Jwt Token Util

Path: com.opus.backend.config.JwtTokenUtil.class

This class provides utility functions for working with JSON Web Tokens (JWTs). It is marked as a Spring component, so it can be automatically discovered and managed by the Spring framework.

1. **@Component**: Marks the class as a Spring component, enabling Spring's component-scanning mechanism to automatically detect and instantiate it as a bean in the application context.
2. **public class** **JwtTokenUtil implements Serializable**: Declares the **JwtTokenUtil** class, which implements the **Serializable** interface.
3. **@Value("${jwt.secret}")**: Injects the value of the "jwt.secret" property from the application's configuration into the **secret** field.
4. **public String getUsernameFromToken(String token)**: Extracts the username (subject) from the given JWT token.
5. **public Date getIssuedAtDateFromToken(String token)**: Extracts the issued-at date from the given JWT token.
6. **public Date getExpirationDateFromToken(String token)**: Extracts the expiration date from the given JWT token.
7. **public <T> T getClaimFromToken(String token, Function<Claims, T> claimsResolver)**: A generic method to extract any claim from the given JWT token using a custom **claimsResolver** function.
8. **private Claims getAllClaimsFromToken(String token)**: Extracts all claims from the given JWT token.
9. **private Boolean isTokenExpired(String token)**: Checks if the given JWT token is expired.
10. **private Boolean ignoreTokenExpiration(String token)**: Specifies whether the expiration of the given JWT token should be ignored. In the current implementation, it always returns **false**, indicating that expiration should not be ignored.
11. **public String generateToken(UserDetails userDetails, long expiration\_seconds)**: Generates a JWT token for the given **UserDetails** object with the specified expiration time in seconds.
12. **private String doGenerateToken(Map<String, Object> claims, String subject, long expiration\_seconds)**: Generates a JWT token with the given claims, subject (username), and expiration time in seconds.
13. **public Boolean canTokenBeRefreshed(String token)**: Determines if the given JWT token can be refreshed. A token can be refreshed if it is not expired or if its expiration should be ignored.
14. **public Boolean validateToken(String token, UserDetails userDetails)**: Validates the given JWT token for the given **UserDetails** object. The token is considered valid if the username extracted from the token matches the username from the **UserDetails** object and the token is not expired.