

Lab4 Agileffy的测试

小组成员:胡志峰、杨俊逸、蔡彦麓、赵伟承、吴钟立

测试与持续集成

Agileffy开发小组为了合作开发的便利性，使用了GitHub¹作为代码协作环境，使用git作为版本控制软件。我们的开发小组的主页为[Agileffy Development Foundation](#)，前端代码仓库为[Agileffy](#)，后端代码仓库为[AgileffyServer](#)，最新的发行版本可以通过[agileffy.github.io](#)访问。

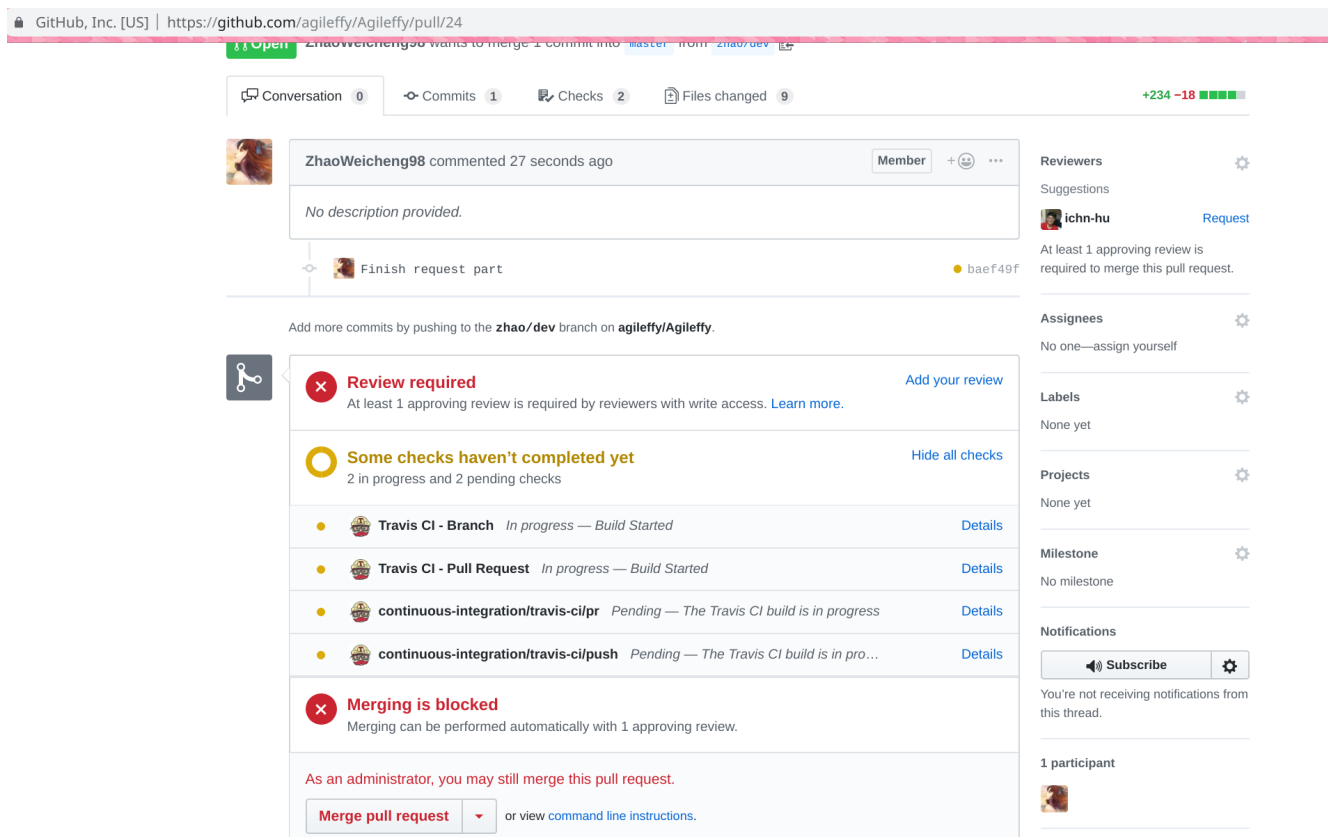
在本Lab中我们对于Agileffy项目已经开发完成的部分进行了系统性的集成测试，注意到虽然我们现在才描述Agileffy的测试，但实际上在项目的一开始我们就留意到了测试对于多人合作的复杂软件开发过程的重要性，因此一开始就构建了鲁棒的测试框架，并持续使用到现在。

