

Introduction



Installation

Install the package via NuGet Package Manager:

```
PM> Install-Package DynamicQueryBuilder
```

Or using .NET CLI:

```
dotnet add package DynamicQueryBuilder
```



Usage

Setting Up DynamicQueryBuilder

1. Add DynamicQueryBuilder to your services in `Program.cs`:

```
var databaseConnectionString =  
builder.Configuration.GetConnectionString("YourConnectionString");  
builder.Services.AddDynamicQueryBuilder(databaseConnectionString);
```

2. Inject and use the service in your code:

```
public class YourController : ControllerBase  
{  
    private readonly IGetDatabaseMetaData _getDatabaseMetaData;  
  
    public YourController(IGetDatabaseMetaData getDatabaseMetaData)  
    {  
        _getDatabaseMetaData = getDatabaseMetaData;  
    }  
  
    /*  
    * This method will return your schema tables metadata.  
    */  
    public async Task<IActionResult> GetDatabaseMetadata()  
    {  
        var metadata = await  
_getDatabaseMetaData.GetDatabaseTablesMetadataAsync(DatabaseDriver.POSTGRESQL); //  
See available drivers.  
        return Ok(metadata);  
    }  
}
```

```
}  
}
```

Method Response

The method returns a structured JSON containing information about various entities in the system. Each table is organized within a `public` object, and each contains an array of columns, where each column provides details such as:

- **Column name** (`column`)
- **Data type** (`columnType`)
- **Primary key status** (`primaryKey`)
- **Relationships with other tables** (`relatedSchema`, `relatedTable`, `relatedColumn`)

Example of Returned Structure:

```
{  
  "public": {  
    "EXAMPLE_TABLE": [  
      {  
        "column": "ID",  
        "columnType": "bigint",  
        "primaryKey": "nextval('\"EXAMPLE_TABLE_ID_seq\"':regclass)",  
        "relatedSchema": null,  
        "relatedTable": null,  
        "relatedColumn": null  
      },  
      {  
        "column": "NAME",  
        "columnType": "character varying",  
        "primaryKey": null,  
        "relatedSchema": null,  
        "relatedTable": null,  
        "relatedColumn": null  
      },  
      {  
        "column": "RELATED_ID",  
        "columnType": "bigint",  
        "primaryKey": null,  
        "relatedSchema": "public",  
        "relatedTable": "ANOTHER_TABLE",  
        "relatedColumn": "ID"  
      }  
    ]  
  }  
}
```

```
}  
}
```

Building Queries

Basic SELECT Query

```
var query = queryBuilder  
    .FromTable("Users")  
    .Columns("Id", "Name", "Email")  
    .GenerateSql();  
// Output: SELECT Id, Name, Email FROM USERS
```

Adding WHERE Filters

```
var query = queryBuilder  
    .FromTable("Users")  
    .Columns("Id", "Name")  
    .FilterBy("Age", ComparisonOperators.GREATER_THAN, "18")  
    .GenerateSql();  
// Output: SELECT Id, Name FROM USERS WHERE Age > 18
```

Using JOINS

```
var query = queryBuilder  
    .FromTable("Users")  
    .Columns("Users.Id", "Users.Name", "Orders.TotalAmount")  
    .Join(JoinOperators.INNER_JOIN, "Orders", "Users.Id = Orders.UserId")  
    .GenerateSql();  
// Output: SELECT Users.Id, Users.Name, Orders.TotalAmount FROM USERS INNER JOIN  
Orders ON Users.Id = Orders.UserId
```