Ask Company

리액트와 함께 장고 시작하기 / 장고 DRF

JWT 인증

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

Token 인증과 JWT 인증

DRF의 Token

단순한 랜덤 문자열

- 각 User와 1:1 매칭
- 유효기간이 없습니다.

```
>>> import binascii
>>> import os
>>> binascii.hexlify(os.urandom(20)).decode()
'ec90f85721dc5f75b6eec47d199e3476c301633f'
```

JWT (JSON Web Token)

데이터베이스를 조회하지 않아도, 로직만으로 인증이 가능

포맷 : "헤더.내용.서명"

서버에서 토큰 발급 시에 비밀키로 서명을 하고, 발급시간을 저장

서명은 암호화가 아닙니다. 누구라도 열어볼 수 있기에, 보안성 데이터는 넣지 말고, 최소한의 필요한 정보만 넣기.

claim : 담는 정보의 한 조각. "key/value" 형식

djangorestframework-jwt에서는 Payload에 user_id, username, email 이름의 claim을 사용

위변조가 불가 → 비밀키를 소중히.

장고에서는 settings.SECRET_KEY를 활용하거나, 별도로 JWT_SECRET_KEY 설정을 합니다.

갱신 (Refresh) 메커니즘을 지원

Token 유효기간 내에 갱신하거나, usernams/password을 통해 재인증

이미 발급된 Token을 폐기(Revoke)하는 것은 불가

Token과 JWT

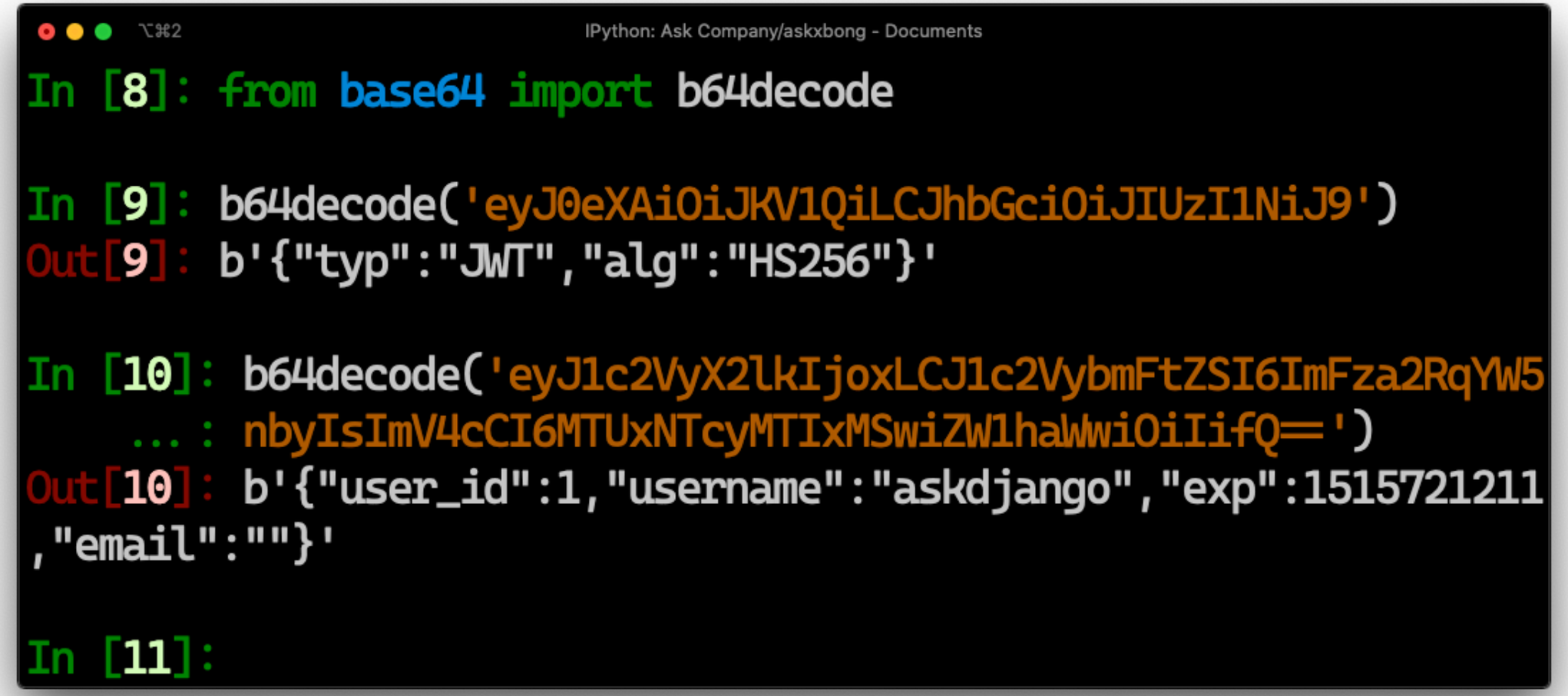
```
8df73dafbde4c669dc37a9ea7620434515b2cc43
```

