Step by Step implementing Dapper

Dapper is an micro ORM for the Microsoft .NET platform: it provides a framework for mapping an object-oriented domain model to a traditional relational database.

Steps

1.Install-Packages ON nugget

1. Dapper 2.System.Data.SqlClient

2.Create DapperContext class

```
public class DapperDBContext {

private readonly string _connectionString;

public DapperDBContext(string connectionString)  {

  _connectionString = connectionString; }

public IDbConnection CreateConnection() => new SqlConnection(_connectionString);

}
```

3.add configuration in Program or startup

```
var DefaultConnection = configuration.GetConnectionString("DefaultConnection");

services.AddSingleton<DapperDBContext>(provider => new

DapperDBContext(DefaultConnection));
```

4.Using Deppar methods (ExecuteScalar, Query, ...)

ExecuteScalar

Executes query that return a single value.

```
public const string sqlGetCountCustomersNew = "SELECT COUNT(*) FROM " + _tablename
+ " where cast( Created as date)= cast(Getdate() as date)";


using (var db = _context.CreateConnection()) {
return await db.ExecuteScalarAsync<int>(SqlCommandConst.sqlGetCountCustomersNew); }
```

Querying Single Row

Executes a query that return a row(first or single)

- QuerySingle(result= a row)
- QuerySingleOrDefault(result= a row or null)
- QueryFirst(result= first row)
- QueryFirstOrDefault(result= first row or null)

```
public const string sqlGetFirstCustomers = " SELECT id,firstname + ' ' + lastName as
FullName  ,Email +' | ' + PhoneNumber Email_PhoneNumber ,BankAccountNumber " +
"FROM " + _tablename + " where cast( Created as date)= cast(Getdate() as date)";


using (var db = _context.CreateConnection())  {
return await
db.QueryFirstOrDefaultAsync<CustomerInfo(SqlCommandConst.sqlGetFirstCustomers); }
```

Query

Executes a query, returning and maps it to a list of dynamic objects

```
 public const string sqlGetAllCustomers =@" SELECT id,firstname + ' ' + lastName as
FullName ,Email +' | ' + PhoneNumber Email_PhoneNumber ,BankAccountNumber " +
"FROM " + _tablename + " where cast( Created as date)= cast(Getdate() as date)";


using (var db = _context.CreateConnection()) {
return await db.QueryAsync<CustomerInfo>(SqlCommandConst.sqlGetAllCustomers); }
```

QueryMultiple

Execute a multi query that returns multiple result sets, and access each in turn.

- Step 1: execute multiple queries using the QueryMultiple
- Step 2: get the returned results with Read, ReadFirst, ReadFirstOrDefault, ReadSingle ,ReadSingleOrDefault

```
  public const string sqlGetFirstAndLastCountCustomers =@" SELECT top 1 id,firstname
+ ' ' + lastName as FullName ,Email +' | ' + PhoneNumber Email_PhoneNumber
,BankAccountNumber FROM " + _tablename + "  order by Id  " +
"SELECT top 1 id,firstname + ' ' + lastName as FullName ,Email +' | ' + PhoneNumber
Email_PhoneNumber ,BankAccountNumber FROM " + _tablename + "  order by Id desc ";


using (var db = _context.CreateConnection()){
using (var result = await
db.QueryMultipleAsync(SqlCommandConst.sqlGetFirstAndLastCountCustomers))  {
   var _first = await result.ReadFirstAsync<CustomerInfo>();
   var _last = await result.ReadFirstAsync<CustomerInfo>();
   return new Tuple<CustomerInfo, CustomerInfo>(_first, _last);
}  }
```

Execute

Execute a query and return the number of affected rows.

It is used to execute INSERT, UPDATE, and DELETE statement.

```
public const string sqlUpdateEmailNullCustomers = @"  update " + _tablename + " set
Email='Empty' where Email is null ";


using (var db = _context.CreateConnection()) {
return await db.ExecuteAsync(SqlCommandConst.sqlUpdateEmailNullCustomers); }
```

ExecuteReader

Execute a query and return an IDataReader .

```
public const string sqlGetValidCsutomer = "SELECT COUNT(*) as count,max(Created) as
maxCreated  FROM " + _tablename;


public async Task<Tuple<long?, DateTime?>> GetValidCsutomer() {
using (var db = _context.CreateConnection()) {
long count = 0; DateTime maxCreateDate = default;
var _result = await db.ExecuteReaderAsync(SqlCommandConst.sqlGetValidCsutomer);
while (_result.Read()) {
     int _f = _result.GetInt32(0); count = _f;
     maxCreateDate = _result.GetDateTime(1); }
    return new Tuple<long?, DateTime?>(count, maxCreateDate);
 } }
```

parameterized query

use Parameter in queries

```
public const string sqlGetCsutomerByEmail =

  "SELECT id,firstname + ' ' + lastName as FullName ,Email +' | ' + PhoneNumber

Email_PhoneNumber ,BankAccountNumber  FROM " + _tablename + " where Email=@Email";


using (var db = _context.CreateConnection())  {

         var _result = await db.QueryAsync<CustomerInfo>

(SqlCommandConst.sqlGetCsutomerByEmail,new {Email = Email});

    return _result;

}
```

Thank you for Reading

Follow me for more Sharing

https://www.linkedin.com/in/abolfazlsadeghi

https://github.com/abolfazlSadeqi

https://stackoverflow.com/users/10193401/abolfazl-sadeghi