Лабораторна робота №7. Розробка сторінкового/віконного WPF-застосунка

Мета : Ознайомитися з основами розробки сторінки WPF-застосунку, що використовує базу даних.

Теми завдань

Рекомендується продовжити тему, що була обрана при виконанні попереднього творчого завдання - в лабораторній роботі №6.

Завдання 1. Освоїти основи розробки сторінкового інтерфейсу

Основні теоретичні відомості

Створення сторінкового додатку

Щоб надати розробникам можливість створювати настільні програми у стилі веб-додатків, до складу WPF була включена власна система посторінкової навігації, яка є напрочуд гнучкою за своєю природою.

У WPF при створенні сторінкових програм контейнером найвищого рівня можуть бути об'єкти ***Frame***, що знаходяться всередині іншого вікна або іншої сторінки.

Для вставки сторінок (***Page***) програми у вікно можна використовувати властивість ***Source*** класу ***Frame***:

<Frame Name="frame1" Margin="3" Source="PageMain.xaml"/>

Клас ***Page*** допускає наявність лише одного вкладеного елемента та має набір властивостей, які дозволяють налаштовувати його зовнішній вигляд, взаємодіяти з контейнером та використовувати навігацію.

Для додавання сторінки в проект необхідно в **браузері рішень** клацнути правою кнопкою миші на ім'я проекту і в контекстному меню вибрати **Додати-Сторінка**. Після цього у вікні додавання нового елемента необхідно вибрати шаблон **Сторінка (WPF)** та задати ім'я сторінки.

Створення гіперпосилань

Для переходу між сторінками використовуються гіперпосилання. Елемент гіперпосилання, що відповідає об'єкту класу ***Hyperlink***, визначається наступним чином:

<Hyperlink NavigateUri="Page1.xaml">Текст Гіперпосилання</Hyperlink>

Властивість ***NavigateUri*** класу ***Hyperlink*** визначає, на яку сторінку буде переходити програму при натисканні на відповідному гіперпосиланні.

У WPF гіперпосилання є потоковими елементами, які повинні розміщуватися всередині іншого елемента, що підтримує їх. Це можна зробити, наприклад, в елементі ***TextBlock***, який для гіперпосилання є контейнером.

Створення основного меню

Клас ***Menu*** містить Windows-елементи керування меню, пов'язані з командами та обробниками подій. Меню формують з об'єктів ***MenuItem*** (ім'я пункту меню) та ***Separator*** (розділювач). Клас ***MenuItem*** має властивість ***Header***, яка визначає текст елемента меню. Цей клас може зберігати колекцію об'єктів

MenuItem, яка відповідає підпунктам меню. Клас ***Separator*** відображає горизонтальну лінію, яка розділяє пункти меню.

Меню зручно розміщувати у контейнері ***StackPanel***.

Приклад XAML - опис меню, яке на верхньому рівні містить, наприклад два пункти, перший з яких включає підпункти з роздільником.

```
<Menu>
    <MenuItem Header="Пункт меню 1" >
        <MenuItem Header="Підпункт меню 1:1" ></MenuItem>
        <Separator></Separator>
        <MenuItem Header=" Підпункт меню 1:2" " ></MenuItem>
    </MenuItem>
    <MenuItem Header="Пункт меню 2"></MenuItem>
</Menu>
```

Створення панелі інструментів

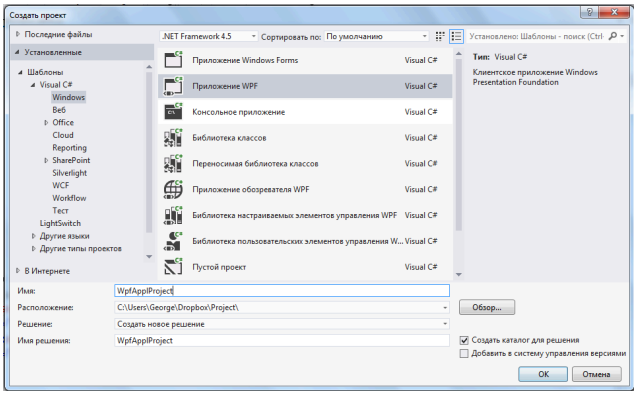
Панель інструментів представляє спеціалізований контейнер для зберігання колекції елементів, зазвичай кнопок:

```
<ToolBar Name="ToolBar1" Margin="3">
    <Button Name="Undo" ToolTip="Скасувати редагування/створення" Margin="5,2,5,2">
        <Image Source="Images/Undo.jpg" />
    </Button>
    ...
</ToolBar>
```

