**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ**

**ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**Отчеты по лабораторным и практическим работам**

**МДК 11.01**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИСПП-35 | |  |  | *Волков Н.А.* |
|  | (Группа) | | (Подпись) | (Дата) | (И.О. Фамилия) |
| Преподаватель | | |  |  | *Маломан Ю. С.* |
|  | |  | (Подпись) | (Дата) | (И.О. Фамилия) |

Архангельск 2024

# **Лабораторная работа №15**

**Применение ORM для доступа к данным БД**

1. **Цель работы** 
   1. Научиться создавать приложение C# для организации доступа к БД.
   2. Научиться проектировать приложение, использующее паттерн репозиторий и Dapper.
2. **Контрольные вопросы**

Вопрос: Что такое Dapper и для чего используется?

Dapper - это легковесное ORM (Object-Relational Mapping) для .NET Framework и .NET Core. Он используется для упрощения работы с базами данных, особенно при выполнении CRUD-операций (Create, Read, Update, Delete) [1][2].

Основные преимущества Dapper:

- Простота использования

- Высокая производительность

- Поддержка асинхронных операций

- Интеграция с Entity Framework

Вопрос: Какие методы Dapper позволяют извлечь данные из БД?

Основные методы Dapper для извлечения данных:

1. Query<T>(): выполняет SQL-запрос и возвращает список объектов типа T [2].

var users = db.Query<User>("SELECT \* FROM Users");

2. QueryFirstOrDefault<T>(): выполняет SQL-запрос и возвращает первый объект типа T или null, если не найдено [2].

User user = db.QueryFirstOrDefault<User>("SELECT \* FROM Users WHERE Id = @Id", new { Id = 1 });

3. Single<T>(): выполняет SQL-запрос и возвращает первый объект типа T или выбрасывает исключение, если больше одного объекта [2].

User user = db.Single<User>("SELECT \* FROM Users WHERE Id = @Id", new { Id = 1 });

4. QueryMultiple<T1, T2>(...): выполняет несколько SQL-запросов и возвращает списки разных типов [2].

Вопрос: Какие методы Dapper позволяют изменить данные в БД?

Основные методы Dapper для изменения данных:

1. Insert<T>(): вставляет новый объект в базу данных [2].

db.Insert(new User { Name = "John Doe", Age = 30 });

2. Update<T>(): обновляет существующий объект в базе данных [2].

db.Update(user);

3. Delete<T>(): удаляет объект из базы данных [2].

db.Delete(user);

4. BulkInsert<T>(): выполняет быструю вставку нескольких объектов [2].

Вопрос: Зачем используется паттерн «репозиторий»?

Паттерн "репозиторий" используется для абстрагирования доступа к хранилищам данных (например, базы данных) и предоставления единого интерфейса для операций CRUD [3].

Основные преимущества паттерна "репозиория":

- Разделение ответственности: бизнес-логика отделена от операций с данными.

- Повторное использование кода: единый интерфейс для разных типов хранилищ.

- Упрощение тестирования: легко создавать моки репозитория для unit-тестов.

- Гибкость: легко заменять реализацию репозитория без изменения остальной части приложения.

Пример использования паттерна "репозитория" с Dapper:

public interface IUserRepository

{

Task<User> GetByIdAsync(int id);

Task AddAsync(User user);

}

public class UserRepository : IUserRepository

{

private readonly IDbConnection \_db;

public UserRepository(IDbConnection db)

{

\_db = db;

}

public async Task<User> GetByIdAsync(int id)

{

return await \_db.QueryFirstOrDefaultAsync<User>(

"SELECT \* FROM Users WHERE Id = @Id",

new { Id = id });

}

public async Task AddAsync(User user)

{

await \_db.InsertAsync(user);

}

}

1. **Вывод**
   1. В ходе лабораторной работы мы научились создавать приложение C# для организации доступа к БД.
   2. Научились проектировать приложение, использующее паттерн репозиторий и Dapper.