Database Module is a module that provide a database template for mapping a relational Database. such as retrieving, inserting, deleting, updating etc.

The Database Module Consists of the following Classes and interface. DbCommand, DbValue, DataArray, TableBuilder, ColumnBuilder and IDbCommand.

Namespace: Dordstream.Data

Module: Database Module

(1) IDbCommand Interface

Represent a connection and DS-SQL statement to execute against Sql Server database.

the interface allow user to connect and execute commands to the database

Constructors

DbCommand(string)

Initialize a new instance of the DbCommand class with a text of ConnectionString to the database.

DbCommand(string,bool)

Initialize a new instance of the DbCommand class with a text of ConnectionString to the database and isConnectionLeftOpen.

DbCommand(IConfiguration)

Initialize a new instance of the DbCommand class with a IConfiguration interface.

DbCommand(IConfiguration bool)

Initialize a new instance of the DbCommand class with a IConfiguration interface and isConnectionLeftOpen .

Properties

Connection

Get the SqlConnection used by this instance of the DbCommand

AffectedRow

Get and Set the Affected row by the ExecuteQuery()

Method

ExecuteQuery(string,DbValue) : void

Executes a Ds-Sql Query statement against the Connection and return void.

Execute with the text of the query such as Ds-SQL Add, Modify, drop and delete statements, and DbValue class consisting the database values.

ExecuteQueryAsync(string,DbValue) : Task

Initiates the asynchronous of ExecuteQuery method

ExecuteReader(string,DbValue?)

Executes commands that return the list the data in table.

Execute with the text of the query such as Ds-SQL Select statements, and DbValue class consisting the database values. Note the DbValue class can be null

ExecuteReaderAsync(string,DbValue?)

Initiates the asynchronous of ExecuteReader method.

Seed(TableBuilder)

Seed the columns and values with TableBuilder class to the database.

the seed method return void.

SeedAsync(TableBuilder)

Initiates the asynchronous of Seed method.

(2) DbValue

Represents a table value(s) in a database.

Constructors

DbValue()

Initialize a new instance of the DbValue class.

DbValue(bool parse)

Initialize a new instance of the DbValue class.

Method

Add(object)

Store the values and return void.

Add(DataArray)

Store the values with DataArray Class and return void

(3) DataArray

Represents a list of data

Constructors

DataArray()

Initialize a new instance of the DataArray class.

Method

Add(object)

Store the values and return void.

(4) TableBuilder

Represent a Table builder in a database

Constructors

TableBuilder(string)

Initialize a new instance of the TableBuilder class with a text representing the Table Name.

Properties

Columns

Set the ColumnBuilder used by this instance of the TableBuilder.

(5) ColumnBuilder

Represent a Column builder in a database table.

Constructors

ColumnBuilder(DataArray)

Initialize a new instance of the ColumnBuilder class with the DataArray Class.

Example

The following example creates a DbCommand, a DataArray, a TableBuilder, and a ColumnBuilder. the example seed (insert) the data into the database and display the result on the web browser.

```
public class DefaultController : Controller
{
```

```
 IDbCommand command;

 public DefaultController(IConfiguration configuration){

 command = new DbCommand(configuration,true);

}

 [Route("/default")]
 public IActionResult Index()
{

 return View();

}



[Route("/table/seed")]
 public IActionResult Seed()
{

 //using TableBuilder class to create a database table and column
//TableBuilder take a string parameter which specify the table name
Dog.
 //We are creating a Dog table
 TableBuilder tableBuilder = new TableBuilder("Dog");
 //using DataArray class to create the list of the Columns and Values
with Anonymous Class
 DataArray dataArray = new DataArray(){
 new { Name="Bingo", Age= 1},
 new {Name ="Winnie", Age=3},
 new {Name ="Mach", Age=4 },
 new {Name ="Monach", Age= 5}

};
```

```
 //using ColumnBuilder class to build the Column of the Table
 //The Dog table is having Name and Age Column
 tableBuilder.Columns = new ColumnBuilder(dataArray);
//using IDbCommand.Seed method  is used to create and insert the  table with columns and //values into the database with TableBuilder object.
command.Seed(tableBuilder);


 return Content("Done");

 }



 }
```

SQL Queries and Examples

Creating of table, Columns and Adding values Query

The query below creates a table with the columns and the column type will depend on the value type and insert the rows/values. if the table does not exist in the database otherwise, it will insert the rows/values.

Add Column1=Value1, Column2=Value2, Column3=Value3 Into TableName.

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize.

