Refactoring Expressions Routes

5 min

Now that you've learned about nested route matching, let's refactor the rest of the /expressions routes into **expressions.js**.

Instructions

1. Checkpoint 1 Passed

1.

Move all your /expressions routes to your router into **expressions.js**. Make sure that they still match the same request paths, and remove the duplicate code from **app.js**.

Move the following routes to **expressions.js**:

- GET /expressions/:id
- PUT /expressions/:id
- POST /expressions
- DELETE /expressions/:id

Remember to change the paths for each route handler as you move them, as they should already be mounted at /expressions inside **expressions.js**.

Make sure that you still require the same helper functions from **utils.js** in **expressions.js**.

Hint

You'll need all these helper functions in **expressions.js**: getElementById, getIndexById, updateElement, seedElements, createElement

app.js

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

const { getElementById, getIndexById, updateElement,
  seedElements, createElement } = require('./utils');

const PORT = process.env.PORT || 4001;
// Use static server to serve the Express Yourself Website
app.use(express.static('public'));
```

```
let animals = [];
seedElements(animals, 'animals');

const expressionsRouter = require('./expressions.js');

app.use('/expressions', expressionsRouter);

// Get all animals
app.get('/animals', (req, res, next) => {
  res.send(animals);
});

// Get a single animal
app.get('/animals/:id', (req, res, next) => {
  const animal = getElementById(req.params.id, animals);
  if (animal) {
    res.send(animal);
  } else {
    res.status(404).send();
  }
});

// Create an animal
app.post('/animals', (req, res, next) => {
  const receivedAnimal = createElement('animals', req.query);
  if (receivedAnimal) {
    animals.push(receivedAnimal);
    res.status(201).send(receivedAnimal);
  } else {
    res.status(400).send();
  }
});

// Update an animal
app.put('/animals/:id', (req, res, next) => {
  const animalIndex = getIndexById(req.params.id, animals);
  if (animalIndex !== -1) {
    updateElement(req.params.id, req.query, animals);
    res.send(animals[animalIndex]);
  } else {
    res.status(404).send();
  }
});

// Delete a single animal
app.delete('/animals/:id', (req, res, next) => {
  const animalIndex = getIndexById(req.params.id, animals);
  if (animalIndex !== -1) {
    animals.splice(animalIndex, 1);
    res.status(204).send();
  } else {
    res.status(404).send();
  }
});

app.listen(PORT, () => {
  console.log(`Server is listening on ${PORT}`);
});
```

expressions.js

```
const express = require('express');

const { seedElements, getElementById, createElement, updateElement,
getIndexById } = require('./utils');


let expressions = [];

seedElements(expressions, 'expressions');


const expressionsRouter = express.Router();


module.exports = expressionsRouter;


// Get all expressions

expressionsRouter.get('/', (req, res, next) => {

  res.send(expressions);

});


// Get a single expression

expressionsRouter.get('/:id', (req, res, next) => {

  const foundExpression = getElementById(req.params.id, expressions);

  if (foundExpression) {

    res.send(foundExpression);

  } else {

    res.status(404).send();

  }

});


// Update an expression

expressionsRouter.put('/:id', (req, res, next) => {

  const expressionIndex = getIndexById(req.params.id, expressions);
  if (expressionIndex !== -1) {
    updateElement(req.params.id, req.query, expressions);
    res.send(expressions[expressionIndex]);
  } else {
    res.status(404).send();
  }
});

// Create an expression
expressionsRouter.post('/', (req, res, next) => {
  const receivedExpression = createElement('expressions', req.query);
  if (receivedExpression) {
    expressions.push(receivedExpression);
    res.status(201).send(receivedExpression);
  } else {
    res.status(400).send();
  }
});

// Delete an expression
expressionsRouter.delete('/:id', (req, res, next) => {
  const expressionIndex = getIndexById(req.params.id, expressions);
  if (expressionIndex !== -1) {
    expressions.splice(expressionIndex, 1);
    res.status(204).send();
  } else {
    res.status(404).send();
  }
});
```