УДК 004

Реализация CRUD-операций на языке программирования C#

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается процесс создания и использования CRUD-операций (Create, Read, Update, Delete) в приложении Windows на языке программирования C#. Основное внимание уделяется интеграции хранимых процедур SQL Server для управления данными, что обеспечивает повышенную безопасность и оптимизацию производительности.
Ключевые слова:SQL, CRUD, операции, C#, Visual Studio

Implementation of CRUD operations in the C# programming language

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the process of creating and using CRUD operations (Create, Read, Update, Delete) in a Windows application in the C# programming language. The focus is on integrating SQL Server stored procedures for data management, which provides increased security and performance optimization.
Keywords: SQL, CRUD, Operations, C#, Visual Studio

В современной разработке программного обеспечения операции CRUD являются фундаментальными для управления данными в базах данных. Эффективная реализация этих операций в приложениях Windows на языке C# имеет решающее значение для создания надежных и масштабируемых систем. Использование хранимых процедур в SQL Server в качестве механизма для выполнения операций CRUD обеспечивает дополнительный уровень абстракции, безопасности и возможность оптимизации запросов. В статье представлены современные методы и техники, которые позволяют разработчикам эффективно интегрировать эти операции в свои приложения, что делает её актуальной для широкого круга специалистов в области программирования.

Целью данной статьи является реализация CRUD-операций в среде разработки «Visual Studio» на языке программирования C#.

В своей работе Н. Н. Додобоев, О. И. Кукарцева, Я. А. Тынченко рассмотрели вопросы появления различных языков программирования (в частности C#), определения особенностей этих языков, а также составления

основных видов и классификаций языков программирования [1]. З. С. Магомадова рассмотрела языки программирования высокого уровня, особенности, недостатки и сложности в изучении, а также описала несколько легких алгоритмов [2]. В. Е. Бажан в своей работе описал применение документно - ориентированных баз данных в CRUD - приложениях. Рассматриваются преимущества и недостатки документно - ориентированных баз данных по сравнению с реляционными [3]. А. А. Рыбанов рассмотрел реализацию CRUD-модели безопасности на основе технологии ROLAP. Предлагаемый подход ориентирован на процессы управления доступом в рамках технологий разработки и защиты баз данных [4].

Создаем проект «Windows Forms App» и называем необходимым именем см. рисунок 1.

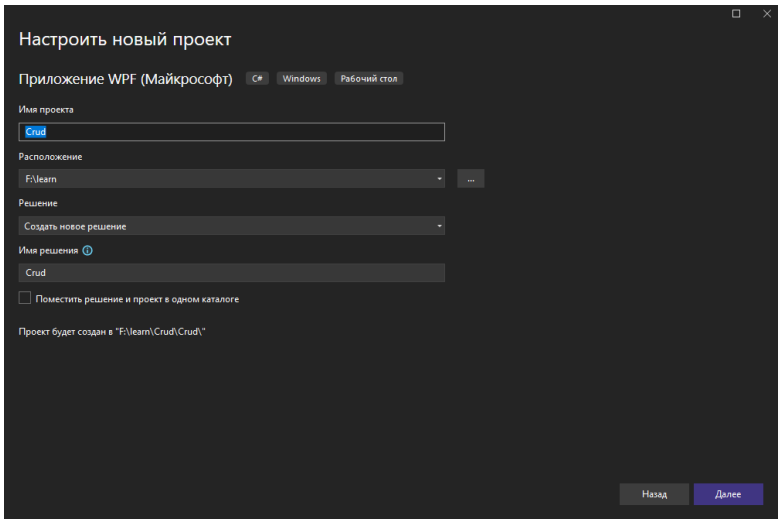

Рис. 1. Создание проекта

Далее заполняем форму необходимыми полями, добавляем: поля ввода данных о сотрудниках, кнопки для поиска, сохранения, обновления, удаления данных о сотрудниках, компонент для вывода таблицы см. рисунке 2.