因为开发过程由多人共同完成代码，测试在这个过程中显得尤为重要，因为我们需要在开发中避免合并代码引入漏洞。如果在开发过程中没有有效的测试，在整个项目完成之后将无法有效地找到漏洞所在，也无法对引入漏洞的提交进行追责。

而引入测试则不可避免地需要我们构建一套方便的工作流程（workflow），因此自动化的持续集成是必要的。我们使用了travis²提供的服务为我们的代码仓库构建了持续集成的支持，对于任何pull request到主分支的代码进行构建和测试，只有在通过构建和测试后才允许合并到主分支中。同时为了保证提交的代码质量，我们额外也使用了代码风格和错误的检测工具（eslint, tslint, pylint），并且通过配置文件规定了统一的代码风格。

我们采用的工作流程如下

1. 开发者（小组成员）pull主代码库，并新建分支 `用户名/dev`，在该分支下进行开发
2. 完成增量开发，commit文件变化，push到GitHub上host的代码仓库，并提起Pull Request
3. 在Pull Request的分支上自动化运行持续集成，通过构建和所有的测试用例后允许Merge
4. 通过GitHub提供的Branch Protection，我们要求至少有另外一个开发者完成代码审核才能Merge
5. Merge代码到主分支上，travis自动构建并部署到[agileffy.github.io](#)



实践中我们十足地受益于该工作流程，具体可以参加我们的[commit历史记录](#)。

前端测试

前端我们采用的技术栈为 Vue+Vuetify+Typescript，我们分别对前端进行单元测试（针对TypeScript语言文件，以白盒为主）和端到端测试（针对Vue文件，以黑盒为主）。

单元测试（Unit Test）

单元测试采用Jest框架和Mocha+Chai断言库进行编写对于ts文件进行测试，并且在Jest框架下进行测试覆盖率检测。

```
===== Coverage summary =====
Statements : 34.34% ( 34/99 )
Branches   : 70% ( 14/20 )
Functions  : 27.78% ( 5/18 )
Lines      : 33.67% ( 33/98 )
=====

Test Suites: 2 passed, 2 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        3.646s, estimated 6s
Ran all test suites.
```

Time 模块的单元测试

对于Time模块的测试，由于Time模块本质上是四个平行的分支结构，因此我们只需要相应的设置对应的测试样例即可，因此既是最大覆盖也是最小覆盖测试。

测试输入	期望输出
今日6：00：00	6：00：00
昨日6：00：00	昨天 6：00：00
一周内某天6：00：00 （假设周六）	星期六 6：00：00
超过一周某天6：00：00 （假设1970-01-01）	1970-01-01 6：00：00

端到端测试（E2E, End to End Testing）

端到端测试采用了Cypress作为基础框架，针对不同页面编写端到端测试，通过模拟用户输入，测试软件是否给出了符合要求的结果。测试过程可见以下两个视频文件：

[LOGIN 测试](#)

[REGISTER 测试](#)

=====

(Run Starting)

```
Cypress: 3.2.0
Browser: Electron 59 (headless)
Specs: 2 found (login.cypress.spec.js, register.cypress.spec.js)
```

Running: login.cypress.spec.js...

(1 of 2)

Login test

```
✓ Test title (2131ms)
✓ Username empty test (125ms)
✓ Password empty test (93ms)
✓ Password invisible test (319ms)
✓ Login button state (75ms)
✓ Register button state (743ms)
✓ Test unsuccessful login (357ms)
✓ Test login with error (127ms)
✓ Test successful login (85ms)
✓ Goto Register Page test (795ms)
```

10 passing (5s)

(Results)

```
Tests: 10
Passing: 10
Failing: 0
Pending: 0
Skipped: 0
Screenshots: 0
Video: true
Duration: 4 seconds
Spec Ran: login.cypress.spec.js
```

Running: register.cypress.spec.js...

