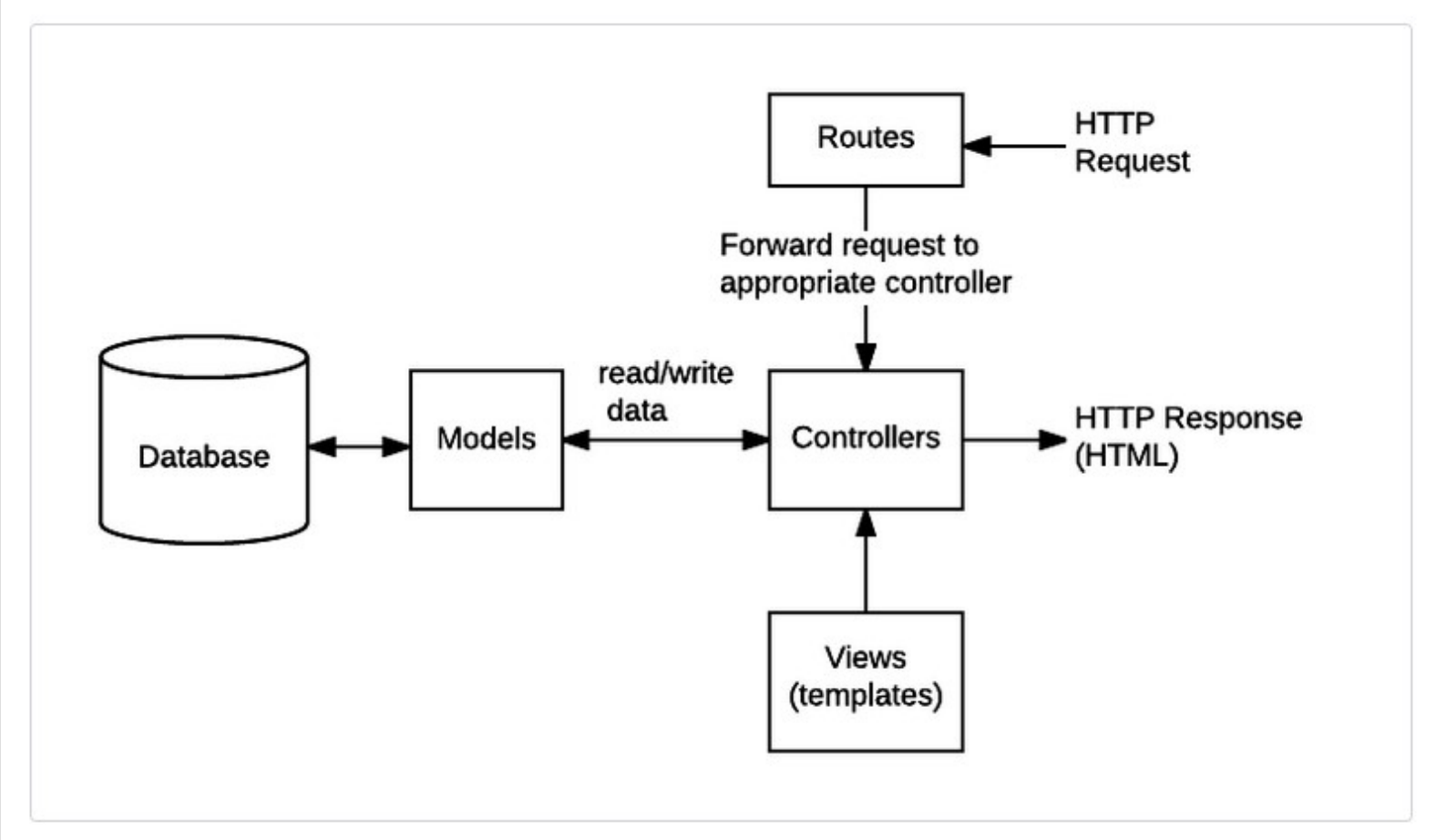
NodeJS : CRUD – Models



Base de données

- On définit la base de données dans laquelle on va placer les informations à manipuler
  => fichier : **database.sql**
- On définit la configuration pour accéder à cette base de données
  => fichier : **dbconfig.js**
- On crée un dossier **/models** où on va regrouper la connexion à la base de données et les fichiers pour les modèles correspondants aux tables

Base de données – configuration

```sql
--
-- Base de données :  `nodejs`
--

-- --------------------------------------------------------

--
-- Structure de la table `messages`
--

DROP TABLE IF EXISTS `messages`;
CREATE TABLE IF NOT EXISTS `messages` (
  `id` smallint(5) UNSIGNED NOT NULL AUTO_INCREMENT,
  `nom` varchar(50) NOT NULL,
  `message` varchar(500) NOT NULL,
  `datemessage` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
COMMIT;
```

```js
module.exports = {
    DB_HOST: "localhost",
    DB_USER: "root",
    DB_PASSWORD: "",
    DB_NAME: "nodejs",
    DB_PORT: 3306
}
```

Dossier models

- On ajoute la connexion à la base de données
  => fichier : **db.js**
- On ajoute un fichier qui va contenir toutes les méthodes pour manipuler les informations dans la base de données
  => fichier : **message.model.js**
- NB : par convention, souvent, le nom de fichier commence par le type d'information ou la table (ici, **message**), et termine par **.model.js**

