Building a Concurrency-Safe Bearer Token Middleware for Swift OpenAPI Clients

Authentication is one of those things that every API client needs, yet implementing it cleanly and safely can be surprisingly tricky—especially in Swift's modern concurrency world. Today, I'm sharing a lightweight middleware solution that handles Bearer token authentication for OpenAPI-generated Swift clients in a thread-safe, composable way.

The Problem

When working with OpenAPI Runtime in Swift, you often need to:

- Inject an `Authorization: Bearer <token>` header into API requests
- Update tokens dynamically (after login, token refresh, etc.)
- Skip authentication for certain endpoints (like login or public routes)
- Ensure thread-safety in concurrent environments

While this sounds straightforward, getting it right with Swift 6's strict concurrency checking requires careful design. You need proper actor isolation, Sendable conformance, and a clean API that doesn't fight the type system.

The Solution: BearerTokenAuthenticationMiddleware

I built `BearerTokenAuthenticationMiddleware`—a minimal, zero-dependency package that solves these problems elegantly. Here's what makes it special:

1. Actor-Isolated Token Storage

The heart of the middleware is a private `TokenStorage` actor that manages the authentication token:

```
private actor TokenStorage {
    var token: String?

    func getToken() -> String? {
        token
    }

    func setToken(_ newToken: String?) {
        token = newToken
    }
}
```

This ensures that token reads and writes are **never concurrent**, eliminating race conditions completely.

2. Clean, Composable API

Setting up the middleware is straightforward:

```
import OpenAPIRuntime
import BearerTokenAuthMiddleware

let authMiddleware = BearerTokenAuthenticationMiddleware(
    initialToken: "my-secret-token"
)

let client = Client(
    serverURL: URL(string: "https://api.example.com")!,
    transport: AsyncHTTPClientTransport(),
    middlewares: [authMiddleware]
)
```

Every request now automatically includes `Authorization: Bearer my-secret-token`.

Wrapping Up

Building authentication middleware might seem like boilerplate, but doing it right—especially with modern Swift concurrency—requires thoughtful design. `BearerTokenAuthenticationMiddleware` provides:

- Thread-safe token management via actors
- Selective authentication with operation-level control
- Dynamic token updates
- Zero dependencies
- Clean composition with other middlewares

Repository: codeberg.org/Mihaela/BearerTokenAuthMiddleware