**Реализация доступа к БД**

Варианты реализации подключения к БД:

1. Применение ADO.Net(см. МДК.01.01)

Требуется подключить NuGet пакет для соответствующей СУБД(провайдер/поставщик данных), Microsoft.Data.SqlClient или MySql.Data или Microsoft.Data.Sqlite

Используется: Connection, Command, DataReader, DataAdapter

1. Применение ORM(Object-Relational Mapping) – объектно-реляционное преобразование, сопоставляет строки таблицы объектам, а таблицы классам. Это технология программирования, связывающая БД с концепцией ООП, через создание виртуальной объектной базы данных

Задачи, которые решает ORM:

1. Работа с данными таблиц, как с объектами
2. Работа с объектами как с подходящими для хранения БД данными
3. Избавление от необходимости написания SQL кода
4. Упрощение разработки и поддержки кода

ORM облегчает создание CRUD приложений:

Create – INSERT

Read – SELECT

Update – UPDATE

Delete – DELETE

Соответствие объектов в БД, элементом программы на объектно-ориентированном языке программирования:

Таблица – Класс

Столбец – Свойство/поле класса

Строка – Объект класса

Первичный ключ – Уникальное свойство

Внешний ключ – Свойство/ссылка на объект другого класса

public class Role

{

}

public class User

{

string Login;

string Password;

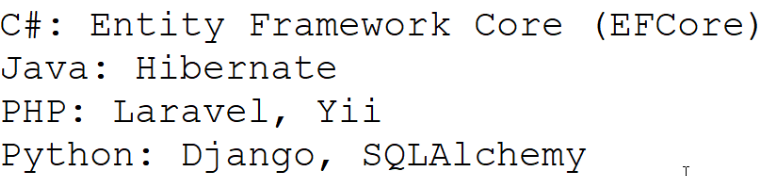
Role Role; // FK

}

SQL запрос, в C# используется LINQ либо аналоги

Подходы, применяемые для работы с ORM:

1. Code First – сначала описываются классы, а потом на их основе с помощью миграций создается БД
2. DataBase First – сначала создается база данных, а потом на ее основе вручную или с помощью мапперов, создается описание класса



1. Применение Micro-ORM, Dapper, требует подключения провайдера данных и подключения пакета Dapper. Обеспечивает большую производительность и легковесность

Применение ADO.Net, алгоритмы:

1. Подключить провайдера используя NuGet
2. Настроить строку подключения
3. Открыть подключение

using SqlConnection connection = new(connectionString);

await connection.OpenAsync();

или

connection.Open();

1. Выполнение SQL запросов

string query = "sql-запрос";

SqlCommand command = new(query, connection);

await command.Execute...Async();

ExecuteNonQuery - DML/DDL

ExecuteScalar - SELECT

ExecuteReader – SELECT

Для избавления от SQL инъекций можно добавить параметры в запрос query и в параметры объекта command

command.Parameters.AddWithValue("@ИмяПараметра", значение);

тип? – допускает null

(int?)значение

**Применение DAPPER**

При создании подключения вместо класса провайдера Connection используется интерфейс IDbConnection, который Dapper дополняет методами Query и Execute, находится в пространстве System.Data.

Алгоритм работы:

1. Создание подключения, в query может быть несколько запросов, параметром может быть объект пользовательского типа или анонимного

SqlConnectionStringBuilder builder = new()

{

DataSource = "PRSERVER\\SQLEXPRESS",

InitialCatalog = "ispp3501",

UserID = "ispp3501",

Password = "3501",

TrustServerCertificate = true

};

using IDbConnection connection = new SqlConnection(builder.ConnectionString);

// создание запроса

string query = "...";

// выполнение запроса

connection.МетодDapper(query, параметры);

await connection.МетодDapperAsync(query, new { ИмяПараметра1 = знач1, ИмяПараметра2 = знач2});

**Методы Dapper:**

* Query<T> - вернет набор строк
* QuerySingleOrDefault<T> - вернёт единственную строку
* QueryFirstOrDefault<T> - вернёт ПЕРВУЮ строку
* Exectute – DML
* ExecuteScalar<T>
* ExecuteMultiple – позволяет вернуть нескольких SELECT

using IDbConnection connection = new SqlConnection(builder.ConnectionString);

// создание запроса

string query = "SELECT \* FROM Category";

var categories = await connection.QueryAsync<Category>(query);

query = "UPDATE Category SET Name = 'РПГ' WHERE Name = @Name";

await connection.ExecuteAsync(query, new { Name = "RPG" });

Console.WriteLine();

query = "INSERT INTO Category(Name) VALUES(@Name); SELECT SCOPE\_IDENTITY()";

Category arcada = new Category { Name="аркада" };

int id = await connection.ExecuteScalarAsync<int>(query, arcada);

Console.WriteLine(id);