SQLServerCoreAPI Documentation

This documentation provides details on how to use the APIs provided by the CoreAPI project. The CoreAPI project offers three main functionalities:

The provided set of APIs constitute a comprehensive framework for interacting with a SQL Server database in a .NET Core environment. Featuring functionalities for retrieving data, executing stored procedures, and executing both select and insert queries, these APIs offer a robust solution for seamless database integration. Leveraging Microsoft technologies such as ASP.NET Core and ADO.NET, the APIs facilitate efficient data manipulation and retrieval. With intuitive endpoints for querying and inserting data, developers can easily integrate these APIs into their applications, streamlining database interactions and enhancing overall performance. Whether executing complex stored procedures or simple queries, these APIs provide a versatile toolkit for database operations within a .NET Core ecosystem.

1. **Get Data API**: Fetch data from the database by executing stored procedures.
2. **Insert Data API**: Insert data into the database by executing stored procedures.
3. **Execute Query API**: Execute custom SQL queries including SELECT and INSERT statements.

Base URL

The base URL for accessing the CoreAPI is:

http://your-api-domain.com/API/

1. Get Data API

Endpoint

POST /GetData

Description

This endpoint executes a stored procedure to fetch data from the database.

Request Body

The request body must be a JSON object with the following properties:

- `SPName`: The name of the stored procedure to execute.
- `Parameters`: An array of parameter names expected by the stored procedure.
- `Values`: An array of parameter values corresponding to the parameter names.

Example Request

```
{

  "SPName": "GetUserData",

  "Parameters": ["UserID"],

  "Values": ["123"]

}
```

Example Response

```
[
  {
    "UserID": 123,
    "UserName": "John Doe",
    "Email": "john.doe@example.com"
  },
  {
    "UserID": 456,
    "UserName": "Jane Smith",
    "Email": "jane.smith@example.com"
  }
]
```

2. Insert Data API

Endpoint

POST /InsertData

Description

This endpoint executes a stored procedure to insert data into the database.

Request Body

The request body must be a JSON object with the following properties:

- **`SPName`**: The name of the stored procedure to execute.

- **`Parameters`**: An array of parameter names expected by the stored procedure.
- **`Values`**: An array of parameter values corresponding to the parameter names.

Example Request

```
{

  "SPName": "InsertUserData",

  "Parameters": ["UserName", "Email"],

  "Values": ["John Doe", "john.doe@example.com"]

}
```

Example Response

```
{
  "StatusCode": 200,
  "Message": "Stored procedure executed successfully."
}
```

3. Execute Query API

Endpoint

```
POST /ExecuteQuery
```

Description

The Execute Query API allows users to execute custom SQL queries against the database. It supports both SELECT and INSERT queries.

Request Body

The request body must be a JSON object with the following properties:

```
  "QueryType": 0 for Select

  "QueryType": 1 for Insert
```

- **`Query`**: The SQL query to execute.
- **`QueryType`**: The type of SQL query, which can be either "Select" or "Insert".

Example Request (Select Query)

```
{
 "Query": "SELECT * FROM Users",
 "QueryType": 0
}
```

Example Request (Insert Query)

```
{
 "Query": "INSERT INTO Users (UserName, Email) VALUES ('John Doe', 'john.doe@example.com')",
 "QueryType": 1
}
```

Response

The API response depends on the type of query executed:

- For SELECT queries, the response will contain an array of objects representing the query results.
- For INSERT queries, the response will contain a message indicating the number of rows affected.

Example Response (Select Query)

```
[
 {
   "UserID": 123,
   "UserName": "John Doe",
   "Email": "john.doe@example.com"
 },
 {
   "UserID": 456,
   "UserName": "Jane Smith",
   "Email": "jane.smith@example.com"
 }
]
```

Example Response (Insert Query)

```
"Inserted 1 row successfully."
```

Conclusion

The Execute Query API provides flexibility for executing custom SQL queries against the database. It is suitable for scenarios where predefined stored procedures are not applicable or when dynamic queries are required.