Ask Django

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

EP 17. Android 앱 샘플 돌려보기 및 JWT 인증

Token 인증과 JWT 인증

What is JWT ?

- DRF에서 기본지원하는 **Token**은 단순한 랜덤 문자열
  - 각 User와 1:1 매칭
  - 유효기간이 없습니다.

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

JWT는 토큰 자체가 데이터를 가지고 있습니다.

- 또 다른 Token의 한 형태
- 즉, 데이터베이스를 조회하지 않아도 **로직만으로도 인증이 가능합니다.**
- 토큰 포맷: "헤더.내용.서명"
- 서버 측에서 토큰 발급 시에 비밀키로 서명을 하고, 발급시간 claim[1]으로서 exp를 씁니다.
- 서명을 하는 것일 뿐, 암호화는 아닙니다. 누구라도 열어볼 수 있으므로 보안에 중요한 데이터를 넣으면 안됩니다. 필요한 최소한의 정보만 담는 것이 권장됩니다.
- `djangorestframework-jwt`에서는 Payload 항목에 디폴트로 `user_id`, `user_name`, `email` 이름의 claim을 사용합니다.
- 위변조가 안됩니다. (비밀키를 소중히 관리해야겠죠.)
- 갱신(Refresh) 메커니즘을 지원
  - Token 유효기간 내에 필히 갱신하거나, username/password를 통해 재인증을 받아야합니다.
- 이미 발급된 Token 폐기 (Revoke) 불가

---

[1] claim - 담는 정보의 한 조각. "key/value" 형식.

Token은 안전한 장소에 보관해야 합니다.

- 일반 Token / JWT 토큰 여부에 상관없습니다.
- 스마트폰 앱은, 설치된 앱 별로 안전한 저장공간이 제공되지만, 웹브라우저에서는 없습니다.
    - Token는 앱 환경에서만 권장.
    - 웹 클라이언트 환경에서는 보안적인 측면에서 세션 인증이 나은 선택일 수 있습니다.

Token 예시

8df73dafbde4c669dc37a9ea7620434515b2cc43

JWT 예시

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxLCJ1c2VybmFtZSI6ImFza2RqYW5nbyIsImV4cCI6MTUxNTcyMTIxMSwiZW1haWwiOiIifQ.Zf_o3S7Q7-cmUzLWlGEQE5s6XoMguf8SLcF-2VdokJQ

- 헤더 (Header)를 base64 인코딩하여, `eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9`
- 내용 (Payload)을 base64 인코딩하여, `eyJ1c2VyX2lkIjoxLCJ1c2VybmFtZSI6ImFza2RqYW5nbyIsImV4cCI6MTUxNTcyMTIxMSwiZW1haWwiOiIifQ`
- 서명 (Signature) : Header/Payload를 조합하고 비밀키[2]로 서명한 후 base64 인코딩하여, `Zf_o3S7Q7-cmUzLWlGEQE5s6XoMguf8SLcF-2VdokJQ`

---

[2] `settings.JWT_SECRET_KEY`를 통해 커스텀 지정할 수 있으며, 미지정시에는 `settings.SECRET_KEY`가 사용됩니다.

기존 프로젝트에 djangorestframework-jwt 셋업하기

- pip install djangorestframework-jwt

```
# 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 = [
    url(r'^api-auth/', include('rest_framework.urls', namespace='rest_framework')),
    url(r'^api-token-auth/$', obtain_auth_token),

    url(r'^api-jwt-auth/$', obtain_jwt_token),
    url(r'^api-jwt-auth/refresh/$', refresh_jwt_token),
    url(r'^api-jwt-auth/verify/$', verify_jwt_token),

    url(r'^blog/', include('blog.urls', namespace='blog')),
]
```

```
# 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,
}
```

HTTPie를 통한 JWT 발급

```
쉘> http POST http://서비스주소/api-jwt-auth/ username="유저명" password="암호"

{
    "token": "인증에 성공할 경우, 토큰응답이 옵니다."
}
```

- 인증에 실패할 경우, "400 Bad Request" 응답을 받습니다.
- 모든 연결에서는 서버와 HTTP**S** 통신이 권장됩니다.
- 발급받은 JWT Token을 jwt.io 를 통해 검증해보세요.

발급받은 JWT Token 확인

```
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

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

- DRF Token에서는 인증헤더 시작문자열로 **"Token"**을 썼지만, 이젠 **"JWT"** 사용
- 인증이 성공할 경우, 해당 API 응답을 받습니다.
- 이제 매 API 요청마다, 필히 JWT 토큰을 인증헤더에 담아 전송해주세요.

JWT Token 유효기간이 지났을 경우에는

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

HTTP/1.0 401 Unauthorized

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

- 토큰 유효기간[3] 내에 토큰을 갱신받아야합니다. 유효기간이 지났을 경우, 위와 같은 "401 Unauthorized" 응답을 받습니다.
  - 유효기간 내에는 token 만으로 갱신을 받을 수 있지만,
  - 유효기간이 지나면 다시 username/password를 통해 인증을 받아야만 합니다.

---

[3] JWT 토큰 유효기간은 `settings.JWT_AUTH`의 `JWT_EXPIRATION_DELTA` 값을 참조합니다. 디폴트 **5분**입니다.

JWT Token 갱신받기

필히 토큰 유효기간 내에 이뤄줘야합니다.

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

{
    "token": "갱신받은 JWT 토큰"
}
```

- settings.JWT_AUTH의 JWT_ALLOW_REFRESH 설정은 디폴트 False입니다. True 설정을 하셔야만 갱신을 진행할 수 있습니다. False인 경우에 요청하면 orig_iat 필드를 찾을 수 없다는 응답을 받게 됩니다.

djangorestframework-jwt의 주요 settings

```
# 디폴트 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),
}
```

Android 앱 샘플 돌려보기

Android 실행환경 구축

1. Android Studio 다운로드/설치
    - 설치 중에 Android SDK까지 같이 설치하실 수 있습니다.
2. Android Studio 실행 후에 "환경설정" 메뉴
    - Appearance & Behavior -> System Settings -> Android SDK 메뉴
        - API Level 26 의 SDK를 필히 설치
        - app/build.gradle 파일의 targetSdkVersion이 26으로 지정되어있습니다.
3. 소스코드 다운로드 & 안드로이드 프로젝트 열기
4. Android Studio 내 Sync Now 메뉴를 통해, 필요한 라이브러리 자동 설치

초기 실행 시 변경 포인트

소스경로: `app/src/main/java/kr/nomade/apivod/`

- BlogManager.kt 파일 36라인
  - baseUrl 부분을 각자의 주소로 변경

서버에서 JWT 지원 후에 변경 포인트

- BlogManager.kt 파일 24라인
  - "Token" -> "JWT" 변경
- BlogService.kt 파일 9라인
  - "api-token-auth" -> "api-jwt-auth" 변경

실제 앱에서의 권장구현

- 로그인 창을 통해 유저로부터 username/password를 입력받아 JWT Token 발행
  - username/password는 저장하지 않습니다.
- 서버에 https 지원 : username/password/token 의 암호화된 송수신
- JWT 토큰을 Private Preference 영역에 저장
- 유효시간 내에 토큰 갱신
- 유효시간이 지나고 Signature가 만료되면, 추가 username/password 입력을 통해 JWT Token 재발행

본 안드로이드 앱 개발과정은 ...

- 1월 말에 새로운 10분 코딩으로 인사드리겠습니다.
- AWS Lambda 서비스 배포 및 Google Play Store 배포까지 진행해보겠습니다.

Life is short,
use Python3/Django.