

Python 模块

目录

1. JSON 模块
2. PICKLE 模块
3. configparser 模块
4. XML
5. Requests 模块
6. Request 调用 Gerrit API

Json 模块

4 个功能: **dump dumps load loads**

Dumps: 将字典转换成字符串

```
>>> dic={'age':23,'job':'student'}
>>> dic_str=json.dumps(dic)
>>> dic_str
'{"age": 23, "job": "student"}'
>>> type(dic_str)
<type 'str'>
>>> type(dic)
<type 'dict'>
```

Loads: 将字符串转换成字典

```
>>> dic_obj=json.loads(dic_str)
>>> dic_obj
{'age': 23, 'job': 'student'}
>>> type(dic_obj)
<type 'dict'>
```

Dump: 字典存储到文件中

```
>>> import json
>>> dic={'age':23,'job':'student'}
>>> with open('abc.json','w') as f:
...     json.dump(dic,f)
...
>>>
```

写入 abc.json 文件中

Load: 读取文件中的内容

```
>>> with open('abc.json','r') as f:
...     obj=json.load(f)
...     print(obj)
...
{'age': 23, 'job': 'student'}
```

pickle 模块

4 个功能: **dumps loads dump load**

dumps: 存入数据

```
>>> dic={'age':23,'job':'student'}
```

```
>>> byte_data=pickle.dumps(dic) 将字典转换成字符串
```

```
>>> byte_data
```

```
"(dp0\nS'age'\nnp1\nI23\nsS'job'\nnp2\nsS'student'\nnp3\ns."
```

loads:读取数据

```
>>> obj=pickle.loads(byte_data)
```

```
>>> obj
```

```
{'age': 23, 'job': 'student'}
```

存储数据到文件中，使用 **dump** 和 **load**。由于 **pickle** 写入的是二进制数据，所以打开方式需要以 **wb** 和 **rb** 的模式。

=====conf

igparser 模块,只能在 **python3** 中使用

Configparser 是用来读取配置文件的包,配置文件格式如下:

```
1 [db]      ([ ]为 section)
```

```
2 db_host = 127.0.0.1  (key=value 为 option)
```

```
3 db_port = 69
```

```
4 db_user = root
```

```
5 db_pass = root
```

```
6 host_port = 69
```

```
7
```

```
8 [concurrent]
```

```
9 thread = 10
```

```
10 processor = 20
```

案例:

(1): config.sections()得到所有 **sections** 的值

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.read("a.ini",encoding="utf-8")
{'a.ini'}
>>> config.sections()
['db', 'concurrent']
```

(2):获取指定 section 中的 option 的值

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.read("a.ini",encoding="utf-8")
{'a.ini'}
>>> r=config.options("db") ← 获取指定section的option的值
>>> r
['db_host', 'db_port', 'db_user', 'db_pass', 'host_port']
>>>
```

(3):获取指定 option 指点的值

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.read("a.ini",encoding="utf-8")
{'a.ini'}
>>> r=config.get("db","db_host") ← 获取指定option指点的值
>>> r
'127.0.0.1'
>>>
```

(4):获取指定 section 的所有值,会把 option 的键和值全部输出

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.read("a.ini",encoding="utf-8")
{'a.ini'}
>>> r=config.items("db") ← 获取指定section的所有值
>>> r
[('db_host', '127.0.0.1'), ('db_port', '69'), ('db_user', 'root'), ('db_pass', 'root'), ('host_port', '69')]
>>>
```

(5): 修改某个 option 的值, 如果不存在则会出创建

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.read("a.ini",encoding="utf-8")
{'a.ini'}
>>> config.set("db","db port","69")
>>> config.set("db","db port","80")
>>> config.write(open("a.ini","w"))
>>>
```

(6): 检查 section 或 option 是否存在, 返回 bool 值

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.has_section("section")
False
>>> config.has_option("section","option")
False
>>>
```

(7):添加 section 节点

```
>>> import configparser
>>> import configparser
>>>
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.add_section("user") ← 添加section节点, 然后把节点写入一个ini文件中
>>> config.write(open("config.ini","w"))
>>>
```

(8): 删除 section 节点

```
>>> import configparser
>>> config=configparser.ConfigParser()
>>> config.add_section("user")
>>> config.write(open("config.ini","w"))
>>> config.remove_section("user") ← 删除使用remove_section
True
>>> config.write(open("config.ini","w"))
```

(9)：删除指定 **section** 组内的 **option**

```
>>> import configparser

>>> config=configparser.ConfigParser()

>>> config.read("a.ini")

>>> config.remove_option("db","host_port")

>>> config.write(open("a.ini","w"))
```

(10)：添加 **option** 的键和值：

```
AttributeError: ConfigParser object has no attribute 'add_option'
>>> config.set("USER","name","gaoyuxia")
>>> config.set("USER","age","25")
>>> config.set("USER","sex","女")
>>> config.write(open("config.ini","w"))
```

添加option的值

XML

解析 XML

读取 **xml** 文件的内容：

```
>>> from xml.etree import ElementTree as ET

>>> tree=ET.parse("xo.xml")    ===》解析 XML

>>> root=tree.getroot()      ==>得到 Root 节点

>>> print(root.tag,root.attrib)    ==》tag 是 string,attrib 是字典{}
```

Root 的子节点

```
>>> for child in root:
...     print(child.tag,child.attrib)
...
('country', {'name': 'Liechtenstein'})
('country', {'name': 'Singapore'})
('country', {'name': 'Panama'})
>>>
```

得到root的子节点

Root 的子节点的 key

```
('country', {'name': 'Panama'})
>>> root[0].tag
'country'
>>> root[1].tag
'country'
>>> root[2].tag
'country'
>>> root[3].tag
```

Root 的子节点的子节点的 key

```

...
>>> root[0][0].tag
'rank'
>>> root[0][1].tag
'year'
>>> root[0][2].tag
'gdppe'
>>> root[0][3].tag
'neighbor'
>>> root[0][4].tag
'neighbor'
>>> root[0][5].tag

```

Element.findall()只查找直接的孩子，返回所有符合要求的 **Tag** 的 **Element**，而 **Element.find()**只返回符合要求的第一个 **Element**。如果查看 **Singapore** 的 **year** 的值，可以

```

>>> for country in root.findall('country'):
...     if country.attrib['name']=='Singapore':
...         years=country.findall('year')
...         print(years[0].text)
...
2026
>>> for country in root.findall('country'):
...     if country.attrib['name']=='Singapore':
...         years=country.findall('year')
...         print(years[1].text)
...

```

```

>>> for country in root.findall('country'):
...     root.remove(country)
...
>>> tree=ET.ElementTree(root)
>>> tree.write('a1.xml',encoding="utf-8")

```

删除指定的属性值

>>> for country in root.findall("country"):

... rank=country.find("rank").text **rank** 的内容

... name=country.get("name") **country** 的 **name** 属性

... print(name,rank)

...

Liechtenstein 2

Singapore 5

Panama 69

修改 **XML**

```

>>> for rank in root.iter("rank"):
...     new_rank=int(rank.text)+1
...     rank.text=str(new_rank)
...     rank.set("update","NO")
...
>>> ET.dump(root)
<data>
  <country name="Liechtenstein">
    <rank update="NO" updated="yes">3</rank>
    <year>2023</year>
    <gdppe>141100</gdppe>
    <neighbor direction="E" name="Austria" />
    <neighbor direction="W" name="Switzerland" />
  </country>
  <country name="Singapore">
    <rank update="NO" updated="yes">6</rank>
    <year>2026</year>
    <gdppe>59900</gdppe>
    <neighbor direction="N" name="Malaysia" />
  </country>
  <country name="Panama">
    <rank update="NO" updated="yes">70</rank>
    <year>2026</year>
    <gdppe>13400</gdppe>
    <neighbor direction="W" name="Costa Rica" />
    <neighbor direction="E" name="Colombia" />
  </country>
</data>
>>> tree.write("output.xml")

