JavaScript Node.js

Web Servers 2

Url: https://github.com/obiSerra/laboratorio-nodejs-2022

Breve riepilogo

Expressjs e’ un node module molto comodo per sviluppare web servers in Node

Utilizza un’architettura basata su middlewares

E’ possibile creare un “boilerplate” per un’applicazione Expressjs con *$ npx express-generator*

Handlebars e’ un template engine che possiamo usare per arricchire l’HTML con del JavaScript

Handlebars continue

Possiamo gestire dei partial template “dinamici” registrando la cartella

```
const hbs = require('hbs')
hbs.registerPartials(path.join(__dirname,'views','partials'))
```

passando una funzione che ritorna il nome del nostro partial all’interno di una variabile

```
dynamicPartial: () => "user"
```

e infine richiamando la variabile nel template

```
{{> (dynamicPartial)}}
```

(ref lezione5/project)

(ref helpers https://handlebarsjs.com/guide/builtin-helpers.html)

Dove eravamo rimasti

Aggiungiamo 2 pagine:

- /users/ che visualizza una lista di utenti (da https://jsonplaceholder.typicode.com/users)
- /users/:id che restituisce lo user con quell'id (dalla finta lista di utenti)

entrambe le pagine utilizzeranno Handlebars per renderizzare i risultati sfruttando diversi templates

Database

- Installiamo MongoDb (https://docs.mongodb.com/manual/administration/install-community/) e NoSqlBooster (https://www.nosqlbooster.com/downloads/)
- MongoDb e' NoSql Database
- Semplice da usare, gestisce dati in formato JSON
- Non essendo un DB relazionale non e' molto performante per alcune tipologie di query
- Dopo aver installato Mongodb possiamo creare una nuova directory per il db e lanciare *$ mongod --dbpath db*

MongoDb e Node

- Per prima cosa dobbiamo installare i driver per Node *$ npm install mongodb --save*
- Creiamo un modulo per gestire le connessioni con il DB
- Adesso possiamo spostare la gestione degli utenti nel DB

ref https://www.npmjs.com/package/mongodb

Unit test (opzionale)

- Il modo piu' semplice per scrivere unit test per Express e' usare il pacchetto supertest (https://www.npmjs.com/package/supertest)

```
const request = require("supertest");

const app = require('../../app')

describe("GET /", function () {

 it("responds with json", function (done) {

   request(app).get("/").set("Accept", "application/json").expect(200, done);

 });

});
```

REST API (opzionale)

- Express permette anche di sviluppare API REST
- Installiamo Postman (https://www.postman.com/) per facilitare lo sviluppo
- Creiamo una nuova applicazione Express con *$ npx express-generator --no-view*
- Possiamo cancellare i file che non ci servono
- Per effettuare risposte in JSON, ci basta usare *res.json()* oppure usando il middleware json, semplicemente *res.send({...})*