Меню

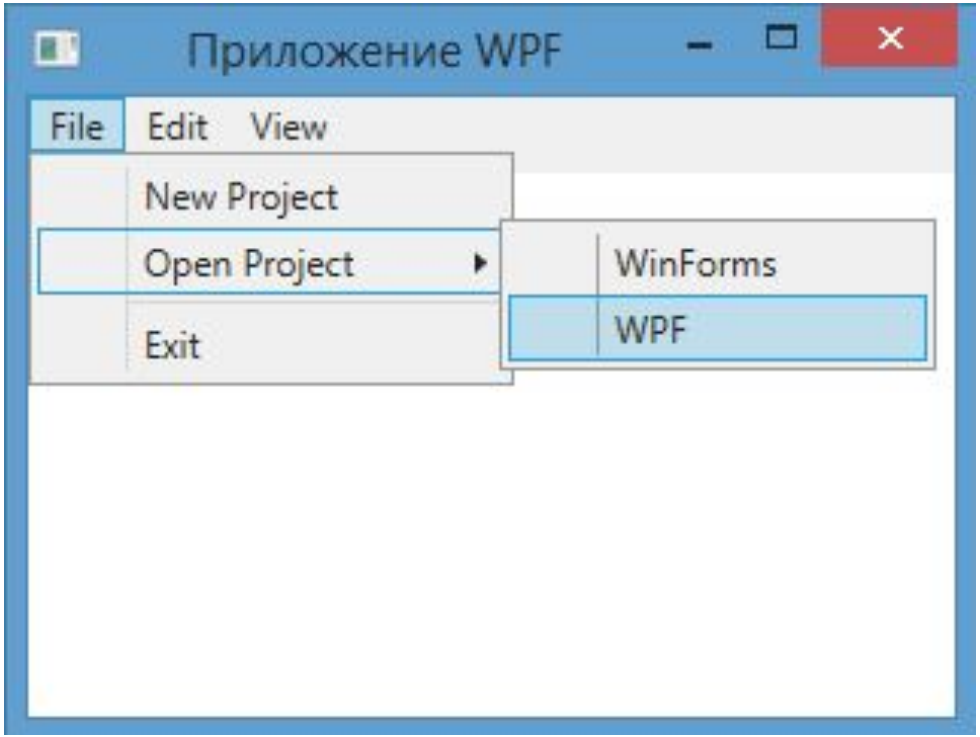
Мухортова Н.Н.

Цель

Научиться добавлять меню приложения

Меню

```
<Menu Height="25" VerticalAlignment="Top">
    <MenuItem Header="File">
        <MenuItem Header="New Project"
></MenuItem>
        <MenuItem Header="Open Project" >
            <MenuItem
Header="WinForms"></MenuItem>
            <MenuItem Header="WPF" ></MenuItem>
        </MenuItem>
        <Separator />
        <MenuItem Header="Exit" ></MenuItem>
    </MenuItem>
    <MenuItem Header="Edit" ></MenuItem>
    <MenuItem Header="View" ></MenuItem>
</Menu>
```


Приложение WPF
File
Edit
View
New Project
Open Project
Exit
WinForms
WPF

MenuItem

Элемент Menu включает набор элементов MenuItem, которые опять же являются элементами управления содержимым и могут включать другие элементы MenuItem и не только. Также мы можем вложить в меню и другие элементы, которые неявно будут преобразованы в MenuItem.

Button

```
<Menu Height="25" VerticalAlignment="Top">
    <MenuItem Header="File">
        <Button Content="Exit" />
    </MenuItem>
    <MenuItem Header="Edit" ></MenuItem>
    <MenuItem Header="View" ></MenuItem>
    <Button Content="Кнопка в меню" />
</Menu>
```

