Express.js ❤️‍🔥 ORM

Apa itu ORM?

- ORM (Object-Relational Mapping) adalah teknik yang menghubungkan objek dalam kode dengan baris dalam database relasional.
- Merupakan layer abstraksi yang memungkinkan kita untuk berinteraksi dengan database menggunakan objek dan metode, bukan SQL langsung.
- Memungkinkan kita untuk berinteraksi dengan database relasional (seperti MySQL, PostgreSQL, dan SQLite) menggunakan objek JavaScript.

Contoh ORM

- Sequelize (https://sequelize.org/)
- Prisma (https://www.prisma.io/)
- Mongoose (https://mongoosejs.com/)
- Bookshelf (https://bookshelfjs.org/)
- Waterline (https://waterlinejs.org/)
- Objection (https://vincit.github.io/objection.js/)

Keunggulan ORM

- Abstraksi database yang kuat.
- Membantu mencegah SQL Injection.
- Mudah berpindah ke database lain tanpa mengubah kode.
- Pemeliharaan yang lebih mudah.
- Kemudahan dalam bekerja dengan objek.

Kelemahan ORM

- Performa yang relatif lebih lambat jika dibandingkan dengan SQL.
- Membutuhkan waktu untuk mempelajari cara kerjanya.
- Tidak semua fitur database didukung.

Key Features Sequelize

- Validasi data.
- Hubungan antar model.
- Migration dan seeding database.
- Query builder yang kuat.
- Dukungan untuk banyak database.
- Model yang mendefinisikan tabel.
- Sinkronisasi struktur database.

Manfaat Menggunakan Sequelize

- Pengkodean yang lebih cepat dan mudah.
- Abstraksi database yang lebih tinggi.
- Dukungan untuk transaksi dan penguncian.
- Mudah mengelola hubungan antar tabel.
- Pemeliharaan yang lebih sederhana.
- Memungkinkan pengembangan yang lebih fleksibel.

Integrasi Sequelize di Express.js

```
npm install sequelize mysql2
```

Integrasi Sequelize di Express.js

```javascript
// config/database.js
const { Sequelize } = require("sequelize");
require("dotenv").config();

const host = process.env.DB_HOST;
const database = process.env.DB_DATABASE;
const username = process.env.DB_USERNAME;
const password = process.env.DB_PASSWORD;

const sequelize = new Sequelize(database, username, password, {
  host: host,
  dialect: "mysql",
});

module.exports = sequelize;
```

Integrasi Sequelize di Express.js

```
// models/book.js
const { DataTypes } = require("sequelize");
const sequelize = require("../config/database");
const Book = sequelize.define(
  "book",
  {
    title: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    isbn: {
      type: DataTypes.STRING,
      allowNull: false,
    },
  },
  {
    timestamps: false,
    underscored: true,
  }
);
module.exports = Book;

// Repeat similar structure for Author, Publisher, and Category models
```

Integrasi Sequelize di Express.js

```
const Book = require("../models/Book");

// Get all books
app.get("/books", (req, res) => {
  Book.findAll().then((books) => {
    res.json(books);
  });
});
```

sequelize-cli

Instalasi:

```
npm install sequelize-cli
```

Penggunaan:

```
npx sequelize [command]

# help
npx sequelize --help
```

sequelize-cli

```
init # Initializes project
init:config # Initializes configuration
init:migrations # Initializes migrations
init:models # Initializes models
init:seeders # Initializes seeders
```

Setelah menjalankan perintah di atas, akan muncul:

- File `config/config.json`
- File `models/index.js`
- Direktori/folder `migrations`
- Direktori/folder `seeders`

.sequelizerc

- Merupakan file konfigurasi untuk sequelize-cli.
- Digunakan untuk mengatur path direktori/folder yang digunakan.

```
// .sequelizerc
const path = require("path");

module.exports = {
  config: path.resolve("config", "database.js"),
  "models-path": path.resolve("app", "models"),
  "seeders-path": path.resolve("database", "seeders"),
  "migrations-path": path.resolve("database", "migrations"),
};
```

Original Config File

```json
{
  "development": {
    "username": "root",
    "password": null,
    "database": "database_development",
    "host": "127.0.0.1",
    "dialect": "mysql"
  },
  "test": {
    "username": "root",
    "password": null,
    "database": "database_test",
    "host": "127.0.0.1",
    "dialect": "mysql"
  },
  "production": {
    "username": "root",
    "password": null,
    "database": "database_production",
    "host": "127.0.0.1",
    "dialect": "mysql"
  }
}
```

Dynamic Config

```
require("dotenv").config()

const host = process.env.DB_HOST
const database = process.env.DB_DATABASE
const username = process.env.DB_USERNAME
const password = process.env.DB_PASSWORD

module.exports = {
  development: {
    username: username,
    password: password,
    database: database,
    host: host,
    dialect: "mysql",
  },
  ...
}
```

.env

```
NODE_ENV=development

APP_NAME="Bookstore API"
APP_ENV={$NODE_ENV}
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost

...
```

sequelize-cli: Generate Model and Migration

```
model:generate # Generates a model and its migration

# Example:
npx sequelize model:generate --name Author --attributes name:string
npx sequelize model:generate --name Category --attributes name:string
npx sequelize model:generate --name Publisher --attributes name:string
npx sequelize model:generate --name Book --attributes title:string,authorId:integer
```

sequelize-cli: Generate Database Seeders

```
seed:generate # Generates a new seed file

# Example:
npx sequelize seed:generate --name author-seeder
npx sequelize seed:generate --name book-seeder
```

sequelize-cli: Run Migration and Seeder

```
migration:generate # Generates a new migration file

db:migrate # Run pending migrations
db:migrate:status # List the status of all migrations
db:migrate:undo # Reverts a migration
db:migrate:undo:all # Revert all migrations ran

db:seed # Run specified seeder
db:seed:undo # Deletes data from the database
db:seed:all # Run every seeder
db:seed:undo:all # Deletes data from the database
```

Model Relationships

```
// models/book.js
const { Model } = require("sequelize");

module.exports = (sequelize, DataTypes) => {
  class Book extends Model {
    static associate(models) {
      models.Book.belongsTo(models.Author); // Define relationship
    }
  }
  Book.init(
    {
      title: DataTypes.STRING,
      authorId: DataTypes.INTEGER,
      isbn: DataTypes.STRING,
    },
    {
      sequelize,
      modelName: "Book",
    }
  );
  return Book;
};
```

Model Relationships

```
// models/author.js
const { Model } = require("sequelize");

module.exports = (sequelize, DataTypes) => {
  class Author extends Model {
    static associate(models) {
      models.Author.hasMany(models.Book); // Define relationship
    }
  }
  Author.init(
    {
      name: DataTypes.STRING,
    },
    {
      sequelize,
      modelName: "Author",
    }
  );
  return Author;
};
```

Including Relationships

```
// index.js
const { Author, Book } = require("./app/models");

app.get("/books", async (req, res) => {
  const books = await Book.findAll({ include: "Author" });
  res.json(books);
});

app.get("/books/:id", async (req, res) => {
  const book = await Book.findByPk(req.params.id, { include: "Author" });
  if (book) {
    res.status(404).json({ error: "Book not found" });
    return;
  }

  res.json(book);
});
```