Connexion

```
const mysql = require("mysql");
const dbConfig = require("../config/dbconfig.js");  // fichier de configuration de la DB

// Création de la connexion avec la DB
// 1. Chargement de la configuration pour la connection
const connection = mysql.createConnection({
    host : dbConfig.DB_HOST,
    user: dbConfig.DB_USER,
    password: dbConfig.DB_PASSWORD,
    database: dbConfig.DB_NAME,
    port: dbConfig.DB_PORT
});

// 2. Ouverture de la connexion
connection.connect(function(error){
    if (error) throw error;         // si erreur de connexion, ça s'arrête ici
    console.log("Connecté avec succès à la base de données !");
});

// On exporte pour les autres modules, la connexion à la base de données
module.exports = connection;
```

Modèle

- On trouve le constructeur avec les attributs, en liaison avec les champs de la table
- On a les différentes méthodes pour le CRUD :
  - Create – Créer un élément
  - Read – Lire un élément / Lire tous les éléments
  - Update – Mettre à jour un élément
  - Delete – Supprimer un élément
- Pour Read, Update et Delete, on doit :
  - 1. choisir l'élément pour déterminer son ID unique
  - 2. traiter l'élément choisi sur base de cet ID

Constructeur

```
const sql = require("./db.js");

console.log("on passe dans models/message.model.js")

// Constructeur
const Message = function(lemessage) {
    this.nom = lemessage.nom;
    this.msg = lemessage.msg;
    this.date_creation = new Date();
};



module.exports = Message;
```

Ecrire un élément

```
// Méthode pour créer un message et le sauvegarder dans la base de données
// newMsg : l'objet Message à créer et sauver dans la DB
// resultat : la réponse du serveur de DB quand je fais l'insertion (OK ou erreur)
Message.create = function(newMsg, resultat){
    sql.query("INSERT INTO messages(nom,message) VALUES (?,?)", [newMsg.nom, newMsg.msg], function(err,res){
        // si on a une erreur lors de l'insertion, on reçoit les données dans err
        // sinon, si tout se passe bien, on reçoit les données dans res
        if (err) {
            console.log("Erreur Message.create : ", err);
            resultat(err,null);
            return;
        }
        console.log("Réponse Message.create : ", res);
        resultat(null,res);
    });
};
```

Lire tous les éléments

```
// Méthode pour lire tous les messages dans la DB
Message.readAll = function(resultat) {
    sql.query("SELECT * FROM messages ORDER BY datemessage DESC", function(err,res){
        // Si erreur dans la lecture des données
        if (err) {
            console.log("Erreur Message.readAll : ", err);
            resultat(err,null);
            return;
        }
        // Si données reçues
        console.log("Réponse Message.readAll : ", res);
        resultat(null,res);
    });
};
```

Lire un élément

```
// Méthode pour lire un message dans la DB, en fonction de son ID
Message.readById = function(id, resultat) {
    sql.query("SELECT * FROM messages WHERE id = ?", id, (err, res) => {
      if (err) {
        console.log("Erreur Message.readById : ", err);
        resultat(err, null);
        return;
      }

      if (res.length) {
        console.log("Message.readById - message trouvé : ", res[0]);
        resultat(null, res[0]);
        return;
      }

      // Pas de message trouvé avec cet ID
      resultat({ type: "ERR_NOT_FOUND" }, null);
    });
};
```

Quel élément à mettre à jour ?

```
// Méthode pour récupérer un message dans la DB, en fonction de son ID, afin de le modifier
Message.updateById = function(id, resultat) {
  sql.query("SELECT * FROM messages WHERE id = ?", id, (err, res) => {
    if (err) {
      console.log("Erreur Message.updateById : ", err);
      resultat(err, null);
      return;
    }

    if (res.length) {
      console.log("Message.updateById - message trouvé : ", res[0]);
      resultat(null, res[0]);
      return;
    }

    // Pas de message trouvé avec cet ID
    resultat({ type: "ERR_NOT_FOUND" }, null);
  });
};
```

Mettre à jour l’élément choisi

```
// Méthode pour modifier un message dans la DB, en fonction de son ID
Message.update = (id, msg, resultat) => {
    console.log(msg);
    sql.query("UPDATE messages SET nom= ?, message= ? WHERE id = ?", [msg.nom, msg.msg, id], (err, res) => {
        if (err) {
          console.log("Erreur Message.update : ", err);
          resultat(err, null);
          return;
        }

        if (res.affectedRows == 0) {
          // Pas de message trouvé avec cet ID
          resultat({ type: "ERR_NOT_FOUND" }, null);
          return;
        }

        console.log("Message.update - message mis à jour : ", { id: id, ...msg });
        resultat(null, { id: id, ...msg });
    });
};
```

Quel élément à supprimer ?

```
// Méthode pour récupérer un message dans la DB, en fonction de son ID, afin de le supprimer
Message.deleteById = function(id, resultat) {
  sql.query("SELECT * FROM messages WHERE id = ?", id, (err, res) => {
    if (err) {
      console.log("Erreur Message.deleteById : ", err);
      resultat(err, null);
      return;
    }

    if (res.length) {
      console.log("Message.deleteById - message trouvé : ", res[0]);
      resultat(null, res[0]);
      return;
    }

    // Pas de message trouvé avec cet ID
    resultat({ type: "ERR_NOT_FOUND" }, null);
  });
};
```

Supprimer l’élément choisi

```
// Méthode pour supprimer un message dans la DB, en fonction de son ID
Message.delete = (id, resultat) => {
  sql.query("DELETE FROM messages WHERE id = ?", [id], (err, res) => {
      if (err) {
        console.log("Erreur Message.delete : ", err);
        resultat(err, null);
        return;
      }

      console.log("RES="+res);

      if (res.affectedRows == 0) {
        // Pas de message trouvé avec cet ID
        resultat({ type: "ERR_NOT_FOUND" }, null);
        return;
      }

      console.log("Message.delete - message " + id + " supprimé");
      resultat(null, res);
  });
};
```