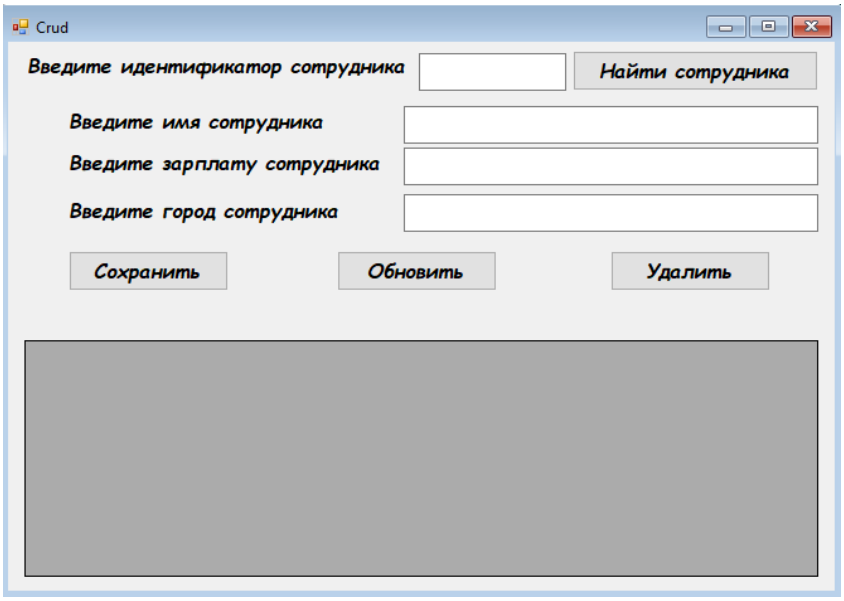

Рис. 2. Компоненты формы

Теперь дважды щелкаем в любом месте формы, чтобы сгенерировать событие Form_Load. Заменяем сгенерированный код на код события, а также импортируем пространство имен System.Data.SqlClient. Также отключаем кнопки обновления и удаления при загрузке, кнопки будут включаться, когда пользователь получит запись об одном сотруднике, нажав на кнопку поиска сотрудника см. рисунок 3.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CrudOperationUsingStoreProcedure
{
    Ссылок: 3
    public partial class Form1 : Form
    {
        Ссылок: 1
        public Form1()
        {
            InitializeComponent();
        }

        SqlConnection cn;
        SqlCommand cmd;
        SqlDataAdapter da;
        SqlDataReader dr;
        Ссылок: 1
        private void Form1_Load(object sender, EventArgs e)
        {
            cn = new SqlConnection(@"Data Source=(local);Initial Catalog=Tutorials;Integrated Security=True");
            cn.Open();
            GetAllEmployeeRecord();

            btnUpdate.Enabled = false;
            btnDelete.Enabled = false;
        }
```

Рис. 3. Событие Form_Load

Теперь создаем метод для получения всех данных из таблицы и установки в представлении сетки данных см. рисунок 4.

```
private void GetAllEmployeeRecord()
{
    cmd = new SqlCommand("EmployeeCrudOperation", cn);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.AddWithValue("@Employeeid", 0);
    cmd.Parameters.AddWithValue("@EmployeeName", "");
    cmd.Parameters.AddWithValue("@EmployeeSalary", 0);
    cmd.Parameters.AddWithValue("@EmployeeCity", "");
    cmd.Parameters.AddWithValue("@OperationType", "5");
    da = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView1.DataSource = dt;

}
```

Рис. 4. Метод GetAllEmployeeRecord

Далее создаем метод сохранения путем двойного щелчка по кнопке сохранения см. рисунок 5.

```
private void Btnsave_Click(object sender, EventArgs e)
{
    if (txtempcity.Text != string.Empty  && txtempname.Text != string.Empty && txtempsalary.Text != string.Empty)
    {
        cmd = new SqlCommand("EmployeeCrudOperation", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@Employeeid", 0);
        cmd.Parameters.AddWithValue("@EmployeeName", txtempname.Text);
        cmd.Parameters.AddWithValue("@EmployeeSalary", txtempsalary.Text);
        cmd.Parameters.AddWithValue("@EmployeeCity", txtempcity.Text);
        cmd.Parameters.AddWithValue("@OperationType", "1");
        cmd.ExecuteNonQuery();
        MessageBox.Show("Record inserted successfully.", "Record Inserted", MessageBoxButtons.OK, MessageBoxIcon.Information);
        GetAllEmployeeRecord();
        txtempcity.Text = "";
        txtempid.Text = "";
        txtempname.Text = "";
        txtempsalary.Text = "";
    }
    else
    {
        MessageBox.Show("Please enter value in all fields", "Invalid Data", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

