动化测试，主要利用 httpclient.jar 包，然后利用 JUnit 或者 TestNG 这样的单元测试工具，进行测试用例的开发，接着在 Jenkins 上创建一个 job，进行持续集成测试。

6.Python 代码做接口测试

和 Java 一样，用 Python 做接口测试，可以利用一个功能强大的第三方库 Requests，它能够方便地创建接口自动化用例。Python 下的单元测试框架，一般采用unittest。生成测试报告，一般选择 HTMLTestRunner.py。同样，可以结合 Jenkins 做持续集成测试。

7.LoadRunner

不要以为 LoadRunner 只能做性能测试，它同样可以做接口自动化和接口压力测试。只是我们很多人，不会利用 LoadRunner 的函数，设计接口测试用例。

8.JMeter

JMeter 同 LoadRunner 一样，都以性能测试出名，一般用 JMeter 也是做接口性能测试。例如 Java+Jmeter+Ant+Jenkins 做接口性能监听测试。JMeter 如何做接口测试，可以查阅官方文档和 CSDN 博客专栏《JMeter 性能测试》 。

上面说了这么多工具，基本覆盖了接口功能测试、接口自动化测试、接口性能测试。这里提一下，推荐 Python 语言下的一个性能测试工具 Locust。自己百度并安装下，很简单的 Web 界面，作为一个轻量级的协程测试工具，感觉很不错。

服务端测试也应该具备一些计算机基础知识的能力，例如下面几个领域的知识。

操作系统和网络

互联网里最大的应用场景就是高并发、高可用、高性能的线上服务，做这类系统实际上考验的是我们对操作系统和网络的理解。任何一个系统最后都是运行在操作系统之上的，也都运行在网络之上的，包括分布式系统，所以，需要在操作系统和网络上一定要有较深的造诣，尤其是高并发和高性能。如果对操作系统原理一无所知，基本很难理解

什么是并发和锁，很难理解高性能用什么指标来衡量，以及怎么实现高并发、高可用和高性能。

对于操作系统，我们必须了解 CPU 的多核体系结构、内存分页和缓存技术、磁盘 IO 的优略和网卡 IO 的情况，并且要理解计算机的工作原理，会根据这些指标粗略评估服务能够输出的性能。

对于网络，必须理解理论上定义的7层模型，了解 TCP/IP 的三次握手。另外我们在分布式服务架构中多数使用应用层的HTTP协议，所以还需要对 HTTP 协议有很深刻的理解。

算法和数据结构

应用层面的小伙伴们可能对算法和数据结构的应用比较少，即使有应用也比较简单，但是算法能力代表了一个人的逻辑思维和思考能力，能把各种基础算法理解的人智商都不会低，能够把程序写好的人逻辑思维一定很强。一般在面试小伙伴的时候，我都会考察一下他会不会高级算法，例如递归、剪枝、贪婪、动态规划。仔细想一下就会知道，会动态规划的人，他不是勤奋的就是聪明的，无论哪一样，你都有录取他的冲动。

线上高并发服务方向

线上高并发服务是个强需求，无论你开发哪类应用和网站，线上服务都是必须的，有了服务才有了功能，才有了产品，我也主要从事互联网后台高并发服务的设计与实现。解决高并发服务其实并没有那么难，这些年高并发服务的技术栈也已经没有了门槛，只要投入足够的成本，构建可伸缩的服务不是问题，达到多大的 TPS 也都不是问题，核心思想就是“分而治之，大而化下，小而化了”。

我们使用的缓存抗读、消息队列抗写、数据库分库分表、缓存分片、应用层伸缩、减少竞争、7层负载均衡、三四层负载均衡等，以及 CDN、DNS 轮训等等，这一切都是“分而治之”的思想。

大数据方向

大数据是最近比较火的方向，大小公司都在使用大数据技术。大公司使用大数据分析技术从众多数据中分析出业务模型，得出有价值的结果，来帮助企业制定市场和销售策略，中小公司多数使用大数据出报表和做风控等。

所有的大数据技术的根基都来自 Mapreduce、GFS 和 Bigtable 这三篇论文，推荐大家阅读这些论文：

- MapReduce: Simplifed Data Processing on Large Clusters
- The Google File System
- Bigtable: A Distributed Storage System for Structured Data