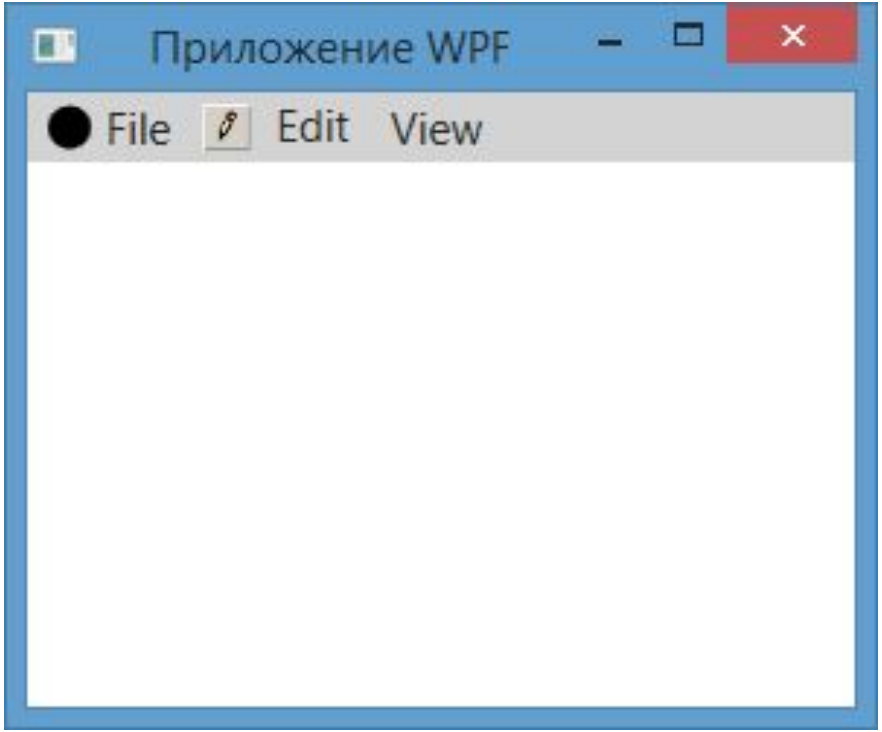
MenuItem

Также для разделения отдельных пунктов меню можно включать элемент Separator, как в примере выше.

Мы также можем настроить внешний вид отображения, задав свойство **MenuItem.Header** или использовав свойство Icons:

```
<Menu Height="25" VerticalAlignment="Top" Background="LightGray">
    <MenuItem>
        <MenuItem.Header>
            <StackPanel Orientation="Horizontal">
                <Ellipse Height="10" Width="10" Fill="Black" Margin="0
0 5 0" />
                <TextBlock>File</TextBlock>
            </StackPanel>
        </MenuItem.Header>
    </MenuItem>
```

```
    <MenuItem Header="Edit">
        <MenuItem.Icon>
            <Image
Source="C:\Users\Eugene\Documents\pen.png"></Image>
        </MenuItem.Icon>
    </MenuItem>
 <MenuItem Header="View"></MenuItem>
</Menu>
```


Приложение WPF
File
Edit
View

Обработчик

Чтобы обработать нажатие пункта меню и произвести определенное действие, можно использовать событие Click, однако в будущем мы познакомимся с еще одним инструментом под названием команды, который также широко применяется для реакции на нажатие кнопок меню.

```
<MenuItem Header="View" Click="MenuItem_Click"></MenuItem>
```

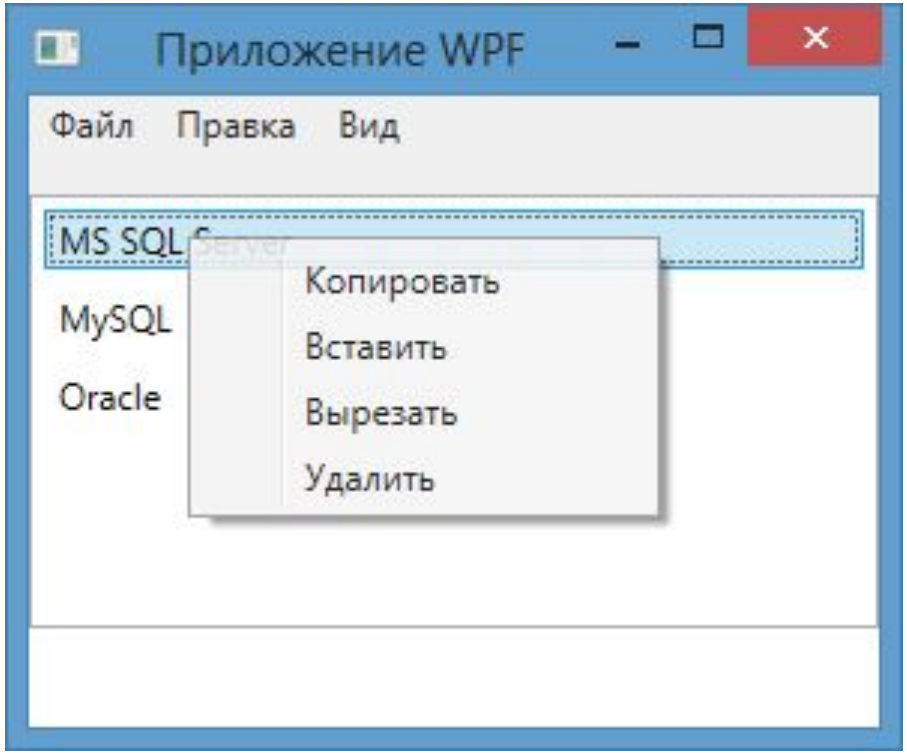
Обработчик

```
private void MenuItem_Click(object sender,
RoutedEventArgs e)
{
    MenuItem menuItem = (MenuItem)sender;
    MessageBox.Show(menuItem.Header.ToString());
}
```