Example

The snippet code below creates student table with their name, age, address and insert their values if student table is not existing in the database otherwise, it will insert their values.

```
 [Route("/student/add")]
 public IActionResult AddStudent()
{

//using add Ds-Sql to command the database table
 string addQuery = "add Name=@1, Age=@2, Address=@3 into Student";

//using DbValue to store the data

DbValue value = new DbValue();

value.Add("DordStream Student");

value.Add(15);

value.Add("No 21 sokoto road zaria Kaduna");

//calling IDbCommand interface which has be initialized  in the Constructor
 //using ExecuteQuery to execute the command in the database

command.ExecuteQuery(addQuery,value);

return Content("Saved");
```

```
}
```

Where the name, age, address are the columns and @number is the value identifier.

Creating of table, Columns and Adding values with Foreign Key Query

The query below creates a table with the columns and the column type will depend on the value type and insert the values, and create relationship with an existing table, if the table does not exist in the database otherwise, it will insert the values.

Add Column1=Value1, Column2=Value2, Column3=Value3 Into TableName Join ForeignTable Where ForeignColumn=ForeignValue

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize.

Example

The snippet code below creates student table with their name, age, address and insert their values and create a relationship with Course table, if student table is not existing in the database otherwise, it will insert their values.

```
[Route("/student/addCourse")]
public IActionResult AddCourse()
{

//using add Ds-Sql to command the database table

string addQuery = "add Name=@1, Title=@2 into Course Join Student where ID=@3";
```

```
 //using DbValue to store the data

DbValue value = new DbValue();

value.Add("Dord112");

value.Add("Introduction To DordStream");

value.Add(121196);

//calling IDbCommand interface which has be initialized  in the Constructor

//using ExecuteQuery to execute the command in the database

command.ExecuteQuery(addQuery,value);

return Content("Saved");
    }
```

Where the name, age, address are the columns and @number is the value identifier.

Adding Unique Key to a Column of an Existing Table Query

The query below adds unique key to a particular or specified column name and the column type will depend on the value type. from an existing table in the database.

Add Unique (ColumnName) Into TableName

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize.

Example

The snippet code below add unique key to name in student table.

```
[Route("/student/adduniquekey")]
public IActionResult AddUniqueKey()
{

//using add Ds-Sql to command the database table

string addQuery = "add unique(Name) into Student";

//using DbValue to store the data

DbValue value = new DbValue();

value.Add("");

//calling IDbCommand interface which has be initialized  in the Constructor
//using ExecuteQuery to execute the command in the database

command.ExecuteQuery(addQuery,value);

return Content("Saved");

}
```

Where the name is the column and @number is the value identifier.

Adding Column of an Existing Table Query

The query below adds new column and the column type will depend on the value type. from an existing table in the database.

Add Column(ColumnName) Into TableName

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize.

Example

The snippet code below add RegistrationDate to student table.

```
 [Route("/student/addreg")]
 public IActionResult AddRegNo()
{
 //using add Ds-Sql to command the database table

 string addQuery = "add Column(RegistrationDate) into Student";

 //using DbValue to store the data

DbValue value = new DbValue();

value.Add("");

 //calling IDbCommand interface which has be initialized  in the Constructor
 //using ExecuteQuery to execute the command in the database

 command.ExecuteQuery(addQuery,value);

 return Content("Saved");

    }
```

Where the Registration is the column and @number is the value identifier.

Adding Primary Key of an Existing Table Query

The query below adds new primary and specify the column type based on the value type. From an existing table in the database. if the table does not has any primary key otherwise, an exception will throw up that primary key already exist.

Add Primary(ColumnName) Into TableName

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize.

Example

The snippet code below adds ID as primary key to student table.

```
  [Route("/student/addprimarykey")]
public IActionResult AddPrimaryKey()
{
//using add Ds-Sql to command the database table

string addQuery = "add Primary(ID) into Student";

//using DbValue to store the data

DbValue value = new DbValue();

value.Add(0);

//calling IDbCommand interface which has be initialized  in the Constructor
//using ExecuteQuery to execute the command in the database

command.ExecuteQuery(addQuery,value);

return Content("Saved");

    }
```

Where the Registration is the column and @number is the value identifier.

Modifying Rows of an Existing Table Query

The query below modifies the row of the specified column with a certain condition. From an existing Table in the database

Modify Column1=Value1, Column2=Value2, Column3=Value3 From TableName Where Column3=Value3.

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize. And where is a conditional keyword which can be used with =, !=, >, <, AND, OR operators.

