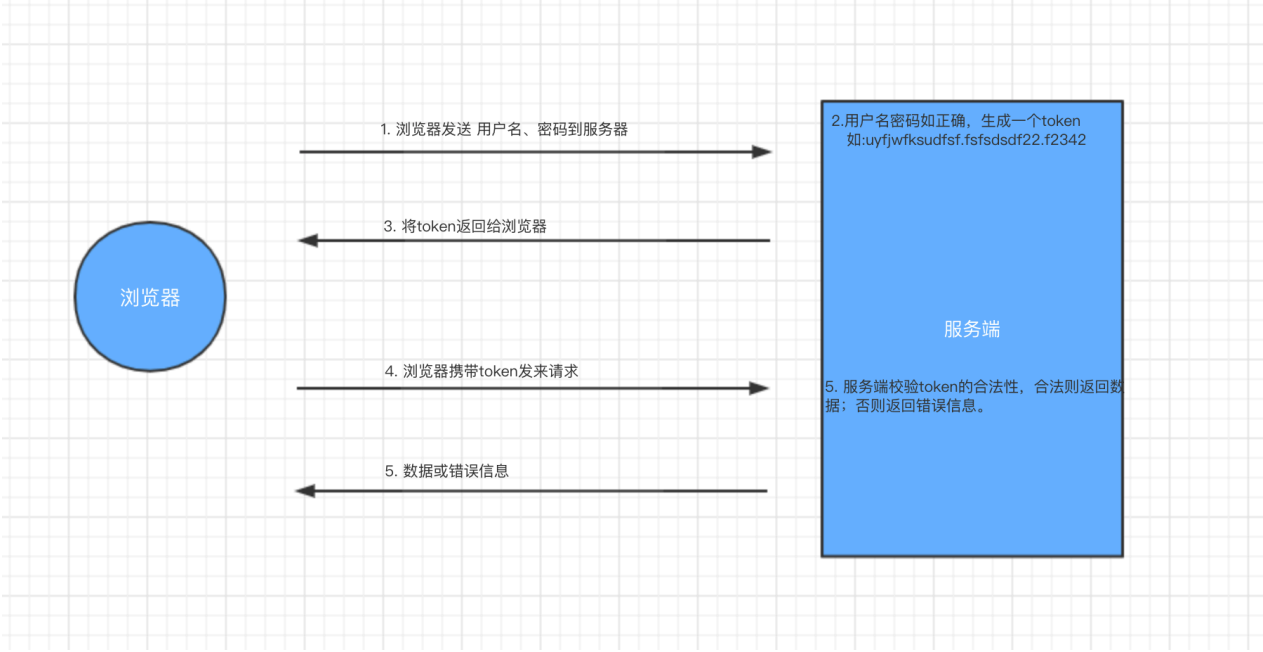
jwt

JSON Web Tokens，是一种开发的行业标准 RFC 7519 ，用于安全的表示双方之间的声明。目前，jwt广泛应用在系统的用户认证方面，特别是现在前后端分离项目。

1. jwt认证流程



在项目开发中，一般会按照上图所示的过程进行认证，即：用户登录成功之后，服务端给用户浏览器返回一个token，以后用户浏览器要携带token再去向服务端发送请求，服务端校验token的合法性，合法则给用户看数据，否则，返回一些错误信息。

传统token方式和jwt在认证方面有什么差异?

- 传统token方式

  > 用户登录成功后，服务端生成一个随机token给用户，并且在服务端(数据库或缓存)中保存一份token，以后用户再来访问时需携带token，服务端接收到token之后，去数据库或缓存中进行校验token的是否超时、是否合法。

- jwt方式

  > 用户登录成功后，服务端通过jwt生成一个随机token给用户（服务端无需保留token），以后用户再来访问时需携带token，服务端接收到token之后，通过jwt对token进行校验是否超时、是否合法。

2. jwt创建token

2.1 原理

jwt的生成token格式如下，即：由 `.` 连接的三段字符串组成。

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

生成规则如下：

- 第一段HEADER部分，固定包含算法和token类型，对此json进行base64url加密，这就是token的第一段。

  ```
  {
    "alg": "HS256",
    "typ": "JWT"
  }
  ```

- 第二段PAYLOAD部分，包含一些数据，对此json进行base64url加密，这就是token的第二段。

  ```
  {
    "sub": "1234567890",
    "name": "John Doe",
    "iat": 1516239022
    ...
  }
  ```

- 第三段SIGNATURE部分，把前两段的base密文通过 `.` 拼接起来，然后对其进行 `HS256` 加密，再然后对 `hs256` 密文进行base64url加密，最终得到token的第三段。

  ```
  base64url(
      HMACSHA256(
        base64UrlEncode(header) + "." + base64UrlEncode(payload),
        your-256-bit-secret （秘钥加盐）
      )
  )
  ```