(2 of 2)

Register test

- ✓ Test title (1328ms)
- ✓ Username empty test (174ms)
- ✓ Username too short test (211ms)
- ✓ Username too long test (709ms)
- ✓ Username counter test (738ms)
- ✓ Username illegal test (556ms)
- ✓ Username illegal test (528ms)
- ✓ Password empty test (114ms)
- ✓ Password invisible test (683ms)
- ✓ Password visible test (999ms)
- ✓ Password illegal test (307ms)
- ✓ Password illegal test (743ms)
- ✓ Password illegal test (430ms)
- ✓ Password illegal test (374ms)
- ✓ Password illegal test (316ms)
- ✓ Password illegal test (413ms)
- ✓ Password invisible test (914ms)
- ✓ Password repeat test (888ms)
- ✓ Email empty test (177ms)
- ✓ Email illegal test (497ms)
- ✓ Checkbox empty test (175ms)
- ✓ Register button state
- ✓ Register button state (1725ms)
- ✓ Test unsuccessful register (422ms)
- ✓ Test register with error (138ms)
- ✓ Test successful register (88ms)
- ✓ Goto login Page test (760ms)

27 passing (15s)

(Results)

```
Tests:      27
Passing:    27
Failing:    0
Pending:    0
Skipped:    0
Screenshots: 0
Video:      true
Duration:   14 seconds
Spec Ran:   register.cypress.spec.js
```

(Run Finished)

Spec	Tests	Passing	Failing	Pending	Skipped
✓ login.cypress.spec.js	10	10	-	-	-
✓ register.cypress.spec.js	27	27	-	-	-
All specs passed!	37	37	-	-	-

Login 页面的端到端测试

首先是针对表单的测试，由于表单都是必填项，因此我们对这些表单进行测试。同时我们需要测试在表单无效时，登录按钮是否可以点击。最后我们使用Cypress启动一个虚假的服务器后台并劫持来自前端的请求，并且返回可能的各种response（包括正常的OK response，正常的验证失败response和404等错误response），然后测试页面对于response的反应是否符合预期。另外，我们还对页面上的链接是否正确的进行了跳转进行了测试。

Register 页面的端到端测试

与Login测试类似，不过测试的重点在于表单项。对于一个表单项，我们分别针对其的require，最短，最长和正则表达式验证以最小覆盖的方法进行测试（用户名，密码，邮箱信息）。同时我们还额外针对密码输入区域的文字可见性，密码再次输入的一致性验证等逐一进行了测试。