```

修改属性的值，dump将结果显示在屏幕上

```

>>> import xml.etree.ElementTree as ET
>>> tree=ET.parse("output.xml")
>>> root=tree.getroot()
>>> for rank in root.iter("rank"):
...     del rank.attrib["updated"]
...
>>> ET.dump(root)
<data>
  <country name="Liechtenstein">
    <rank update="NO">3</rank>
    <year>2023</year>
    <gdpp>141100</gdpp>
    <neighbor direction="E" name="Austria" />
    <neighbor direction="W" name="Switzerland" />
  </country>
  <country name="Singapore">
    <rank update="NO">6</rank>
    <year>2026</year>
    <gdpp>59900</gdpp>
    <neighbor direction="N" name="Malaysia" />
  </country>
  <country name="Panama">
    <rank update="NO">70</rank>
    <year>2026</year>
    <gdpp>1360</gdpp>
    <neighbor direction="W" name="Costa Rica" />
    <neighbor direction="E" name="Colombia" />
  </country>
</data>
>>>

```

删除指定的属性值

小结: 关于 `class xml.etree.ElementTree.Element` 属性相关

- **attrib** 为包含元素属性的字典
- **keys()** 返回元素属性名称列表
- **items()** 返回(**name,value**)列表
- **get(key, default=None)** 获取属性
- **set(key, value)** # 跟新/添加 属性
- **del xxx.attrib[key]** # 删除对应的属性
- **ET.dump(root)** 很有用可以在屏幕上显示输出结果,(括号中输入不同的值, 会输出不同的结果)

OS 模块学习

for root,dirs,files in os.walk(".",topdown=False):

... print(root,dirs,files)

Root: 当前正在遍历的这个文件夹的本身的地址

Dirs.: 是一个 **list** , 内容是该文件夹中所有的目录的名字(不包括子目录)

Files: 是一个 **list** , 内容是该文件夹中所有的文件(不包括子目录)

1: 使用 **python** 遍历一个目录 查找文件。(在一套源码中查找 **Version** 文件, 并返回它所在的目录)

1 `#!/usr/bin/python`

2 `# _ _ coding:utf-8 _ _`

3

```

4 import os

5

6 for root,dirs,files in os.walk("wind",topdown=True):

7     for file in files:

8         if filename=="version":

9             print(root,"找到了！ ")

```

2：使用 **python** 给 **manifest.xml** 添加 **path** 属性值

分析：首先使用 **ElementTree** 解析 **XML** 文件，并读取 **XML**

代码如下：

```

#!/usr/bin/python

#_*_ coding:utf-8 _*_

#给 xml 添加 path 属性

from xml.etree import ElementTree as ET

#解析 XML 文件

tree=ET.parse("manifest.xml")

#得到根节点

root=tree.getroot()

for node in root.iter("project"):

    list1=node.attrib

    if "path" not in list1.keys():

        name=list1["name"]

        #node.set("path",list1["name"])

        node.attrib["path"]=name

```

```
tree.write("abc.xml")
```

requests 模块

安装: `sudo apt-get install python-requests`

请求方式:

Get

(1): 基本 **get** 请求

```
import requests
```

```
response = requests.get('http://httpbin.org/get')
```

```
print(response.text)
```

(2): 带参数 **get** 请求

方法一:

```
>>> import requests
```

```
>>> response=requests.get("http://httpbin.org/get?name=germey&age=22")
```

参数的传

递方式

方法二:

```
>>> import requests
```

```
>>> data={
```

```
... "name":"germey",
```

```
... "age":23
```

```
... }
```

```
>>> response=requests.get("http://httpbin.org/get",params=data)
```

```
>>> print(response.text)
```

Put

Delete

Post:

```
>>> import requests

>>> data={"name":"tom","age":"22"}

>>> response=requests.post("http://httpbin.org/post",data=data)

>>> print(response.text)
```

Gerrit rest api 学习

```
>>> import requests

>>> from requests.auth import HTTPBasicAuth

>>> from requests.auth import HTTPDigestAuth

>>> import json

>>>
auth=HTTPBasicAuth("gerrit","WspgVEG3Bp0XOqqox6km2h2a+H9WZXAQJ3
Gc0yGPGA")

>>> data=json.dumps({"name"})

>>> header={'content-type':'application/json;charset=UTF-8'}
```

上面的几行话，为调用 **rest api** 的指定格式

1：查询指定用户的 ID

语法：

```
'GET / accounts / {account-id} '
```

```
>>> import requests

>>> from requests.auth import HTTPBasicAuth

>>> from requests.auth import HTTPDigestAuth
```

```

>>>
auth=HTTPBasicAuth("gerrit","FfP+4QVtxspyEj2+VXb8VUKpQxzLPad+Tffh
AGrYw")

>>>
response=requests.get("http://192.168.56.101:8081/a/accounts/?q=name:
gerrit",auth=auth)

>>> print(response.text)

}}'

[

{

    "_account_id": 1

}

]

```

12: 查询指定用户的信息

语法:

'GET / accounts / {account-id} '

```

>>> import requests

>>> from requests.auth import HTTPBasicAuth

>>> from requests.auth import HTTPDigestAuth

>>>
auth=HTTPBasicAuth("gerrit","FfP+4QVtxspyEj2+VXb8VUKpQxzLPad+Tffh
AGrYw")

>>>
response=requests.get("http://192.168.56.101:8081/a/accounts/gerrit",au
th=auth)

>>> print(response.text)

}}'

{

```

```

    "_account_id": 1,

    "name": "gerrit",

    "email": "gerrit@byd.com",

    "username": "gerrit"
  }
}]}'

{

  "_account_id": 1,

  "name": "gerrit",

  "email": "gerrit@byd.com",

  "username": "gerrit"

}

```

注意：默认情况下，所有 **REST** 端点都假定匿名访问和过滤结果与匿名用户可以读取的内容相对应（可能根本没有任何内容）。

用户（和程序）可以通过为端点 **URL** 添加前缀来进行身份验证 **/a/**。例如，要进行身份验证 **/projects/**，请求 **URL/a/projects/**。

Gerrit 使用 **HTTP** 基本身份验证和用户帐户设置页面中的 **HTTP** 密码

3：列出 **All-Projects** 的访问权限

```

>>> import requests

>>> from requests.auth import HTTPBasicAuth

>>> from requests.auth import HTTPDigestAuth

>>> import json

>>>
auth=HTTPBasicAuth("gerrit","WspgVEG3Bp0XOqqox6km2h2a+H9WZXAQJ3
Gc0yGPGA")

>>> data=json.dumps({"name"})

>>> header={'content-type':'application/json;charset=UTF-8'}

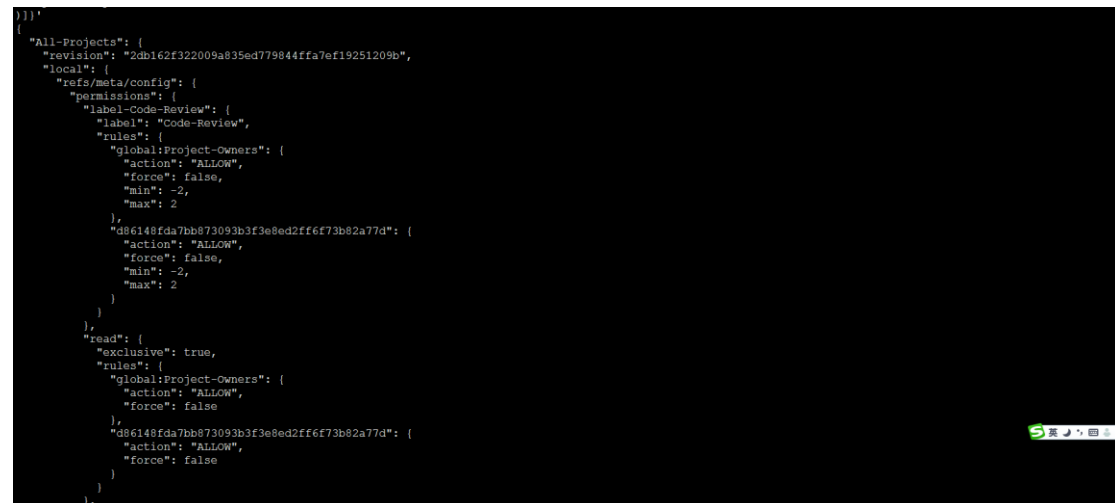
```

```
>>>
```

```
response=requests.get("http://192.168.56.101:8081/a/access/?project=All-Projects",auth=auth)
```

```
>>> print(response.text)、
```