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxLCJ1c2VybmFtZSI6ImFza2RqYW5nbyIsImV4cCI6MTUxNTcyMTIxMSwiZW1haWwiOiIifQ.Zf_o3S7Q7-cmUzLWlGEQE5s6XoMguf8SLcF-2VdokJQ
```

Header를 base64 인코딩

Payload를 base64 인코딩

Signature = Header/Payload를 조합하고, 비밀키로 서명한 후, base64 인코딩



JWT의 Life cycle

JWT는 만료시간이 있고, Refresh를 지원합니다.

Token은 안전한 장소에 보관하기

일반 Token / JWT 토큰 여부에 상관없습니다.

스마트폰 앱은, 설치된 앱 별로 안전한 저장공간이 제공되지만, 웹브라우저에는 없습니다.

Token은 앱 환경에서만 권장하기도 합니다.

웹 클라이언트 환경에서는 세션 인증이 나은 선택일 수 있습니다. 단 장고/웹클라이언트가 같은 호스트명을 가져야.

통신은 필히 https !!!

https://stormpath.com/blog/where-to-store-your-jwts-cookies-vs-html5-web-storage

djangorestframework-jwt

설치

설치 : pip3 install djangorestframework-jwt

셋업 (1/2)

```
# 프로젝트/settings.py

REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.BasicAuthentication',
        'rest_framework.authentication.SessionAuthentication',
        'rest_framework.authentication.TokenAuthentication',
        'rest_framework_jwt.authentication.JSONWebTokenAuthentication',
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ],
}

JWT_AUTH = {
    'JWT_ALLOW_REFRESH': True,
}
```

셋업 (2/2)

```
# 프로젝트/urls.py
from rest_framework.authtoken.views import obtain_auth_token
from rest_framework_jwt.views import obtain_jwt_token, refresh_jwt_token, verify_jwt_token

urlpatterns = [
    path('api-auth/', include('rest_framework.urls')),
    path('api-token-auth/$', obtain_auth_token),

    path('api-jwt-auth/$', obtain_jwt_token),
    path('api-jwt-auth/refresh/$', refresh_jwt_token),
    path('api-jwt-auth/verify/$', verify_jwt_token),

    path('blog/', include('blog.urls')),
]
```

rest_framework_jwt의 뷰 구현 (1)

```
class JSONWebTokenAPIView(APIView):
    # 중략 ...

    def post(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)

        if serializer.is_valid():
            user = serializer.object.get('user') or request.user
            token = serializer.object.get('token')
            response_data = jwt_response_payload_handler(token, user, request)
            response = Response(response_data)
            if api_settings.JWT_AUTH_COOKIE:
                expiration = (datetime.utcnow() + api_settings.JWT_EXPIRATION_DELTA)
                response.set_cookie(api_settings.JWT_AUTH_COOKIE, token, expires=expiration,
                                    httponly=True)
            return response

