06/04/2022, 14:01 QA Community

COURSEWARE

Professional Skills Agile Fundamentals Jira Git Databases Introduction Java Beginner Maven Testing (Foundation) Java Intermediate HTML CSS Javascript Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express O Intro O Handling Requests O Middleware O Error Handling O Router O Mongoose NodeJS React Express-Testing Networking	
Git Databases Introduction Java Beginner Maven Testing (Foundation) Java Intermediate HTML CSS Javascript Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express O Intro O Handling Requests O Middleware O Error Handling O Router O Mongoose NodeJS React Express-Testing	Professional Skills
Git Databases Introduction Java Beginner Maven Testing (Foundation) Java Intermediate HTML CSS Javascript Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	Agile Fundamentals
Databases Introduction Java Beginner Maven Testing (Foundation) Java Intermediate HTML CSS Javascript Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express O Intro O Handling Requests O Middleware O Error Handling O Router Mongoose NodeJS React Express-Testing	Jira
Java Beginner Maven Testing (Foundation) Java Intermediate HTML CSS Javascript Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	Git
Maven Testing (Foundation) Java Intermediate HTML CSS Javascript Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express O Intro O Handling Requests O Middleware O Error Handling O Router O Mongoose NodeJS React Express-Testing	Databases Introduction
Testing (Foundation) Java Intermediate HTML CSS Javascript Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	Java Beginner
Java Intermediate HTML CSS Javascript Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	Maven
HTML CSS Javascript Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	Testing (Foundation)
CSS Javascript Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	Java Intermediate
Javascript Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	HTML
Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	CSS
Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	Javascript
Sonarqube Advanced Testing (Theory) Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	Spring Boot
Advanced Testing (Theory) Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	Selenium
Cucumber MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	Sonarqube
MongoDB Express Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing	Advanced Testing (Theory)
Express O Intro O Handling Requests O Middleware O Error Handling O Router O Mongoose NodeJS React Express-Testing	Cucumber
 Intro Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing 	MongoDB
 Handling Requests Middleware Error Handling Router Mongoose NodeJS React Express-Testing 	Express
MiddlewareError HandlingRouterMongooseNodeJSReactExpress-Testing	Intro
O Error Handling O Router O Mongoose NodeJS React Express-Testing	 Handling Requests
O Router O Mongoose NodeJS React Express-Testing	Middleware
O Mongoose NodeJS React Express-Testing	O Error Handling
NodeJS React Express-Testing	O Router
React Express-Testing	Mongoose
Express-Testing	NodeJS
	React
Networking	Express-Testing
	Networking

Middleware

Contents

- Overview
 - <u>Using Middleware</u>
 - <u>next</u>
 - Creating middleware
 - Nesting middleware
 - Common middleware
 - cors
 - body-parser
- <u>Tutorial</u>
- Exercises

Overview

Middleware is software that facilitates communication between two different technologies (such as a database and an application).

Middleware often provides extra functionality to the software it is placed between.

Using Middleware

Middleware in Express is just a function - these functions should all have a similar structure with three parameters: req, res and next.

```
app.use((req, res, next) => {
    //do something
    next();
})
```

This syntax should be very familiar to you from request handling.

next

A handler function that calls the next piece of middleware in the chain. Every middleware function should either terminate with **res.send()** or call **next()**, otherwise the request *will hang*.

Creating middleware

Remember that middleware is just a function, creating middleware is as simple as making a function that follows the convention and then use()-ing it.

For example, here is a simple piece of middleware that logs the current date to the console:

```
const logger = (req, res, next) => {
    console.log(new Date());
    next();
}
app.use(logger);
```

Now, whenever a route further down is hit this function will be called first, log the date and then hand over to the next function in the chain.

Nesting middleware

Security

06/04/2022, 14:01 QA Community

Cloud Fundamentals
AWS Foundations
AWS Intermediate
Linux
DevOps
Jenkins Introduction
Jenkins Pipeline
Markdown
IDE Cheatsheet

So far we have applied middleware to *every* route in the chain but sometimes we only want to apply middleware to *some* routes - we can do this by *nesting* the middleware.

```
const logger = (req, res, next) => {
   console.log(new Date());
   next();
}
app.get('/', logger, (req, res) => {
   res.send('Hello, world!');
});
```

In this example the logger middleware is *only* applied to that route and any other routes in the express app are unaffected.

Common middleware

Some pieces of middleware are so common that the Expressis team have their own managed versions (note that they still need to be installed via npm as they are separate *modules*), a few examples of these are:

cors

Used to disable restrictions on Cross Origin Resource Sharing.

```
const cors = require('cors');
app.use(cors());
```

body-parser

Allows for the parsing of request bodies into JS objects. Comes with several body parsers but json is probably the most useful.

```
const bodyParser = require('body-parser');
app.use(bodyParser.json());
```

Tutorial

There is no tutorial for this module.

Exercises

- 1. Go back to your *request_handling*.

 Implement some middleware *before* the routing that logs the current time to the console *but still passes the request on to the next function*.
- ▶ Show solution