Using Queries

17 min

You may have noticed in the previous exercise that our PUT route had no information about how to update the specified expression, just the id of which expression to update. It turns out that there was more information in the request in the form of a *query string*. Query strings appear at the end of the path in URLs, and they are indicated with a ? character. For instance, in /monsters/1?name=chimera&age=1, the query string is name=chimera&age=1 and the path is /monsters/1/

Query

Preview: Docs Loading link description

strings

do not count as part of the route path. Instead, the Express server parses them into a JavaScript object and attaches it to the request body as the value of req.query. The key: value relationship is indicated by the = character in a query string, and key-value pairs are separated by &. In the above example route, the req.query object would be { name: 'chimera', age: '1' }.

```
const monsters = { '1': { name: 'cerberus', age: '4'  } };
// PUT /monsters/1?name=chimera&age=1
app.put('/monsters/:id', (req, res, next) => {
  const monsterUpdates = req.query;
  monsters[req.params.id] = monsterUpdates;
  res.send(monsters[req.params.id]);
});
```

Here, we have a route for updating monsters by ID. When a PUT /monsters/1?name=chimera&age=1 request arrives, our callback function is called and, we create a monsterUpdates variable to store req.query. Since req.params.id is '1', we replace monsters['1']'s value with monsterUpdates . Finally, Express sends back the new monsters['1'].

When updating, many servers will send back the updated resource after the updates are applied so that the client has the exact same version of the resource as the server and database.

Instructions

1. Checkpoint 1 Passed

1.

Use req.query to update the proper element in the expressions array.

We've imported a helper function from **/utils.js** to help with this task.

You can use the updateElement() helper function in your PUT /expressions/:id route.

It takes three arguments:

- id (the ID number of the element)
- queryArguments (the new, updated expression object from req.query)
- elementList (the array which contains the element to update)

updateElement() updates that specific element in the elementList array (you'll pass in the expressions array), and then returns the updated element.

Be sure to check that an expression with the id you provide exists in the expressions array (getIndexById() can help)!

To test your functionality with the Express Yourself machine, make sure your server is running, get all expressions, and then use the UPDATE tab to select an individual expression, select updates, and send the PUT request.

Hint

Remember:

- getIndexById will return -1 if the expressions array doesn't contain an element with that id.
- updateElement will throw an error if you pass in an id that doesn't exist in the array

app.js

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

// Serves Express Yourself website
app.use(express.static('public'));

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

const expressions = [];
seedElements(expressions, 'expressions');
```

```
const PORT = process.env.PORT || 4001;

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

app.get('/expressions/:id', (req, res, next) => {
  const foundExpression = getElementById(req.params.id, expressions);
  if (foundExpression) {
    res.send(foundExpression);
  } else {
    res.status(404).send();
  }
});

app.put('/expressions/:id', (req, res, next) => {
  const indexToUpdate = getIndexById(req.params.id, expressions);

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

app.listen(PORT, () => {
  console.log(`Listening on port ${PORT}`);
```

```
});
```

utils.js

```
let expressionIdCounter = 0;
let animalIdCounter = 0;

const getElementById = (id, elementList) => {
  return elementList.find((element) => {
    return element.id === Number(id);
  });
};

const getIndexById = (id, elementList) => {
  return elementList.findIndex((element) => {
    return element.id === Number(id);
  });
};

const createElement = (elementType, queryArguments) => {
  if (queryArguments.hasOwnProperty('emoji') &&
      queryArguments.hasOwnProperty('name')) {
    let currentId;
    if (elementType === 'expressions') {
      expressionIdCounter += 1;
      currentId = expressionIdCounter;
    } else {
      animalIdCounter += 1;
      currentId = animalIdCounter;
    }
    return {
```

```
      'id':    currentId,
      'emoji': queryArguments.emoji,
      'name':  queryArguments.name,
    };
  } else {
    return false;
  }
};

const updateElement = (id, queryArguments, elementList) => {
  const elementIndex = getIndexById(id, elementList);
  if (elementIndex === -1) {
    throw new Error('updateElement must be called with a valid id
parameter');
  }
  if (queryArguments.id) {
    queryArguments.id = Number(queryArguments.id);
  }
  Object.assign(elementList[elementIndex], queryArguments);
  return elementList[elementIndex];
};

const seedElements = (arr, type) => {
  if (type === 'expressions') {
    arr.push(createElement('expressions', {'emoji': '😀', 'name':
'happy'}));
    arr.push(createElement('expressions', {'emoji': '😎', 'name':
'shades'}));
    arr.push(createElement('expressions', {'emoji': '😴', 'name':
'sleepy'}));
  } else if (type === 'animals') {
```

```
    arr.push(createElement('animals', {'emoji': '🐶', 'name':
'Pupper'}));

    arr.push(createElement('animals', {'emoji': '🐍', 'name':
'Snek'}));

    arr.push(createElement('animals', {'emoji': '😺', 'name':
'Maru'}));

  } else {

    throw new Error(`seed type must be either 'expression' or
'animal'`);

  }

};


module.exports = {

  createElement: createElement,

  getIndexById: getIndexById,

  getElementById: getElementById,

  updateElement: updateElement,

  seedElements: seedElements,

};
```