**Шпаргалка: Импорт данных в WPF-приложении (аналог import\_data.py, но на C# и WPF)**

Если вам нужно реализовать импорт данных из Excel в WPF-приложении (аналог import\_data.py), вот пошаговая инструкция с кодом.

**1. Подготовка проекта**

**Требования:**

* Visual Studio (2019 или выше).
* Установленные NuGet-пакеты:
  + Microsoft.EntityFrameworkCore.Sqlite (для работы с SQLite).
  + ExcelDataReader (для чтения Excel-файлов).
  + System.Data.SQLite (драйвер SQLite).

**Структура проекта:**

Copy

Download

/WpfApp

│

├── /Data

│ ├── Materials\_import.xlsx

│ ├── Material\_type\_import.xlsx

│ ├── ...

│

├── /Models

│ ├── Material.cs

│ ├── MaterialType.cs

│ ├── Product.cs

│ ├── ...

│

├── /Services

│ ├── DatabaseService.cs

│ ├── ExcelImporter.cs

│

├── MainWindow.xaml

└── MainWindow.xaml.cs

**2. Модели данных (C# классы)**

**Material.cs**

csharp

Copy

Download

public class Material

{

public int Id { get; set; }

public string Name { get; set; }

public int TypeId { get; set; }

public decimal Price { get; set; }

public string Unit { get; set; }

public double QuantityInPack { get; set; }

public double QuantityInStock { get; set; }

public double MinQuantity { get; set; }

public MaterialType Type { get; set; } // Навигационное свойство

}

**MaterialType.cs**

csharp

Copy

Download

public class MaterialType

{

public int Id { get; set; }

public string Name { get; set; }

public double LossPercentage { get; set; }

public List<Material> Materials { get; set; } // Навигационное свойство

}

**Product.cs**

csharp

Copy

Download

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public string Article { get; set; }

public int TypeId { get; set; }

public decimal MinPartnerPrice { get; set; }

public ProductType Type { get; set; } // Навигационное свойство

}

**MaterialProduct.cs (связь многие-ко-многим)**

csharp

Copy

Download

public class MaterialProduct

{

public int MaterialId { get; set; }

public int ProductId { get; set; }

public double RequiredQuantity { get; set; }

public Material Material { get; set; }

public Product Product { get; set; }

}

**3. Сервис работы с базой данных (DatabaseService.cs)**

csharp

Copy

Download

using Microsoft.EntityFrameworkCore;

public class AppDbContext : DbContext

{

public DbSet<Material> Materials { get; set; }

public DbSet<MaterialType> MaterialTypes { get; set; }

public DbSet<Product> Products { get; set; }

public DbSet<ProductType> ProductTypes { get; set; }

public DbSet<MaterialProduct> MaterialProducts { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder options)

=> options.UseSqlite("Data Source=materials.db");

}

public class DatabaseService

{

private readonly AppDbContext \_db;

public DatabaseService()

{

\_db = new AppDbContext();

\_db.Database.EnsureCreated(); // Создаёт БД, если её нет

}

// Метод для сохранения списка материалов

public void SaveMaterials(List<Material> materials)

{

\_db.Materials.AddRange(materials);

\_db.SaveChanges();

}

// Аналогично для других сущностей...

}

**4. Импорт данных из Excel (ExcelImporter.cs)**

csharp

Copy

Download

using ExcelDataReader;

using System.Data;

using System.IO;

public class ExcelImporter

{

public List<Material> ImportMaterials(string filePath)

{

var materials = new List<Material>();

using (var stream = File.Open(filePath, FileMode.Open, FileAccess.Read))

using (var reader = ExcelReaderFactory.CreateReader(stream))

{

var result = reader.AsDataSet();

var table = result.Tables[0]; // Первый лист Excel

// Пропускаем заголовок (первую строку)

for (int i = 1; i < table.Rows.Count; i++)

{

var row = table.Rows[i];

materials.Add(new Material

{

Name = row[0].ToString(),

TypeId = GetMaterialTypeId(row[1].ToString()), // Получаем ID типа материала

Price = decimal.Parse(row[2].ToString()),

Unit = row[6].ToString(),

QuantityInPack = double.Parse(row[5].ToString()),

QuantityInStock = double.Parse(row[3].ToString()),

MinQuantity = double.Parse(row[4].ToString())

});

}

}

return materials;

}

private int GetMaterialTypeId(string typeName)

{

// Здесь должна быть логика получения ID типа материала из БД

// Например:

using (var db = new AppDbContext())

{

var type = db.MaterialTypes.FirstOrDefault(t => t.Name == typeName);

return type?.Id ?? 0;

}

}

}

**5. Пример интерфейса в WPF (MainWindow.xaml)**

xml

Copy

Download

Run

<Window x:Class="WpfApp.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Импорт данных из Excel" Height="450" Width="800">

<Grid>

<Button Content="Импорт материалов"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Click="ImportMaterials\_Click"/>

</Grid>

</Window>

**MainWindow.xaml.cs**

csharp

Copy

Download

public partial class MainWindow : Window

{

private readonly DatabaseService \_dbService;

private readonly ExcelImporter \_excelImporter;

public MainWindow()

{

InitializeComponent();

\_dbService = new DatabaseService();

\_excelImporter = new ExcelImporter();

}

private void ImportMaterials\_Click(object sender, RoutedEventArgs e)

{

try

{

var openFileDialog = new OpenFileDialog

{

Filter = "Excel Files|\*.xlsx;\*.xls",

Title = "Выберите файл с материалами"

};

if (openFileDialog.ShowDialog() == true)

{

var materials = \_excelImporter.ImportMaterials(openFileDialog.FileName);

\_dbService.SaveMaterials(materials);

MessageBox.Show("Данные успешно импортированы!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

**6. Итог**

1. **База данных** создаётся автоматически при запуске (AppDbContext).
2. **Excel-файлы** читаются с помощью ExcelDataReader.
3. **Данные сохраняются** в SQLite через Entity Framework Core.
4. **Интерфейс WPF** позволяет выбрать файл и запустить импорт.

**Что можно улучшить?**

* Добавить прогресс-бар для отображения хода импорта.
* Реализовать валидацию данных перед сохранением.
* Добавить логирование ошибок.

Если нужно что-то доработать — пишите! 🚀