Express - 3

Environment Variables and POST requests

Environment Variables

Setting up environment variables in an Express.js application is essential for securely configuring sensitive information like API keys, database URIs, and other configuration details.

Hardcoding sensitive information in the source code is not recommended.

Using environment variables also allows us to manage all the sensitive data at one place

Steps to create and use Environment Variables

1. Create a file named .**env** in the root directory of your project. This fill will store your environment variables.

```
PORT=3000
DB_URI=mongodb://localhost/mydatabase
API_KEY=your_api_key_here
```

2. You can access environment variables anywhere in your application using **process.env.VARIABLE_NAME**

```
const port = process.env.PORT || 3000;
```

Best Practice

You should never commit your **.env** file to version control.
Always add .env to your .gitignore file

Because env file contains your sensitive data and you do not want to push your sensitive data to version control such as Github, Gitlab, etc

POST Requests

POST Requests

Definition: POST requests are HTTP methods used to submit data to a server to create or update or resource.

Usage: Commonly used for forms, file uploads, and APIs to send data to server

Setting up POST requests

To start using POST request we need to use the body-parser middleware

Body-parser middleware is used to parse incoming request bodies in middleware before your handlers.

Two common body-parser middleware that we use

1. json() method parses JSON-formatted data
2. urlencoded() method parses URL-encoded data

```
app.use(express.json());
app.use(express.urlencoded({extended: true}));
```

Handling a POST request

1. Create a route using the app.post() method.
2. In the route you can access the data sent to the POST route using req.body,
3. Process the data and then respond accordingly

Creating a new user using POST request

```
app.post("/users/register", (req, res) => {
  const user = req.body

  if(!user.id || !user.username)
    return res.status(400).json({message: "Id or Username is not present"})

  users.push(user);
  res.json({message: "Registration Successfull"});

})
```

Updating a users data using POST request

```
app.put('/users/update', (req, res) => {
  console.log(req.body)
  const id = Number(req.body.id);
  const username = req.body.username;
  const userIndex = users.findIndex(user => user.id == id);

  if(userIndex === -1)
      return res.status(404).json({message: "User not found"})

  users[userIndex].username = username;

  res.json(users)

})
```