Project Overview (NLP to SQL) Backend Documentation

This application converts natural language queries into SQL queries using OpenAI's GPT-3.5 model. It processes a user's input, generates a corresponding SQL query, and fetches results from a database using that SQL query.

Components

1. Controller: DatabaseController

The DatabaseController handles HTTP requests related to the NLP-to-SQL query conversion. It contains the following endpoints:

- **GET /api/health**: A simple health check to confirm that the service is running.
- **POST /api/query**: Accepts a QueryRequest containing a natural language query. It uses the QueryService to process the query and returns a QueryResponse with the generated SQL query and the results of executing that query.

Key Fields:

- **QueryService**: Injected service that processes natural language queries and executes SQL queries on the database.

---

2. Model: QueryRequest

The QueryRequest class is used to capture a natural language query from the user.

Key Fields:

- **naturalLanguage**: The natural language query provided by the user. It is annotated with @NotNull to ensure the query is not null.

---

3. Model: QueryResponse

The QueryResponse class holds the response data:

- The generated SQL query.
- The results fetched by executing the SQL query on the database.

Key Fields:

- **sqlQuery**: The SQL query generated from the natural language input.
- **results**: A list of maps where each map represents a row of results fetched from the database.

---

4. Service: QueryService

The QueryService class is responsible for:

- Converting natural language into an SQL query.
- Executing the SQL query on a database and returning the results.

Key Methods:

- **processQuery(QueryRequest request)**: Processes the natural language query, converts it into SQL, and executes the SQL query on the database using JdbcTemplate.

- **convertToSQL(String naturalLanguage)**: Sends a request to the OpenAI API to convert the natural language query into a valid SQL query. The method formats the query and sends it to the OpenAI API, which returns the generated SQL query.

Key Fields:

- **openaiApiKey**: The OpenAI API key for authenticating the API request.
- **jdbcTemplate**: Used to execute the SQL query against the database.

---

5. Application: NlPtoSqlQueryConverterApplication

This is the main class that runs the Spring Boot application.

Key Annotations:

- **@SpringBootApplication**: Marks this class as the entry point for the Spring Boot application.
- **@CrossOrigin**: Allows Cross-Origin Resource Sharing (CORS) from http://localhost:3000, enabling the front-end (React, for example) to communicate with the back-end.

Error Handling: ErrorResponse

In case of an error, the application returns a structured error response with the error message.

Key Fields:

- **message**: The error message explaining the issue.

Workflow

1. The user sends a natural language query via a POST request to /api/query.
2. The DatabaseController invokes QueryService to process the query.
3. QueryService sends the natural language query to OpenAI's GPT-3.5 model to generate the corresponding SQL query.
4. The SQL query is executed using JdbcTemplate on the database.
5. A QueryResponse is returned, containing the SQL query and the database results.

Dependencies

- **Spring Boot**: The framework used for building the application.
- **JdbcTemplate**: A part of Spring for querying a relational database.
- **OpenAI API**: Used to convert natural language to SQL using the GPT-3.5 model.
- **Lombok**: Used to automatically generate getters and setters in the model classes.

Error Handling

If the OpenAI API fails or if there's an issue with executing the SQL query, the application returns a 400 Bad Request with a structured error message.

CORS Configuration

The application is configured to accept requests only from http://localhost:3000, which is the default URL for a React front-end during local development.