Лабораторная работа №32
Разработка приложения с использованием элементов отображения списков

1Цель работы

1.1 Изучить свойства и процесс обработки событий элементов отображения списков в приложениях WPF.

2Литература

2.1 https://metanit.com/sharp/wpf/5.7.php ListBox
2.2 https://metanit.com/sharp/wpf/5.8.php ComboBox
2.3 https://metanit.com/sharp/wpf/5.9.php ListView

3Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).
3.2 Изучить описание лабораторной работы.

4Основное оборудование

4.1 Персональный компьютер.

5Задание

ListBox – простой список – предназначен для отображения раскрытого списка.
Особенности:
- содержит коллекцию элементов **Items** типа **ListBoxItem**
- вместо элементов ListBoxItem или внутри ListBoxItem могут быть указаны другие типы элементов управления (включая StackPanel)
- допускает множественный выбор, если свойству **SelectionMode** присвоить значение **Multiple** (для выбора используется нажатие) или **Extended** (для выбора используются нажатие и Shift и/или Ctrl)
- для выбора элемента(ов) нужно указать атрибут **IsSelected**="True"
- для получения выделенного элемента используется **SelectedItem**
- для получения всех выделенных элементов используется коллекция **SelectedItems**
- если нужно определить, с какого элемента был снят выбор, можно воспользоваться свойством **RemovedItems** объекта **SelectionChangedEventArgs**.

ComboBox – комбинированный или выпадающий список.
Особенности:
- содержит коллекцию элементов **Items** типа **ComboBoxItem**.
- вместо элементов ComboBoxItem или внутри ComboBoxItem могут быть указаны другие типы элементов управления (включая StackPanel)
- установка свойства **IsEditable**="True" позволяет вводить в поле списка начальные символы, а затем функция автозаполнения подставит подходящий результат
- для выбора элемента нужно указать атрибут **IsSelected**="True"
- для получения выделенного элемента используется **SelectedItem**.

Привязка данных

контрол.ItemsSource = список;
контрол.DisplayMemberPath = "Свойство объекта"; // отображаемое свойство

5.1 Создание аналога CheckedListBox (раскрытого списка с флажками)

5.1.1 Создать WPF-приложение.

5.1.2 Добавить на форму ListBox и заполнить в дизайнере его коллекцию Items элементами типа CheckBox (в списке должно быть как минимум 5 элементов), у каждого CheckBox указать свое значение свойства Content.

5.1.3 Добавить на форму кнопку, при нажатии на которую вывести значения отмеченных флажком элементов списка в одном MessageBox.

Пример перебора всех элементов типа CheckBox в списке:

```
foreach (CheckBox item in контрол.Items)
{
    // …
}
```

5.2 Программное добавление элементов в элементы-селекторы

5.2.1 Создать WPF-приложение.

5.2.2 Добавить на форму ListBox и ComboBox, поле ввода и кнопку «Добавить».

5.2.3 При нажатии на кнопку добавлять элемент из поля ввода в коллекцию Items элементов ListBox и ComboBox:

```
список.Items.Add(значение);
```

5.3 Привязка элементов-селекторов к списку пользовательского типа данных

5.3.1 Создать WPF-приложение. Для компоновки использовать Grid

5.3.2 Добавить на форму элементы-селекторы: ComboBox, ListBox, ListView. Предоставить пользователю возможность выбора более одного значения в ListBox.

5.3.3 Добавить в приложение источник данных:

- создать класс User для хранения идентификатора, логина и пароля пользователя:

```
class User
{
   public int Id { get; set; }
   public string Login { get; set;  }
   public string Password { get; set;  }
}
```

- в конструкторе окна после инициализации заполнить список List<User> данными (не менее 5 записей)

- у элементов-селекторов указать в качестве источника данных список пользователей

5.3.4 Настроить отображение логинов пользователей, указав у элементов-селекторов DisplayMemberPath

5.4 Получение выбранных элементов

Пример приведения выбранного элемента к требуемому типу:

```
Тип объект = (Тип)контрол.SelectedItem; // или контрол.SelectedItem as Тип
```

5.4.1 Рядом с каждым элементом-селектором разместить метку для отображения выбранных значений

5.4.2 При выборе элемента в ComboBox отобразить значения идентификатора, логина и пароля,

5.4.3 При выборе элементов в ListBox, ListView отобразить все выбранные логины для соответствующего элемента-селектора.

5.5 Настройка внешнего вида элементов-селекторов

5.5.1 Создать WPF-приложение. Для компоновки использовать Grid, на котором разместить элементы-селекторы ComboBox, ListBox, ListView и связать их со списком пользователей (как в п.5.3).

5.5.2 Добавить в ресурсы окна (Window.Resources) DataTemplate, которому задать x:Key. В шаблоне описать структуру отображения элемента списка:

- слева разместить картинку
- справа от картинки в один столбец разместить идентификатор, логин, кнопку «Показать пароль» (при нажатии отображать пароль соответствующего пользователя в MessageBox),
- указать разный цвет и размер текста идентификатора и логина,
- настроить границу вокруг элемента, отступы и ширину элемента.

5.5.3 Применить шаблон DataTemplate в элементах-селекторах:

ItemTemplate="{StaticResource имяШаблона}"

5.5.4 Изменить панель элементов-селекторов:

- настроить ItemsPanelTemplate типа WrapPanel у ListBox, ListView, чтобы элементы сдвигались, как при режиме «Плитка» в проводнике,
- отключить у элементов-селекторов горизонтальную прокрутку.

6Порядок выполнения работы

6.1 Выполнить все задания из п.5 в одном решении LabWork32. Каждый проект – приложение WPF.

6.2 Ответить на контрольные вопросы.

7Содержание отчета

7.1 Титульный лист
7.2 Цель работы
7.3 Ответы на контрольные вопросы
7.4 Вывод

8Контрольные вопросы

8.1 Что такое ComboBox и для чего он используется?
8.2 Что такое ListBox и для чего он используется?
8.3 Какое событие срабатывает при выборе элемента в селекторе?
8.4 В каком свойстве хранятся элементы селекторов?
8.5 Какого типа элементы могут быть в селекторе?
8.6 Какое свойство позволяет привязать селектор к набору данных?
8.7 Для чего используется свойство DisplayMemberPath в селекторе?