КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. А.Н.Туполева - КАИ

(КНИТУ-КАИ)

Отделение среднего профессионального образования института

Компьютерных технологий и защиты информации

«Колледж информационных технологий»

Лабораторная работа №5

по предмету:

Наладчик технологического оборудования

Выполнил:

студент 4 курса

Группы 4441

Зарипов Ислам Радифович

Казань 2019 г.

**Цель работы:**

Разработка программы позволяющий пользователю создавать, редактировать, сохранять документы формата «XLS».

**Решение:**

Были установлены слушатели на кнопки «Новый», «Открыть», «Сохранить», «Закрыть». При нажатии на копку «Новый» в памяти инициализируется пустой документ, открывается возможность редактировать документ (см. Рис. 3). Если было вызвано событие открытия файла, то отображается окно выбора файла (см. Рис. 4), затем – содержание файла можно редактировать в текстовом поле программы. После изменения, файл можно сохранить, нажав на кнопку «Сохранить». Так же файл можно закрыть нажатие кнопки «Закрыть». Предусмотрена проверка изменения файла при закрытии файла (см. Рис. 5). Для чтения данных из .xls фала была выбрана библиотека OpenXML, подключение через NuGet:

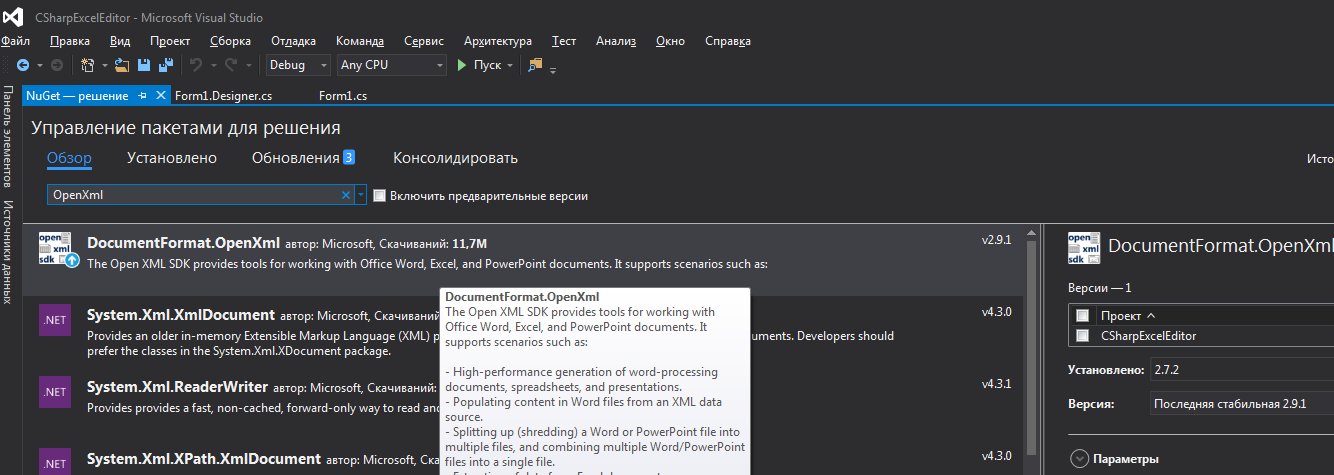


Рис. 1. Установка библиотеки NuGet

**Скриншоты работы программы на C#:**

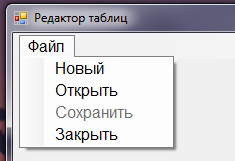
****

Рис. 2. Контекстное меню программы

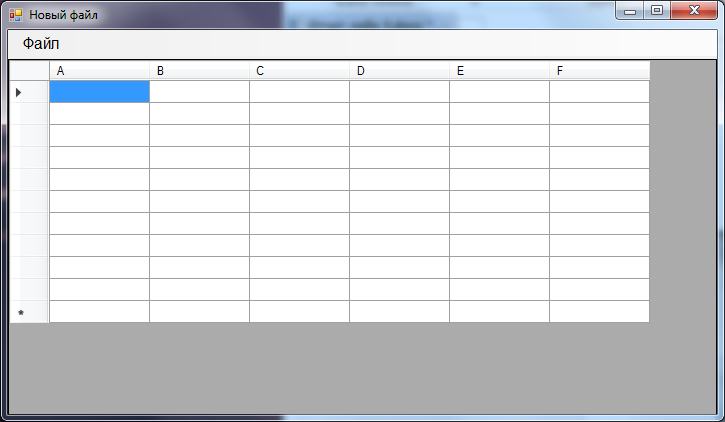


Рис. 3. Окно редактирования нового файла

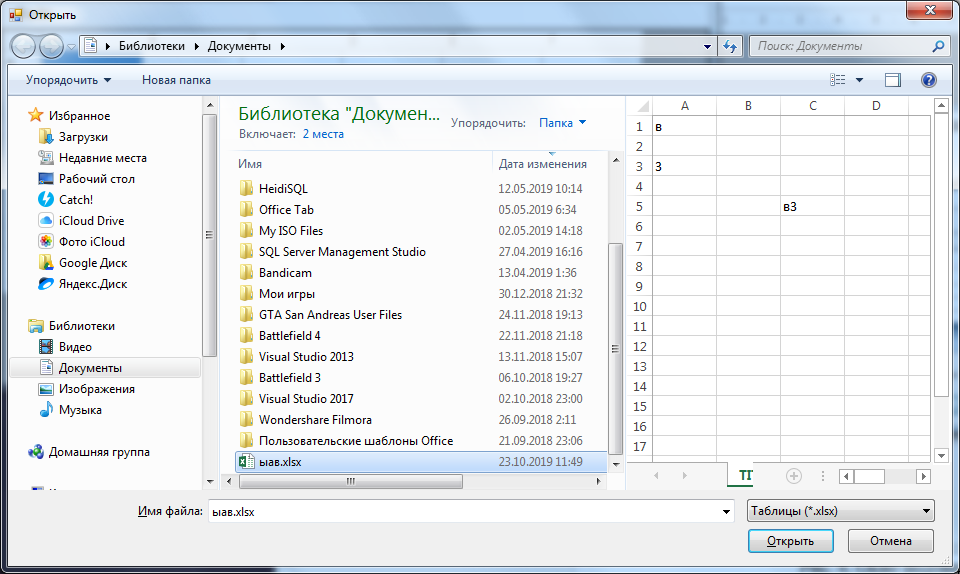


Рис. 4. Окно выбора файла для открытия

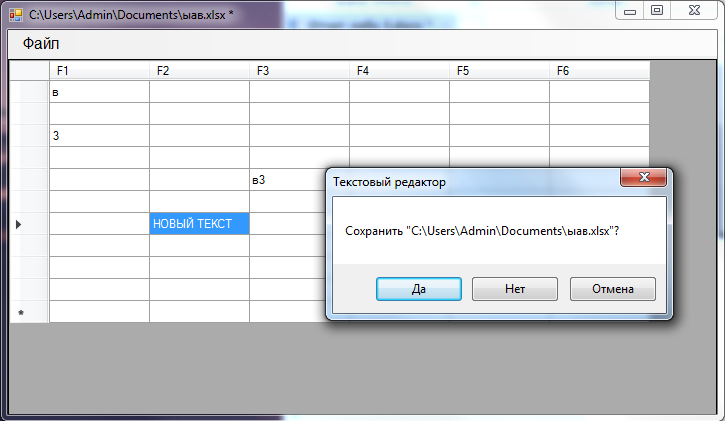


Рис. 5. Окно подтверждения «сохранить или нет»

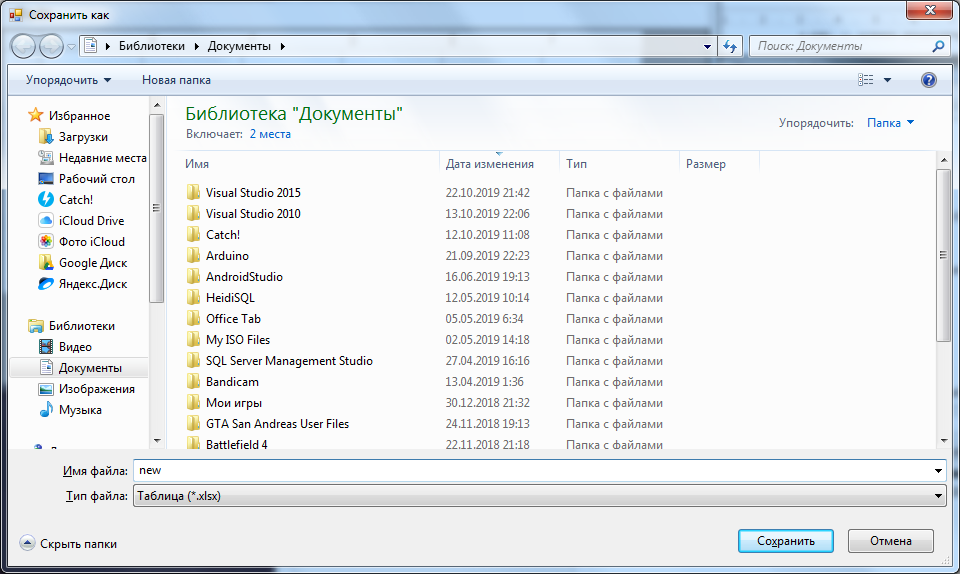


Рис. 6. Окно сохранения нового файла

**Вывод:**

В ходе лабораторной работы была разработана программа, позволяющая пользователю создавать, редактировать, сохранять документы формата «XLS».