Example

The snippet code below modify name, address where the age is 15 from student table

```
 [Route("/student/updatestudent")]
public IActionResult UpdateStudent()
{


//using modify Ds-Sql to command the database table


string modifyQuery = "modify Name=@1, Address=@2 from Student where Age=@3";


//using DbValue to store the data


DbValue value = new DbValue();

value.Add("NewDordStreamName");


value.Add("n0 85 zaria road. kaduna");


value.Add(15);
 //calling IDbCommand interface which has be initialized  in the Constructor
 //using ExecuteQuery to execute the command in the database
```

```
command.ExecuteQuery(modifyQuery,value);

return Content("Saved");



    }
```

Where the name, age, address are the columns and @number is the value identifier.

Modifying Primary Key to a Column of an Existing Table Query

The query below modifies the primary key of an existing table in the database with the specified column.

Modify Primary(ColumnName) from TableName.

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize.

Example

The snippet code below change primary key to Name from Course table

```
 [Route("/student/modifyprimarykey")]
public IActionResult ModifyPrimaryKey()
{
//using modify Ds-Sql to command the database table

string modifyQuery = "modify primary(Name) from Course";

//using DbValue to store the data

DbValue value = new DbValue();
```

```
value.Add("");

 //calling IDbCommand interface which has be initialized  in the Constructor
 //using ExecuteQuery to execute the command in the database

command.ExecuteQuery(modifyQuery,value);

return Content("Saved");

    }
```

Note: Primary key won't be modified if the column has been used for foreign Key.

Note: the value of the column is set to empty string which mean the modified column is a string type (NVARCHAR(MAX)).

Where the Age is the column and @number is the value identifier.

Modifying Unique Key to a Column of an Existing Table Query

The query below modifies the unique key of an existing table in the database with the specified column.

Modify Unique(ColumnName) from TableName.

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize.

Example

The snippet code below change unique key to age from student table

```
   [Route("/student/modifyuniquekey")]
public IActionResult ModifyuniqueKey()
{
```

```
//using modify Ds-Sql to command the database table

string modifyQuery = "modify unique(Age) from Student";

//using DbValue to store the data

DbValue value = new DbValue();

value.Add(0);



//calling IDbCommand interface which has be initialized  in the Constructor
 //using ExecuteQuery to execute the command in the database

command.ExecuteQuery(modifyQuery,value);
return Content("Saved");

 }
```

Where the Age is the column and @number is the value identifier.

Note: the value of the column is set to 0 which mean the modified column is a integer type (int).

Deleting Row of an Existing Table Query

The query below delete the row with a certain condition in an existing Table in the database

Delete from TableName Where Column1=Value1.

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize. And where is a conditional keyword which can be used with =, !=, >, <, AND, OR operators.

Example

The snippet code below delete row where the age is 16 from student table

```
 [Route("/student/delete")]
public IActionResult DeleteStudent()
{

//using Delete Ds-Sql to command the database table

string deleteQuery = "delete from Student where Age=@1";

//using DbValue to store the data

DbValue value = new DbValue();

value.Add(16);

 //calling IDbCommand interface which has be initialized  in the Constructor
 //using ExecuteQuery to execute the command in the database

command.ExecuteQuery(deleteQuery,value);

return Content("Saved");

 }
```

Where age is the column and @number is the value identifier.

Drop Unique Key from a Column of an Existing Table Query

The query below removes unique key from an existing table in the database with the specified column.

Drop Unique(ColumnName) from TableName.

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize.

Example

The snippet code below removes unique key from Age in student table

```
 [Route("/student/dropuniquekey")]
 public IActionResult DropUniqueKey()
{

//using Delete Ds-Sql to command the database table

string deleteQuery = "drop unique(Age) from Student";


//using DbValue to store the data

DbValue value = new DbValue();

value.Add(0);


//calling IDbCommand interface which has be initialized  in the Constructor
 //using ExecuteQuery to execute the command in the database

command.ExecuteQuery(deleteQuery,value);
```

```
return Content("Saved");

   }
```

Where the Age is the column and @number is the value identifier.

Note: the value of the column is set to 0 which mean the modified column is a integer type (int).

Drop Primary Key from a Column of an Existing Table Query

The query below removes primary key from an existing table in the database with the specified column.

Drop Primary(ColumnName) from TableName.

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize.

Example

The snippet code below removes primary key from Name in Course table

```
 [Route("/student/dropprimarykey")]
 public IActionResult DropPrimaryKey()
{

//using Delete Ds-Sql to command the database table
 string deleteQuery = "drop primary(Name) from Course";

//using DbValue to store the data
DbValue value = new DbValue();

value.Add("");
```

```
//calling IDbCommand interface which has be initialized  in the Constructor
 //using ExecuteQuery to execute the command in the database

command.ExecuteQuery(deleteQuery,value);

return Content("Saved");

}
```

Where the Name is the column and @number is the value identifier.