理解了这些基础技术原理，再去学习 Hadoop、HBase、Storm、Spark、Cassandra、MongoDB、ES 等都不是问题。

接口测试常用的工具、平台、框架

自动化测试可以快速自动完成大量测试用例，节约巨大的人工测试成本；同时需要拥有专业开发技能的人才能完成开发，且需要大量时间进行维护（在需求经常变化的情况下），所以大部分具有很好开发技能的人员不是很愿意编写自动化用例。但由于软件规模的高速增长，人力资源的逐步稀缺，自动化测试已是势在必行。

下面是我精选的8个 Java 测试工具和框架。

1.Arquillian

Arquillian 是一个基于 JVM 高度创新性和可扩展的测试平台，允许 Java 开发人员轻松创建自动化集成、功能和验收测试。Arquillian 允许在运行时间执行测试。可以用来管理单个或多个容器的生命周期，捆扎测试用例、从属类和资源。它还能够部署归档到容器中，在容器中执行测试、捕获结果，并创建报告。它集成了常见的测试框架，如 JUnit 4、TestNG 5，并允许使用现有的 IDE 发布测试，并且由于模块化的设计使其能够运行 Ant 和 Maven 测试插件。

2.JTest

JTest 也被称为“Parasoft JTest”，是一款由 Parasoft 公司推出的自动化 Java 软件测试和静态分析软件。JTest包含的功能有：单元测试情况下的生成和执行、静态代码分析、数据流的静态分析、度量分析、回归测试、运行时错误检测。此外，它还具备了同行代码审查流程自动化和运行时错误检测的功能，如竞态条件、异常、资源和内存泄漏、安全漏洞攻击。

3.The Grinder

The Grinder 是一个 Java 负载测试框架，运行简单，且其分布式测试采用了许多负载注入机器。只要有 Java API，The Grinder 就可以进行负载测试。这包括 HTTP Web 服务器、SOAP 和 REST Web 服务器、应用程序服务器，以及用强大的 Jython 和 Clojure 语言写的包含了自定义协议的测试脚本。The Grinder 的 GUI 控制台允许对多个负载注射器进行监测和控制，并自动管理客户端连接和 Cookies、SSL、代理感知和节流连接。 The Grinder 在 BSD 风格的开源许可下是免费的。
官方网站：Downloading The Grinder

4.TestNG

TestNG 是一款为 Java 编程语言设计的测试框架，灵感来自于 JUnit 和 NUnit。TestNG 可覆盖范围更广的测试类型，如单元、功能性、端到端、一体化等。它还有一些新的功能，可以使之更强大和更容易使用，如注解、具备大型线程池各种策略的运行测试、多线程的代码测试、灵活的测试配置、参数化数据驱动的测试支持等等。TestNG 支持各种各样的工具和插件，比如 Eclipse、IDEA、Maven 等等。

5.JUnit

JUnit 是一个为 Java 编程语言设计的单元测试框架。JUnit 为测试驱动开发框架的发展发挥了重要作用。它是现在被统称为 xUnit 的单元测试框架大家庭的组成成员之一，源于 SUnit。在编译时，JUnit 可以连接作为 JAR，用于编写可重复的测试。

6.Powermock

PowerMock 是一款支持单元测试源代码的 Java 框架。虽然 PowerMock 可以作为 Mocking 框架，例如作为 Mockito 和 EasyMock 的扩展而运行，但它还具备更强大的能力。PowerMock 利用自定义的类加载器和字节码操纵器，来确保静态方法的模拟、静态初始化的删除、函数构造、最终的类和方法以及私有方法。它的主要目的是通过最少的方法和注释来扩展现有的 API，以获得额外的功能。

7.Cucumber

Cucumber 是 BDD 模式下实现可执行规范（Executable Specifications）的开源工具，但是它的使命并不局限于做自动化验收测试，更加重要的在于其能够在团队成员之间构建统一的交流基础（feature 文件）、规范交流用语（Domain Specific Language）、提高各个利益相关方（Business Stakeholders）沟通效率和效果，从而达到提升产品质量、做出客户期望得到的产品这一最终目标。

