**Express Middlewares**

**High Level Goals**

By the end of this lesson, you will be familiar with the following:

* Express middlewares
* Application-level middlewares
* Built-in middlewares
* Router-level middlewares

## What Is an Express Middleware

* البرمجيات الوسيطة هي وظيفة تصل إلى كائنات الطلب والاستجابة والبرمجيات الوسيطة التالية في دورة الطلب والاستجابة للتطبيق ، بمعنى آخر ، البرمجيات الوسيطة هي وظيفة يمكن استدعاؤها قبل وظيفة معالجة الطلب ومن الممكن ربط وظائف وسيطة متعددة معًا ، للتمرير من وظيفة إلى أخرى ، يجب عليك استدعاء المعلمة الثالثة التي يشار إليها عادةً باسم التالي وإلا إذا لم يتم استدعاء الوظيفة التالية ، فسيتم ترك الطلب معلقًا.
* An example of a middleware:
* // this middleware function have access to three parameters:
* // req: the request object
* // res: the response object
* // next: the next function in the request-response cycle
* const ourMiddleware = (req, res, next) => {
* const name = req.body.name;
* if (name === "John") {
* // by invoking the next we will exit the function and go to the next one
* next();
* }
* res.send("Invalid Name");
* };
* // we separate middlewares with commas, the invocation order will be from left to right
* app.post("/create/item" ,ourMiddleware ,(req, res) => {
* // this function will only be invoked if the one before it was invoked and it invoked the next function
* // send a response back
* res.send("test create/items endpoint")
* } ;

## Middleware Types

* هناك أنواع مختلفة من البرامج الوسيطة التي يتم استخدامها اعتمادًا على حالة الاستخدام ، إليك بعض أنواع البرامج الوسيطة:

1. Application-level middleware: مقيد في مثيل التطبيق
2. Built-in middleware: مدمج في البرامج الوسيطة على مستوى التطبيق
3. Router-level middleware: مرتبط بمثيل جهاز التوجيه
4. Error-handling middleware: تحتوي هذه البرامج الوسيطة على توقيع دالة مختلف يتكون من أربعة معاملات بدلاً من ثلاثة
5. Third-party middleware: البرامج الوسيطة لجهات خارجية لتوسيع وظائف جديدة للتعبير

### Application-Level Middleware

* Application-level middleware are bound on the application instance, it can be done by using app.use()

An example on application-level middleware:

* const express = require("express");
* const app = express();
* // في كل مرة يتلقى فيها التطبيق طلبًا ، سيتم تنفيذ هذه الوظيفة الوسيطة
* app.use((req, res, next) => {
* console.log("Hello world");
* next();
* });
* // سيتم تشغيل هذه الطريقة في كل مرة يتلقى فيها التطبيق أي نوع من طلبات HTTP /users
* app.use("/users", (req, res, next) => {
* console.log(req.method);
* next();
* });

### Builtin middleware

* Builtin Middleware are Builtin Application-level middlewares.

An example on builtin middlewares:

const express = require("express");

const app = express();

// express .static() is a middleware that serves static assets such as HTML files, images, etc

// it will read files from a folder called public in the same directory

App .use(express .static("public"));

// express . json() is a middleware that parses incoming requests with JSON payloads

app.use(express.json());

* : express .static() عبارة عن برمجية وسيطة تخدم أصولًا ثابتة مثل ملفات HTML والصور وما إلى ذلك
* Express .json () : هو برنامج وسيط يوزع الطلبات الواردة بحمولات JSON

### Router-Level middleware

* The router object allows us to create an isolated instance of middleware and routes. it is similar to the express application but capable of only performing middleware and routing functions, and to use the router it must be used as a middleware in the application.

An example on router-level middlewares:

const express = require("express");

const app = express();

// insatiate router object

const authRouter = express.Router();

// a middleware function that logs the type of the HTTP method

const methodType = (req, res, next) => {

console.log(req.method);

next();

};

// a middleware function that will be executed for every request made to the router

authRouter.use((req, res) => {

console.log("Test Router");

});

// a middleware that will be executed for every type of HTTP request made to the / endpoint

authRouter.use("/", (req, res) => {

console.log("Another middleware");

});

// a middleware that will be executed for every POST requests to the /login endpoint

authRouter.post("/login", methodType, (req, res) => {

res.send("Login successful");

});

// mount the authentication router as an application-level middleware

// this authRouter will be called on the routes that match /auth, and in order to invoke the login, the

// endpoint must be /auth/login, and if we request that route both of the middlewares will be invoked before the login handler

app.use("/auth", authRouter);

### Error-handling middleware

Error-handling middleware are used as the name applies for error handling the only difference between it and a normal middleware is the amount of parameters.

const express = require("express");

const app = express();

app.get("/", (req, res, next) => {

// create a new error

const err = new Error("Internal server error");

err.status = 500;

// pass it to next

next(err);

});

// since this middleware comes at the end then if the previous route called next then it will be called

app.use((err, req, res, next) => {

// set the status code

res.status(err.status);

// send the response in JSON format

res.json({

error: {

status: err.status,

message: err.message,

},

### Third-Party middleware

* Third-party middleware are used to extend the functionality of express and it is usually installed using npm the used in the application.
* In the example below we will be using a package called morgan npm i morgan.

An example on third-party middlewares:

const express = require("express");

const morgan = require("morgan");

const app = express();

// morgan is used to log information in the console, by default it can long information about the incoming requests

// using morgan as an application-level middleware

app.use(morgan("default"));

app.get("/", (req, res, next) => {

res.send("Testing endpoint");

});