输出结果如下图：



4：获取指定用户信息

语法：

方法一：（通过用户名称获取用户信息）

```
>>>
```

```
response=requests.get("http://192.168.56.101:8081/a/accounts/?suggest  
&q=gerri",auth=auth)
```

```
>>> print(response.text)
```

方法二：（通过用户 ID 获取用户信息）

```
>>>
```

```
response=requests.get("http://192.168.56.101:8081/a/accounts/self",auth  
=auth)
```

```
>>> print(response.text)
```

5：创建一个新帐户

语法:

'PUT / accounts / {username} '

>>>

```
response=requests.put("http://192.168.56.101:8081/a/accounts/chuisholi  
ng",auth=auth)
```

```
>>> print(response.text)
```

6: 获取用户的详细信息

语法:

'GET / accounts / {account-id} / detail'

>>>

```
response=requests.get("http://192.168.56.101:8081/a/accounts/chuisholi  
ng/detail",auth=auth)
```

```
>>> print(response.text)
```

获取当前用户的详细信息

>>>

```
response=requests.get("http://192.168.56.101:8081/a/accounts/self/detai  
l",auth=auth)
```

```
>>> print(response.text)
```

7: 获取帐户的名称;

语法:

'GET / accounts / {account-id} / name'

>>>

```
response=requests.get("http://192.168.56.101:8081/a/accounts/2/name",  
auth=auth)
```

```
>>> print(response.text)
```

8: 删除帐户名称

语法:

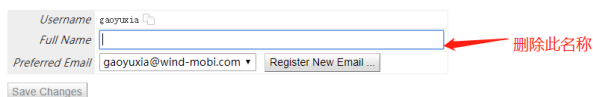
'DELETE / accounts / {account-id} / name'

>>>

```
response=requests.delete("http://192.168.56.101:8081/a/accounts/4/name",auth=auth)
```

```
>>> print(response.text)
```

这里删除的是 **FULL NAME**



9: 设置帐户名称: (没看明白)

语法:

'PUT / accounts / {account-id} / name'

```
>>> data=json.dumps({"name":"admin"})
```

>>>

```
res=requests.put("http://192.168.56.101:8081/a/accounts/5/admin",auth=auth,data=data,headers=header)
```

```
>>> print(res.text)
```

9: 获取帐户状态

语法:

'GET /accounts/{account-id}/status'

>>>

```
response=requests.get("http://192.168.56.101:8081/a/accounts/1/status",auth=auth)
```

```
>>> print(response.text)
```

10: 获取用户名

语法:

'GET /accounts/{account-id}/username'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/accounts/1/username",auth=auth)
```

```
>>> print(res.text)
```

11：检查用户是否有效

语法：

'GET /accounts/{account-id}/active'

```
#!/usr/bin/python
```

```
#_*_ coding:utf-8 _*_
```

```
import requests
```

```
from requests.auth import HTTPBasicAuth
```

```
from requests.auth import HTTPDigestAuth
```

```
import json
```

```
auth=HTTPBasicAuth("gaoyuxia","d2Tz5eu2D9bcsdSpywD/RAe4nFBCUrx7uJhPY/cjWQ")
```

```
data=json.dumps({"username":"gaoyuxia"})
```

```
header={'content-type': 'application/json;charset=UTF-8'}
```

```
response=requests.get("http://192.168.56.101:8081/a/accounts/2/active",auth=auth)
```

```
print(response.text)
```

12：列出帐户电子邮件

语法：

'GET /accounts/{account-id}/emails'

```
>>> res=requests.get("http://192.168.56.101:8081/a/accounts/1/emails",auth=auth)
>>> print(res.text)
{'email': 'gerrit@byd.com',
 'preferred': true}
```

得到电子邮件

13: 创建帐户电子邮件 (试验未成功)

语法:

'PUT /accounts/{account-id}/emails/{email-id}'

```
>>>
```

```
res=requests.put("http://192.168.56.101:8081/a/accounts/self/emails/yaoyuanchun@wind-mobi.com",auth=auth)
```

```
>>> print(res.text)
```

14: 删除用户的电子邮件 (可删除)

语法:

'DELETE /accounts/{account-id}/emails/{email-id}'

```
>>>
```

```
res=requests.delete("http://192.168.56.101:8081/a/accounts/self/emails/gaoyuxia@wind-mobi.com",auth=auth)
```

```
>>> print(res.text)
```

15: 列出用户的 SSH 密钥

语法:

'GET /accounts/{account-id}/sshkeys'

```
>>> import requests
>>> from requests.auth import HTTPBasicAuth
>>> from requests.auth import HTTPDigestAuth
>>> import json
>>> auth=HTTPBasicAuth('gaoyuxia', 'ogHrKugEJskD1Dq$thLGtatGn3IQqaEE4MhwWijCFg')
>>> res=requests.get("http://192.168.56.101:8081/a/accounts/self/sshkeys",auth=auth)
>>> print(res.text)
{'seq': 1,
 'ssh_public_key': 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDQgRr4b1YVV3BIQjKZEYaDKGT/CH2kr24LInARtv6kVXMB6zhvJ4dRMs9cQq426XRG9vX7c72jSRwQX
RImbSx2Yq137zWkD8Ah4fU9Tg8cgy3G2ag6/OSKXI4B7WJgUddQ2hOfqj+o/JtcWac2ib2aEVS2or1j4p2UB84AAAh/KSn1jUjpdmtmBPWWJTaYpDRNW8KlrJ7S91xw207g3/fBcthe
mKvU0Gn3g9x6PSPAFthqB/Qr/H2mtP+uqb22BgFXn37WFkxh6O2clJYaTxchONGTbuIRq3/HgdjwsvxF4ntB3GLdHnsuxhkeL3CCUBow4SwFUX4WQ2cmKQgkI1l gaoyuxia@gerrit-
VirtualBox",
 'encoded_key': "AAAAAB3NzaC1yc2EAAAADAQABAAQDQgRr4b1YVV3BIQjKZEYaDKGT/CH2kr24LInARtv6kVXMB6zhvJ4dRMs9cQq426XRG9vX7c72jSRwQXRI
mbSx2Yq137zWkD8Ah4fU9Tg8cgy3G2ag6/OSKXI4B7WJgUddQ2hOfqj+o/JtcWac2ib2aEVS2or1j4p2UB84AAAh/KSn1jUjpdmtmBPWWJTaYpDRNW8KlrJ7S91xw207g3/fBcthemKvU0Gn3g9x
6PSPAFthqB/Qr/H2mtP+uqb22BgFXn37WFkxh6O2clJYaTxchONGTbuIRq3/HgdjwsvxF4ntB3GLdHnsuxhkeL3CCUBow4SwFUX4WQ2cmKQgkI1l",
 'algorithm': "ssh-rsa",
 'comment': "gaoyuxia@gerrit-VirtualBox",
 'valid': true}
```

16: 列出帐户的功能

语法:

'GET /accounts/{account-id}/capabilities'

```
>>> res=requests.get("http://192.168.56.101:8081/a/accounts/self/capabilities",auth=auth)
>>> print(res.text)
}}'
{
  "administrateServer": true,
  "queryLimit": {
    "min": 0,
    "max": 500
  },
  "createAccount": true,
  "createGroup": true,
  "createProject": true,
  "emailReviewers": true,
  "flushCaches": true,
  "killTask": true,
  "maintainServer": true,
  "modifyAccount": true,
  "runGC": true,
  "streamEvents": true,
  "viewAllAccounts": true,
  "viewCaches": true,
  "viewConnections": true,
  "viewPlugins": true,
  "viewQueue": true
}
```

17: 获取用户指定的功能:

语法:

'GET /accounts/{account-id}/capabilities/{capability-id}'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/accounts/self/capabilities?q=createAccount&q=createGroup",auth=auth)
```

```
>>> print(res.text)
```

只获取 **createAccount** 和 **createGroup** 功能，这里的 **q** 用来传递参数

18: 检查指定用户是否有某项功能:

语法:

```
res=requests.get("http://192.168.56.101:8081/a/accounts/self/capabilities?q=createGroup",auth=auth)
```

```
>>> print(res.text)
```

```
}}'
```

```
{
```

```
  "createGroup": true
```

```
}
```

返回 **true** 有此功能，返回 **false** 没有此功能

19：列出指定用户所在的所有组

语法：

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/accounts/self/groups/",auth=auth)
```