后端测试

后端测试以白盒测试为主，本次针对注册和登录两个功能模块进行测试。

注册模块

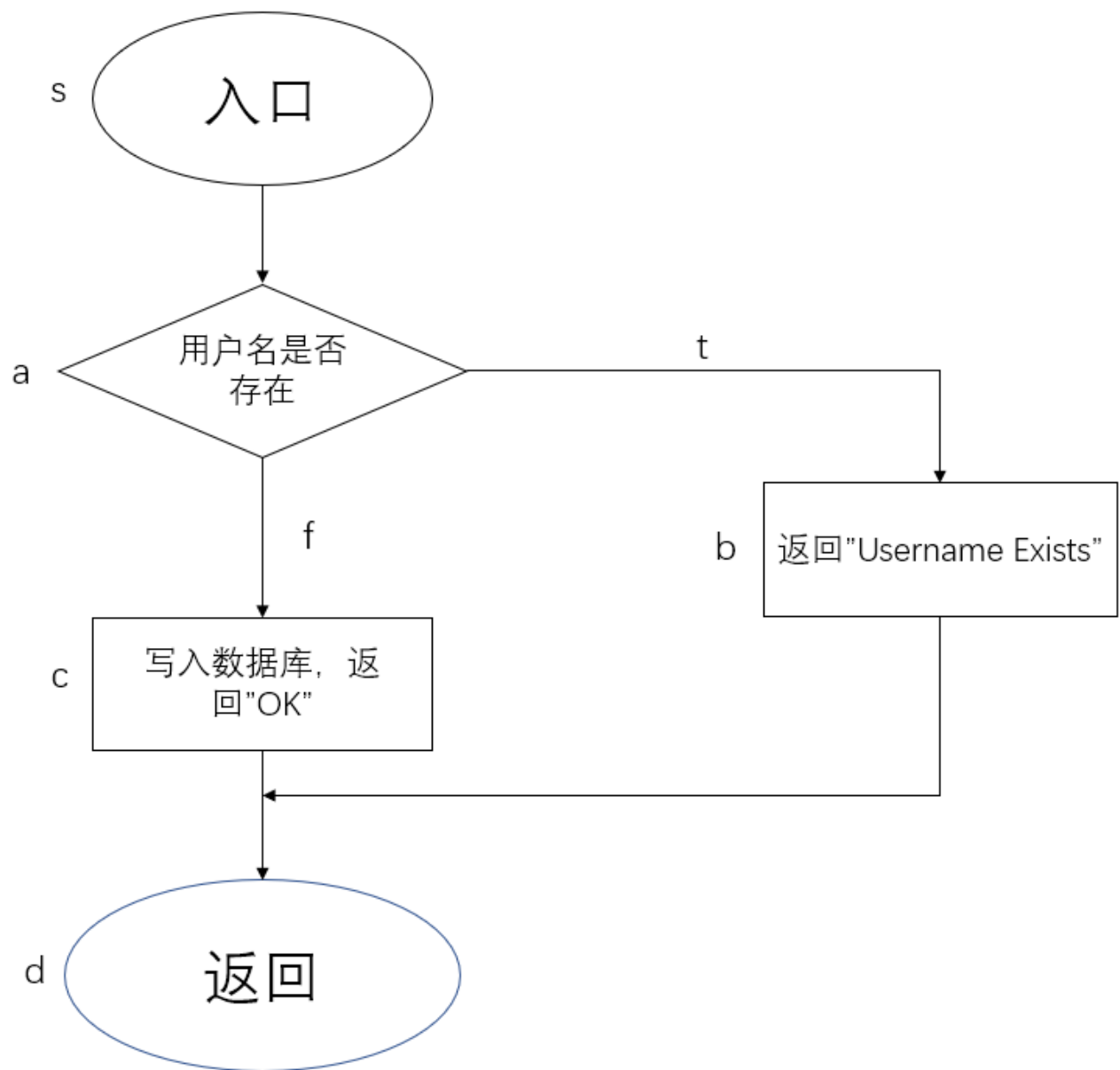
逻辑

注册模块代码详见[此链接](#)

注册模块接受来自前端的用户注册信息，包括用户名、密码、邮箱；其合法性由前端保证。后端的注册模块只检查用户名是否已存在，若用户名不存在则将用户注册信息写入数据库，若写入成功则返回"OK"信息，写入失败则返回"An Error occured on the server"信息；若用户名已存在则返回"Username Exists"信息。

测试

除了写入新用户信息失败的分支(该分支无法直接通过测试样例到达)，测试实现了判定——条件覆盖。在数据库为空时传入一条注册信息以进入注册成功分支，再传入一条相同的注册信息以进入用户名已存在的分支，再传入密码/邮箱改变而用户名不变的注册信息验证用户名已存在的分支只对用户名有效，再传入一条用户名不同的注册信息以再次验证注册成功分支，检查其输出。路径示意图、测试用例表以及测试代码如下：



