

`rest_framework_simplejwt.authentication.JWTAuthentication`
如何生成信息

使用 DRF Simple JWT 生成 JWT 令牌指南

`rest_framework_simplejwt.authentication.JWTAuthentication` 是 Django REST Framework 的一个认证类，用于处理基于 JSON Web Token (JWT) 的用户认证。要生成 JWT 令牌，你需要完成以下几个步骤：

1. 安装和配置 Simple JWT

安装包

Bash



```
pip install djangorestframework-simplejwt
```

配置 settings.py

Python



```

# settings.py
INSTALLED_APPS = [
    # ...
    'rest_framework',
    'rest_framework_simplejwt',
]

REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        # ...
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    )
}

# JWT 配置
from datetime import timedelta

SIMPLE_JWT = {
    'ACCESS_TOKEN_LIFETIME': timedelta(minutes=15), # Access Token 有效期
    'REFRESH_TOKEN_LIFETIME': timedelta(days=7), # Refresh Token 有效期
    'ROTATE_REFRESH_TOKENS': False, # 刷新 Refresh Token
    'BLACKLIST_AFTER_ROTATION': True, # 刷新后黑名单旧令牌

    'ALGORITHM': 'HS256', # 加密算法
    'SIGNING_KEY': SECRET_KEY, # 签名密钥

    'AUTH_HEADER_TYPES': ('Bearer',), # 认证头类型
    'USER_ID_FIELD': 'id', # 用户ID字段
    'USER_ID_CLAIM': 'user_id', # 用户ID在JWT中的声明
}

```

2. 创建登录视图生成 JWT 令牌

使用内置视图生成 Token

Python



```

# urls.py
from rest_framework_simplejwt.views import (
    TokenObtainPairView,
    TokenRefreshView,
)

urlpatterns = [
    # 获取Token对 (Access + Refresh)
    path('api/token/', TokenObtainPairView.as_view(), name='token_obtain_pair'),

    # 刷新Access Token
    path('api/token/refresh/', TokenRefreshView.as_view(), name='token_refresh'),
]

```

自定义登录视图生成 Token

Python



```
# views.py
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from rest_framework_simplejwt.tokens import RefreshToken
from django.contrib.auth import authenticate

class LoginView(APIView):
    def post(self, request):
        username = request.data.get('username')
        password = request.data.get('password')

        user = authenticate(username=username, password=password)

        if user is None:
            return Response(
                {'error': 'Invalid credentials'},
                status=status.HTTP_401_UNAUTHORIZED
            )

        # 为用户生成 JWT 令牌
        refresh = RefreshToken.for_user(user)

        return Response({
            'access': str(refresh.access_token),
            'refresh': str(refresh),
            'user_id': user.id,
            'username': user.username,
            'expires_in': int(SIMPLE_JWT['ACCESS_TOKEN_LIFETIME'].total_seconds())
        })
```

3. 生成 JWT 令牌的流程

令牌获取流程

1. 客户端发送登录请求:

Json



```
POST /api/token/
{
  "username": "admin",
  "password": "password123"
}
```

2. 服务器响应令牌:

Json



```
{  
  "access": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",  
  "refresh": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."  
}
```

3. 客户端使用令牌访问API:



```
GET /api/protected/  
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

令牌刷新流程

1. 客户端发送刷新请求:

Json



```
POST /api/token/refresh/  
{  
  "refresh": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."  
}
```

2. 服务器响应新令牌:

Json



```
{  
  "access": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."  
}
```

4. 自定义令牌内容

添加自定义声明

Python



```
# serializers.py
from rest_framework_simplejwt.serializers import TokenObtainPairSerializer

class CustomTokenObtainPairSerializer(TokenObtainPairSerializer):
    @classmethod
    def get_token(cls, user):
        token = super().get_token(user)

        # 添加自定义声明
        token['username'] = user.username
        token['email'] = user.email
        token['is_staff'] = user.is_staff
        token['custom_data'] = {
            'department': user.department,
            'role': user.get_role_display()
        }

        return token
```

使用自定义序列化器

Python



```
# views.py
from rest_framework_simplejwt.views import TokenObtainPairView
from .serializers import CustomTokenObtainPairSerializer

class CustomTokenObtainPairView(TokenObtainPairView):
    serializer_class = CustomTokenObtainPairSerializer
```

Python