8.Python Unittest

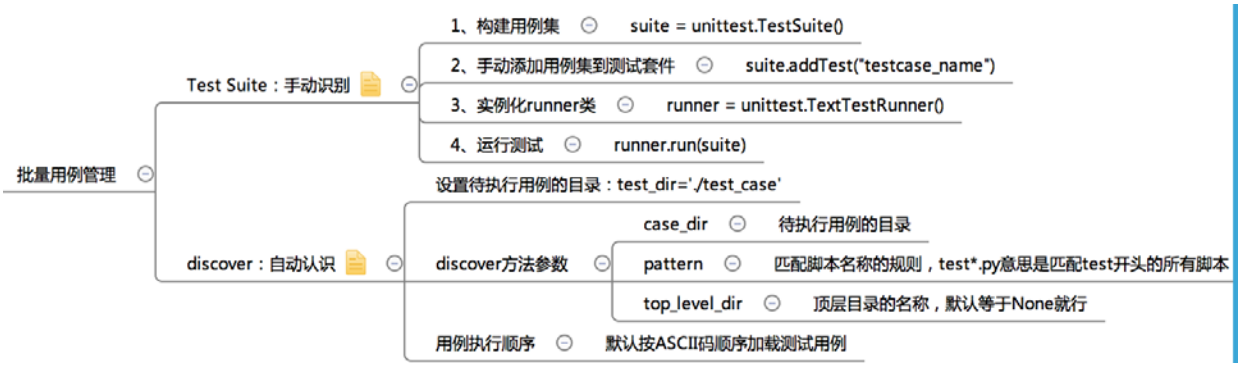
Python 单元测试框架 Unittest，是基于 Java 中流行单元测试框架 JUnit 设计的，其功能强大且灵活，对于熟悉 JUnit 的人来说，掌握 Unittest 很简单。

Unittest 涉及的知识点较多，但核心的只有一部分，本文将介绍它最核心和基础的内容。

类似 JUnit，使用 Unittest 编写 Python 的单元测试代码，包括如下几个步骤：

1. 编写一个 Python 类，继承 Unittest 模块中的 TestCase 类，这就是一个测试类。
2. 在上面编写的测试类中定义测试方法（这个就是指的测试用例），每个方法的方法名要求以 test 打头，没有额外的参数。在该测试方法中调用被测试代码，校验测试结果，TestCase 类中提供了很多标准的校验方法，如最常见的 assertEqual。
3. 执行 unittest.main()，该函数会负责运行测试，它会实例化所有 TestCase 的子类，并运行其中所有以 test 打头的方法。

以下是Python Unittest 测试用例执行的关系图。



我们下面看一些例子，编写如下的 Python文件。

（1）手工加载批量用例。

```
import unittest

class TestOne(unittest.TestCase):
    def setUp(self):
        print '\ncases before'
        pass

    def test_add(self):
        '''test add method'''
        print 'add...'
        a = 3 + 4
        b = 7
        self.assertEqual(a, b)

    def test_sub(self):
        '''test sub method'''
        print 'sub...'
        a = 10 - 5
        b = 5
        self.assertEqual(a, b)

    def tearDown(self):
        print 'case after'
        pass

if __name__ == '__main__':
    # 1、构造用例集
    suite = unittest.TestSuite()

    # 2、执行顺序是安加载顺序：先执行test_sub，再执行test_add
    suite.addTest(TestOne("test_sub"))
    suite.addTest(TestOne("test_add"))

    # 3、实例化runner类
    runner = unittest.TextTestRunner()

    # 4、执行测试
    runner.run(suite)
```


（2）自动加载批量用例。

```
import unittest
import os

class TestOne(unittest.TestCase):
    def setUp(self):
        print '\ncases before'
        pass
```

```
    def test_add(self):
        '''test add method'''
        print 'add...'
        a = 3 + 4
        b = 7
        self.assertEqual(a, b)

    def test_sub(self):
        '''test sub method'''
        print 'sub...'
        a = 10 - 5
        b = 5
        self.assertEqual(a, b)

    def tearDown(self):
        print 'case after'
        pass

if __name__ == '__main__':

    # 1、设置待执行用例的目录
    test_dir = os.path.join(os.getcwd())

    # 2、自动搜索指定目录下的cas，构造测试集,执行顺序是命名顺序：先执行
test_add，再执行test_sub
    discover = unittest.defaultTestLoader.discover(test_dir,
pattern='test_*.py')

    # 实例化TextTestRunner类
    runner = unittest.TextTestRunner()

    # 使用run()方法运行测试套件（即运行测试套件中的所有用例）
    runner.run(discover)
```


