ADO.NET Core

(SqlConnection, SqlCommand, SqlDataReader)

Артём Трофимушкин

NuGet-пакет Microsoft.Data.SqlClient

Данный NuGet-пакет содержит большинство классов, необходимых для обеспечения доступа к базе данных MS SQL Server. Основные классы:

- SqlConnection — класс обслуживающий соединение с базой данных.
  - Для соединения использует строку подключения к БД, хранящую в себе
    - Адрес сервера
    - Название БД
    - Аутентификационные данные
    - Дополнительные опциональные детали устанавливаемого подключения
- SqlCommand — класс, позволяющий выполнять SQL-команды
  - Требует наличия открытого соединения
  - Бывает
-

SqlConnection

Представляет подключение к базе данных SQL Server. Реализует интерфейс IDisposable — требует закрытия соединения после работы.

ConnectionString это просто строка собранная по определенным правилам:

```
string connectionString = "Server=tcp:Server,Port;Initial Catalog=Database;
Persist Security Info=False;User ID=User;Password=Pass;Encrypt=True;"

using (var connection = new SqlConnection(connectionString))
{
    connection.Open();
    // Работа с БД...
}
```

SqlCommand (как инструкция T-SQL)

Представляет инструкцию Transact-SQL или хранимую процедуру, выполняемую над базой данных SQL Server.

Содержит ряд методов ExecuteXXX для различных результатов.

```
using (var connection = new SqlConnection(connectionString))
{
    connection.Open();
    var sqlCommand = connection.CreateCommand();
    sqlCommand.CommandType = System.Data.CommandType.Text;
    sqlCommand.CommandText = "SELECT COUNT(*) FROM [TableName]";

    var result = (int) sqlCommand.ExecuteScalar();
    return result;
}
```

SqlDataReader (чтение данных запроса)

Предоставляет способ чтения строк запроса базы данных SQL Server:

```
using (var reader = sqlCommand.ExecuteReader())
{
     if (!reader.HasRows)
          return result;

     int idColumn1Index = reader.GetOrdinal("Column1");
     int idColumn2Index = reader.GetOrdinal("Column2");

     while (reader.Read())
     {
          var int32Value = reader.GetInt32(idColumnIndex);
          var stringValue = reader.GetString(nameColumnIndex);
     }
}
```

SqlCommand (как хранимая процедура)

```
using (var connection = new SqlConnection(connectionString))
{
    connection.Open();
    var sqlCommand = connection.CreateCommand();
    sqlCommand.CommandType = System.Data.CommandType.StoredProcedure;
    sqlCommand.CommandText = "StoredProcedureName";

    sqlCommand.Parameters.AddWithValue("@inputParam1", inputParam1);
    sqlCommand.Parameters.AddWithValue("@inputParam2", inputParam2);

    var outputParameter = new SqlParameter("@outParam", System.Data.SqlDbType.Int, 1);
    outputParameter.Direction = System.Data.ParameterDirection.Output;
    sqlCommand.Parameters.Add(outputParameter);

    sqlCommand.ExecuteNonQuery();
    int outputValue = (int)outputIdParameter.Value;
}
```

Совместная работа

Реализуем интерфейс IProductRepository:

```
public interface IProductRepository
{
    int GetCount();
    Product GetById(int id);
    List<Product> GetAll();
    int Insert(CreateProductCommand dto);
}
```

Определение класса `Product` содержит поля:

- Идентификатор (Id)
- Название (Name)
- Цена (Price)

Совместная работа

Реализуем интерфейс IOrderRepository:

```
public interface IOrderRepository
{
      int GetCount();
      Order GetById(int id);
      List<Order> GetAll();
      int Insert(CreateOrderCommand dto);
}
```

Определение класса `Order` содержит поля:

- Идентификатор (Id)
- Имя покупателя (Customer)
- Дату заказа (OrderDate)
- Скидку (Discount)
- Общую сумму за заказ с учетом скидки (Total)

Домашняя работа

Реализовать полностью интерфейсы IOrderRepository и IProductRepository

Спасибо за внимание.