

# Requests

## einfache http-Abfragen

- <https://requests.readthedocs.io/en/master/user/quickstart/>
- <https://realpython.com/python-requests/>

```
import requests #scriptname must not be requests.py or http.py!

url="https://api.github.com"

try:
    response = requests.get(url)
    # If the response was successful, no Exception will be raised
    response.raise_for_status()
except requests.HTTPError as e:
    #print(f'HTTP error occurred: {e}')
    status_code = e.response.status_code
    print("Status:", status_code)
except requests.Timeout as e:
    print('The request timed out')
except requests.ConnectionError as e:
    print(f"connection-error {e}")
except Exception as e:
    print(f'Other error occurred: {e}') # Python 3.6
else:
    print('Success!')
    #print(response.text)
    print(response.headers['content-type'])
    json_data=response.json()
    print(json_data['issues_url'])

#get-param
response = requests.get(
    'https://api.github.com/search/repositories',
    params={'q': 'requests+language:python'},
    headers={'Accept': 'application/vnd.github.v3.text-match+json'},
)
json_response = response.json()
repository = json_response['items'][0]
print(f'Repository name: {repository["name"]}')

requests.post('https://httpbin.org/post', data={'key': 'value'})
requests.put('https://httpbin.org/put', data={'key': 'value'})
requests.delete('https://httpbin.org/delete')
requests.head('https://httpbin.org/get')
requests.patch('https://httpbin.org/patch', data={'key': 'value'})
```

```
requests.options('https://httpbin.org/get')
```

<https://stackoverflow.com/a/47007419>

```
print(response.status_code)
```

Asynchrone (parallele) Abfragen:

- <https://medium.com/hackernoon/how-to-run-asynchronous-web-requests-in-parallel-with-python-3-5-without-aihttp-264dc0f8546>
- <https://dev.to/matteo/async-request-with-python-1hpo>

## Authentifikation

```
response = requests.get(url, verify=False,  
auth=HTTPBasicAuth(self.user,self.pass))
```

## parallele Abfragen

```
import requests  
import asyncio  
import concurrent.futures  
  
class multiplerequests():  
  
    def basequery(self,url):  
        print(url)  
        response=requests.get(url)  
        #print(response)  
        ctype=response.headers['content-type']  
        #print(ctype)  
        data=response.text  
        #print(data)  
        ret={"url":url,"type":ctype,"data":data}  
        #print("basequery",ret)  
        return ret #{"type":ctype,"data":data}  
  
    async def multiquery_helper(self,urls):  
        results=[]  
        with concurrent.futures.ThreadPoolExecutor(max_workers=20) as  
executor:  
  
        loop = asyncio.get_event_loop()  
        futures = [  
            loop.run_in_executor(  
                executor,  
                self.basequery,  
                u  
            )  
        ]
```

```
        for u in urls
    ]
    for response in await asyncio.gather(*futures):
        #pass
        results.append(response)

    #print("mq_helper:", results)
    return results
def multiquery(self, urls):
    loop = asyncio.get_event_loop()
    data=loop.run_until_complete(self.multiquery_helper(urls))
    return data
```

apibase.py

## github

```
import requests

token="your_token_here"
headers = {
    'User-Agent': 'Script',
    'Authorization': 'token '+token,
}

def ghrequest(url, session=None):
    # headers = {
    #     'User-Agent': 'Script',
    #     'Authorization': 'token '+token,
    # }
    if not 'headers' in locals():
        headers=None

    if session:
        response = session.get(url)
    else:
        response = requests.get(url, headers=headers)
    return response

def getBranches(repo, session=None):
    url="https://api.github.com/repos/"+repo+"/branches?per_page=100"
    return ghrequest(url, session)

def getForks(repo, session=None):
    url="https://api.github.com/repos/"+repo+"/forks"
    return ghrequest(url, session)

def getClones(repo, session=None):
    url="https://api.github.com/repos/"+repo+"/traffic/clones"
    return ghrequest(url, session)
```

```
session = requests.Session()
session.headers.update(headers)

repos={'frank-w/BPI-R2-4.14', 'frank-w/u-boot'}
for r in repos:
    #res=getBranches(r)
    #print(res.status_code, "\n", res.headers, "\n", res.text)
    #res=getForks(r)
    #print(res.status_code, "\n", res.headers, "\n", res.text)
    res=getClones(r, session)
    print(res.status_code, "\n", res.headers, "\n", res.text)
```

From:

<http://fw-web.de/dokuwiki/> - **FW-WEB Wiki**

Permanent link:

<http://fw-web.de/dokuwiki/doku.php?id=programming:python:requests>

Last update: **2023/06/08 17:06**