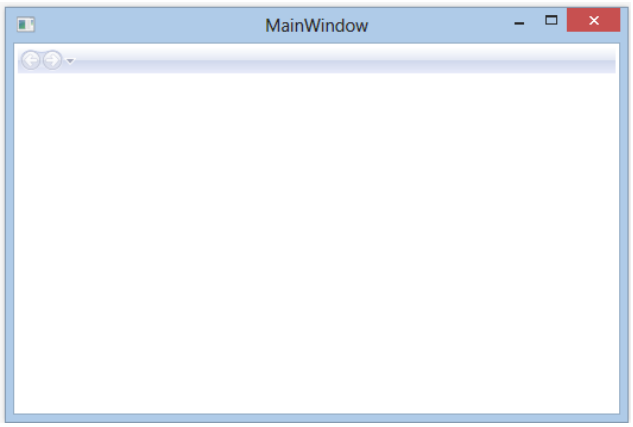
Для кожної кнопки задаються властивості: ***Name*** - ім'я об'єкта в програмі, ***ToolTip*** -з текстом підказки при наведенні покажчика миші на кнопку, ***Margin*** - визначає зовнішні відступи для кнопки. Завдання графічного об'єкта для кнопки здійснюється визначенням об'єкта ***Image*** джерела ***Source***, який повинен відповідати повному шляху до графічного файлу.

Хід виконання

1. Запустіть Visual Studio.
2. Натисніть комбінацію клавіш ***Ctrl-Shift-N*** або скористайтеся командою ***File*** - ***New*** - ***New Project*** для відкриття вікна створення проекту.
3. У діалозі перевірте установку бібліотеки .NET Framework не нижче 4.0 і виберіть ***WPF Application***. Вкажіть як ім'я проекту ***WpfApplProject*** і натисніть ***OK*** .



4. Додайте в проект початкову сторінку під ім'ям ***PageMain.xaml***.
5. Перейдіть у вікно XAML-розмітки основного вікна ***MainWind.xaml***.
6. Змініть властивість ***Title*** вікна відповідно до варіанта лабораторної роботи.
7. У контейнер ***Grid*** вставте екземпляр класу ***Frame*** для розміщення сторінок програми.
8. Вкажіть такі властивості екземпляра класу ***Frame***: ***Name="frame1"***, ***Margin="3", Source="PageMain.xaml", NavigationUIVisibility="Visible"*** .
9. Скомпілюйте проект і переконайтеся, що отримано наступний результат:



10. Додайте до проекту інші сторінки програми відповідно до варіанта завдання. Далі вони позначаються як ***<PageTable1>.xaml***, ***<PageTable2>.xaml*** тощо.
11. Перейдіть у вікно початкової сторінки ***PageMain.xaml*** та створіть гіперпосилання для переходу на сторінки програми.
12. Переконайтеся у працездатності створених гіперпосилань.
13. Додайте до сторінки ***<PageTable1>. xaml*** контейнер ***StackPanel***.
14. Додайте до ***StackPanel*** сторінки програми XAML опис меню, яке на верхньому рівні буде містити, два пункти ***Дія*** та ***Звіт***. Пункт ***Дія*** включає підпункти ***Скасувати***, ***Створити***, ***Редагувати***, ***Зберегти***, ***Знайти*** та ***Видалити***. Додайте розділові лінії між пунктами ***Скасувати***, ***Створити*** та пунктами ***Знайти***, ***Видалити.***
15. Створіть у проекті папку ***Images*** для розміщення графічних елементів, скориставшись командою ***Проект-Створити папку***.
16. Скопіюйте у папку ***Images*** файли іконок із зображеннями для кнопок панелі інструментів.
17. Додайте іконки до проекту, скориставшись командою ***Проект-Додати існуючий елемент…*** і вказавши файли з папки ***Images***.
18. Додайте до ***StackPanel*** сторінки програми XAML контейнер ***ToolBar*** для створення панелі інструментів.
19. Розташуйте в панелі інструментів кнопки, функціональне призначення яких відповідає підпунктам меню ***Дія***, тобто ***Скасувати***, ***Створити***, ***Редагувати***, ***Зберегти***, ***Знайти*** та ***Видалити***. Використовуйте іконки з папки ***Images*** для графічного зображення відповідної дії.

20. Код XAML-розмітки на головній сторінці програми скопіюйте у звіт про виконання завдання 1.

Завдання 2. Освоїти основи розробки бізнес-логіки програми

Основні теоретичні відомості

У WPF команда являє собою завдання додатка та механізм стеження за тим, коли вона може бути виконана. Одна і та ж команда може бути прив'язана до одного або кількох інтерфейсних елементів програми. Ініціюють команду джерела, які можуть бути різними елементами керування (наприклад ***MenuItem*** або ***Button***). Цільовим об'єктом команди є елемент, для котрого призначена ця команда.