Note: Primary key won't be modified if the column has been used for foreign Key.

Note: the value of the column is set to empty string which mean the modified column is a string type (NVARCHAR(MAX)).

Drop Column from an Existing Table Query

The query below drop the specified column from an existing Table in the database

Drop Column(ColumnName) from TableName.

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize.

Example

The snippet code below remove RegistrationDate from student table

```
 [Route("/student/dropcolumn")]
 public IActionResult DropColumnKey()
{

//using Delete Ds-Sql to command the database table
```

```
string deleteQuery = "drop Column(RegistrationDate) from Student";

//using DbValue to store the data
DbValue value = new DbValue();

value.Add(0);

 //calling IDbCommand interface which has be initialized  in the Constructor
 //using ExecuteQuery to execute the command in the database

command.ExecuteQuery(deleteQuery,value);

return Content("Saved");

    }
```

Where the RegistrationDate is the column and @number is the value identifier.

Selecting Rows from an Existing Table Query

The query below selects rows from an existing Table in the database.

Query 1: Select all from TableName. and

Query 2: Select number from TableName Where Column1=Value1 And Column2=Value2.

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize. And where is a conditional keyword which can be used with =, !=, >, <, AND, OR operators.

Example 1

The snippet code below selects all rows from student table

```
 [Route("/student/get")]
 public IActionResult SelectStudent()
{

//using Select Ds-Sql to command the database table
 string selectQuery = "select all from Student";



//calling IDbCommand interface which has be initialized  in the Constructor

//using ExecuteReader to execute the command and return the rows from the
database
 var reader = command.ExecuteReader(selectQuery);

//Json method is used to parse the result to json object

return Json(reader);

  }
```

Example 2

The snippet code below select 5 rows from the student table

```
 [Route("/student/get")]
 public IActionResult SelectStudent()
{

//using Select Ds-Sql to command the database table
 string selectQuery = "select 5 from Student";



 //calling IDbCommand interface which has be initilized in the Constructor
 //using ExecuteReader to execute the command and return the rows from the
```

```
database

var reader = command.ExecuteReader(selectQuery);


//Json method is used to parse the result to json object

return Json(reader);




 }
```

Example 3

The snippet code below select 5 rows for a given condition from the student table

```
[Route("/student/get")]
 public IActionResult SelectStudent()
{

//using Select Ds-Sql to command the database table
 string selectQuery = "select all from Student where Age=@1";

//using DbValue to store the data
DbValue value = new DbValue();

value.Add(30);

//calling IDbCommand interface which has be initialized  in the Constructor
 //using ExecuteReader to execute the command and return the rows from the
database
 var reader = command.ExecuteReader(selectQuery,value);
```

```
return Json(reader);


  }
```

Select Max, Min, Avg, Count, Distinct and OrderBy from an Existing Table Query

The query below selects rows from an existing Table in the database.

Select Max(ColumnName) from TableName.

Select Min(ColumnName) from TableName.

Select Avg(ColumnName) from TableName.

Select Count(ColumnName) from TableName.

Select Distinct(ColumnName or ColumnName1, ColumnName2) from TableName.

Select all from TableName Orderby(Column).

Select all from TableName Orderby(Column) ASC | DESC.

Select all from TableName where condition Orderby(Column) ASC | DESC.

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize. And where is a conditional keyword which can be used with =, !=, >, <, AND, OR operators.

Example

The snippet code below selects all rows from student table in descending order

```
[Route("/student/get")]
 public IActionResult SelectStudent()
{
```

```
//using Select Ds-Sql to command the database table
 string selectQuery = "select all from Student where ID>@1 Orderby(Name) desc";


//using DbValue to store the data
DbValue value = new DbValue();

value.Add(30);




//calling IDbCommand interface which has be initialized   in the Constructor
 //using ExecuteReader to execute the command and return the rows from the
database

 var reader = command.ExecuteReader(selectQuery,value);

//Json method is used to parse the result to json object

 return Json(reader);



    }
```

Drop an Existing Table Query

The query below drops an existing Table in the database

Drop TableName.

Note: the keywords are not case sensitive. i.e. it can be written Lowercase, Uppercase and Capitalize.

Example

The snippet code below drop course table

```
 [Route("/student/drop")]
public IActionResult DropStudent()
{

//using drop Table Ds-Sql to command the database table
string dropQuery = "drop Course";

//calling IDbCommand interface which has be initialized in the Constructor
//using ExecuteQuery to execute the command to the database


command.ExecuteQuery(dropQuery,null);


//Json method is used to parse the result to json object

return Content("Table Dropped");


  }
```