概述

在http.client模块中，我们主要使用HTTPConnection和HTTPResponse对象来处理整个HTTP交互过程，所以我们接下里主要介绍以下内容:

- HTTPConnection
- HTTPResponse
- 基本示例

HTTPConnection

先看一下HTTPConnection初始化定义函数

> HTTPConnection(host, port=None, [timeout,] source_address=None)
>
> 参数说明
>
> host: 目标服务器IP或url
> port: 目标服务端口(默认值 http: 80, https: 443), 可选参数 timeout： 超时参数， 可选 source_address: 用于标识链接的来源，格式为(host, port)

一个HTTPConnection实例代表着一个http客户端与服务器端的交互事务，在实例化HTTPConnection时，应该传入目标服务主机（IP或url）和端口，如果没有传递端口，则会从主机字符串中解析端口号（例如www.baidu.com:80，则会尝试从该字符串中去解析出80作为交互端口），如果主机字符串中也无端口号，则使用http.client中定义的默认端口(http: 80, https: 443)。如果给出了可选的超时参数(即timeout)，则阻塞操作(如连接尝试)将在许多秒之后超时(如果没有给出，则使用全局缺省超时设置)。

下面看下几种初始化的方式:

> h1 = http.client.HTTPConnection('www.python.org')
>
> h2 = http.client.HTTPConnection('www.python.org:80')
>
> h3 = http.client.HTTPConnection('www.python.org', 80)
>
> h4 = http.client.HTTPConnection('www.python.org', 80, timeout=10)
>
> h5 = http.client.HTTPConnection('www.python.org', 80, timeout=10, source_address=
> ("www.baidu.com", 80) >

下面一起看看HTTPConnection提供出来的API，我们主要对常用的API进行简要说明：

```
# http链接初始化
# 返回一个HTTPConnetion实例对象
HTTPConnection(host, port=None, [timeout, ]source_address=None)

# https链接初始化
# 返回一个HTTPSConnection实例对象
```

```
HTTPSConnection(host, port=None, key_file=None, cert_file=None, [timeout,
]source_address=None, *, context=None, check_hostname=None)

# 发送http请求
HTTPConnection.request(method, url, body=None, headers={}, *,
encode_chunked=False)

# 获取返回值，获取到的是一个HTTPResponse实例对象
HTTPConnection.getresponse()

# 设置调试级别,默认为0， 即不输出调试信息
# 用于链接出现问题时，打开调试信息，方便定位
HTTPConnection.set_debuglevel(level)

# 设置HTTP隧道，即运行通过代理服务器运行连接
# 注意这里的host、port指定是目标服务端的host和端口
# 不是代理的host和端口
# 代理的host和端口，应当在初始化时指定
# import http.client
# 代理服务器: localhost, 端口 8080
# conn = http.client.HTTPSConnection("localhost", 8080)
# www.python.org为我们的目标交互服务
# conn.set_tunnel("www.python.org")
# conn.request("HEAD","/index.html")
HTTPConnection.set_tunnel(host, port=None, headers=None)

# 连接到创建对象时指定的服务器。
# 默认情况下，如果客户端尚未有连接，则在发出请求时自动调用此功能。
HTTPConnection.connect()

# 关闭连接
HTTPConnection.close()

# 向服务器发送RFC 822样式的头。
# 它向服务器发送一条行，包括头、冒号和空格，以及第一个参数。
# 如果给出更多的参数，则会发送延续行，每个行包含一个选项卡和一个参数。
HTTPConnection.putheader(header, argument[, …])

# 向服务器发送一条空行，表示头的尾。
# 可选的messagebody参数可用于传递与该请求相关联的消息体。
HTTPConnection.endheaders(message_body=None, *, encode_chunked=False)

# 将数据发送到服务器。
# 在调用endheader()方法和调用getresponse()之前，
# 应该直接使用该方法。
HTTPConnection.send(data)
```

HTTPResponse

HTTPResponse实例代表着一个从服务器中获得HTTP响应的实例。

它提供对请求头和实体主体的访问。

响应是一个可迭代的对象。

下面我们一起看下其主要的API，并对API进行简要的说明：

```
# 读取并返回响应主体
HTTPResponse.read()

# 将响应主体的下一个len(b)字节读取到缓冲区b中，
# 返回读取的字节数。
HTTPResponse.readinto(b)

# 返回头名的值，如果没有标题匹配名称，则返回默认值。
# 如果有不止一个带有name名称的头，则返回由''所连接的所有值。
# 如果“default”是除单个字符串以外的任何可迭代的，它的元素也会以逗号的方式返回。
HTTPResponse.getheader(name, default=None)

# 返回一个(header, value)元组的列表
HTTPResponse.getheaders()


# 返回服务器使用的HTTP协议版本。
# 10为http/1.0，11为http/1.1。
HTTPResponse.version

# 返回服务器返回的状态码
# 例如200
HTTPResponse.status

# 返回服务器返回的reason描述
# 例如 OK
HTTPResponse.reason

# 返回流的状态
# True表示流已关闭
HTTPResponse.closed
```

示例

下面我们演示下如何使用http.client进行http的GET、POST、HEAD方法

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

__author__ = "苦叶子"

import http.client, urllib.parse

if __name__ == "__main__":
```

```
print("http.client基本示例")

print("http.client GET方法示例")

# 初始化
conn = http.client.HTTPSConnection("www.python.org")

# 发送GET请求
conn.request("GET", "/")
# 获取响应
r1 = conn.getresponse()
# 打印状态码、对应说明、协议版本
print(r1.status, r1.reason, r1.version)
# 读取整个响应内容
data1 = r1.read()
# 下面代码演示如何分chunck读取内容
conn.request("GET", "/")
r1 = conn.getresponse()
while not r1.closed:
    # 每次读取200bytes
    r1_data = r1.read(200)
    if len(r1_data) == 0:
        break
    print(r1_data)

# 请求不存在的url
conn.request("GET", "/parrot.spam")
r2 = conn.getresponse()
print(r2.status, r2.reason)
data2 = r2.read()
# 断开连接
conn.close()

print("http.client HEAD方法")
conn = http.client.HTTPSConnection("www.python.org")
conn.request("HEAD", "/")
res = conn.getresponse()
print(res.status, res.reason)

data = res.read()
print(len(data))
conn.close()

print("http.client POST方法")
# 请注意这里设置http headers的方法
params = urllib.parse.urlencode({'@number': 19999,
    '@type': 'issue',
    '@action': 'show'})

# http头参数
headers = {"Content-type": "application/x-www-form-urlencoded",
           "Accept": "text/plain"}
```

```
conn = http.client.HTTPConnection("bugs.python.org")

# 把请求的data和头参数一起传入
conn.request("POST", "", params, headers)
# 获取响应对象
response = conn.getresponse()
# 打印响应状态
print(response.status, response.reason)
# 读取响应内容
data = response.read()
print(data)
# 关闭连接
conn.close()
```

对于其他方面请参照上述示例进行举一反三。

扫一扫关注微信公众号：