Класи, що реалізують команди, повинні підтримувати інтерфейс ***ICommand***. У цьому інтерфейсі визначено два методи ***Execute***, ***CanExecute*** та подію ***CanExecuteChanged***. Метод ***Execute*** може містити код, що реалізує бізнес-логіку програми. Метод ***CanExecute*** повертає інформацію про стан команди. П*одія* **CanExecuteChanged** викликається при зміні стану команди.

Для створення команд користувача використовується клас ***RoutedCommand*** з простору імен **System.Windows.Input** , який має реалізацію інтерфейсу ***ICommand***.

Для розробки команд користувача рекомендується наступний порядок дій:

1. Додайте до проекту нову папку ***Commands*** та розмістіть в ній новий файл класу ***DataCommands.cs*** для опису команд користувача.
2. У класі **DataCommands** для кожної команди оголосіть статичну властивість типу ***RoutedCommand***, наприклад: ***public static RoutedCommand Delete {get; set;}.***
3. У класі **DataCommands** створіть конструктор класу, де визначте об'єкт типу ***InputGestureCollection***. Клас ***InputGestureCollection*** - впорядкована колекція об'єктів ***InputGesture***, які дозволяють за допомогою класу ***KeyGesture*** встановити комбінацію клавіш для виклику команд. Приклад фрагмента конструктора класу:

```
static DataCommands() {

        InputGestureCollection inputs = new InputGestureCollection();

        inputs.Add(new KeyGesture(Key.D, ModifierKeys.Control, "Ctrl+D"));

        delete = new RoutedCommand("Delete", typeof(DataCommands), inputs);

        //…

}
```

4. Перейдіть у файл коду сторінки (***Page<Table1>.xaml.cs***) і додайте до нього логічне поле ***isDirty*** для керування доступністю команд: ***private bool isDirty = true*** .
5. Для кожної команди додайте обробник з реалізацією методу ***CanExecute*** :

```
public void DeleteCommandBinding_CanExecute(object sender, CanExecuteRoutedEventArgs e) {

e.CanExecute = isDirty;

}

//…
```

6. Для кожної команди додайте обробник з реалізацією методу ***Executed*** :

```
private void DeleteCommandBinding_Executed(object sender, ExecutedRoutedEventArgs e) {

MessageBox.Show("Видалення");

isDirty = true;

}

//…
```

7. Перейдіть до файлу XAML-документа сторінки (***Page<Table1>.xaml***) і додайте до нього простір імен, в якому розташований клас команд (***DataCommands***):
   xmlns:command="clr-namespace:WpfApplProject.Commands" .
8. У файлі XAML-документа сторінки (***Page<Table1>.xaml***) сформуйте колекцію об'єктів ***CommandBinding*** для здійснення прив'язки команд для даного елемента та оголосіть зв'язок між командою, її подіями та обробниками:

```
<Page.CommandBindings>

<CommandBinding Command="Delete"

Executed="DeleteCommandBinding_Executed"

        CanExecute="DeleteCommandBinding_CanExecute" />

        <!--…-->

</Page.CommandBindings>
```

9. Модифікуйте XAML-документ у частині завдання властивості ***Command*** під час опису пунктів меню:

```
<Menu>

        <MenuItem Header="Дія" BorderThickness="3" >

        <MenuItem Header="Видалити" Command="Delete"></MenuItem>

        <!--…→

</Menu>
```

10. Модифікуйте документ XAML у частині завдання властивості ***Command*** при описі панелі інструментів:

```
<ToolBar Name="ToolBar1" Margin="3">

        <Button Name="Видалити" Margin="5,2,5,2" Command="Delete" ToolTip="Видалення">

                <Image Source="Images/Delete.jpg"></Image>

        </Button>

        <!--…–>

</ToolBar>
```

1. Додайте в проект папку ***Commands,*** а до неї - файл класу під ім'ям ***DataCommands.cs*** для опису команд користувача.
2. Підключіть за допомогою директиви ***using*** простір імен ***System.Windows.Input***, що містить класи ***RoutedCommand*** і ***InputGestureCollection***.
3. У класі оголосіть статичні властивості типу ***RoutedCommand*** для всіх команд меню та кнопок панелі інструментів: ***Undo*** (*Скасувати)*, ***New*** (*Створити)*, ***Replace*** (*Редагувати)*, ***Save*** (*Зберегти)*, ***Find*** (*Знайти)* і ***Delete*** (*Видалити)* .
4. Створіть конструктор класу, де визначте комбінації клавіш для виклику команд.
5. Перейдіть у файл коду головної сторінки (***Page<Table1>.xaml.cs***) і додайте до нього логічне поле ***isDirty*** (для керування доступністю команд) та обробники з реалізацією методів ***CanExecute()*** та ***Executed()*** (для кожної команди).
6. Переконайтеся, що меню та кнопки панелі інструментів працюють.
7. Код XAML-розмітки та коду головної сторінки програми, а також код класу команд користувача скопіюйте у звіт про виконання завдання 2.