测试数据	预期结果	路径			覆盖的条件
'username':'rebel', 'password':'123456', 'email': 'cylnb@cylnb.com'	"OK"	sacd			用户名尚未存在
'username':'rebel', 'password':'123456', 'email': 'cylnb@cylnb.com'	"Username Exists"	sabd			用户名已存在
'username':'rebel_', 'password':'123456', 'email': 'cylnb@cylnb.com'	"OK"	sacd			用户名尚未存在

```
import requests
```

```
url = 'http://127.0.0.1:5000/register'
```

```
d = {'username':'rebel',
      'password':'123456',
      'email' : 'cylnb@cylnb.com'}

r = requests.post(url, data=d)
print(r.text)

d = {'username':'rebel',
      'password':'123456',
      'email' : 'cylnb@cylnb.com'}

r = requests.post(url, data=d)
print(r.text)

d = {'username':'rebel',
      'password':'654321',
      'email' : 'cylnb@cylnb.com'}

r = requests.post(url, data=d)
print(r.text)

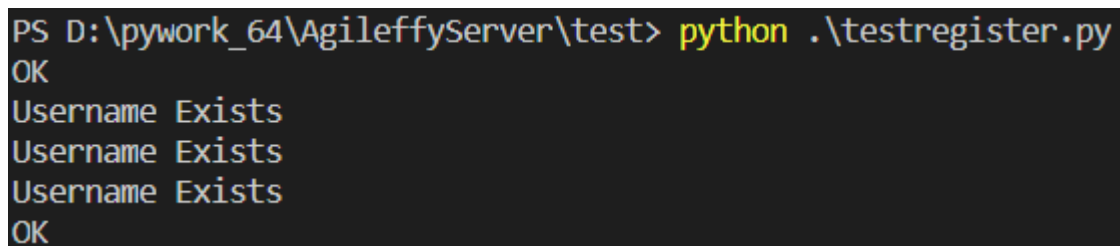
d = {'username':'rebel',
      'password':'123456',
      'email' : 'rebel@cylnb.com'}

r = requests.post(url, data=d)
print(r.text)

d = {'username':'rebel_',
      'password':'123456',
      'email' : 'cylnb@cylnb.com'}

r = requests.post(url, data=d)
print(r.text)
```

测试结果如下：



```
PS D:\pywork_64\AgileffyServer\test> python .\testregister.py
OK
Username Exists
Username Exists
Username Exists
OK
```

测试结果符合预期。

登录模块

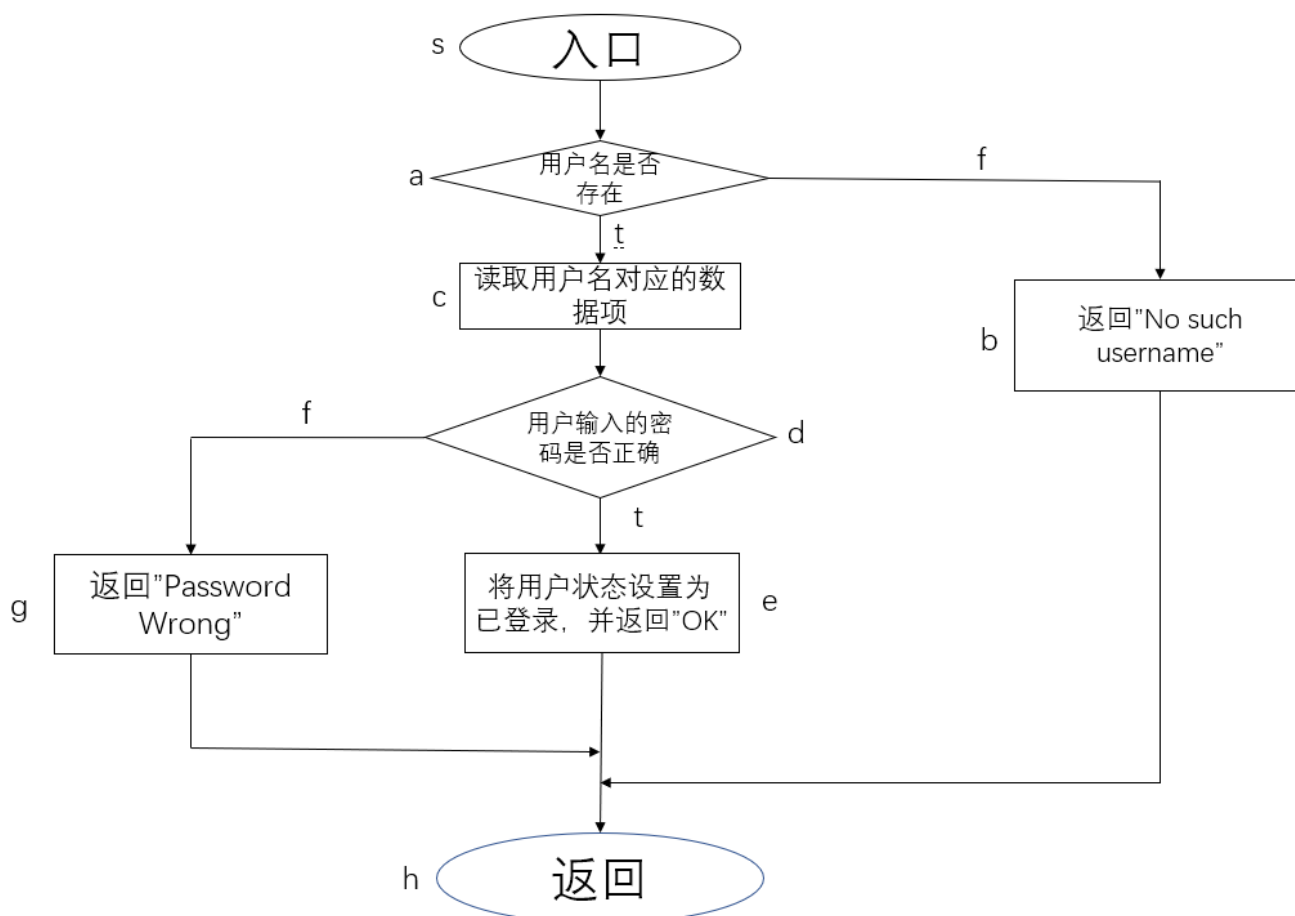
逻辑

登录模块代码详见[此链接](#)

登录模块接受用户的登录请求，然后检查数据库中是否存在用户名对应的数据项，若不存在相应的数据项，则返回"No such username"信息；否则找到相应的数据项，并将数据库中存储的用户密码与输入的密码进行比较，若密码一致，则将用户的状态设置为已登录，并返回"OK"信息；否则返回"Password Wrong"信息。

测试

测试实现了判定——条件覆盖。基于上一步已写入数据库的两条记录，先后输入：正确的用户名和密码以进入登陆成功分支、正确的用户名和错误的密码以进入用户名存在而密码错误的分支、不存在的用户名和密码以进入用户名不存在分支，以及另一条记录的正确的用户名和密码以再次验证登陆成功分支，检查其输出。路径示意图、测试用例表以及测试代码如下：



测试数据	预期结果	路径		覆盖的条件
'username':'rebel', 'password':'123456'	"OK"	sacdeh		用户名存在且密码正确
'username':'rebel', 'password':'654321'	"Password Wrong"	sacdgh		用户名存在且密码不正确
'username':'rebel0', 'password':'123456'	"No Such username"	sabh		用户名不存在

```
import requests

url = 'http://127.0.0.1:5000/login'
d = {'username':'rebel', 'password':'123456'}
```

```
r = requests.post(url, data=d)
print(r.text)

d = {'username':'rebel', 'password':'654321'}

r = requests.post(url, data=d)
print(r.text)

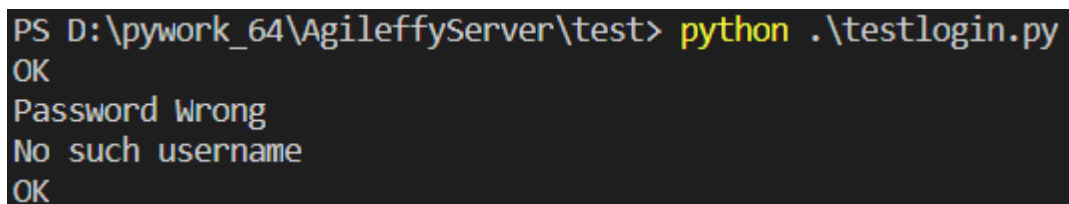
d = {'username':'rebel0', 'password':'123456'}

r = requests.post(url, data=d)
print(r.text)

d = {'username':'rebel_', 'password':'123456'}

r = requests.post(url, data=d)
print(r.text)
```

测试结果如下：

A terminal window with a black background and white text. The prompt is 'PS D:\pywork_64\AgileffyServer\test>'. The command 'python .\testlogin.py' has been executed, resulting in four lines of output: 'OK', 'Password Wrong', 'No such username', and 'OK'.

```
PS D:\pywork_64\AgileffyServer\test> python .\testlogin.py
OK
Password Wrong
No such username
OK
```

测试结果符合预期。

1. <https://github.com/>

2. <https://github.com/apps/travis-ci>