Рис. 5. Метод Btnsave_Click

Генерируем событие щелчка по кнопке поиска сотрудника, чтобы получить запись об одном сотруднике, передав идентификатор, и отобразив данные в текстовом поле см. рисунок 6.

```
private void Btnfind_Click(object sender, EventArgs e)
{
    if (txtempid.Text != string.Empty)
    {

        cmd = new SqlCommand("EmployeeCrudOperation", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@Employeeid", txtempid.Text);
        cmd.Parameters.AddWithValue("@EmployeeName", "");
        cmd.Parameters.AddWithValue("@EmployeeSalary", 0);
        cmd.Parameters.AddWithValue("@EmployeeCity", "");
        cmd.Parameters.AddWithValue("@OperationType", "4");
        dr = cmd.ExecuteReader();
        if (dr.Read())
        {
            txtempname.Text = dr["EmployeeName"].ToString();
            txtempsalary.Text = dr["EmployeeSalary"].ToString();
            txtempcity.Text = dr["EmployeeCity"].ToString();
            btnUpdate.Enabled = true;
            btnDelete.Enabled = true;
        }
        else
        {
            MessageBox.Show("No record found with this id", "No Data Found", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        dr.Close();
    }
    else
    {
        MessageBox.Show("Please enter employee id ", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Рис. 6. Метод Btnfind_Click

Создаем метод BtnUpdate_Click запускающий событие нажатия на кнопку обновления см. рисунок 7.

```
private void BtnUpdate_Click(object sender, EventArgs e)
{
    if (txtempcity.Text != string.Empty && txtempid.Text != string.Empty && txtempname.Text != string.Empty && txtempsalary.Text != string.Empty)
    {
        cmd = new SqlCommand("EmployeeCrudOperation", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@Employeeid", txtempid.Text);
        cmd.Parameters.AddWithValue("@EmployeeName", txtempname.Text);
        cmd.Parameters.AddWithValue("@EmployeeSalary", txtempsalary.Text);
        cmd.Parameters.AddWithValue("@EmployeeCity", txtempcity.Text);
        cmd.Parameters.AddWithValue("@OperationType", "2");
        cmd.ExecuteNonQuery();
        MessageBox.Show("Record update successfully.", "Record Updated", MessageBoxButtons.OK, MessageBoxIcon.Information);
        GetAllEmployeeRecord();
        btnDelete.Enabled = false;
        btnUpdate.Enabled = false;
    }
    else
    {
        MessageBox.Show("Please enter value in all fields", "Invalid Data", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

