Сообщить об ошибке.

# Модуль http.client в Python





## Клиентская сторона для протоколов HTTP и HTTPS

Модуль http.client определяет классы, реализующие клиентскую сторону протоколов HTTP и HTTPS.

Обычно этот модуль не используется напрямую - его использует модуль urllib.request для обработки URL-адресов, использующих HTTP и HTTPS.

Содержание:



## Пример GET-запроса при помощи модуля http.client.

```
>>> import http.client
>>> conn = http.client.HTTPSConnection("www.python.org")
>>> conn.request("GET", "/")
>>> r1 = conn.getresponse()
>>> print(r1.status, r1.reason)
# 200 OK

# Прочитаем страницу целиком
>>> data1 = r1.read()
# чтение данных по частям
>>> conn.request("GET", "/")
>>> r1 = conn.getresponse()
>>> while chunk := r1.read(200):
...     print(repr(chunk))
# b'<!doctype html>\n<!--[if"...
# ...
```

Пример недопустимого GET-запроса:

```
>>> conn = http.client.HTTPSConnection("docs.python.org")
>>> conn.request("GET", "/parrot.spam")
>>> r2 = conn.getresponse()
>>> print(r2.status, r2.reason)
# 404 Not Found
>>> data2 = r2.read()
>>> conn.close()
```

## Пример HEAD-запроса при помощи модуля http.client.

```
>>> import http.client
>>> conn = http.client.HTTPSConnection("www.python.org")
>>> conn.request("HEAD", "/")
>>> res = conn.getresponse()
```

```
>>> print(res.status, res.reason)
# 200 OK
>>> data = res.read()
>>> print(len(data))
# 0
>>> data == b''
# True
```

## Пример POST-запроса при помощи модуля http.client.

```
>>> import http.client, urllib.parse
>>> params = urllib.parse.urlencode({'@number': 12524, '@type': 'issue', '@action': 'show'})
>>> headers = {"Content-type": "application/x-www-form-urlencoded", "Accept": "text/plain"}
>>> conn = http.client.HTTPConnection("bugs.python.org")
>>> conn.request("POST", "", params, headers)
>>> response = conn.getresponse()
>>> print(response.status, response.reason)
# 302 Found
>>> data = response.read()
>>> data
# b'Redirecting to <a href="http://bugs.python.org/issue12524">http://bugs.python.org/issue12524</a>'
>>> conn.close()
```

## Пример PUT-запроса при помощи модуля http.client.

HTTP-запросы PUT на стороне клиента очень похожи на запросы POST. Разница заключается только на стороне сервера, где HTTP-сервер позволяет создавать ресурсы с помощью запроса PUT. Следует отметить, что пользовательские методы HTTP также обрабатываются в urllib.request.Request путем установки соответствующего атрибута метода. Вот пример сеанса, который показывает, как отправить запрос PUT с помощью http.client:

```
# Этот код создает HTTP-сообщение с содержимым
# BODY в качестве закрытого представления
#  для ресурса http://localhost:8080/file.
>>> import http.client
>>> BODY = "***filecontents***"
>>> conn = http.client.HTTPConnection("localhost", 8080)
>>> conn.request("PUT", "/file.txt", BODY)
>>> response = conn.getresponse()
>>> print(response.status, response.reason)
# 200, OK
```

## Пример базовой аутентификацией с модулем http.client.

```
from http.client import HTTPSConnection
from base64 import b64encode
# Установим соединение https
con = HTTPSConnection("www.google.com")
# Необходимо закодировать `username:password` в base64, а затем
#  декодировать в acsii, т.к. python3 сохраняет их как байтовую строку
user_pass = b64encode(b"username:password").decode("ascii")
headers = {'Authorization': f'Basic {user_pass}'}
# подключаемся
con.request('GET', '/', headers=headers)
# Получаем ответ
res = con.getresponse()
# на этом этапе можно проверить статус и т. д.
# и получить текст страницы
data = res.read()
```

## Пример JSON запроса и чтение JSON ответа с модулем http.client.

```
import http.client, json

# Установим соединение https
connection = http.client.HTTPSConnection('api.github.com')
# необходимо сообщить серверу, в каком виде поступит запрос
```

```
# для этого установим заголовки
headers = {'Content-type': 'application/json'}
# сам JSON запрос
foo = {'text': 'Hello world github/linguist#1 **cool**, and #1!'}
json_foo = json.dumps(foo)
# отправляем запрос на сервер
connection.request('POST', '/markdown', json_foo, headers)
# читаем ответ сервера
response = connection.getresponse()
print(response.read().decode())
```

Содержание раздела: