Ask Company

리액트와 함께 장고 시작하기 / 장고 DRF

Authentication과 Permission

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

인증 (Authentication)

인증

유입되는 요청을 허용/거부하는 것을 결정하는 것이 아니라,

단순히 인증정보로 유저를 식별하는 것입니다.

Authentication : 유저 식별

Permissions : 각 요청에 대한 허용/거부

Throttling : 일정 기간 동안에 허용할 최대 요청 횟수

https://www.django-rest-framework.org/api-guide/authentication/

인증 처리 순서

1. 매 요청 시마다 APIView의 dispatch(request) 호출
2. APIView의 initial(request) 호출
3. APIView의 perform_authentication(request) 호출
4. request의 user 속성 호출 (rest_framework.request.Request 타입)
5. request의 _authenticate() 호출

APIView 클래스

```
395     def initial(self, request, *args, **kwargs):
396         """
397         Runs anything that needs to occur prior to calling the method handler.
398         """
399         self.format_kwarg = self.get_format_suffix(**kwargs)
400
401         # Perform content negotiation and store the accepted info on the request
402         neg = self.perform_content_negotiation(request)
403         request.accepted_renderer, request.accepted_media_type = neg
404
405         # Determine the API version, if versioning is in use.
406         version, scheme = self.determine_version(request, *args, **kwargs)
407         request.version, request.versioning_scheme = version, scheme
408
409         # Ensure that the incoming request is permitted
410         self.perform_authentication(request)
411         self.check_permissions(request)
412         self.check_throttles(request)
```

Request 클래스

```
366     def _authenticate(self):
367         """
368         Attempt to authenticate the request using each authentication instance
369         in turn.
370         """
371         for authenticator in self.authenticators:
372             try:
373                 user_auth_tuple = authenticator.authenticate(self)
374             except exceptions.APIException:
375                 self._not_authenticated()
376                 raise
377
378             if user_auth_tuple is not None:
379                 self._authenticator = authenticator
380                 self.user, self.auth = user_auth_tuple
381                 return
382
383         self._not_authenticated()
```

지원하는 인증 (Authentication)의 종류

SessionAuthentication

- 세션을 통한 인증, APIView에서 디폴트 지정

BasicAuthentication

- Basic 인증 헤더를 통한 인증 (예 → Authorization: Basic YWxsaWV1czE6MTAyOXNoYWtl)
- HTTPie를 통한 요청 : 쉘> http --auth 유저명:암호 --form POST :8000 필드명1:값1 필드명2:값2

TokenAuthentication

base64 인코딩되어 지정

- Token 헤더를 통한 인증 (예 → Authorization: Token 401f7ac837da42b97f613d789819ff93537bee6a)

RemoteUserAuthentication

- User가 다른 서비스에서 관리될 때, Remote 인증
- Remote-User 헤더를 통한 인증 수행

웹브라우저를 통한 API 접근에서 로그인/로그아웃 지원

장고 기본 앱 django.contrib.auth를 통한 지원 → 세션 인증

rest_framework/urls.py

```
from django.conf.urls import url
from django.contrib.auth import views

app_name = 'rest_framework'
urlpatterns = [
    url(r'^login/$', views.LoginView.as_view(template_name='rest_framework/login.html'),
        name='login'),
    url(r'^logout/$', views.LogoutView.as_view(), name='logout'),
]
```

프로젝트 내, urls.py

```
urlpatterns += [
    path('api-auth/', include('rest_framework.urls', namespace='rest_framework')),
]
```

허가와 권한
(Authorizations and Permissions)

인증과 허가

개체(정보/코드 등)에 대한 접근을 허용하기 위해서, 인증/식별만으로는 충분하지 않습니다. 추가로 각 개체에 대한 허가가 필요합니다.

> *Authentication or identification by itself is not usually sufficient to gain access to information or code. For that, the entity requesting access must have authorization.*
>
> — *Apple Developer Documentation*

DRF의 Permission 시스템

현재 요청에 대한 허용/거부를 결정합니다. APIView 단위로 지정할 수 있습니다.