最后将三段字符串通过 `.` 拼接起来就生成了jwt的token。

注意：base64url加密是先做base64加密，然后再将 `-` 替代 `+` 及 `_` 替代 `/` 。

2.2 代码实现

基于Python的pyjwt模块创建jwt的token。

- 安装

  ```
  pip3 install pyjwt
  ```

- 实现

```
import jwt
import datetime
from jwt import exceptions

SALT = 'iv%x6xo7l7_u9bf_u!9#g#m*)*=ej@bek5)(@u3kh*72+unjv='

def create_token():

    # 构造header
    headers = {
        'typ': 'jwt',
        'alg': 'HS256'
    }
    # 构造payload
    payload = {
        'user_id': 1, # 自定义用户ID
        'username': 'wupeiqi', # 自定义用户名
        'exp': datetime.datetime.utcnow() + datetime.timedelta(minutes=5)
# 超时时间
    }

    result = jwt.encode(payload=payload, key=SALT, algorithm="HS256",
headers=headers).decode('utf-8')
    return result

if __name__ == '__main__':
    token = create_token()
    print(token)
```

3. jwt校验token

一般在认证成功后，把jwt生成的token返回给用户，以后用户再次访问时候需要携带token，此时jwt需要对token进行`超时`及`合法性`校验。

获取token之后，会按照以下步骤进行校验：

- 将token分割成`header_segment`、`payload_segment`、`crypto_segment`三部分

```
jwt_token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6
IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_
adQssw5c"

signing_input, crypto_segment = jwt_token.rsplit(b'.', 1)
header_segment, payload_segment = signing_input.split(b'.', 1)
```

- 对第一部分`header_segment`进行base64url解密，得到`header`
- 对第二部分`payload_segment`进行base64url解密，得到`payload`

- 对第三部分`crypto_segment`进行base64url解密，得到`signature`
- 对第三部分`signature`部分数据进行合法性校验
    - 拼接前两段密文，即：`signing_input`
    - 从第一段明文中获取加密算法，默认：`HS256`
    - 使用 算法+盐 对`signing_input`进行加密，将得到的结果和`signature`密文进行比较。

```
import jwt
import datetime
from jwt import exceptions

def get_payload(token):
    """
    根据token获取payload
    :param token:
    :return:
    """
    try:
        # 从token中获取payload【不校验合法性】
        # unverified_payload = jwt.decode(token, None, False)
        # print(unverified_payload)

        # 从token中获取payload【校验合法性】
        verified_payload = jwt.decode(token, SALT, True)
        return verified_payload
    except exceptions.ExpiredSignatureError:
        print('token已失效')
    except jwt.DecodeError:
        print('token认证失败')
    except jwt.InvalidTokenError:
        print('非法的token')

if __name__ == '__main__':
    token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1NzM1NTU1NzksInVzZXJuYW1lIjoid3VwZWlxaSIsInVzZXJfaWQiOjF9.xj-7qSts6Yg5Ui55-aUOHJS4KSaeLq5weXMui2IIEJU"
    payload = get_payload(token)
```

