

ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЙ БАЗ ДАННЫХ

В материалах предыдущих практикумов вы получили общее представление о технологии ADO.NET в контексте windows и web- форм, ознакомились с доступом к данным, основанном на соединениях (создание соединений, команд, чтение данных, вызов хранимых процедур). Теперь необходимо собрать описание структуры и код доступа к данным в хорошо спроектированное приложение.

В правильно организованном приложении код доступа к данным никогда не включается непосредственно в отделенный код. Вместо этого он выделяется в *компонент данных*.

Создание компонента данных

Для построения хорошо инкапсулированного, оптимизированного компонента базы данных, можно учитывать следующие рекомендации:

- Открывайте соединение с базой данных при каждом вызове метода и закрывайте перед его завершением.
- Передавайте всю необходимую методу информацию в его параметрах и возвращайте все извлекаемые им данные через возвращаемое значение.
- Каждый запрос должен оптимально выбирать лишь те строки, которые ему действительно нужны, для чего ограничивайте результаты с помощью соответствующих конструкций, например, WHERE, TOP.

Хорошее, прямолинейное проектное решение компонентов баз данных использует отдельный класс для каждой таблицы базы (или логически связанной группы таблиц). Общие методы доступа к данным, такие как вставка, удаление и модификация записей, должны быть помещены в отдельные, не поддерживающие состояние методы. И, наконец, каждое обращение к базе данных должно использовать выделенную хранимую процедуру.

Прежде чем приступить к написанию логики доступа к данным, потребуется создать набор хранимых процедур, необходимых для извлечения, вставки и обновления информации.

Хранимые процедуры

Пример хранимой процедуры для вставки новой строки в таблицу Emp, у которой первичный ключ EmpId генерируется автоматически.

```
CREATE PROCEDURE InsertEmp
    @depId int,
    @fio varchar(50),
    @birthDate date,
    @EmpID int OUTPUT
AS
BEGIN
    INSERT INTO Emp (DepId, FIO, DateOfBirth)
    VALUES (@depId, @fio, @birthDate);
    SET @EmpID = @@IDENTITY;
END
```

Пример хранимой процедуры для удаления строки из таблицы Emp с обеспечением удаления или изменения всех связанных записей.

```

CREATE PROCEDURE DeleteEmp
    @EmpID int
AS
BEGIN
    DELETE FROM Salaries WHERE EmpID = @EmpID;
    DELETE FROM CourseAttendees WHERE EmpID = @EmpID;
    DELETE FROM Payments WHERE EmpID = @EmpID;
    UPDATE Dep Set DepartmentHeadId=null WHERE DepartmentHeadId = @EmpID;
    UPDATE Emp Set ManagerId=null WHERE ManagerId = @EmpID;
    DELETE FROM Emp WHERE EmpID = @EmpID
END

```

Класс данных

Для облегчения перемещения информации в базу данных и обратно имеет смысл создать класс Emp, который представит все поля в виде общедоступных свойств. Ниже приведен его полный код:

```

public class Emp
{
    private int empId;
    private string fio;
    private int depId;
    private DateTime dateOfBirth;

    public int EmpId { get => empId; set => empId = value; }
    public string Fio { get => fio; set => fio = value; }
    public int DepId { get => depId; set => depId = value; }
    public DateTime DateOfBirth { get => dateOfBirth; set => dateOfBirth = value; }
    //Конструкторы класса
    public Emp() { }
    public Emp(int empId, string fio, int depId, DateTime dateOfBirth)
    {
        this.EmpId = empId;
        this.Fio = fio;
        this.DepId = depId;
        this.DateOfBirth = dateOfBirth;
    }
    public Emp(string fio, int depId, DateTime dateOfBirth)
    {
        this.Fio = fio;
        this.DepId = depId;
        this.DateOfBirth = dateOfBirth;
    }
}

```

Служебный класс базы данных

И, наконец, необходим класс, выполняющий операции над данными в базе. Этот класс использует хранимые процедуры, разработанные в базе данных.

В рассматриваемом примере служебный класс называется EmpDB. Он инкапсулирует весь код доступа к данным и специфичные для конкретного использования методов детали. Вот его структура:

```

public class EmpDB
{
    //Строка соединения извлекается при каждом создании экземпляра класса, но не при
    //каждом вызове метода.
    private string connStr;
    //Конструкторы класса
    public EmpDB()...
    public EmpDB(string connectionString)...)
    //Методы для чтения
    public Emp GetEmp(int empId)...)
    public List<Emp> GetEmps()...)
    public SqlCommand cmdGetEmps()...)
    //Методы для вставки новой строки
    public SqlCommand InsertEmp()...)
    public int InsertEmp(Emp emp)...)
    //Методы для обновления строки
    public SqlCommand UpdateEmp()...)
    public void UpdateEmp(Emp emp)...)
    public void UpdateEmp(int EmpId, int DepId, string fio, DateTime birth)...)
    //Методы для удаления строки
    public SqlCommand DeleteEmp()...)
    public void DeleteEmp(int empId)...)
}

```

Тестирование компонента базы данных

Теперь, когда компонент данных создан, понадобится простая тестовая форма или страница, чтобы испытать его. На рисунке 1 пример формы для тестирования всех версий разработанных методов.

winForms

Объекты ado.net Интерфейсы Классы

Пакеты данных

	EmpID	FIO	DepId	DateOfBirth
	1	Петров А.А.	1	14.01.1977
	11	Арбузова Г.Д.	2	01.12.1980
▶	29	Волнушкин Е.О.	3	16.11.1957
	32	Широков Р.Д.	1	03.12.1997
	37	Корнеев В.В.	2	01.01.1990
*				

Таб. номер ФИО Номер отдела Дата рождения

29 Волнушкин Е.О. 3 16 ноября 1957 г.

Добавить новую запись X

Рисунок 1 – Win-форма для проверки компонента данных

```

//Добавить новую запись из текстовых окон
private void button2_Click(object sender, EventArgs e)
{
    EmpDB empDB = new EmpDB();
    textBox1.Text=empDB.InsertEmp(new Emp(textBox2.Text, Convert.ToInt32
        (textBox3.Text), dateTimePicker1.Value)).ToString();
}
//Чтение указанной записи методом GetEmp()
private void dataGridView1_DoubleClick(object sender, EventArgs e)
{
    EmpDB empDB = new EmpDB();
    int i = (int)dataGridView1[dataGridView1.CurrentCellAddress.X,
        dataGridView1.CurrentCellAddress.Y].Value;
    if (dataGridView1.CurrentCell.ColumnIndex == dataGridView1.Columns
        [0].DisplayIndex)
    {
        Emp emp = empDB.GetEmp(i);
        textBox1.Text = emp.EmpId.ToString();
        textBox2.Text = emp.Fio;
        textBox3.Text = emp.DepId.ToString();
        dateTimePicker1.Value = emp.DateOfBirth;
    }
}

```

Задание для самостоятельной работы

В своей базе данных выберите таблицу с разнообразными типами данных: числа, строки, даты, логика, деньги.

Разработайте комплект хранимых процедур для выполнения всех операций по чтению, обновлению, удалению и вставке данных в таблицу.

Разработайте классы для работы с данными таблицы.

Протестируйте все свойства и методы в форме или странице.

В отчете опишите назначение и особенности использования все разработанных элементов.