```
# urls.py
urlpatterns = [
    path('api/token/', CustomTokenObtainPairView.as_view(), name='token_obtain_pair'),
    # ...
]
```

5. 验证和解码 JWT 令牌

手动验证令牌

Python



```

from rest_framework_simplejwt.tokens import AccessToken

def get_user_from_token(token):
    try:
        access_token = AccessToken(token)
        user_id = access_token['user_id']
        # 根据 user_id 获取用户对象
        user = User.objects.get(id=user_id)
        return user
    except Exception as e:
        # 处理令牌无效或过期的情况
        return None

```

查看令牌内容

Python



```

token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoxLC..."
decoded = AccessToken(token)

print(decoded.payload)
# 输出: {
#   'token_type': 'access',
#   'exp': 1742894400,
#   'iat': 1742894100,
#   'jti': 'c6f5d825ac1d4fec8ae9c2b3b4f1b2fa',
#   'user_id': 1,
#   'username': 'admin'
# }

```

6. 保护 API 视图

类视图保护

Python



```

from rest_framework.views import APIView
from rest_framework.permissions import IsAuthenticated

class ProtectedAPIView(APIView):
    permission_classes = [IsAuthenticated]

    def get(self, request):
        # 访问当前用户: request.user
        return Response({"message": f'Hello, {request.user.username}!'})

```

函数视图保护

Python



```
from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import IsAuthenticated

@api_view(['GET'])
@permission_classes([IsAuthenticated])
def protected_function_view(request):
    return Response({"data": "Authenticated access"})
```

7. JWT 令牌结构解析

Access Token 结构

Json



```
{
  "token_type": "access",
  "exp": 1742894400,    // 过期时间戳
  "iat": 1742894100,    // 签发时间戳
  "jti": "c6f5d...",    // 令牌唯一标识
  "user_id": 1,         // 用户ID
  "username": "admin",   // 用户名
  "custom_data": {      // 自定义数据
    "department": "IT",
    "role": "Admin"
  }
}
```

Refresh Token 结构

Json



```
{
  "token_type": "refresh",
  "exp": 1743498900,    // 过期时间戳 (通常比ACCESS长)
  "iat": 1742894100,    // 签发时间戳
  "jti": "a3e4b...",    // 令牌唯一标识
  "user_id": 1          // 用户ID
}
```

8. JWT 生命周期管理

最佳实践建议

1. 设置合理的有效期:

- Access Token: 15分钟-1小时

- Refresh Token: 7-30天

2. 安全存储令牌:

- 前端: 使用 HttpOnly Cookie 存储 Refresh Token

- 移动端: 使用安全存储 (Keychain/Keystore)

3. 令牌轮换:

Python



```
SIMPLE_JWT = {  
    'ROTATE_REFRESH_TOKENS': True, # 刷新时生成新Refresh Token  
    'BLACKLIST_AFTER_ROTATION': True, # 将旧Token加入黑名单  
}
```

4. 实现令牌黑名单:

Bash



```
pip install drf-simplejwt-blacklist
```

Python



```
# settings.py  
INSTALLED_APPS = [  
    # ...  
    'rest_framework_simplejwt.token_blacklist',  
]
```

5. 自定义令牌验证:

Python




```
# authentication.py
from rest_framework_simplejwt.authentication import JWTAuthentication
from rest_framework_simplejwt.exceptions import InvalidToken

class CustomJWTAuthentication(JWTAuthentication):
    def get_validated_token(self, raw_token):
        try:
            return super().get_validated_token(raw_token)
        except InvalidToken as e:
            # 自定义错误处理逻辑
            logger.error(f'Invalid token: {raw_token}')
            raise
```

总结

使用 `rest_framework_simplejwt` 生成和使用 JWT 令牌的完整流程如下：

1. **配置**：安装包并在 `settings.py` 中配置 `SIMPLE_JWT`
2. **生成**：使用 `TokenObtainPairView` 或自定义视图生成令牌对
3. **验证**：`JWTAuthentication` 自动验证请求中的令牌
4. **刷新**：使用 `TokenRefreshView` 刷新过期的 Access Token
5. **自定义**：添加自定义声明到令牌中
6. **保护**：使用权限类保护需要认证的视图
7. **管理**：实现令牌轮换和黑名单管理

通过 JWT 认证，你可以为你的 DRF API 添加安全、无状态的身份验证机制，特别适合构建 RESTful API 和前后端分离的应用架构。