ADO.NET Core

(SqlConnection, SqlCommand, SqlDataReader)

Андрей Голяков

NuGet-пакет System.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 опрелённым правилам:

```
string connectionString =
    "Data Source=SqlServer;Initial Catalog=DbName;Integrated Security=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 (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    var sqlCommand = connection.CreateCommand();
    sqlCommand.CommandType = System.Data.CommandType.Text;
    sqlCommand.CommandText = "SELECT COUNT(*) FROM dbo.SomeTable";

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

SqlDataReader (чтение данных запроса)

Предоставляет способ чтения строк запроса базы данных SQL Server:

```
using (SqlDataReader 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);
     }
}
```

Самостоятельная работа #1

Создать новый интерфейс IOrderRepository:

```
public interface IOrderRepository
{
	int GetOrderCount();

	List<string> GetOrderDeiscountList();
}
```

Расширить класс OnlineStoreRepository созданным интерфейсом и реализовать его в отдельном файле OnlineStoreRepository.Order.cs.

SqlCommand (как хранимая процедура)

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

Самостоятельная работа #2

Написать SQL-запросы для создания двух хранимых процедур:

- `dbo.AddOrder(@customerId AS INT, @orderDate AS DATETIMEOFFSET, @discount AS FLOAT = NULL, @id AS INT OUTPUT)`
- `dbo.AddOrderItem(@orderId AS INT, @productId AS INT, @numberOfItems AS INT)`

Расширить интерфейс IOrderRepository новым методом AddOrder:

```
public interface IOrderRepository
{
    int GetOrderCount();

    List<string> GetOrderIdList();

    int AddOrder(
        int customerId,
        DateTimeOffset orderDate,
        float? discount,
        List<Tuple<int, int>>
        productIdCountList);
}
```

Реализовать новый метод в классе OnlineStoreRepository.

Домашняя работа

Попытаться самостоятельно реализовать методы интерфейса `IReminderStorage` в новой сборке `Class Library (.NET Standard)` `Reminder.Storage.SqlServer.ADO`.

Спасибо за внимание.