Mise en place du CRUD

2019/2020

APERCU

Dans ce TP vous serez amenés à mettre en place un modèle CRUD

OBJECTIFS

1. Mettre en place les routes
2. Communiquer avec la base de données

1. Mise en place du CRUD

Dans ce TP nous allons mettre en place un modèle CRUD pour un objet utilisateur. Le modèle CRUD peut s'appliquer sur n'importe quel type d'objet. A vous de comprendre ce modèle afin d'ensuite pouvoir le décliner selon vos besoins.

Avant de procéder à la suite du TP, veuillez créer une nouvelle table **users** dans votre base de donnée avec deux colonnes **id** et **username**. Avec **id** un **int** en auto incrémentation et **username** un **varchar** de 255 caractères.

1.1 Création des routes

Tout d'abord, nous allons créer les différentes routes de nos modèles. Il faut donc au moins quatre routes pour le **Create**, **Read**, **Update** et **Delete**.

Chaque route utilisera une méthode différentes pour l'accès:

- **Create**: POST
- **Read**: GET
- **Update**: PUT
- **Delete**: DELETE

Attention aussi aux paramètres de routage. Seule la route **Create** n'a pas besoin de paramètres. Quant aux routes **Read** et **Delete**, leurs paramètres de routage sont facultatifs. Cela veut donc dire avec **Express** qu'il faut créer deux routes: une avec le paramètre et une sans.

```
app.post('/users', function(req, res) {
    res.send(JSON.stringify("Create"));
});

app.get('/users', function(req, res) {
    res.send(JSON.stringify("Read"));
});
app.get('/users/:id', function(req, res) {
    res.send(JSON.stringify("Read"));
});

app.put('/users/:id', function(req, res) {
    res.send(JSON.stringify("Update"));
});

app.delete('/users', function(req, res) {
    res.send(JSON.stringify("Delete"));
});
app.delete('/users/:id', function(req, res) {
    res.send(JSON.stringify("Delete"));
});
```

Nous avons donc toutes nos routes ici pour notre modèle CRUD. Vous pouvez les tester avec Postman pour vérifier la bonne implémentation. Il faut donc désormais implémenter les requêtes SQL dans chacune des fonctions associées à la route.

Pour la route **Create**, il faut déjà récupérer les informations de la requête POST puis créer la requête SQL qui ajoutera la ligne dans la table:

```
const express = require('express');
const mysql = require('mysql');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.urlencoded({ extended: true }));

let db = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "root",
    database: "courses",
    port: "3306"
});
app.post('/users', function(req, res) {
    let username = req.body.username;
    let query = `INSERT INTO users (username) VALUES ('${username}')`;

    db.query(query, function(err, result, fields) {
        if (err) throw err;

        res.send(JSON.stringify("Success"));
    });
});
```

Ici nous utilisons donc la librairie **body-parser** qui nous permet d'extraire les données de notre requête POST de type **x-www-form-urlencoded**. Pour utiliser la bibliothèque **body-parser** à notre application nous utilisons la méthode suivante:

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

Pour tester cette route avec Postman, sélectionnez dans la catégorie "Body" (en dessous de la zone de texte pour l'URL) le type **x-www-form-urlencoded**, puis entrez la pair clef/valeur avec la clef **username** et la valeur de votre choix. La clef **username** est récupérée dans votre script NodeJS grâce au dictionnaire **req.body** qui stocke toutes les pairs clef/valeur de votre requête **x-www-form-urlencoded**.

Ensuite, pour les routes **Read**, il faut simplement récupérer les données depuis notre base de données:

```
app.get('/users', function(req, res) {
    let query = "SELECT * FROM users"
    db.query(query, function(err, result, fields) {
        if (err) throw err;

        res.send(JSON.stringify(result));
    });
});
app.get('/users/:id', function(req, res) {
    let id = req.params.id;
    let query = `SELECT * FROM users WHERE id=${id}`;
    db.query(query, function(err, result, fields) {
        if (err) throw err;

        res.send(JSON.stringify(result));
    });
});
```

Dans la première route, la table **users** est récupérée en entier puis retournée en réponse JSON. Dans la deuxième route, on sélectionne dans la table **users** l'utilisateur avec l'id passé en paramètre de routage. On utilise donc le dictionnaire **req.params** pour récupérer la variable **id**.

Ensuite, pour la route **Update**:

```
app.put('/users/:id', function(req, res) {
    let id = req.params.id;
    let username = req.body.username;
    let query = `UPDATE users SET username = '${username}' WHERE id=${id}`;
    db.query(query, function(err, result, fields) {
        if (err) throw err;

        res.send(JSON.stringify("Success"));
    });
});
```

Et enfin pour les routes **Delete**:

```
app.delete('/users', function(req, res) {
    let query = "DELETE FROM users";
    db.query(query, function(err, result, fields) {
        if (err) throw err;

        res.send(JSON.stringify("Success"));
    });
});
app.delete('/users/:id', function(req, res) {
    let id = req.params.id;
    let query = `DELETE FROM users WHERE id=${id}`;
    db.query(query, function(err, result, fields) {
        if (err) throw err;

        res.send(JSON.stringify("Success"));
    });
});
```

Vous avez désormais votre modèle **CRUD** fonctionnel grâce auquel vous avez un contrôle complet sur vos objets.