**Приложение**

**Файл Form1.cs:**

using ClosedXML.Excel;

using DocumentFormat.OpenXml;

using DocumentFormat.OpenXml.Packaging;

using DocumentFormat.OpenXml.Spreadsheet;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CSharpExcelEditor

{

public partial class Form1 : Form

{

string currentFilePath = null;

string loadedFileText = null;

bool edited = false;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

this.Resize += new EventHandler(Form1\_OnResize);

this.FormClosing += Form1\_FormClosing;

dataGrid.CellEndEdit += onCellEditEnding;

dataGrid.Visible = false;

}

private void onCellEditEnding(object sender, DataGridViewCellEventArgs e)

{

if (currentFilePath == null)

this.Text = "Новый файл \*";

else

this.Text = currentFilePath + " \*";

edited = true;

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

if (!checkIsSaved()) e.Cancel = true;

}

private void Form1\_OnResize(object sender, System.EventArgs e)

{

dataGrid.Size = new Size(this.Size.Width - 16, this.Size.Height - 65);

}

private void newMenuItem\_Click(object sender, EventArgs e)

{

if (!checkIsSaved()) return;

currentFilePath = null;

loadedFileText = "";

initTable(null);

this.Text = "Новый файл";

edited = false;

dataGrid.Visible = true;

saveMenuItem.Enabled = true;

}

private DataTable readExcelToTable(string path)

{

string connstring = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + path + ";Extended Properties='Excel 8.0;HDR=NO;IMEX=1';";

using (OleDbConnection conn = new OleDbConnection(connstring))

{

conn.Open();

DataTable sheetsName = conn.GetOleDbSchemaTable(OleDbSchemaGuid.Tables, new object[] { null, null, null, "Table" });

string firstSheetName = sheetsName.Rows[0][2].ToString();

string sql = string.Format("SELECT \* FROM [{0}]", firstSheetName);

OleDbDataAdapter ada = new OleDbDataAdapter(sql, connstring);

DataSet set = new DataSet();

ada.Fill(set);

return set.Tables[0];

}

}

private void initTable(String file)

{

if (file != null)

{

DataTable dt = readExcelToTable(file);

dataGrid.DataSource = dt;

} else

{

DataTable dt = new DataTable();

dt.Columns.Add("A");

dt.Columns.Add("B");

dt.Columns.Add("C");

dt.Columns.Add("D");

dt.Columns.Add("E");

dt.Columns.Add("F");

for (int i = 0; i < 10; i++)

{

DataRow r = dt.NewRow();

r["A"] = ""; r["B"] = ""; r["C"] = ""; r["D"] = ""; r["E"] = ""; r["F"] = "";

dt.Rows.Add(r);

}

dataGrid.DataSource = dt;

}

dataGrid.Visible = true;

}

private void exportDataSet(DataTable table, string destination)

{

using (var workbook = SpreadsheetDocument.Create(destination, DocumentFormat.OpenXml.SpreadsheetDocumentType.Workbook))

{

var workbookPart = workbook.AddWorkbookPart();

workbook.WorkbookPart.Workbook = new DocumentFormat.OpenXml.Spreadsheet.Workbook();

workbook.WorkbookPart.Workbook.Sheets = new DocumentFormat.OpenXml.Spreadsheet.Sheets();

var sheetPart = workbook.WorkbookPart.AddNewPart<WorksheetPart>();

var sheetData = new DocumentFormat.OpenXml.Spreadsheet.SheetData();

sheetPart.Worksheet = new DocumentFormat.OpenXml.Spreadsheet.Worksheet(sheetData);

DocumentFormat.OpenXml.Spreadsheet.Sheets sheets = workbook.WorkbookPart.Workbook.GetFirstChild<DocumentFormat.OpenXml.Spreadsheet.Sheets>();

string relationshipId = workbook.WorkbookPart.GetIdOfPart(sheetPart);

uint sheetId = 1;

if (sheets.Elements<DocumentFormat.OpenXml.Spreadsheet.Sheet>().Count() > 0)

{

sheetId =

sheets.Elements<DocumentFormat.OpenXml.Spreadsheet.Sheet>().Select(s => s.SheetId.Value).Max() + 1;

}

DocumentFormat.OpenXml.Spreadsheet.Sheet sheet = new DocumentFormat.OpenXml.Spreadsheet.Sheet() { Id = relationshipId, SheetId = sheetId, Name = table.TableName };

sheets.Append(sheet);

DocumentFormat.OpenXml.Spreadsheet.Row headerRow = new DocumentFormat.OpenXml.Spreadsheet.Row();