```
>>> print(res.text)
```

CHANGES

1：查询已经 **merged** 的修改

语法：

```
'GET /changes/q=status:merge
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/changes/?q=status:merged",auth=auth)
```

```
>>> print(res.text)
```

2：查询 **open** 状态的修改

语法：

```
'GET /changes/q=status:merge
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/changes/?q=status:open",auth=auth)
```

```
>>> print(res.text)
```

3：查看最新的公开提交

语法：


```
'GET /changes/'
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/changes/?q=status:merged&n=2",auth=auth)
```

```
>>> print(res.text)
```

```
>>> res=requests.get("http://192.168.56.101:8081/a/changes/?q=status:merged&n=2",auth=auth)
>>> print(res.text)
]]]]'
```

merged:表示提交状态

可以通过添加 **o** 参数来获取其他字段，

参数如下：

- **LABELS**：提交所需的每个标签的摘要，以及已授予（或拒绝）该标签的批准者。
- **DETAILED_LABELS**：详细的标签信息，包括所有现有批准的数值，已识别的标签值，当前用户允许设置的值，各州的所有审阅者以及当前用户可能删除的审阅者。
- **CURRENT_REVISION**：描述更改的当前修订版（补丁集），包括提交SHA-1和要从中获取的URL。
- **ALL_REVISIONS**：描述所有修订，而不仅仅是当前修订。
- **DOWNLOAD_COMMANDS**：`commands` 在[FetchInfo](#)中包含字段以进行修订。仅在选择 **CURRENT_REVISION** 或 **ALL_REVISIONS** 选项时有效。
- **CURRENT_COMMIT**：解析并输出提交对象中的所有头字段，包括消息。仅在选择 **CURRENT_REVISION** 或 **ALL_REVISIONS** 选项时有效。
- **ALL_COMMITS**：解析并输出输出修订中的所有标题字段。如果仅 **CURRENT_REVISION** 请求，则仅输出当前修订的提交数据。
- **CURRENT_FILES**：列出由提交和魔术文件修改的文件，包括每个文件插入/删除的基本行数。仅在选择 **CURRENT_REVISION** 或 **ALL_REVISIONS** 选项时有效。
- **ALL_FILES**：列出由提交和魔术文件修改的文件，包括每个文件插入/删除的基本行数。如果仅 **CURRENT_REVISION** 请求，则仅输出该提交的已修改文件。
- **DETAILED_ACCOUNTS**：包括 `_account_id`，`email` 和 `username` 字段引用帐户时。
- **REVIEWER_UPDATES**：包括设置为 [ReviewerUpdateInfo](#) 实体的审阅者的更新。
- **MESSAGES**：包括与更改关联的消息。
- **CURRENT_ACTIONS**：包括有关更改的可用操作及其当前修订的信息。如果呼叫者未经过身份验证，则忽略。
- **CHANGE_ACTIONS**：包括有关更改的可用更改操作的信息。如果呼叫者未经过身份验证，则忽略。
- **REVIEWED**：`reviewed` 如果满足以下所有条件，则包括该字段：
 - 变化是开放的
 - 呼叫者已通过身份验证
 - 调用者对变更的评论最近比变更所有者的上次更新，即此更改将显示在 `reviewby: self` 的结果中。
- **SUBMITTABLE**：`submittable` 在 [ChangeInfo](#) 中包含该字段，该字段可用于判断是否已审核更改并准备提交。
- **WEB_LINKS**：`web_links` 在 [CommitInfo](#) 中包含该字段，因此仅与 **CURRENT_COMMIT** or 结合使用 **ALL_COMMITS**。
- **CHECK**：包括变化的潜在问题。
- **COMMIT_FOOTERS**：在 [RevisionInfo](#) 中包含具有Gerrit特定提交页脚的完整提交消息。
- **PUSH_CERTIFICATES**：在 [RevisionInfo](#) 中包含推送证书信息。如果未在服务器上[启用](#)签名推送，则忽略。

4：获取指定 **change-Id** 的信息

语法：

```
'GET /changes/{change-id}'
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/changes/I12d48c0758cce5092167aa8162968bf8c1b20a12",auth=auth)
```

```
>>> print(res.text)
```

5: 创建合并补丁集以进行更改

语法:

'POST /changes/{change-id}/merge'

```
>>>
```

```
res=requests.post("http://192.168.56.101:8081/a/changes/I12d48c0758c  
ce5092167aa8162968bf8c1b20a12/merge",auth=auth)
```

```
>>> print(res.text)
```

6: 获取 **change-ID** 的更改细节

语法:

'GET /changes/{change-id}/detail'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/changes/I12d48c0758cce  
5092167aa8162968bf8c1b20a12/detail",auth=auth)
```

```
>>> print(res.text)
```

8: **revert change**

语法:

'POST /changes/{change-id}/revert'

```
>>> res=requests.get("http://192.168.56.101:8081/a/changes/Iffa0873f5148231c0a0db29b3d8813e7b58fbaea/assignee",auth=auth)
>>> print(res.text)
{
  "change_id": "Iffa0873f5148231c0a0db29b3d8813e7b58fbaea",
  "assignee": "root"
}

根据change-ID revert掉一笔提交

>>> res=requests.put("http://192.168.56.101:8081/a/changes/Iffa0873f5148231c0a0db29b3d8813e7b58fbaea/revision/84575cf4c8f87a2fd5501ea55769f  
628e1e20093/drafts",auth=auth)
>>> print(res.text)
path must be non-empty

>>>
>>> res=requests.post("http://192.168.56.101:8081/a/changes/Iffa0873f5148231c0a0db29b3d8813e7b58fbaea/revert",auth=auth)
p>>>
>>> print(res.text)
{
  "id": "yuanchuntest-dev-I0107f474eb41d5c6a250b01c9547fc83ec13c0f4",
  "project": "yuanchuntest",
  "branch": "dev",
  "hashtags": [],
  "change_id": "I0107f474eb41d5c6a250b01c9547fc83ec13c0f4",
  "subject": "Revert \"Insert the description of the change.\",",
  "status": "NEW",
  "created": "2019-01-29 02:29:58.000000000",
  "updated": "2019-01-29 02:29:58.000000000",
  "submit_type": "MERGE_IF_NECESSARY",
  "mergeable": true,
  "insertions": 0,
  "deletions": 0,
  "unresolved_comment_count": 0,
  "number": 11,
  "owner": {
    "_account_id": 2
  }
}
```

此执行结果可以在如下图中获取

All	My	Projects	People	Plugins	Documentation	
Open	Merged	Abandoned				status:open
Search for status:open						
Subject	Status	Owner	Project			
Revert "Insert the description of the change."		gaoyuxia	yuanchuntest			

9: abandon change

语法:

'POST /changes/{change-id}/abandon'

>>>

```
res=requests.post("http://192.168.56.101:8081/a/changes/I0107f474eb41d5c6a250b01c9547fc83ec13c0f4/abandon",auth=auth)
```

>>> print(res.text)

执行结果如下图:

All	My	Projects	People	Plugins	Documentation	
Open	Merged	Abandoned				status:abandon
Search for status:abandoned						
Subject	Status	Owner	Project			
Revert "Insert the description of the change."	Abandoned	gaoyuxia				

10: restore change

语法:

'POST /changes/{change-id}/restore'

>>>

```
res=requests.post("http://192.168.56.101:8081/a/changes/I0107f474eb41d5c6a250b01c9547fc83ec13c0f4/restore",auth=auth)
```

>>> print(res.text)

输出结果如下图:

All	My	Projects	People	Plugins	Documentation	
Open	Merged	Abandoned				status:open
Search for status:open						
Subject	Status	Owner	Project			
Revert "Insert the description of the change."		gaoyuxia	yuanchuntest			

11: rebase change

语法:

'POST /changes/{change-id}/rebase'

```
>>>
```

```
res=requests.post("http://192.168.56.101:8081/a/changes/I0107f474eb41d5c6a250b01c9547fc83ec13c0f4/rebase",auth=auth)
```

```
>>> print(res.text)
```

12: get change

语法:

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/changes/I0107f474eb41d5c6a250b01c9547fc83ec13c0f4",auth=auth)
```

```
>>> print(res.text)
```

13: get commit

语法:

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/changes/I0107f474eb41d5c6a250b01c9547fc83ec13c0f4/revisions/37d35774f4c1049abc8e0c34d6bd7fb4d23cc7cf/commit",auth=auth)
```

```
>>> print(res.text)
```