AllowAny (디폴트 전역 설정) : 인증 여부에 상관없이, 뷰 호출 허용

IsAuthenticated : 인증된 요청에 한해서, 뷰 호출 허용

IsAdminUser : Staff 인증 요청에 한해서, 뷰 호출 허용

IsAuthenticatedOrReadOnly : 비인증 요청에게는 읽기 권한만 허용

DjangoModelPermissions : 인증된 요청에 한해 뷰 호출을 허용하고, 추가로 장고의 모델단위 Permissions 체크

DjangoModelPermissionsOrAnonReadOnly

DjangoModelPermissions과 유사하나, 비인증 요청에게는 읽기만 허용

DjangoObjectPermissions

비인증 요청은 거부하고, 인증된 요청은 Object에 대한 권한 체크를 수행

permission_classes 지정

APIView에는 permission_classes 설정

```
from rest_framework.permissions import IsAuthenticated

class ExampleView(APIView):
    permission_classes = [IsAuthenticated]

    def get(self, request, format=None):
        content = { 'status': 'request was permitted' }
        return Response(content)
```

@api_view에는 @permission_classes 장식자 설정

```
from rest_framework.decorators import permission_classes

@api_view(['GET'])
@permission_classes([IsAuthenticated])
def example_view(request, format=None):
    content = { 'status': 'request was permitted' }
    return Response(content)
```

디폴트 전역 설정

```
REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.AllowAny',
    ]
}
```

커스텀 Permission

모든 Permission 클래스는 다음 2가지 함수를 선택적으로 구현

- has_permission(request, view)
  - APIView 접근 시, 체크
  - 거의 모든 Permission 클래스에서 구현하며, 로직에 따라 True/False 반환
- has_object_permission(request, view, obj)
  - APIView의 get_object 함수를 통해 object 획득 시에 체크
  - 브라우저를 통한 API 접근에서 CREATE/UPDATE Form 노출 시에 체크
  - DjangoObjectPermissions에서 구현하며, 로직에 따라 True/False 반환

dry-rest-permissions : 룰 기반 퍼미션 지원
https://github.com/dbkaplan/dry-rest-permissions

DRF의 Permissions 코드 살펴보기 (1/2)

```
SAFE_METHODS = ('GET', 'HEAD', 'OPTIONS')

class AllowAny(BasePermission):
    def has_permission(self, request, view):
        return True

class IsAuthenticated(BasePermission):
    def has_permission(self, request, view):
        return request.user and request.user.is_authenticated

class IsAdminUser(BasePermission):
    def has_permission(self, request, view):
        return request.user and request.user.is_staff
```

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

DRF의 Permissions 코드 살펴보기 (2/2)

```python
class DjangoObjectPermissions(DjangoModelPermissions):
    # ...
    def has_object_permission(self, request, view, obj):
        # authentication checks have already executed via has_permission
        queryset = self._queryset(view)
        model_cls = queryset.model
        user = request.user

        perms = self.get_required_object_permissions(request.method, model_cls)

        if not user.has_perms(perms, obj):
            if request.method in SAFE_METHODS:
                raise Http404

            read_perms = self.get_required_object_permissions('GET', model_cls)
            if not user.has_perms(read_perms, obj):
                raise Http404

            return False

        return True
```

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

포스팅 작성자가 아니라면, 읽기 권한만 부여하기

모델에 author 필드가 있다고 가정

```python
from rest_framework import permissions

class IsAuthorOrReadonly(permissions.BasePermission):
    # 인증된 유저에 한해, 목록조회/포스팅등록을 허용
    def has_permission(self, request, view):
        return request.user.is_authenticated

    # 작성자에 한해, Record에 대한 수정/삭제 허용
    def has_object_permission(self, request, view, obj):
        # 조회 요청(GET, HEAD, OPTIONS) 에 대해서는 인증여부에 상관없이 허용
        if request.method in permissions.SAFE_METHODS:
            return True

        # PUT, DELETE 요청에 대해, 작성자일 경우에만 요청 허용
        return obj.author == request.user
```

포스팅 작성자에게 수정권한만 부여하고, 삭제는 superuser에게만.

```python
from rest_framework import permissions

class IsAuthorUpdateOrReadonly(permissions.BasePermission):
    def has_permission(self, request, view):
        return request.user.is_authenticated

    def has_object_permission(self, request, view, obj):
        if request.method in permissions.SAFE_METHODS:
            return True

        if (request.method == 'DELETE'):
            return request.user.is_superuser  # 또는 request.user.is_staff

        return obj.author == request.user
```

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

-

Ask Company