# Практическая работа №23 Передача и обработка данных

1. Цель работы
   1. Научиться выполнять обработку данных в приложениях Avalonia UI.
   2. Научиться передавать данные между частями приложения при помощи внедрения зависимостей в приложениях Avalonia UI.
2. Литература
   1. Avalonia documentation – Текст : электронный // AvaloniaUI, 2024. – URL: https://docs.avaloniaui.net/
3. Подготовка к работе
   1. Повторить теоретический материал (см.п.2).
   2. Изучить описание лабораторной работы.
4. Основное оборудование
   1. Персональный компьютер.
5. Задание
   1. Конфигурация приложения
      1. Установите пакеты:

Microsoft.Extensions.Configuration,

Microsoft.Extensions.Configuration.Json,

* + 1. В файле App.axaml.cs добавьте свойство

public static IConfiguration Configuration { get; private set; }

* + 1. Добавьте в проект файл appsettings.json

{

"ApiKeys": {

"SomeApi": "YourApiKey"

}

}

* + 1. Создайте в файле App.axaml.cs ConfigurationBuilder для настроек вашего приложения

ConfigurationBuilder builder = new ConfigurationBuilder();

//добавляем файл с настройками

builder.AddJsonFile("appsettings.json");

Configuration = builder.Build();

* + 1. При помощи обращения к конфигурации приложения отобразите данные из конфигурации на странице приложения

var key = App.Configuration.GetSection("ApiKeys")["SomeApi"];

* 1. Внедрение зависимостей
     1. Установите пакет Microsoft.Extensions.DependencyInjection
     2. В файле App.axaml.cs добавьте свойство

public static ServiceProvider Services { get; private set; }

* + 1. Создайте в файле App.axaml.cs ServiceCollection для вашего приложения

var collection = new ServiceCollection();

//Добавить сервисы

Services = collection.BuildServiceProvider();

* 1. Модифицировать сервис навигации из ПР№3 с использованием внедрения зависимостей
     1. Добавить в сервис навигации наследование от ObservableObject, а также вызовы OnPropertyChanged() для свойства текущей вершины стека навигации при вызове методов перехода между страницами.
     2. Добавить перегруженный метод, получающий страницу для навигации через внедрение зависимостей с помощью GetRequiredService()

public void NavigateTo<T>(Action<T>? action = null) where T : ViewModelBase

{

var viewModel = App.Services.GetRequiredService<T>();

NavigateTo(viewModel, action);

}

* + 1. Зарегистрируйте сервис навигации и страницы в приложении в коллекции сервисов

//Singleton – один на все приложение

collection.AddSingleton<NavigationService>();

collection.AddSingleton<MainViewModel>();

//Transient – создается новый при каждом обращении

collection.AddTransient<RegistrationViewModel>();

collection.AddTransient<AuthorizationViewModel>();

* 1. Применение внедрения зависимостей
     1. В конструкторы RegistrationViewModel и AuthorizationViewModel добавьте параметр NavigationService navigation, значение параметра присвойте приватному полю класса. Аналогичные действия выполните в MainViewModel.
     2. На страницах расположите кнопки для перехода от одной страницы к другой, к ним необходимо привязать команды, которые будут вызывать метод навигации. В параметры передайте значение свойства для настройки свойств целевой страницы.

\_navigation.NavigateTo<AuthorizationViewModel>(

x=>x.Login = this.Login);

* + 1. В MainViewModel измените структуру отображения страницы

public ViewModelBase CurrentPage => navigation.Current;

* + 1. В конструкторе добавьте подписку на событие PropertyChanged сервиса навигации, и соответствующий обработчик, который будет вызывать обновление значения CurrentPage

navigation.PropertyChanged += navigation\_PropertyChanged;

private void navigation\_PropertyChanged(object? sender, PropertyChangedEventArgs e)

{

OnPropertyChanged(nameof(CurrentPage));

}

* 1. Конверторы значений
     1. Для кастомизации отображения и преобразования привязанных данных используются конверторы значений. Создайте новый класс CelsiusToFarenhateConverter, наследующий IValueConverter

public class CelsiusToFarenhateConverter : IValueConverter

{

public object? Convert(object? value, Type targetType, object? parameter, CultureInfo culture)

{

if (value == null)

return null;

return value; //преобразуйте value из градусов C в градусы F

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

return value; //преобразуйте value из градусов F в градусы C

}

}

* + 1. Для применения конвертера добавьте на страницу

<UserControl.Resources>

<local:CelsiusToFarenhateConverter x:Key="CelsiusToFarenhate"/>

</UserControl.Resources>

Применение в привязке:

{Binding Celsius, Converter={StaticResource CelsiusToFarenhate}}

* + 1. Проверьте работу конвертора

1. Порядок выполнения работы
   1. Выполнить все задания из п.5.
   2. Ответить на контрольные вопросы.
2. Содержание отчета
   1. Титульный лист
   2. Цель работы
   3. Ответы на контрольные вопросы
   4. Вывод
3. Контрольные вопросы
   1. Для чего необходимо внедрение зависимостей
   2. Как использовать конфигурацию IConfiguration
   3. Как получить сервис из коллекции сервисов.
   4. Для чего применяются конверторы значений