        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

rest_framework_jwt의 뷰 구현 (2)

```python
class ObtainJSONWebToken(JSONWebTokenAPIView):
    """username/password를 POST로 받아 인증을 수행하고, JSON Web Token을 응답"""
    serializer_class = JSONWebTokenSerializer


class VerifyJSONWebToken(JSONWebTokenAPIView):
    """전달받은 Token의 진실성(veracity)를 검증"""
    serializer_class = VerifyJSONWebTokenSerializer


class RefreshJSONWebToken(JSONWebTokenAPIView):
    """token을 통한 token 갱신
    'orig_iat' (original issued-at-time) 필드가 필수.
    현재. 시각이 orig_iat 범위 안에서 갱신 허용.
    갱신 후에 orig_iat 값은 그대로 복사.
    """
    serializer_class = RefreshJSONWebTokenSerializer


obtain_jwt_token = ObtainJSONWebToken.as_view()
refresh_jwt_token = RefreshJSONWebToken.as_view()
verify_jwt_token = VerifyJSONWebToken.as_view()
```

```python
# 모두 정상 처리 시에 아래 포맷의 응답
{
    "token": token,
    "user": user
}
```

HTTPie를 통한 JWT 발급

인증에 실패할 경우, "400 Bad Request" 응답

모든 연결에서는 서버와 HTTPS 통신을 권장

발급받은 JWT Token을 jwt.io 서비스를 통해 검증해보세요.

```
쉘> http POST http://서비스주소/api-jwt-auth/ username="유저명" password="암호"
{
    "token": "인증에 성공할 경우, 토큰응답이 옵니다."
}
```

```
쉘> http POST http://서비스주소/api-jwt-auth/verify/ token="토큰"
{
    "token": "검증에 성공할 경우, 검증한 토큰 응답이 옵니다."
}
```

발급받은 JWT Token으로 포스팅 목록 API 요청

DRF Token에서는 인증헤더 시작문자열로 Token을 썼지만, 이제 JWT 사용

매 요청시마다 인증을 수행합니다.

```
쉘> http http://서비스주소/blog/api/post/ "Authorization: JWT 토큰"
```

JWT Token 유효기간이 지났다면?

JWT Token 유효기간 내에 갱신을 해야만 합니다.

유효기간이 지난 Token은 아래와 같이 "401 Unauthorized" 응답

유효기간 내에는 Token 만으로 갱신 가능

유효기간이 지나면 다시 username/password를 통해 인증을 받아야만 합니다.

유효기간

settings.JWT_AUTH의 JWT_EXPIRATION_DELTA 참조 → 디폴트 5분

```
쉘> http http://서비스주소/blog/api/post/ "Authorization: JWT 토큰"`

HTTP/1.0 401 Unauthorized

{
    "detail": "Signature has expired."
}
```

JWT Token 갱신받기

Token 유효기간 내에만 가능

settings.JWT_AUTH의 JWT_ALLOW_REFRESH 설정은 디폴트 False

- True 설정에서만 갱신 지원. False인 경우에는 orig_iat 필드를 찾을 수 없다는 응답.

```
쉘> http POST http://서비스주소/api-jwt-auth/refresh/ token="토큰"

{
    "token": "갱신받은 JWT 토큰"
}
```

djangorestframework-jwt의 주요 settings

```
JWT_AUTH = {
    'JWT_SECRET_KEY': settings.SECRET_KEY,
    'JWT_ALGORITHM': 'HS256',
    'JWT_EXPIRATION_DELTA': datetime.timedelta(seconds=300),
    'JWT_ALLOW_REFRESH': False,
    'JWT_REFRESH_EXPIRATION_DELTA': datetime.timedelta(days=7),
}
```

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

-

Ask Company