List<String> columns = new List<string>();

foreach (System.Data.DataColumn column in table.Columns)

{

columns.Add(column.ColumnName);

DocumentFormat.OpenXml.Spreadsheet.Cell cell = new DocumentFormat.OpenXml.Spreadsheet.Cell();

cell.DataType = DocumentFormat.OpenXml.Spreadsheet.CellValues.String;

cell.CellValue = new DocumentFormat.OpenXml.Spreadsheet.CellValue(column.ColumnName);

headerRow.AppendChild(cell);

}

sheetData.AppendChild(headerRow);

foreach (System.Data.DataRow dsrow in table.Rows)

{

DocumentFormat.OpenXml.Spreadsheet.Row newRow = new DocumentFormat.OpenXml.Spreadsheet.Row();

foreach (String col in columns)

{

DocumentFormat.OpenXml.Spreadsheet.Cell cell = new DocumentFormat.OpenXml.Spreadsheet.Cell();

cell.DataType = DocumentFormat.OpenXml.Spreadsheet.CellValues.String;

cell.CellValue = new DocumentFormat.OpenXml.Spreadsheet.CellValue(dsrow[col].ToString());

newRow.AppendChild(cell);

}

sheetData.AppendChild(newRow);

}

}

}

private void openMenuItem\_Click(object sender, EventArgs e)

{

if (!checkIsSaved()) return;

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.InitialDirectory = ".";

openFileDialog.Filter = "Таблицы (\*.xlsx)|\*.xlsx";

openFileDialog.FilterIndex = 0;

openFileDialog.RestoreDirectory = true;

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

string selectedFileName = openFileDialog.FileName;

string readText = File.ReadAllText(selectedFileName);

currentFilePath = selectedFileName;

this.Text = currentFilePath;

loadedFileText = readText;

initTable(selectedFileName);

dataGrid.Enabled = true;

saveMenuItem.Enabled = true;

}

}

private void saveMenuItem\_Click(object sender, EventArgs e)

{

saveCurrentFile(true);

}

// return: true - продолжить; false - отмена

private bool checkIsSaved()

{

if (!edited) return true;

string what = currentFilePath == null ? "новый файл" : "\"" + currentFilePath + "\"";

DialogResult dialogResult = MessageBox.Show("Сохранить " + what + "?", "Текстовый редактор", MessageBoxButtons.YesNoCancel);

if (dialogResult == DialogResult.Yes)

{

saveCurrentFile(false);

return true;

}

else if (dialogResult == DialogResult.No)

return true;

else

return false;

}

private void saveCurrentFile(bool andOpen)

{

if (currentFilePath != null)

{

exceldata(currentFilePath);

this.Text = currentFilePath;

}

else

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.InitialDirectory = ".";

saveFileDialog.Filter = "Таблица (\*.xlsx)|\*.xlsx";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

string filename = saveFileDialog.FileName;

currentFilePath = filename;

exceldata(filename);

}

}

edited = false;

}

public void exceldata(String docName)

{

using (SpreadsheetDocument spreadsheetDocument = SpreadsheetDocument.Create(docName, SpreadsheetDocumentType.Workbook))

{

WorkbookPart workbookpart = spreadsheetDocument.AddWorkbookPart();

workbookpart.Workbook = new Workbook();

WorksheetPart worksheetPart = workbookpart.AddNewPart<WorksheetPart>();

SheetData sheetData = new SheetData();

worksheetPart.Worksheet = new Worksheet(sheetData);

Sheets sheets = spreadsheetDocument.WorkbookPart.Workbook.

AppendChild<Sheets>(new Sheets());

Sheet sheet = new Sheet()

{

Id = spreadsheetDocument.WorkbookPart.

GetIdOfPart(worksheetPart),

SheetId = 1,

Name = "TITLE"

};

for (int i = 0; i < dataGrid.Rows.Count - 1; i++)

{

UInt32Value v = new UInt32Value((uint)(i + 1));

Row row = new Row() { RowIndex = v };

for (int j = 0; j < dataGrid.Columns.Count; j++)

{

char c = (char)(j + 65);

string cellRef = c + "" + v;

Cell cell = new Cell() { CellReference = cellRef, CellValue = new CellValue(dataGrid.Rows[i].Cells[j].Value.ToString()), DataType = CellValues.String };

row.Append(cell);

}

sheetData.Append(row);

}

sheets.Append(sheet);

workbookpart.Workbook.Save();

spreadsheetDocument.Close();

}

}

private void closeMenuItem\_Click(object sender, EventArgs e)

{

if (!checkIsSaved()) return;

this.Text = "Текстовый редактор";

currentFilePath = null;

loadedFileText = null;

dataGrid.Visible = false;

saveMenuItem.Enabled = false;

edited = false;

}

}

}