Express.js

Express.js je routing i middleware web radni okvir. Express aplikacija je u suštini samo sljed poziva raznih middleware funkcija. Kako bi se razumio Express, potrebno je shvatiti što su middleware-i. Laičkim riječnikom to bi bile funkcije koje su zadužene za obradu request-a prije nego li se on prosljedi konačnoj funkciji koja vraća response.

Preciznje, middleware-i su funkcije koje imaju pristup **request** i **response** objektu te **next()** middleware funkciji u request/response ciklusu aplikacije (ciklus je gotov kada jedan od middleware-a vrati response).
Više pročitati na linku https://expressjs.com/en/guide/using-middleware.html#middleware.application

Middleware funkcije mogu:

- Izvršiti kod
- Napraviti promjenu na request i response objektu
- Prekinuti request/response ciklus
- Pozvati sljedeću middleware funkciju

Ako trenutna middleware funkcija na završava request/response ciklus (ako ne vraća response), potrebno je pozvati sljedeću middleware funkciju. U protivnom će request ostati „u zraku“. Pozivom funkcije next(), radi se prosljeđivanje requesta na obradu sljedećoj middleweare funkciji.

Express aplikacija može koristiti sljedeće tipove middleware-ova:

Application-level middleware
https://expressjs.com/en/guide/using-middleware.html#middleware.application

Router-level middleware
https://expressjs.com/en/guide/using-middleware.html#middleware.router

Error-handling middleware
https://expressjs.com/en/guide/using-middleware.html#middleware.error-handling

Build-in middleware
https://expressjs.com/en/guide/using-middleware.html#middleware.built-in

Third party middleware
https://expressjs.com/en/guide/using-middleware.html#middleware.third-party

U primjerima sa predavanja su korišteni niže navedeni koncepti:

Application level middleware

Aplikacijski middleware (application level middleware) se veže za instancu aplikacije preko app.use() i app.METHOD() funkcija (METHOD je HTTP metoda request-a koju middleware funkcija handle-a). Middleware se može primjeniti nad svim zahtjevima koji dolaze na aplikaciju (kad specifična putanja nije naznačena) ili samo nad zahtjevima koji su vezani za specifične putanje (kad je specifična putanja naznačena).

Moguće je definirati više ruta (route handler-a) za istu putanju (dio koji se upisuje u adress bar). Koji route handler će preuzeti request i obraditi ga sa svojim middleware-om ili svojih više middleware-a? Kod se izvršava sljedno te će prvi route handler za specifičnu putanju preuzeti request i napraviti obradu. Hoće li tada obrada završiti ili će se preći na idući route handler ovisi i o tome je li zahtjev-odgovor ciklus završio. Ukoliko se zahtjev-odgovor ciklus za konkretnu putanju („adresu“) završi sa prvim route handler-om i njegovom funkcijom/funkcijama za obradu, do idućeg se neće ni doći. Ovdje je potrebno uočiti kako poziv next() metode preusmjerava request na obradu sljedećem middlewareu.

Ukoliko se želi preskočiti slijed middleware funkcija koji obrađuju request, u next() funkciju se ubacuje route parametar (next('route')) kako bi se kontrola prebacila na sljedeću rutu. Napomena **next('route') će raditi ispravno jedino u middleware funkcijama koje su generirane preko app.METHOD() ili router.METHOD().**

Router-level middleware

Radi na istom principu kao i Application level middleware, samo što je vezan za instancu express.Router(). Više o Router-u pročitati na http://expressjs.com/en/5x/api.html#router

Express.js se u poptunosti sastoji od middleware funkcija. Osim što je moguće koristiti express-ove middleware, može se napisati i svoje https://expressjs.com/en/guide/using-middleware.html#middleware.application .

Node.js Moduli

U Node.js-u modul predstavlja jedinicu (jednostavnu ili kompeksnu), koja je organizirana unutar jednog ili više js fileova. Modul se može iznova iskoristiti (reuse) kroz Node.js aplikaciju.

module.exports je poseban objekt, koji je po defaultu uključen u svaki Node.js projekt. Kada se funkcija, objekt ili varijabla žele izložiti kao jedan specifičan modul u Node.js aplikaciji, koristi se module.exports objekt. Kada se specifičan modul želi iskoristiti u željenom file-u se to radi preko funkcije require(), čiji je argument putanja do željenog modula.

Postoje tri osnovne vrste modula u Node.js-u:

1. Core moduli - dolaze sa node.js-om i potrebno ih je uključiti sa ključnom riječi require
2. Lokalni moduli – moduli koji se kreiraju lokalno unutar projekta na kojem se radi
3. Third party moduli – moduli koji se skinu sa repozitorija

Više pročitati na https://www.tutorialsteacher.com/nodejs/nodejs-module-exports

U Primjeru1 (primjer sa predavanje) u datoteci *app.js* korišteni niže navedeni koncepti:

http://expressjs.com/en/api.html#req.query

http://expressjs.com/en/api.html#req.baseUrl

U primjeru `movieRouter.route('/movies')`

Prikazano je dohvaćanje svih podataka iz baze kao i dohvaćanje podataka prema navedenom kriteriju. Korištena je metoda find (iz paketa mongoose), koja kao prvi argument može primiti kriterij fitriranja. Ako ne primi nijedan kriterij, vratit će se svi podaci iz baze objekta Movie.

U primjeru `movieRouter.route('/movies/:movieId')`
Korištena je metoda findById(iz paketa mongoose), koja kao prvi argument prima id za pretragu.

U Primjer1 (primjeri sa predavanja) u datoteci *models/MovieModel* iz paketa mongoose, korišteni su niže navedeni koncepti:

https://mongoosejs.com/docs/api/mongoose.html#mongoose_Mongoose

https://mongoosejs.com/docs/api/schema.html

https://mongoosejs.com/docs/api/model.html

U Primjeru 2 je dodana metoda post. A u Primjeru 3 put, patch (za editiranje) i delete. Time je zaokružena izrada primitivnog REST-like Web API-a. U primjerima 2 i 3 su rute izdvojene u odvojeni file. Time se demonstrira organiziranje ruta u zasebne cjeline, što je korisno kod web mjesta sa velikom brojem ruta. U našem primjeru jedna je osnovna ruta /api, a na nju se dodaju /movies/... rute. Mogla je primjeric u app.js postojati i ruta /products, pa bi se na nju ulančao drugi skup ruta (organiziranje koda vezanog uz rute na više smislenih cjelina).