14: get review 信息

语法:

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/changes/I0107f474eb41d5c6a250b01c9547fc83ec13c0f4/revisions/37d35774f4c1049abc8e0c34d6bd7fb4d23cc7cf/review",auth=auth)
```

```
>>> print(res.text)
```

15: 提交修改(根据 **change-ID** 和 **revision-ID**,提交前需要先 **code-review** 和 **verified**)

语法:

```
>>>
res=requests.post("http://192.168.56.101:8081/a/changes/I0107f474eb41d5c6a250b01c9547fc83ec13c0f4/revisions/37d35774f4c1049abc8e0c34d6bd7fb4d23cc7cf/submit",auth=auth)

>>> print(res.text)
```

16: get patch

语法:

```
>>>
res=requests.get("http://192.168.56.101:8081/a/changes/I0107f474eb41d5c6a250b01c9547fc83ec13c0f4/revisions/37d35774f4c1049abc8e0c34d6bd7fb4d23cc7cf/patch",auth=auth)

>>> print(res.text)
```

17: 获取提交类型

语法:

```
>>>
res=requests.get("http://192.168.56.101:8081/a/changes/Ia3f03b1050419e91143e9d4a5f507ed891e63431/revisions/f7948cde5c39730e1ac0be66c8ed86809b6ac04e/submit_type",auth=auth)

>>> print(res.text)
```

```
}}'
```

```
"MERGE_IF_NECESSARY"
```

7: 获取 topic

语法:

'GET /changes/{change-id}/topic'

>>>

```
res=requests.get("http://192.168.56.101:8081/a/changes/I12d48c0758cce5092167aa8162968bf8c1b20a12/topic",auth=auth)
```

```
>>> print(res.text)
```

8:设置 topic

语法:

'PUT /changes/{change-id}/topic'

```
>>> header={'content-type': 'application/json;charset=UTF-8'}
```

```
>>> data=json.dumps({"topic": "Documentation"})
```

>>>

```
res=requests.put("http://192.168.56.101:8081/a/changes/I0107f474eb41d5c6a250b01c9547fc83ec13c0f4/topic",auth=auth,data=data,headers=header)
```

```
>>> print(res.text)
```

9:删除 topic(表示没有看懂)

语法:

'DELETE /changes/{change-id}/topic'

>>>

```
res=requests.delete("http://192.168.56.101:8081/a/changes/I0107f474eb41d5c6a250b01c9547fc83ec13c0f4/topic",auth=auth)
```

```
>>> print(res.text)
```

10: Set Assignee

语法:

'PUT /changes/{change-id}/assignee'


```
>>> data=json.dumps({"assignee": "gaoyuxia"})
>>> res=requests.put("http://192.168.56.101:8081/a/changes/Iffa0873f5148231c0a0db29b3d8813e7b58fbaea/assignee", auth=auth, data=data, headers=headers)
>>> print(res.text)
{'account_id': 2,
 'name': 'gaoyuxia',
 'email': 'gaoyuxia@wind-mobi.com',
 'username': 'gaoyuxia'}
```

11: get Assignee

语法:

'GET /changes/{change-id}/assignee'

```
>>> res=requests.get("http://192.168.56.101:8081/a/changes/Iffa0873f5148231c0a0db29b3d8813e7b58fbaea/assignee", auth=auth)
>>> print(res.text)
{'account_id': 2,
 'name': 'gaoyuxia',
 'email': 'gaoyuxia@wind-mobi.com',
 'username': 'gaoyuxia'}
```

12: delete Assignee

语法:

'DELETE /changes/{change-id}/assignee'

```
>>> res=requests.delete("http://192.168.56.101:8081/a/changes/Iffa0873f5148231c0a0db29b3d8813e7b58fbaea/assignee", auth=auth)
>>> print(res.text)
{'account_id': 2,
 'name': 'gaoyuxia',
 'email': 'gaoyuxia@wind-mobi.com',
 'username': 'gaoyuxia'}
```

13: Changes Submitted Together

语法:

'GET /changes/{change-id}/submitted_together?o=NON_VISIBLE_CHANGES'

```
>>> res=requests.get("http://192.168.56.101:8081/a/changes/Iffa0873f5148231c0a0db29b3d8813e7b58fbaea/submitted_together?o=NON_VISIBLE_CHANGE", auth=auth)
>>> print(res.text)
{'changes': [],
 'non_visible_changes': 0}
```

(没看明白楼上是什么意思)

14: Publish Draft Change(发布一个草稿)

语法:

'POST /changes/{change-id}/publish'

>>>

```
res=requests.post("http://192.168.56.101:8081/a/changes/Idc21039d8474216139c48b19783af240367a89fc/publish", auth=auth)
```

```
>>> print(res.text)
```

15:delete change(只有管理员才可以删除)

语法:

'DELETE /changes/{change-id}'

>>>

```
res=requests.delete("http://192.168.56.101:8081/a/changes/Idc21039d8474216139c48b19783af240367a89fc",auth=auth)
```

>>> print(res.text)

16:发布更改的 edit

语法:

'POST /changes/{change-id}/edit:publish

未发布前的状态显示:

The screenshot shows the Gerrit web interface for a change. The change is in the 'Draft' state. The 'Change Edit' section shows the change ID 'Idc21039d8474216139c48b19783af240367a89fc'. The 'Related Changes' section shows two related changes. The 'Code-Review' section shows the change is 'Verified'. The 'Files' section shows a table of files with their paths and commit messages.

File Path	Commit Message	Comments	Size
src/A.a.txt		1	
src/A.b.txt		1	
src/A.c.txt		0	
src/A.d.txt		0	
		->	0

发布后的状态显示:

The screenshot shows the Gerrit web interface for a change. The change is in the 'Needs Verified' state. The 'Change Edit' section shows the change ID 'Idc21039d8474216139c48b19783af240367a89fc'. The 'Related Changes' section shows two related changes. The 'Code-Review' section shows the change is 'Verified'. The 'Files' section shows a table of files with their paths and commit messages.

File Path	Commit Message	Comments	Size
src/A.a.txt		1	
src/A.b.txt		1	
src/A.c.txt		0	
src/A.d.txt		0	
		-2	0

发布指令:

>>>

```
res=requests.post("http://192.168.56.101:8081/a/changes/I7d7ff591d57d78cb95bb31f50ab19ceca96fdb93/edit:publish",auth=auth)
```

```
>>> print(res.text)
```

注意:如果要传递参数,需要加入 **headers=header** 这个指令

```
data=json.dumps({'name':'gaoyuxia'})
```

```
header={'content-type': 'application/json;charset=UTF-8'}
```

```
r =
```

```
requests.put('http://192.168.56.102:8083/a/accounts/self/name',auth=auth,data=data,headers=header)
```

17:list reviewers

语法:

'GET /changes/{change-id}/reviewers/'

```
>>> res=requests.get("http://192.168.56.101:8081/a/changes/Ic008ble73ef71d2da4a5bd8a13f183de066255cc/reviewers/",auth=auth)
>>> print(res.text)
)))'
{
  (
    {
      "approvals": {
        "Code-Review": " 0",
        "Verified": " 0"
      },
      "account_id": 1,
      "name": "gerrit",
      "email": "gerrit@byd.com",
      "username": "gerrit"
    },
    {
      "approvals": {
        "Code-Review": " 0",
        "Verified": " 0"
      },
      "account_id": 3,
      "name": "yaoyuanchun",
      "username": "yaoyuanchun"
    }
  )
}
```

18:get reviewer

语法:

'POST /changes/{change-id}/reviewers/1'

```
>>> res=requests.get("http://192.168.56.101:8081/a/changes/Ic008ble73ef71d2da4a5bd8a13f183de066255cc/reviewers/1",auth=auth)
>>> print(res.text)
)))'
{
  (
    {
      "approvals": {
        "Code-Review": " 0",
        "Verified": " 0"
      },
      "account_id": 1,
      "name": "gerrit",
      "email": "gerrit@byd.com",
      "username": "gerrit"
    }
  )
}
```

19:add reviewer

语法:

'POST /changes/{change-id}/reviewers/1'

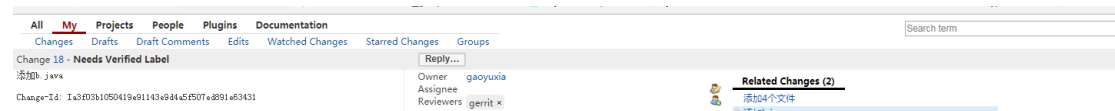
```
>>> user1=json.dumps({"reviewer": "gerrit"})
```

