# Django Middleware Explained

## 1. Introduction

Middleware in Django is a framework of hooks into Django’s request/response processing. It is a lightweight plugin system for globally altering Django’s input or output. Each middleware component is responsible for doing some specific function. For example, authentication, session management, or security.

## 2. How Middleware Works

When a request comes in, Django processes it through the middleware stack defined in the MIDDLEWARE setting. Each middleware can:  
- Modify the request before it reaches the view  
- Stop the request and return a response early  
- Modify the response before it is sent back to the client  
  
The order of middleware in the MIDDLEWARE list is very important.

## 3. Example Flow

If there are two middlewares configured in settings:  
  
MIDDLEWARE = [  
 'myapp.middleware.FirstMiddleware',  
 'myapp.middleware.SecondMiddleware'  
]  
  
The flow will be:  
FirstMiddleware (before) → SecondMiddleware (before) → View → SecondMiddleware (after) → FirstMiddleware (after).

## 4. JWT Authentication Middleware Example

import jwt  
from django.http import JsonResponse  
from django.conf import settings  
  
class JWTAuthenticationMiddleware:  
 def \_\_init\_\_(self, get\_response):  
 self.get\_response = get\_response  
  
 def \_\_call\_\_(self, request):  
 auth\_header = request.headers.get("Authorization")  
  
 if not auth\_header or not auth\_header.startswith("Bearer "):  
 return JsonResponse({"error": "Authorization header missing"}, status=401)  
  
 token = auth\_header.split(" ")[1]  
  
 try:  
 payload = jwt.decode(token, settings.SECRET\_KEY, algorithms=["HS256"])  
 request.user = payload.get("user")  
 except jwt.ExpiredSignatureError:  
 return JsonResponse({"error": "Token expired"}, status=401)  
 except jwt.InvalidTokenError:  
 return JsonResponse({"error": "Invalid token"}, status=401)  
  
 response = self.get\_response(request)  
 return response

## 5. Code Explanation

1. Import required modules: jwt, JsonResponse, and settings.  
2. Define a class JWTAuthenticationMiddleware.  
3. The \_\_init\_\_ method stores the next callable (get\_response).  
4. The \_\_call\_\_ method runs on every request.  
5. It checks for the Authorization header and validates the JWT.  
6. If the token is invalid or expired, it returns a 401 response.  
7. If valid, it attaches user info to request.user and continues to the view.

## 6. Who Uses Middleware?

- Developers: Write custom middleware for logging, authentication, caching, etc.  
- Security teams: Enforce rules like JWT validation, CSRF protection.  
- Django itself: Comes with built-in middlewares (AuthenticationMiddleware, SessionMiddleware, SecurityMiddleware, etc.).

## 7. Key Points

- Middleware is called on every request.  
- The order in the MIDDLEWARE list matters.  
- Middleware can short-circuit the request and return a response.  
- It is best practice to keep middleware small and focused.