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

Требуется разработать WPF-приложение, отображающее каталог товаров, используя различные элементы отображения списков для настройки фильтров и отображения товаров.

Примерное расположение элементов:

<table>
<tr><td colspan="3">Список категорий товаров (ListBox горизонтальный с полосой прокрутки (настроено свойство ItemsPanelTemplate))</td></tr>
<tr>
<td>Наличие (ListBox с флажками):
• в наличии
• под заказ: сегодня
• под заказ: завтра
• под заказ: позже
• нет в продаже

Кнопка «Применить»</td>
<td>Сортировка по наличию (ComboBox)

Сортировка: _
Категория: _
Наличие: _
ToggleButton «Список» / «Плитка»

Список товаров (ListView)
- Товар1
- Товар2
…</td>
<td>Корзина (ListBox)
- Товар1
- Товар2
…

Кнопка «Удалить»</td>
</tr>
</table>

5.1 Настройка отображения элементов списка

5.1.1 Добавить в приложение отображение вариантов сортировки в виде выпадающего списка. Значения заполнить в дизайнере:

- сначала недорогие
- сначала дорогие
- по новинкам
- по скидке
- по количеству отзывов
- сначала с лучшей оценкой.

Один из вариантов выбрать в дизайнере (настроить IsSelected).

5.1.2 Добавить в приложение отображение фильтрации по наличию в виде раскрытого списка. Значения заполнить в дизайнере, каждый элемент: CheckBox, у которого в свойстве Content - вариант фильтра. Все, кроме “нет в продаже”, должны быть выбраны (настроить IsChecked).

5.1.3 Добавить в приложение отображение фильтрации по категории в виде раскрытого списка. Значения заполнить программно, указав источник данных (список из 10 названий категория). Сменить стиль ItemsPanelTemplate на StackPanel с горизонтальной ориентацией.

5.2 Получение выбранных/отмеченных элементов списка

5.2.1 Выбранный вариант сортировки отобразить в TextBlock в формате “Сортировка: выбранный вариант”

5.2.2 Выбранную категорию отобразить в TextBlock в формате “Категория: выбранная категория”

5.2.3 Добавить под списком с вариантами наличия кнопку «Применить», при нажатии на которую вывести значения отмеченных флажком элементов списка в TextBlock.

Пример перебора всех элементов типа CheckBox в списке:

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

5.3 Настройка отображения элементов в виде карточек

5.3.1 Создать класс Product с автосвойствами Id, Name, Category, Price.

5.3.2 Добавить в разметку приложения ListView. Добавить в приложение список List<Product> из пяти товаров и использовать его как ItemsSource в ListView.

5.3.3 Настроить у него отображение элементов в виде карточек, используя ItemTemplate. Добавить в ресурсы приложения картинку, которая будет использоваться как заглушка для изображения всех товаров.

5.4 Изменение содержимого списков

5.4.1 Добавить в разметку приложения ListBox для отображения названий товаров в корзине. Настроить у него DisplayMemberPath и разрешить выделение нескольких элементов с нажатием Shift/Control.

5.4.2 При нажатии на кнопку «Купить» товар должен добавляться в корзину.

5.4.3 При нажатии на кнопку «Удалить» выделенные в корзине товары должны удаляться из нее (может потребоваться удаление элементов с конца списка).

5.5 Смена стиля отображения элементов списка

5.5.1 Добавить в ресурсы окна шаблоны панели элементов ItemsPanelTemplate и применить шаблон для списка в разметке:

- WrapPanel (для отображения элементов в виде плитки)
- VirtualizingStackPanel (для отображения элементов в виде списка)

5.5.2 Добавить в ресурсы окна шаблоны элементов списка DataTemplate согласно макету и применить шаблон для списка в разметке:

<table>
<tr>
<td>Вариант 1. Отображение элемента (список):
<table><tr><td>Изображение</td><td>Название</td><td>Цена

«Купить»</td></tr></table></td>
<td>Вариант 2. Отображение элемента (плитка, указать ширину):
<table><tr><td>Изображение

Название

Цена «Купить»</td></tr></table></td>
</tr>
</table>

5.5.3 Настроить переключение внешнего вида со списка на плитку и обратно при нажатии на соответствующую кнопку:

ИмяСписка.ItemsPanel = (ItemsPanelTemplate)FindResource("НазваниеTemplate");

ИмяСписка.ItemTemplate = (DataTemplate)FindResource("НазваниеTemplate");

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?

9 Приложение

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 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

Программное добавление элемента в список:

контрол.Items.Add(значение);

Пример приведения выбранного элемента к требуемому типу:

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

Перебор в списке всех элементов определенного типа:

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

Настройка шаблона панели элементов списка:

```
<ItemsPanelTemplate x:Key="НазваниеTemplate">
      Настройка шаблона
</ItemsPanelTemplate>
```

Использование: <Список … ItemsPanel="{StaticResource НазваниеTemplate}" …>
Программная смена шаблона:
ИмяСписка.ItemsPanel = (ItemsPanelTemplate)FindResource("НазваниеTemplate");

Настройка шаблона элемента списка:

```
<DataTemplate x:Key="НазваниеTemplate">
      Настройка шаблона
</DataTemplate>
```

Использование: <Список … ItemTemplate="{StaticResource НазваниеTemplate}" …>
Программная смена шаблона:
ИмяСписка.ItemTemplate = (DataTemplate)FindResource("НазваниеTemplate");