```
>>>
```

```
res=requests.post("http://192.168.56.101:8081/a/changes/Ia3f03b105041",auth=auth,data=user1,headers=header)
```

```
9e91143e9d4a5f507ed891e63431/reviewers",auth=auth,data=user1,headers=header)
```

```
>>> print(res.text)
```



20:delete reviewer

语法:

```
'DELETE /changes/{change-id}/reviewers/{account-id}'
```

```
'POST /changes/{change-id}/reviewers/{account-id}/delete'
```

```
>>>
```

```
res=requests.delete("http://192.168.56.101:8081/a/changes/Ia3f03b1050419e91143e9d4a5f507ed891e63431/reviewers/1",auth=auth,data=user1,headers=header)
```

```
>>>
```

```
res=requests.delete("http://192.168.56.101:8081/a/changes/Ia3f03b1050419e91143e9d4a5f507ed891e63431/reviewers/1",auth=auth)
```

```
>>> print(res.text)
```

21:get commit

语法:

```
'GET /changes/{change-id}/revisions/{revision-id}/commit'
```

```
>>> res=requests.get("http://192.168.56.101:8081/a/changes/Ia3f03b1050419e91143e9d4a5f507ed891e63431/revisions/1/commit",auth=auth)
>>> print(res.text)
)))'
{
  "commit": "f7948cde5c39730e1ac0be66c8ed86809b6ac04e",
  "parents": [
    {
      "commit": "86d566cad6ee6dd27dc5e5d327c0a4b3bad7b984",
      "subject": "Merge \"add a.java file\""
    }
  ],
  "author": {
    "name": "gaoyuxia",
    "email": "gaoyuxia@wind-mobi.com",
    "date": "2019-01-29 03:24:56.000000000",
    "tz": 480
  },
  "committer": {
    "name": "gaoyuxia",
    "email": "gaoyuxia@wind-mobi.com",
    "date": "2019-01-29 03:24:56.000000000",
    "tz": 480
  },
  "subject": "添加b.java",
  "message": "添加b.java\n\nChange-Id: Ia3f03b1050419e91143e9d4a5f507ed891e63431\n"
```

22:get patch

语法:

'GET /changes/{change-id}/revisions/{revision-id}/patch'

>>>

```
res=requests.get("http://192.168.56.101:8081/a/changes/Ic008b1e73ef71d2da4a5bd8a13f183de066255cc/revisions/c772c741fc80fe64f0bb47ebe6149444278ce8ba/patch",auth=auth)
```

```
>>> print(res.text)
```

23:get submit type

语法:

'GET /changes/{change-id}/revisions/{revision-id}/submit_type'

>>>

```
res=requests.get("http://192.168.56.101:8081/a/changes/Ic008b1e73ef71d2da4a5bd8a13f183de066255cc/revisions/c772c741fc80fe64f0bb47ebe6149444278ce8ba/submit_type",auth=auth)
```

```
>>> print(res.text)
```

}}'

"FAST_FORWARD_ONLY"

24:list files (得到此笔提交的修改的文件)

语法:

'GET /changes/{change-id}/revisions/{revision-id}/files/'

>>>

```
res=requests.get("http://192.168.56.101:8081/a/changes/I7d7ff591d57d78cb95bb31f50ab19ceca96fdb93/revisions/c82d594d64e1c259cbe8c88a2cdfde746046c686/files",auth=auth)
```

```
>>> print(res.text)
```

25:get content(得到修改文件的内容, 内容以 **base64** 编码的字符串形式返回)

语法:

'GET /changes/{change-id}/revisions/{revision-id}/files/{file-id}/content'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/changes/Id2a64084f1f811b307c01b5b81cc0dcf6784a352/revisions/2ff57de3e9bc192a6940df8157815e94dd8954fb/files/a.java/content",auth=auth)
```

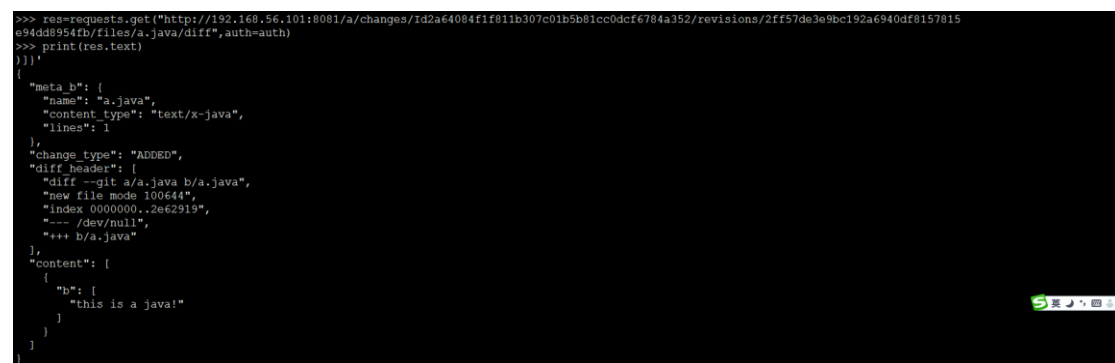
```
>>> print(res.text)
```

dGhpcyBpcyBhIGphdmEhCg==

26: get diff(此命令可以得到修改文件的内容)

语法:

'GET /changes/{change-id}/revisions/{revision-id}/files/{file-id}/diff'



```
>>> res=requests.get("http://192.168.56.101:8081/a/changes/Id2a64084f1f811b307c01b5b81cc0dcf6784a352/revisions/2ff57de3e9bc192a6940df8157815e94dd8954fb/files/a.java/diff",auth=auth)
>>> print(res.text)
)))'
{
  "meta_b": {
    "name": "a.java",
    "content_type": "text/x-java",
    "lines": 1
  },
  "change_type": "ADDED",
  "diff_header": {
    "diff --git a/a.java b/a.java",
    "new file mode 100644",
    "index 0000000..2e62919",
    "--- /dev/null",
    "+++ b/a.java"
  },
  "content": {
    "b": {
      "this is a java!"
    }
  }
}
```

27: get blame(此命令相当于 **git blame**,可以得到文件每一行的详细修改信息)

语法:

'GET /changes/{change-id}/revisions/{revision-id}/files/{file-id}/blame'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/changes/Id2a64084f1f811b307c01b5b81cc0dcf6784a352/revisions/2ff57de3e9bc192a6940df8157815e94dd8954fb/files/a.java/blame",auth=auth)
```

```
>>> print(res.text)
```

28: Cherry Pick Revision

语法:

'POST /changes/{change-id}/revisions/{revision-id}/cherrypick'#传入参数

```
>>> data=json.dumps({"message":"gaoyuxia.txt","destination":"master"})
```

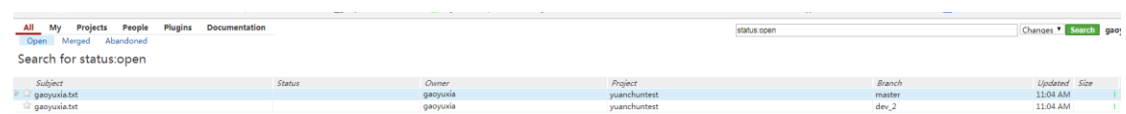
#输入原 PICK 的 **change-id** 和 **revision-id**

```
>>>
```

```
res=requests.post("http://192.168.56.101:8081/a/changes/I36c4f6ec7bb69a161a2782321388790efa69fcf2/revisions/9c1c0886aad10b8d8d6e8e99af74931e155c3305/cherrypick",auth=auth,data=data,headers=header)
```

```
>>> print(res.text)
```

输出结果如下图:



Subject	Status	Owner	Project	Branch	Updated	Size
gaoyuxia.txt	open	gaoyuxia	yuanchuntest	master	11:04 AM	1

29:list revision reviewers

语法:

'GET /changes/{change-id}/revisions/{revision-id}/reviewers/'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/changes/I36c4f6ec7bb69a161a2782321388790efa69fcf2/revisions/9c1c0886aad10b8d8d6e8e99af74931e155c3305/reviewers/",auth=auth)
```

```
>>> print(res.text)
```