接口测试的一些问题和实践

以 TestNG 为例，它是一套根据 JUnit 和 NUnit 思想而构建的利用注释来强化测试功能的一个测试框架。TestNG 设计涵盖所有类型的测试，如单元、功能、端到端、集成等。学习 TestNG 之前需要先学习编程语言 Java、配置本地 JDK 环境（ JDK1.5 版本或以上 ）和安装 Java 开发工具 Eclipse。

接下来，我们一起来学习 TestNG。

1.在 Eclipse 中安装 TestNG。

打开 Eclipse Help -> MarketPlace，在搜索框里面输入 TestNG 搜索，然后安装 TestNG 插件。

2.安装成功后，在项目的 package 上右键可以看到 TestNG -> Create TestNG class。

这里可以勾选 TestNG 的注解方法，主要注解方式有：

- **@BeforeSuite**：被此注解的方法将在所有测试运行之前运行该方法。
- **@AfterSuite**：被此注解的方法将在所有测试运行之后运行该方法。
- **@BeforeTest**：被此注解的方法，将在测试运行之前运行。
- **@AfterTest**：被此注解的方法，将在测试运行之后运行。
- **@BeforeClass**：被此注解的方法，将在当前类的第一个测试方法调用之前运行。
- **@AfterClass**：被此注解的方法，将在当前类的所有测试方法调用之后运行。
- **@BeforeMethod**：被此注解的方法，将在每个测试方法调用之前运行。
- **@AfterMethod**：被此注解的方法，将在每个测试方法调用之后运行。
- **@DataProvider**：标志着一个方法，提供数据的一个测试方法。

至此环境搭建完成，可以开始写测试的 case 了。

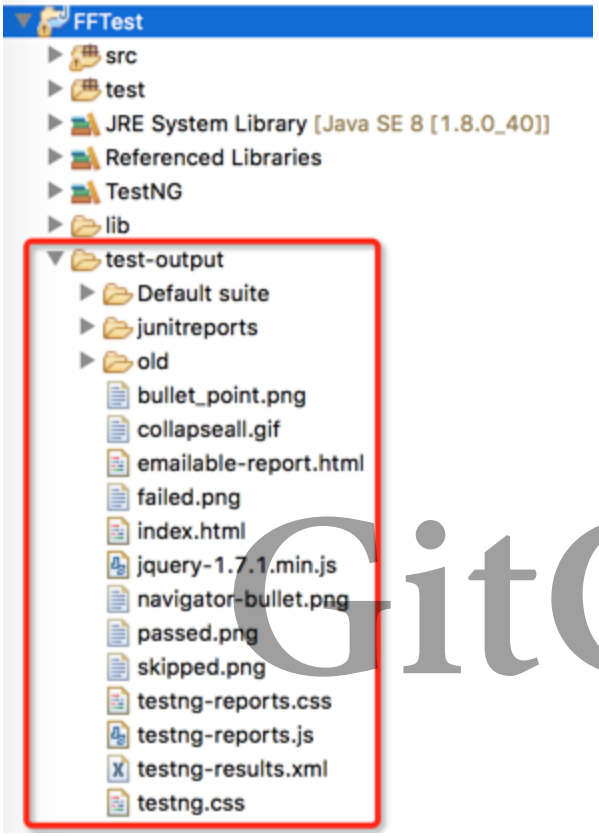
3.从一个简单的测试 case 开始入手，代码如下：

```
package com.pingan.ff.zijin;

import org.testng.annotations.Test;
import org.testng.annotations.DataProvider;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.AfterTest;

public class NewTest {
  @Test(dataProvider = "dp")
  public void f(Integer n, String s) {
      System.out.println("第一个参数是"+n+",第二个参数是"+s);
  }

  @DataProvider
  public Object[][] dp() {
    return new Object[][] {
      new Object[] { 1, "a" },
      new Object[] { 2, "b" },
    };
  }
  @BeforeTest
  public void beforeTest() {
      System.out.println("------------开始测试------------");
  }

  @AfterTest
  public void afterTest() {
      System.out.println("------------结束测试------------");
  }
}
```

