N-Gitea单元测试

说明书

[v1.1]

北京航空航天大学计算机学院

二〇二〇年五月

文档修改记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本号 | 日期 | 修改章节 | 修改说明 | 修改人 | 审核人 |
| 1.0 | 2020.5.19 |  | 初稿 | 王伟民 | 麦梓健 |
| 1.1 | 2020.5.21 | 第二、三章 | 修改格式，完善内容 | 王伟民 | 麦梓健 |
| 1.2 | 2020.5.28 | 第二章 | 缩进统一，页码规范化 | 王伟民 | 麦梓健 |

目录

[1介绍 3](#_Toc385589981)

[2 测试框架选型与简介 3](#_Toc1652647668)

[2.1 Python单元测试框架 3](#_Toc495865778)

[2.2 Go单元测试框架 4](#_Toc1779580486)

[3 N-Gitea的测试 6](#_Toc1404476433)

[3.1 Python模块的测试 6](#_Toc2042645254)

[3.2 Go模块的测试 7](#_Toc1065203036)

# 1介绍

单元测试是软件测试的一种类型。测试的目的是确认目标代码在给定的场景下，有没有按照期望工作。顾名思义，单元测试就是测试的对象是程序中的最小单元的测试。程序的最小单元可以是一个函数、一个类，也可以是函数的组合、类的组合。

由于单元测试的特点，在单元测试中经常使用测试框架来编写测试用例，而测试框架会完成自动化测试、自动化测试用例执行、自动化测试报告生成等工作。

由于本项目开发涉及到Go和Python两种语言，因此本文档介绍本项目中Go和Python的单元测试框架的选型与使用。

# 2 测试框架选型与简介

## 2.1 Python单元测试框架

unittest是Python标准库中自带的单元测试框架，是Python语言的标准单元测试框架。unittest支持自动化测试、测试用例的初始化和关闭、测试用例的聚合等功能。

本项目没有选用unittest进行单元测试，选用的是pytest测试框架。pytest是Python另一个第三方单元测试库。它的目的是让单元测试变得更容易，并且也能扩展到支持应用层面复杂的功能测试。

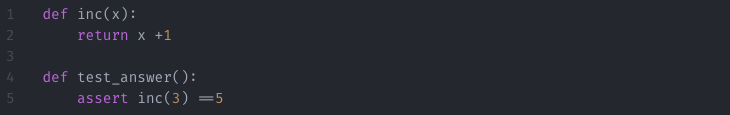
pytest的特性有：

* 支持用简单的assert语句实现丰富的断言
* 自动识别测试模块和测试函数
* 兼容unittest测试集
* 支持Python3和PyPy3
* 丰富的插件生态，已有300多个各式各样的插件，和活跃的社区

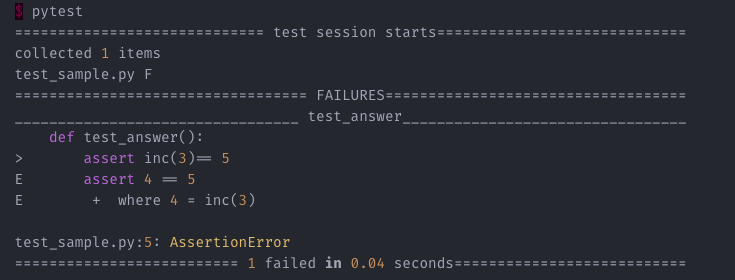
虽然unittest是Python自带的单元测试库，但是unittest的测试用例是基于类的，既需要了解testrunner、testsuite、testcase等基本概念，还需要熟悉面向对象编程，入门难度大于pytest。而pytest则为测试编写人员隐藏了这些细节，因而能够降低入门单元测试的难度。

pytest支持的测试方法有使用assert进行断言测试以及直接使用raise语句抛出异常。

一个简单的pytest示例如下：



执行结果如下：



可以看到测试用例测试没有通过，pytest输出了测试不通过的代码上下文和相关变量值便于排错。

## 2.2 Go单元测试框架

和Python语言一样，Go语言也自带了测试框架和工具，也就是testing包。在本项目使用testing包对Go模块进行测试。Go语言的testing包支持进行单元测试和性能测试。在本项目中，对testing包的使用主要关注单元测试部分。

