



Middlewares in next 1 of 5

What are middlewares?

Middlewares are code that runs before / after your request handler.

It's commonly used for things like

1. Analytics
2. Authentication
3. Redirecting the user

```
src > TS index.ts > ...
1
2 import express from "express";
3
4 const app = express();
5
6 let requestCount = 0;
7
8 app.use(
9   function middleware(req, res, next) {
10     requestCount++;
11     next();
12   }
13 );
14
15 app.get("/", (req, res) => {
16   res.send("Hello world");
17 })
18
19 app.get("/requestCount", (req, res) => {
20   res.json({
21     requestCount
22   })
23 })
24
25 app.listen(3000);
```

▼ Code



```
import express from "express";
```

```
const app = express();
```

```
let requestCount = 0;
```

```
(  
  // Middleware in next 1 of 5  
  (req, res, next) {  
    requestCount++;  
    next()  
  }  
);  
  
app.get("/", (req, res) => {  
  res.send("Hello world");  
})  
  
app.get("/requestCount", (req, res) => {  
  res.json({  
    requestCount  
  })  
})  
  
app.listen(3000);
```

src > TS index.ts > ...

```
1  
2 import express from "express";  
3 import jwt from "jsonwebtoken";  
4  
5 const app = express();  
6  
7 //@ts-ignore  
8 async function authMiddleware(req, res, next) {  
9   const token = req.headers.authorization.split(" ")[1];  
10  const decoded = jwt.verify(token, "secret");  
11  if (decoded) {  
12    next();  
13  } else {  
14    res.status(401).send("Unauthorised");  
15  }  
16 }  
17  
18 app.get("/", authMiddleware, (req, res) => {  
19   res.send("You are logged in");  
20 })  
21
```



```
import express from "express";
import jwt from "jsonwebtoken";

const app = express();

//@ts-ignore
async function authMiddleware(req, res, next) {
  const token = req.headers.authorization.split(" ")[1];
  const decoded = jwt.verify(token, "secret");
  if (decoded) {
    next();
  } else {
    res.status(401).send("Unauthorised");
  }
}

app.get("/", authMiddleware, (req, res) => {
  res.send("You are logged in");
})

app.listen(3000);
```

middlewares + Next

Ref – <https://nextjs.org/docs/app/building-your-application/routing/middleware>

Middleware allows you to run code before a request is completed.

1 You can modify the response by

1 rewriting



ii Middlewares in next 1 of 5

3. modifying the request or response headers

4. or responding directly.

Use cases

- Authentication and Authorization: Ensure user identity and check session cookies before granting access to specific pages or API routes.
- Logging and Analytics: Capture and analyze request data for insights before processing by the page or API.

Code

Create a request count middleware

- Create an empty NextJS project



```
→ Projects npx create-next-app
Next.js will install the following packages:
Middleware in next 1 of 5
✓ What is your project named? ... middleware-test
✓ Would you like to use TypeScript? ... No / Yes
✓ Would you like to use ESLint? ... No / Yes
✓ Would you like to use Tailwind CSS? ... No / Yes
✓ Would you like to use `src/` directory? ... No / Yes
✓ Would you like to use App Router? (recommended) ... No / Yes
✓ Would you like to customize the default import alias (@/*)? ... No / Yes
Creating a new Next.js app in /Users/harkiratsingh/Projects/middleware-test.
```

- Create middleware.ts in the root folder

Note: While only one `middleware.ts` file is supported per project, you can still organize your middleware logic modularly. Break out middleware functionalities into separate `.ts` or `.js` files and import them into your main `middleware.ts` file. This allows for cleaner management of route-specific middleware, aggregated in the `middleware.ts` for centralized control. By enforcing a single middleware file, it simplifies configuration, prevents potential conflicts, and optimizes performance by avoiding multiple middleware layers.

- Add code to track the number of requests

```
import { NextResponse } from 'next/server'
import type { NextRequest } from 'next/server'

let requestCount = 0;
export function middleware(request: NextRequest) {
  requestCount++;
  console.log("number of requests is " + requestCount);
  return NextResponse.next()
}
```

- Try visiting the website



Create a request count middleware to track only requests that start with `/api`

- Add a dummy API route (`api/user/route.ts`)

```
import { NextResponse } from "next/server";

export function GET() {
  return NextResponse.json({
    message: "Hi there"
  })
}
```

- Update `middleware.ts`

```
import { NextResponse } from 'next/server'
import type { NextRequest } from 'next/server'

let requestCount = 0;
export function middleware(request: NextRequest) {
  requestCount++;
  console.log("number of requests is " + requestCount);
  return NextResponse.next()
}

// See "Matching Paths" below to learn more
export const config = {
  matcher: '/api/:path*',
}
```

running middlewares

```
import { NextResponse } from 'next/server'
import type { NextRequest } from 'next/server'

export function middleware(request: NextRequest) {
  console.log(request.nextUrl.pathname)
  if (request.nextUrl.pathname.startsWith('/admin')) {
    return NextResponse.redirect(new URL('/signin', request.url))
  }

  if (request.nextUrl.pathname.startsWith('/dashboard')) {
    return NextResponse.next()
  }
}
```

Ref - <https://github.com/code100x/cms/blob/main/src/middleware.ts>