4. jwt实战

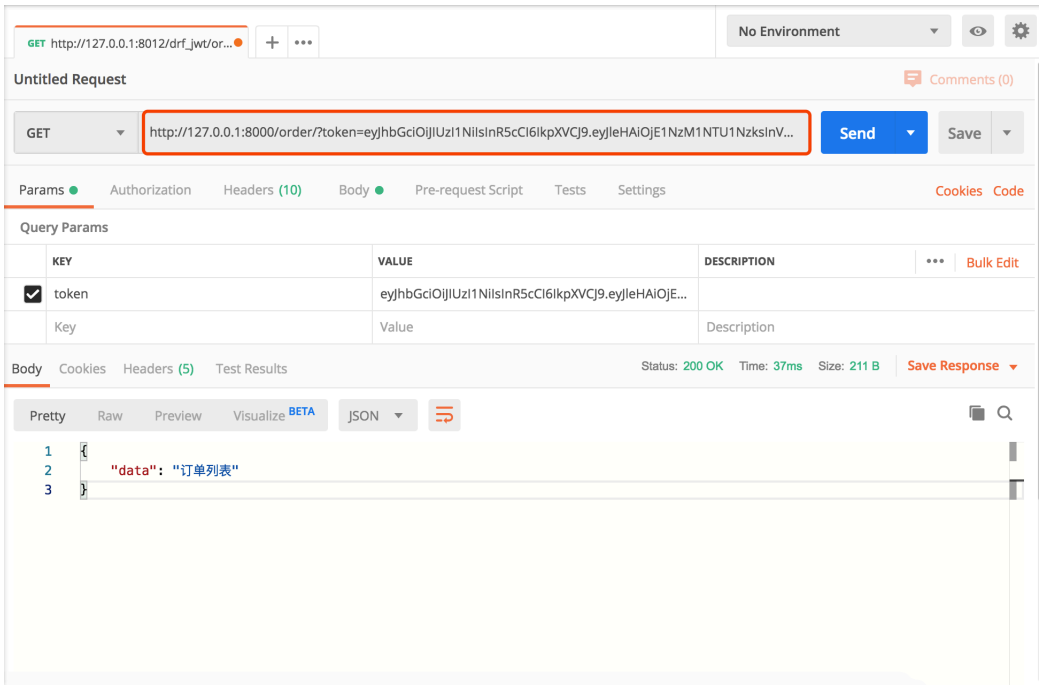
4.1 django 案例

在用户登录成功之后，生成token并返回，用户再次来访问时需携带token。

此示例在django的中间件中对tokne进行校验，内部编写了两个中间件来支持用户通过两种方式传递token。

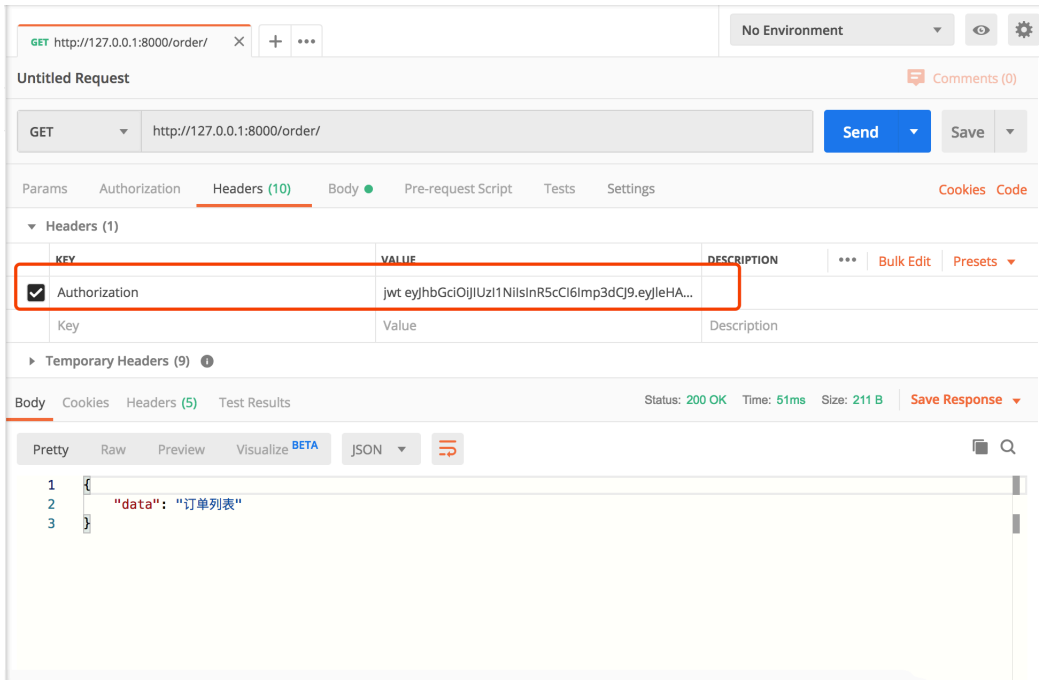
- `url`传参

```
http://www.pythonav.com?token=eyJhbGciOiJIUzI1N...
```



- `Authorization`请求头

```
GET /something/ HTTP/1.1
Host: pythonav.com
Authorization: JWT eyJhbGciOiAiSFMyNTYiLCAidHlwIj
```



源码下载：https://pan.baidu.com/s/1ANibEXYocu6V1JfDUydRHw

4.2 django rest framework 案例

在用户登录成功之后，生成token并返回，用户再次来访问时需携带token。

此示例在drf的认证组件中对token进行校验，内部编写了两个认证组件来支持用户通过两种方式传递token。

- `url`传参
- `Authorization`请求头

源码下载：https://pan.baidu.com/s/14dxnH7YvVNVFwpHEjcBLbg

4.3 flask 案例

在用户登录成功之后，生成token并返回，用户再次来访问时需携带token。

此示例在flask的before_request中对token进行校验，内部编写了两个函数来支持用户通过两种方式传递token。

- `url`传参
- `Authorization`请求头

源码下载：https://pan.baidu.com/s/13w8on_OBBxAd1Pp4KXETaQ

特别提醒：示例源码地址失效，请联系作者 **武沛齐（QQ:424662508）** 进行获取。