**testing包的使用**

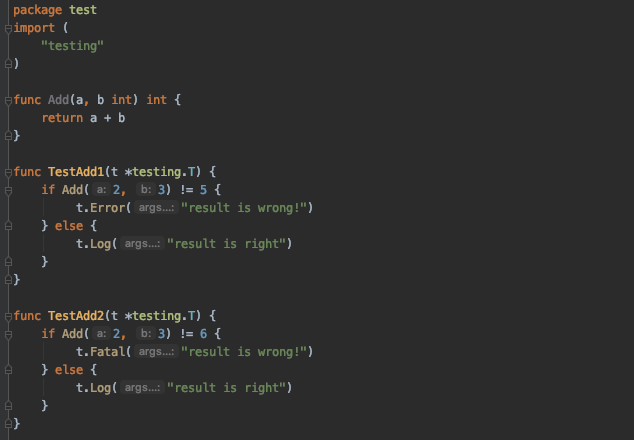
使用Go testing包编写测试用例需要遵守一些约定：

1. 测试代码放在\*\_test.go文件中，与被测代码放于同一个包中。
2. 每个测试用例为一个函数，测试函数必须是公开的函数，函数名以 Test 单词开头。
3. 测试函数的签名必须接收一个指向 testing.T 类型的指针，并且不返回任何值。

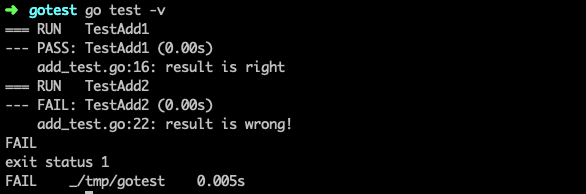
Go testing包支持的测试方法有：

* Fail: 标记失败，但继续执行当前测试函数
* FailNow: 失败，立即终止当前测试函数执行
* Log: 输出错误信息
* Error: 同时具有Fail和Log的功能
* Fatal: 同时具有FailNow和Log的功能
* Skip: 跳过当前函数，通常用于未完成的测试用例

一个简单的Go单元测试示例如下：



使用 go test -v 运行测试用例，测试结果如下：



可以看到TestAdd1测试用例测试通过，TestAdd2测试用例测试失败。输出结果还包含了出错的测试函数名称，执行时长和错误信息。

# 3 N-Gitea的测试

## 3.1 Python模块的测试

N-Gitea中的module\_server模块和用户自定义的通知模块是使用Python进行编写的，这一部分使用pytest进行测试。

测试开始前需要搭建测试环境：

1. 安装依赖，在项目根目录下运行：

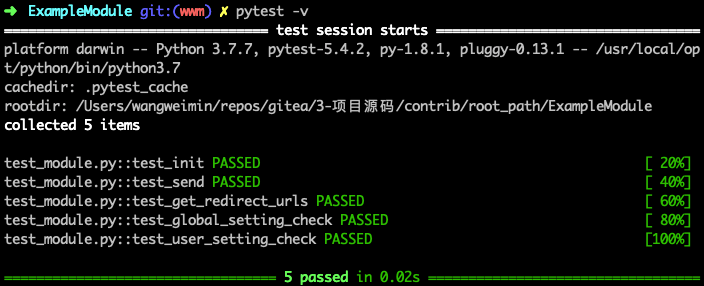
pip install -r 3-项目源码/modules/notification/module\_server/requirements.txt

pip install pytest==5.4.2

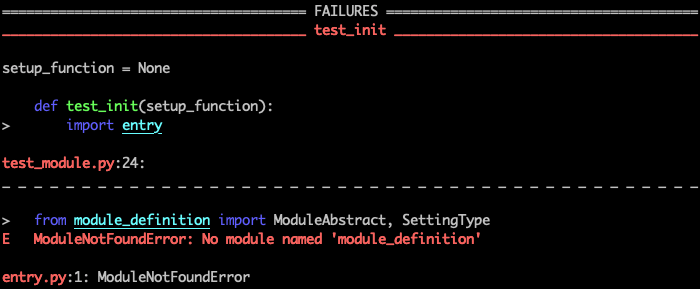
2. 编译gRPC需要类型定义和存根(Stub)代码：

python -m grpc\_tools.protoc -I"3-项目源码/modules/notification/proto" --python\_out="3-项目源码/modules/notification/module\_server" --grpc\_python\_out="3-项目源码/modules/notification/module\_server" service.proto

测试环境准备完成后就可以进行单元测试，在Python模块根目录运行 pytest -v 运行Python模块的测试：



若测试通过pytest会以绿色文本显示测试摘要，上例对项目的示例Python动态模块的各个方法进行了测试，可以看到模块的5个方法全部测试通过。若测试过程失败，pytest会对错误消息进行详细输出：