Рис. 7. Метод BtnUpdate_Click

Теперь создадим событие щелчка по кнопке удаления см. рисунок 8.

```
private void BtnDelete_Click(object sender, EventArgs e)
{
    if (txtempid.Text != string.Empty)
    {
        DialogResult dialogResult = MessageBox.Show("Are you sure you want to delete this employee ? ",
            "Delete Employee", MessageBoxButtons.YesNo, MessageBoxIcon.Asterisk);
        if (dialogResult == DialogResult.Yes)
        {

            cmd = new SqlCommand("EmployeeCrudOperation", cn);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@Employeeid", txtempid.Text);
            cmd.Parameters.AddWithValue("@EmployeeName", "");
            cmd.Parameters.AddWithValue("@EmployeeSalary", 0);
            cmd.Parameters.AddWithValue("@EmployeeCity", "");
            cmd.Parameters.AddWithValue("@OperationType", "3");
            cmd.ExecuteNonQuery();
            MessageBox.Show("Record deleted successfully.", "Record Deleted", MessageBoxButtons.OK, MessageBoxIcon.Information);
            GetAllEmployeeRecord();
            txtempcity.Text = "";
            txtempid.Text = "";
            txtempname.Text = "";
            txtempsalary.Text = "";
            btnDelete.Enabled = false;
            btnUpdate.Enabled = false;
        }
    }
    else
    {
        MessageBox.Show("Please enter employee id", "Invalid Data", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

Рис. 8. Метод BtnDelete_Click

Проверяем работу программу см. рисунок 9-11.

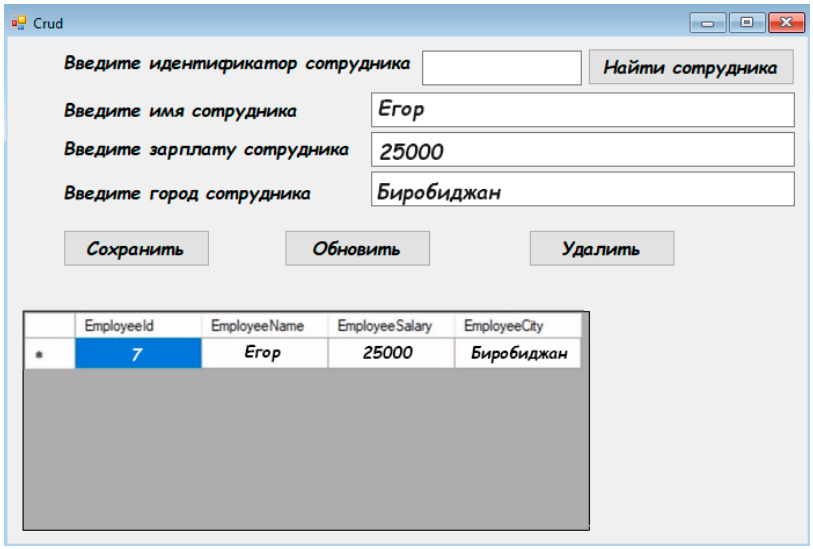

Рис. 9. Добавление сотрудника

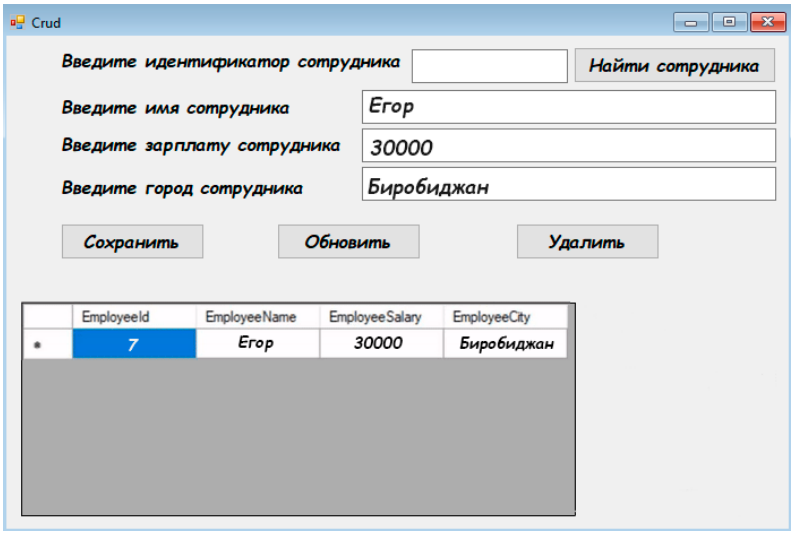

Рис. 10. Обновление данных сотрудника

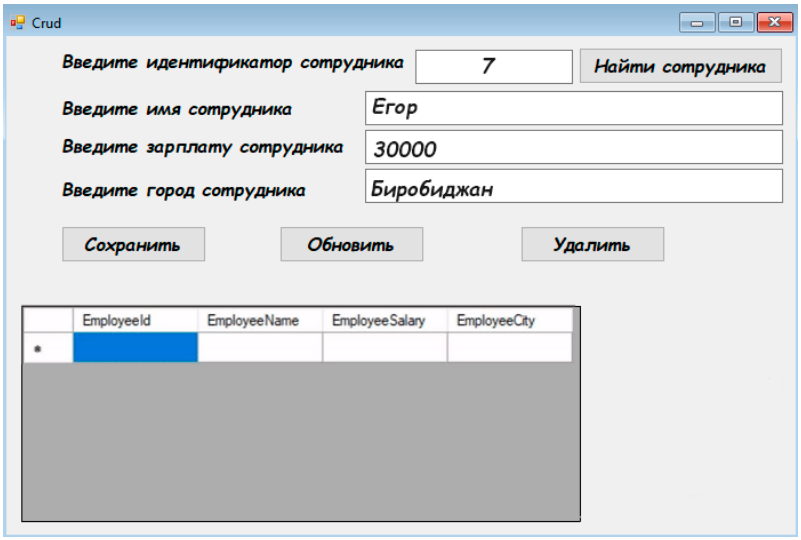

Рис. 11. Удаление сотрудника

Статья предоставляет обзор реализации CRUD-операций в приложениях Windows на языке C#. Описанные методы и подходы позволяют разработчикам создавать эффективные и безопасные системы управления данными. Применение хранимых процедур SQL Server значительно улучшает производительность и обеспечивает высокий уровень безопасности при работе с базами данных.

Библиографический список

1. Додобоев Н. Н., Кукарцева О. И., Тынченко Я. А. Современные языки программирования // Современные технологии: актуальные вопросы, достижения и инновации. 2014. №5. С. 81-85.
2. Магомадова З. С. Языки программирования высокого уровня // Разработка и применение наукоёмких технологий в эпоху глобальных трансформаций. 2020. №8. С. 94-96.
3. Бажан, В. Е. Применение документно-ориентированных баз данных в CRUD-приложениях // Молодежь. Наука. Инновации. 2021. Т. 1. С. 259-261.
4. Рыбанов, А. А. CRUD-модель безопасности как базовый элемент контроля и управления доступом к данным в информационных системах // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки. 2022. № 4(311). С. 45-51.