Config 篇

1:得到当前 **Gerrit** 的版本:

语法:

'GET / config / server / version'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/config/server/version",auth=auth)
```

```
>>> print(res.text)
```

```
}}'
```

```
"2.14"
```

2: get server info

语法:

'GET / config / server / info'

>>>

```
res=requests.get("http://192.168.56.101:8081/a/config/server/info",auth=auth)
```

```
>>> print(res.text)
```

3: List Tasks

语法:

'GET / config / server / tasks /'

>>>

```
res=requests.get("http://192.168.56.101:8081/a/config/server/tasks",auth=auth)
```

```
>>> print(res.text)
```

Groups 篇

1: 得到所有组(相当于 Ssh 命令的 `ls-groups`)

语法:

'GET / groups /'

```
>>> res=requests.get("http://192.168.56.101:8081/a/groups/",auth=auth)
```

```
>>> print(res.text)
```

```
>>> res=requests.get("http://192.168.56.101:8081/a/groups/?n=2",auth=auth)
>>> print(res.text)
}}}}
{
  "Administrators": {
    "url": "#/admin/groups/uuid-d86148fda7bb873093b3f3e8ed2ff6f73b82a77d",
    "options": {},
    "description": "Gerrit Site Administrators",
    "group_id": 1,
    "owner": "Administrators",
    "owner_id": "d86148fda7bb873093b3f3e8ed2ff6f73b82a77d",
    "id": "d86148fda7bb873093b3f3e8ed2ff6f73b82a77d"
  },
  "Non-Interactive Users": {
    "url": "#/admin/groups/uuid-678be8620ef5e6f4bell131a9bd78a1703b0efa09",
    "options": {},
    "description": "Users who perform batch actions on Gerrit",
    "group_id": 2,
    "owner": "Administrators",
    "owner_id": "d86148fda7bb873093b3f3e8ed2ff6f73b82a77d",
    "id": "678be8620ef5e6f4bell131a9bd78a1703b0efa09"
  }
}
```

2: 创建权限组

语法:

```
'PUT / groups / {group-name} '
```

```
>>>
```

```
res=requests.put("http://192.168.56.101:8081/a/groups/product",auth=auth)
```

```
>>> print(res.text)
```

输出结果如下图:

Groups

Filter

Group Name	Description
Administrators	Gerrit Site Administrators
Non-Interactive Users	Users who perform batch actions on Gerrit
develop	开发小组
product	

3:get group detail

语法:

```
'GET / groups / {group-id} / detail'
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/groups/4/detail",auth=auth)
```

```
>>> print(res.text)
```

4:根据组的 ID 得到组的名称

语法:

```
'GET / groups / {group-id} / name'
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/groups/4/name",auth=auth)
```

```
>>> print(res.text)
```

```
}}'
```

```
"product"
```

5:rename group(对组进行重命名)

语法:

`'PUT / groups / {group-id} / name'`

```
>>> data=json.dumps({"name":"Review_Group"})
```

```
>>>
```

```
res=requests.put("http://192.168.56.101:8081/a/groups/4/name",auth=auth,data=data,headers=header)
```

```
>>> print(res.text)
```

```
}}'
```

```
"product"
```

6:得到组的描述信息(get group desripriton)

语法:

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/groups/3/description",auth=auth)
```

```
>>> print(res.text)
```

```
}}'
```

```
"开发小组"
```

7:设置组的描述信息

语法:

`'GET / groups / {group-id} / description'`

```
>>> data=json.dumps({"description":"此组的成员，拥有读取代码，和下载代码的权限"})
```

```
>>>
```

```
res=requests.put("http://192.168.56.101:8081/a/groups/4/description",auth=auth,data=data,headers=header)
```

```
>>> print(res.text)
```

```
}}'
```

"此组的成员，拥有读取代码，和下载代码的权限"

8: Delete Group Description(删除组的描述信息)

语法:

```
'DELETE / groups / {group-id} / description'
```

```
>>>
```

```
res=requests.delete("http://192.168.56.101:8081/a/groups/3/description",auth=auth)
```

```
>>> print(res.text)
```

9:设置组选项

语法:

```
'PUT / groups / {group-id} / options'
```

```
>>> data=json.dumps({"visible_to_all":"true"})
```

```
>>>
```

```
res=requests.put("http://192.168.56.101:8081/a/groups/4/options",auth=auth,data=data,headers=header)
```

```
>>> print(res.text)
```

```
}}'
```

```
{
```

```
    "visible_to_all": true
```

```
}
```

10:得到组选项(不知道此选项功能)

语法:

```
'GET / groups / {group-id} / options'
```

```
>>>
res=requests.get("http://192.168.56.101:8081/a/groups/4/options",auth=
auth)

>>> print(res.text)

}}'

{

    "visible_to_all": true

}
```

11:列出组的所有成员

语法:

'GET / groups / {group-id} / members /'

```
>>>
res=requests.get("http://192.168.56.101:8081/a/groups/1/members/",aut
h=auth)

>>> print(res.text)
```

12:添加小组成员

语法:

'PUT / groups / {group-id} / members / {account-id} '

'POST /groups/{group-id}/members'

```
>>>
data=json.dumps({"username":"gerrit","email":"gerrit@byd.com","name":"ger
rit"})

#4:group_id,1: _account_id

>>res=requests.put("http://192.168.56.101:8081/a/groups/4/members/1/
",auth=auth,data=data,headers=header)

>>> print(res.text)
```

13: 删除组成员

语法:

'DELETE / groups / {group-id} / members / {account-id} '

'POST / groups / {group-id} /members.delete'

方法一:

```
>>>
```

```
res=requests.delete("http://192.168.56.101:8081/a/groups/4/members/gerrit",auth=auth)
```

```
>>> print(res.text)
```

方法二:

```
>>> data=json.dumps({"members":["gerrit@byd.com"]})
```

```
>>>
```

```
res=requests.post("http://192.168.56.101:8081/a/groups/4/members.delete",auth=auth,data=data,headers=header)
```

```
>>> print(res.text)
```

13:list include group

语法:

'GET / groups / {group-id} / groups /'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/groups/4/groups",auth=auth)
```

```
>>> print(res.text)
```

plugins

1:得到已经安装的插件列表

语法:

'GET / plugins /'

```
>>> res=requests.get("http://192.168.56.101:8081/a/plugins/",auth=auth)
```

```
>>> print(res.text)
```

2:获取插件状态

语法:

```
'GET / plugins / {plugin-id} / gerrit~status'
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/plugins/reviewnotes/gerrit~status",auth=auth)
```

```
>>> print(res.text)
```

Project

1:列出所有项目:

语法:

```
'GET /projects/'
```

```
>>> res=requests.get("http://192.168.56.101:8081/a/projects/",auth=auth)
```

```
>>> print(res.text)
```

2:get project

语法:

```
'GET /projects/{project-name}'
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest",auth=auth)
```

```
>>> print(res.text)
```

```
}}'
```

```
{
```

```
    "id": "yuanchuntest",
```

```
    "name": "yuanchuntest",
```

```
    "parent": "All-Projects",
```

```
"state": "ACTIVE"
```

3:create project

语法:

```
'PUT /projects/{project-name}'
```

```
>>> data=json.dumps({"description": "This is a demo  
project","submit_type":"FAST_FORWARD_ONLY"})
```

```
>>>
```

```
res=requests.put("http://192.168.56.101:8081/a/projects/MyProject",auth  
=auth,data=data,headers=header)----->(这里的 MyProject 就是要创建  
的项目名称，可以根据自己的需求填写)
```

```
>>> print(res.text)
```

创建完成后可在 **project list** 中看到，如下图：

 yuanchuntest	
 yuxiaTest	This is a demo project

4:get project description

语法:

```
'GET /projects/{project-name}/description'
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuxiaTest/descr  
iption",auth=auth)
```

```
>>> print(res.text)
```

5:set project description

语法:

```
'PUT /projects/{project-name}/description'
```

```
>>> data=json.dumps({"description": "This is a manifest repository"})
```

```
>>>
```

```
res=requests.put("http://192.168.56.101:8081/a/projects/manifest/dscri  
ption",auth=auth,data=data,headers=header)
```

```
>>> print(res.text)
```

输出结果如下：

 manifest	This is a manifest repository
--	-------------------------------

6: delete project description

语法：