## 3.2 Go模块的测试

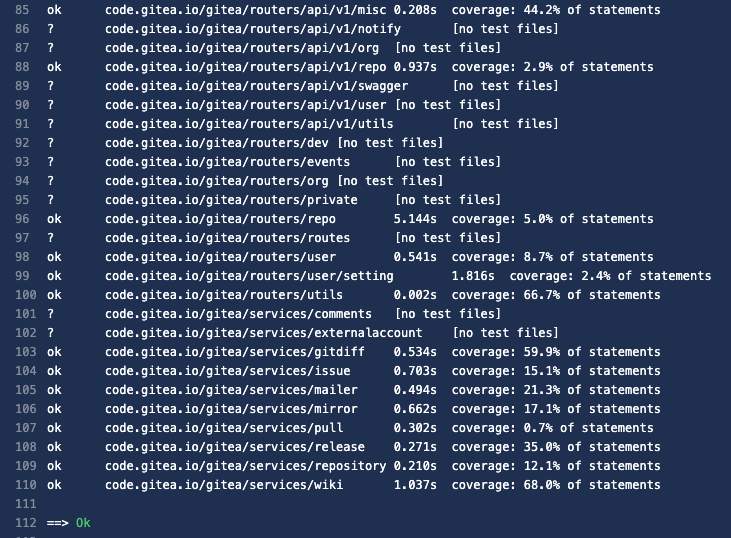
本项目继承了Gitea原项目的测试套件，可以直接使用以下命令进行单元测试，并生成测试覆盖报告。

make unit-test-coverage test-check

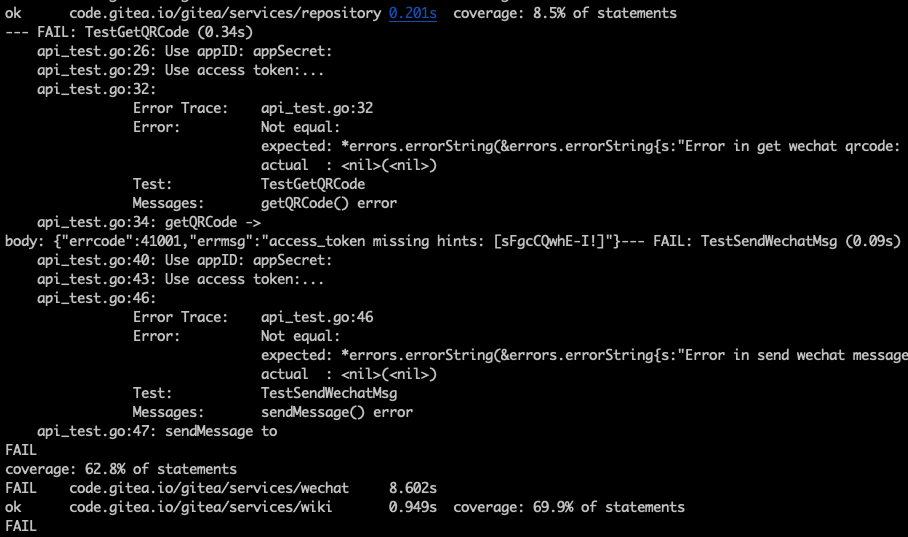
该命令底层使用了上文介绍的Go testing包进行测试用例的查找和执行：



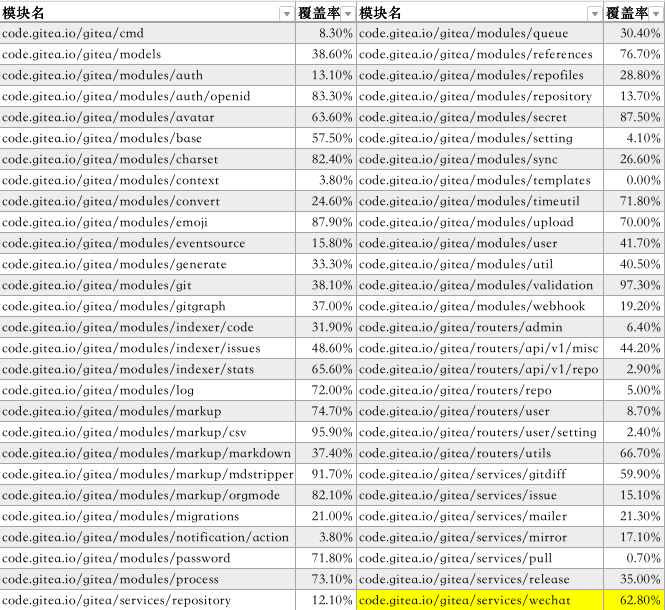
测试进行过程中，每运行完一个模块的测试用例，测试框架会输出模块的测试结果以及测试覆盖率：



当全部模块都通过时，测试结束，测试命令正常返回；当有模块测试不通过时，测试框架输出测试不通过的测试用例位置以及错误提示：



经过单元测试，得到Gitea项目各模块的单元覆盖率如下：



其中，我们扩展的微信通知模块的代码覆盖率有62.8% 。