Ask Company

리액트와 함께 장고 시작하기 / 장고 DRF

Token 인증 적용하기

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

DRF에서 지원하는 인증

rest_framework.authentication.SessionAuthentication

웹 프론트엔드와 장고가 같은 호스트를 쓴다면, 세션 인증을 사용할 수 있습니다. (nginx 등 활용)

외부 서비스/앱에서 세션인증을 못 쓰죠.

rest_framework.authentication.BasicAuthentication

외부 서비스/앱에서 매번 username/password를 넘기는 것은 보안상 위험하고, 못할 짓.

rest_framework.authentication.TokenAuthentication

초기에 username/password으로 Token을 발급받고,

이 Token을 매 API요청에 담아서 보내어 인증을 처리

http://www.django-rest-framework.org/api-guide/authentication/#tokenauthentication

Token 모델 (1/2)

User모델과 1:1 관계

각 User별 Token은 수동으로 생성해줘야합니다.

Token은 User별로 유일하며, Token만으로 인증을 수행합니다.

https://github.com/encode/django-rest-framework/blob/3.10.1/rest_framework/authtoken/models.py

Token 모델 (2/2)

```
class Token(models.Model):
    key = models.CharField(_("Key"), max_length=40, primary_key=True)
    user = models.OneToOneField(
        settings.AUTH_USER_MODEL, related_name='auth_token',
        on_delete=models.CASCADE, verbose_name=_("User"))
    created = models.DateTimeField(_("Created"), auto_now_add=True)

    class Meta:
        abstract = 'rest_framework.authtoken' not in settings.INSTALLED_APPS

    def save(self, *args, **kwargs):
        if not self.key:
            self.key = self.generate_key()
        return super().save(*args, **kwargs)

    def generate_key(self):
        return binascii.hexlify(os.urandom(20)).decode()

    def __str__(self):
        return self.key
```

self.key에 값이 없을 경우,
랜덤 문자열을 지정합니다.

Token 생성

Token 생성 (1/3)

방법1) ObtainAuthToken 뷰를 통한 획득 및 생성 → URL Pattern 매핑 필요

```python
# rest_framework/authtoken/views.py
class ObtainAuthToken(APIView):
    def post(self, request, *args, **kwargs):
        # ...
        token, created = Token.objects.get_or_create(user=user)
        return Response({'token': token.key})
```

https://github.com/encode/django-rest-framework/blob/3.10.1/rest_framework/authtoken/views.py#L10

Token 생성 (2/3)

방법2) Signal을 통한 자동 생성

```
from django.conf import settings
from django.db.models.signals import post_save
from django.dispatch import receiver
from rest_framework.authtoken.models import Token

@receiver(post_save, sender=settings.AUTH_USER_MODEL)
def create_auth_token(sender, instance=None, created=False, **kwargs):
    if created:
        Token.objects.create(user=instance)
```

Token 생성 (3/3)

방법3) Management 명령을 통한 생성

```
# 본 명령은 생성된 Token을 변경하지 않습니다. 필수는 아님.
python3 manage.py drf_create_token <username>

# 강제로 Token 재생성하기
python3 manage.py drf_create_token -r <username>
```

Token 획득

먼저 obtain_auth_token을 노출하기

```
# 프로젝트 / urls.py
from rest_framework.authtoken.views import obtain_auth_token

urlpatterns += [
    path(r'api-token-auth/', obtain_auth_token),
]
```

HTTPie를 통한 Token 획득

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

HTTP/1.0 200 OK
Allow: POST, OPTIONS
Content-Type: application/json
Date: Fri, 01 Dec 2017 06:49:35 GMT
Server: WSGIServer/0.2 CPython/3.6.1

{
"token": "9cdd705c0a0e5adb8671d22bd6c7a99bbacab227"
}
```

Token과 HTTPie 활용 (bash)

```bash
export HOST="http://localhost:8000"
export TOKEN="9cdd705c0a0e5adb8671d22bd6c7a99bbacab227"

# Post List
http GET $HOST/api/post/ "Authorization: Token $TOKEN"

# Post Create
http POST $HOST/api/post/ "Authorization: Token $TOKEN" message="hello"

# Post Create with Photo
http --form POST $HOST/api/post/ "Authorization: Token $TOKEN" message="hello" photo@"f1.jpg"

# Post#16 Detail
http GET $HOST/api/post/16/ "Authorization: Token $TOKEN"

# Post#16 Update
http PATCH $HOST/api/post/16/ "Authorization: Token $TOKEN" message="patched"
http PUT $HOST/api/post/16/ "Authorization: Token $TOKEN" message="updated"

# Post#16 Delete
http DELETE $HOST/api/post/16/ "Authorization: Token $TOKEN"
```

파이썬에서의 처리 (1/3)

```
import requests

HOST = 'http://localhost:8000'
res = requests.post(HOST + '/api-token-auth/', {
    'username': '유저명',   # FIXME: 기입해주세요.
    'password': '암호',    # FIXME: 기입해주세요.
})
res.raise_for_status()

token = res.json()['token']
print(token)
```

이제 인증이 필요한 요청에는 다음 헤더를 붙여주세요.

```
headers = {
  'Authorization': 'Token ' + token, # 필히 띄워쓰기
}
```

파이썬에서의 처리 (2/3)

```
# Post List
res = requests.get(HOST + '/api/post/', headers=headers)
res.raise_for_status()
print(res.json())

# Post Create
data = {'message': 'hello requests'}
res = requests.post(HOST + '/api/post/', data=data, headers=headers)
print(res)
print(res.json())

# Post Create with Photo
files = {'photo': open('f1.jpg', 'rb')}
data = {'message': 'hello requests'}
res = requests.post(HOST + '/api/post/', files=files, data=data, headers=headers)
print(res)
print(res.json())
```

파이썬에서의 처리 (3/3)

```
# Post#16 Detail
res = requests.get(HOST + '/api/post/', headers=headers)
res = requests.get(HOST + '/api/post/16/', headers=headers)
res.raise_for_status()
print(res.json())

# Post#16 Patch
res = requests.patch(HOST + '/api/post/16/', headers=headers, data={'message': 'hello'})
res.raise_for_status()
print(res.json())

# Post#16 Update
res = requests.put(HOST + '/api/post/16/', headers=headers, data={'message': 'hello'})
res.raise_for_status()
print(res.json())

# Post#16 Delete
res = requests.delete(HOST + '/api/post/16/', headers=headers, data={'message': 'hello'})
res.raise_for_status()
print(res.ok)
```

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

-

Ask Company