右键 Run As -> TestNG Test，运行后结果如下图所示。

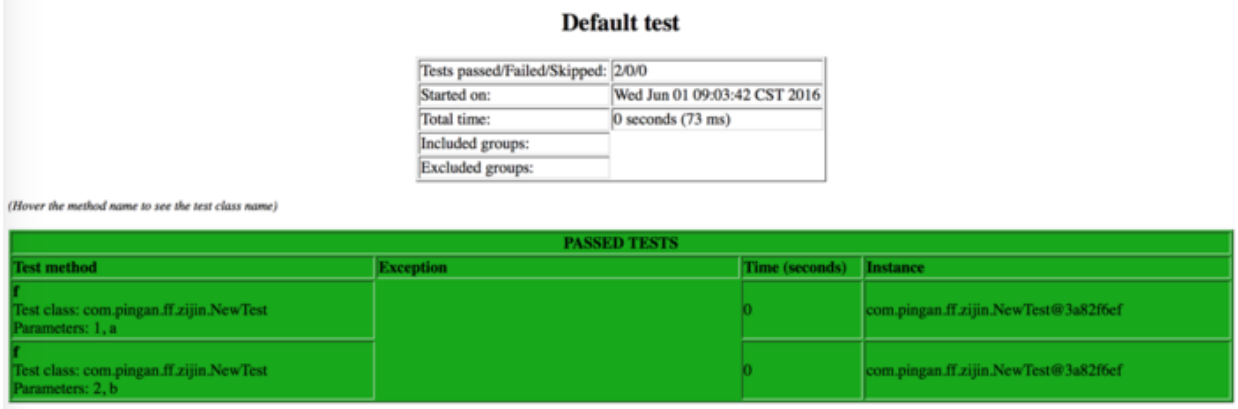
```
------------开始测试------------
第一个参数是1,第二个参数是a
第一个参数是2,第二个参数是b
------------结束测试------------
```

从测试的结果可以看到执行的顺序是 beforeTest() -> Test() -> afterTest()，同时 Test() 方法从 dataProvider dp 里面接收参数。如下图所示。



4.TestNG 默认情况下，会生成两种类型的测试报告 HTML 和 XML，测试报告位于 test-output 目录下。右键项目刷新一下项目就可以看到。

用浏览器打开 /test-output/Default suite/Default test.html 可以看到如下图的测试报告。



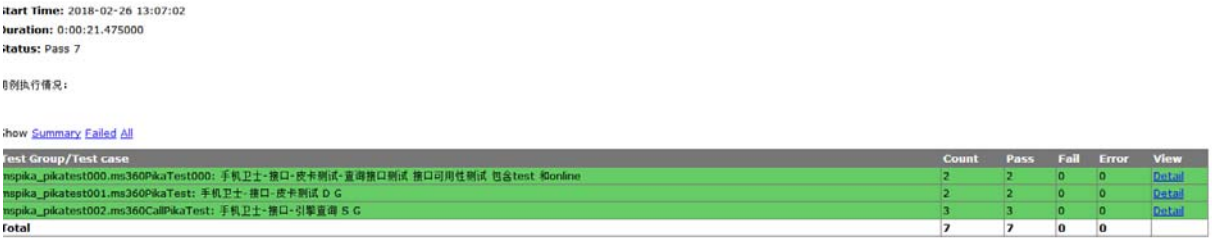
Default test

Tests passed/Failed/Skipped:	2/0/0
Started on:	Wed Jun 01 09:03:42 CST 2016
Total time:	0 seconds (73 ms)
Included groups:	
Excluded groups:	

(Hover the method name to see the test class name)

PASSED TESTS			
Test method	**Exception**	**Time (seconds)**	**Instance**
f Test class: com.pingan.ff.zijin.NewTest Parameters: 1, a		0	com.pingan.ff.zijin.NewTest@3a82f6ef
f Test class: com.pingan.ff.zijin.NewTest Parameters: 2, b		0	com.pingan.ff.zijin.NewTest@3a82f6ef

以 Pythonunit 为例，测试报告如下。



tart Time: 2018-02-26 13:07:02
uration: 0:00:21.475000
tatus: Pass 7

例执行情况：

how Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
nspika_pikatest000.ms360PikaTest000: 手机卫士-接口-皮卡测试-直调接口测试 接口可用性测试 包含test 和online	2	2	0	0	Detail
nspika_pikatest001.ms360PikaTest: 手机卫士-接口-皮卡测试 D G	2	2	0	0	Detail
nspika_pikatest002.ms360CallPikaTest: 手机卫士-接口-引擎直调 S G	3	3	0	0	Detail
Total	7	7	0	0	