```
'DELETE /projects/{project-name}/description'
```

```
>>>
```

```
res=requests.delete("http://192.168.56.101:8081/a/projects/manifest/description",auth=auth)
```

```
>>> print(res.text)
```

7: Get Project Parent

语法：

```
'GET /projects/{project-name}/parent'
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/manifest/parent",auth=auth)
```

```
>>> print(res.text)
```

8:set project parent

语法：

```
'PUT /projects/{project-name}/parent'
```

```
>>> data=json.dumps({"parent": "yuanchuntest"})
```

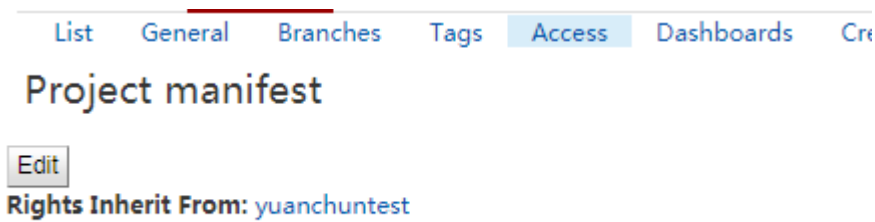
```
>>>
```

```
res=requests.put("http://192.168.56.101:8081/a/projects/manifest/parent",auth=auth,data=data,headers=header)
```

```
>>> print(res.text)
```

```
}}'
```

```
"yuanchuntest"
```

9:”get HEAD(检索项目的分支名称 HEAD)

语法:

'GET /projects/{project-name}/HEAD'

>>>

```
res=requests.get("http://192.168.56.101:8081/a/projects/manifest/HEAD",auth=auth)
```

```
>>> print(res.text)
```

10:set HEAD

语法:

'PUT /projects/{project-name}/HEAD'

>>>

```
res=requests.get("http://192.168.56.101:8081/a/projects/manifest/HEAD",auth=auth)
```

```
>>> print(res.text)
```

11: 获取存储库统计信息

语法:

'GET /projects/{project-name}/statistics.git'

>>>

```
res=requests.get("http://192.168.56.101:8081/a/projects/manifest/statistics.git",auth=auth)
```

```
>>> print(res.text)
```

12:get config

语法:

'GET /projects/{project-name}/config'

```
>>> auth=HTTPBasicAuth('gaoyuxia',  
'+F9MWakj02djuNvPZKdBhuBcY46p50xTJAAeedPCDQ')  
  
>>>  
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/c  
onfig",auth=auth)  
  
>>> print(res.text)
```

13:设置配置

语法:

'PUT /projects/{project-name}/config'

```
>>> data=json.dumps({"submit_type":"FAST_FORWARD_ONLY"})  
  
>>>  
res=requests.put("http://192.168.56.101:8081/a/projects/yuanchuntest/c  
onfig",auth=auth,data=data,headers=header)  
  
>>> print(res.text)
```

14:运行 gc

语法:

'POST /projects/{project-name}/gc'

```
>>>  
res=requests.post("http://192.168.56.101:8081/a/projects/yuanchuntest/  
gc",auth=auth)  
  
>>> print(res.text)
```

15:列出项目的访问权限

语法:

'GET /projects/{project-name}/access'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/access",auth=auth)
```

```
>>> print(res.text)
```

16:列出项目的所有分支

语法:

'GET /projects/{project-name}/branches/'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/branches/",auth=auth)
```

```
>>> print(res.text)
```

17:指定列出分支的数量

Limit(n): 传递分支数量

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/branches?n=2",auth=auth)
```

```
>>> print(res.text)
```

18:Skip(s)跳过给定数量的分支

s=2: 指定跳过前 2 个分支

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/branches?n=2&s=2",auth=auth)
```

```
>>> print(res.text)
```

19:m 参数（没明白有什么用）

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/branches?m=master",auth=auth)
```

```
>>> print(res.text)
```

20:列出所有以 **dev** 开头的分支

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/branches?r=dev.*",auth=auth)
```

```
>>> print(res.text)
```

21: 得到分支 (可以得到此分支提交的最新的 **commit-id**)

语法:

'GET /projects/{project-name}/branches/{branch-id}'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/branches/master",auth=auth)
```

```
>>> print(res.text)
```

22: 创建分支

语法:

'PUT /projects/{project-name}/branches/{branch-id}'

```
>>>
```

```
res=requests.put("http://192.168.56.101:8081/a/projects/yuanchuntest/branches/dev_3",auth=auth)
```

```
>>> print(res.text)
```

23: 删除分支 (根据分支名称删除)

语法:

'POST /projects/{project-name}/branches:delete'

'DELETE /projects/{project-name}/branches/{branch-id}'

```
>>>
```

```
res=requests.delete("http://192.168.56.101:8081/a/projects/yuanchuntest/branches/f7948cde5c39730e1ac0be66c8ed86809b6ac04e",auth=auth)
```

```
>>> print(res.text)
```

24: 获取文件内容（返回的文件内容为 **base64** 加密内容）

语法:

```
'GET  
/projects/{project-name}/branches/{branch-id}/files/{file-id}/content'
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/branches/master/files/a.java/content",auth=auth)
```

```
>>> print(res.text)
```

25: 获取 **reflog**(感觉没什么用)

语法:

```
'GET /projects/{project-name}/branches/{branch-id}/reflog'
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/branches/master/reflog",auth=auth)
```

```
>>> print(res.text)
```

26: 列出子项目(列出继承 **All-Projects** 的所有项目)

语法:

```
'GET /projects/{project-name}/children/'
```

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/All-Projects/children",auth=auth)
```

```
>>> print(res.text)
```

27: 创建 **tag**

语法:

```
'PUT /projects/{project-name}/tags/{tag-id}'
```

```
>>>
data=json.dumps({"revision":"f7948cde5c39730e1ac0be66c8ed86809b6ac0
4e"})

>>>
res=requests.put("http://192.168.56.101:8081/a/projects/yuanchuntest/t
ags/v1.0",auth=auth,data=data,headers=header)

>>> print(res.text)
```

28: 列出 Tag

语法:

'GET /projects/{project-name}/tags/'

```
>>>
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/t
ags",auth=auth)

>>> print(res.text)
```

29: limit n(输出指定数量的 tag)

语法:

GET /projects/work%2Fmy-project/tags?n=2

```
>>>
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/t
ags?n=2",auth=auth)

>>> print(res.text)
```

30: 获取 tag

语法:

'GET /projects/{project-name}/tags/{tag-id}'

```
>>>
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/t
ags/v1.0",auth=auth)

>>> print(res.text)
```

31:删除 tag

语法:

'DELETE /projects/{project-name}/tags/{tag-id}'

'POST /projects/{project-name}/tags:delete'

```
>>> import requests
```

```
>>> from requests.auth import HTTPBasicAuth
```

```
>>> from requests.auth import HTTPDigestAuth
```

```
>>> import json
```

```
>>> auth=HTTPBasicAuth('gaoyuxia',  
'iFo5opoSBYTbEkByIR3Qd5mR5yDCANaAdLlcKf08vA')
```

```
>>> header={'content-type': 'application/json;charset=UTF-8'}
```

```
>>>
```

```
res=requests.delete("http://192.168.56.101:8081/a/projects/yuanchuntes  
t/tags/v1.0",auth=auth)
```

```
>>> print(res.text)
```

32:get commit

语法:

'GET /projects/{project-name}/commits/{commit-id}'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/c  
ommits/6ad6ac4b1e90b93e624af4d4308d57fef29189ec",auth=auth)
```

```
>>> print(res.text)
```

33:get tag in(输出 Tag 所在分支，以及 tag 名称等相关信息)

语法:

'GET /projects/{project-name}/commits/{commit-id}/in'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/commits/6ad6ac4b1e90b93e624af4d4308d57fef29189ec/in",auth=auth)
```

```
>>> print(res.text)
```

34:获取指定文件的内容

语法:

'GET

/projects/{project-name}/commits/{commit-id}/files/{file-id}/content'

```
>>>
```

```
res=requests.get("http://192.168.56.101:8081/a/projects/yuanchuntest/commits/6ad6ac4b1e90b93e624af4d4308d57fef29189ec/files/readme.txt/content",auth=auth)
```

```
>>> print(res.text)
```