ExpressJS

Introduction

ExpressJS est un **framework backend minimaliste** pour **Node.js**. Il permet de créer facilement un serveur et de définir des routes pour répondre à des requêtes HTTP.

Pourquoi utiliser Express ?

- Simple à prendre en main
- Très flexible
- Idéal pour créer des API REST
- Offre un système de middleware puissant
- Grande communauté et beaucoup d'exemples

installation

- créer un repo github
- récupérer ce repo via votre commande `git clone`
- faites la commande `npm init` on initalise le projet nodejs
- faites la commande `npm install express` on installe express dans notre projet avec nodeJS (npm)
- faites la commande `npm install -g nodemon` on installe nodemon globalement pour relancer le serveur automatiquement à chaque changement de code

Les routes en Express

Une route représente une URL + une méthode HTTP. Les méthodes les plus courantes sont :

- `GET → lire une ressource`
- `POST → créer une ressource`
- `PUT → remplacer une ressource`
- `PATCH → modifier partiellement une ressource`
- `DELETE → supprimer une ressource`

Pour définir une route par defaut dans Express, on utilise la méthode `app.get()`, que l'on va appliquer dans notre fichier `server.js` :

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

app.get("/hello", (req, res) => {
  res.send("Bonjour !");
});

app.listen(3000, () => {
  console.log("Serveur démarré sur http://localhost:3000");
});
```

si nous tappons dans le navigateur `http://localhost:3000/hello` nous verrons le message "Bonjour !".

Les paramètres d'URL

il est possible de recuperer un id avec `/:id`

```
app.get("/users/:id", (req, res) => {
  res.send("Utilisateur n° " + req.params.id);
});
```

`/users/42` → `req.params.id` vaut "42"

Router : pour organiser les routes

Pour éviter que app.js devienne trop gros, on utilise express.Router().

Nous allons créer un fichier `users.js` dans un dossier `routes` :

```
const express = require("express");
const router = express.Router();

router.get("/", (req, res) => {
  res.json({ message: "liste des utilisateurs" });
});

module.exports = router;
```

Puis dans `server.js` :

```
app.use("/users", router);
```

niveau arborescence nous aurons donc :

```
└── src
  ├── routes
  │   └── users.js
  ├── db.js
  └── server.js
├── package.json
├── package-lock.json
├── .env
```

dans notre exemple, je dis que toutes les routes définies dans users.js seront préfixées par `/users`.

Les Query Params (paramètres GET)

Les query params sont des paires clé/valeur ajoutées à l'URL après un `?`. Exemple d'URL avec query params :

`/users?role=admin&order=desc`

```
app.get("/users", (req, res) => {
  console.log(req.query.role); // "admin"
  console.log(req.query.order); // "desc"
});
```

Envoyer une réponse

Tu peux répondre de plusieurs manières :

- `res.send()` => Texte, HTML, etc.
- `res.json()` => Format JSON (le plus courant)
- `res.status()` => Changer le code HTTP :

```
res.status(404).send("Not found");
```

status codes courants :

- 200 : OK
- 201 : Créé
- 400 : Requête incorrecte
- 401 : Non autorisé
- 403 : Interdit
- 404 : Non trouvé
- 500 : Erreur serveur

Qu'est-ce qu'un middleware ?

Un middleware est une fonction qui s'exécute entre la réception de la requête et l'envoi de la réponse.

Il peut :

- lire ou modifier req (la requête)
- lire ou modifier res (la réponse)
- décider de continuer (avec next())
- ou stopper la requête

Exemple de middleware logger :

```
app.use((req, res, next) => {
  console.log(`→ ${req.method} ${req.url}`);
  next();
});
```

Gestion des erreurs

```
app.use((err, req, res, next) => {
  try {
    // Logique de gestion des erreurs
  } catch (err) {
    console.error(err.stack);
    res.status(500).json({ error: "Erreur interne" });
  }
});
```

Exemple complet

```
const express = require("express");
const app = express();
// Body parser JSON
app.use(express.json());
// Logger middleware
app.use((req, res, next) => {
  console.log(`[${req.method}] ${req.url}`);
  next();
});
// Route GET
app.get("/", (req, res) => {
  res.json({ message: "Hello world" });
});

// Route GET pour une seule id
app.get("/:id", (req, res) => {
  res.json({ message: "Hello world" });
});

// Route POST
app.post("/user", (req, res) => {
  const { name } = req.body;
  res.json({ message: `Bienvenue ${name}` });
});

// Route PUT pour une seule id
app.put("/:id", (req, res) => {
  res.json({ message: "Hello world" });
});
```

```
// Route DELETE pour une seule id
app.delete("/:id", (req, res) => {
  res.json({ message: "Hello world" });
});

// Lancement du serveur
app.listen(3000, () => {
  console.log("Serveur lancé sur http://localhost:3000");
});
```