ContextMenu

Класс ContextMenu служит для создания контекстных всплывающих меню, отображающихся после нажатия на правую кнопку мыши. Этот элемент также содержит коллекцию элементов MenuItem. Однако сам по себе ContextMenu существовать не может и должен быть прикреплен к другому элементу управления. Для этого у элементов есть свойство ContextMenu

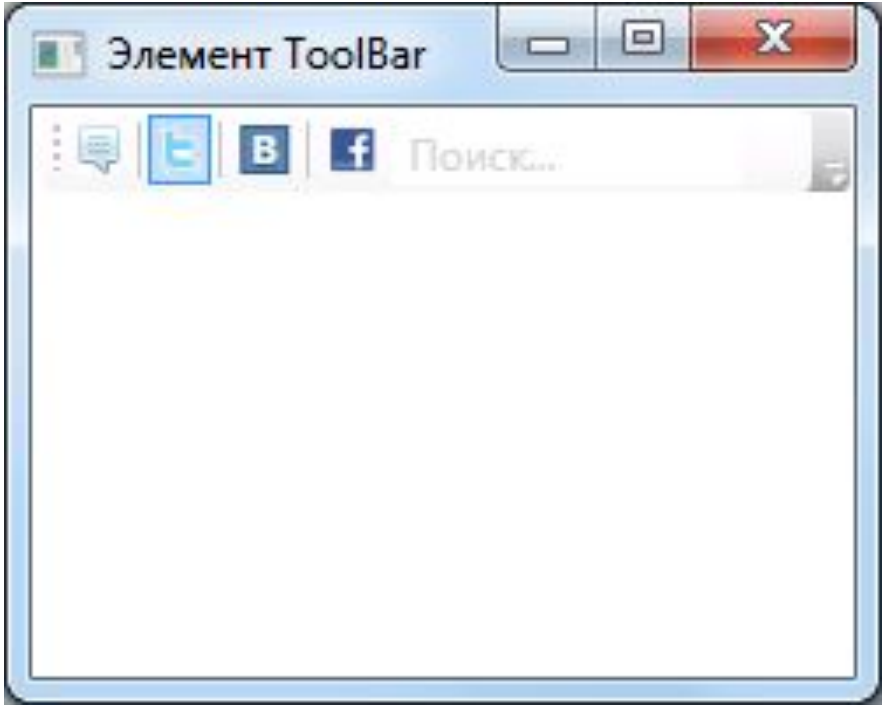
```
<ListBox Name="list" Height="145">
    <ListBoxItem Margin="3">MS SQL Server</ListBoxItem>
    <ListBoxItem Margin="3">MySQL</ListBoxItem>
    <ListBoxItem Margin="3">Oracle</ListBoxItem>
    <ListBox.ContextMenu>
        <ContextMenu>
            <MenuItem Header="Копировать"></MenuItem>
            <MenuItem Header="Вставить"></MenuItem>
            <MenuItem Header="Вырезать"></MenuItem>
            <MenuItem Header="Удалить"></MenuItem>
        </ContextMenu>
    </ListBox.ContextMenu>
</ListBox>
```


Приложение WPF
Файл
Правка
Вид
MS SQL Server
MySQL
Oracle
Копировать
Вставить
Вырезать
Удалить

ToolBar

Этот элемент, как правило, применяется для обеспечения быстрого доступа к наиболее часто используемым операциям. Он может содержать прочие элементы как кнопки, текстовые поля, объекты Menu и др.

```
<ToolBar Height="25" VerticalAlignment="Top">
    <ToggleButton><Image Source="icon0.gif" /></ToggleButton>
    <Separator />
    <Button><Image Source="icon1.gif" /></Button>
    <Separator />
    <Button><Image Source="icon2.png" /></Button>
    <Separator />
    <Button><Image Source="icon3.png" /></Button>
    <TextBox Foreground="LightGray" Width="100">Поиск...</TextBox>
</ToolBar>
```


Элемент ToolBar
Поиск...

Диалоговые окна

WPF поддерживает возможность создания модальных диалоговых окон. При вызове модальное окно блокирует доступ к родительскому окну, пока пользователь не закроет модальное окно.

Для работы добавим в проект новое окно, которое назовем **PasswordWindow**. Это окно будет выполнять роль модального.

```
<Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="20" />
            <RowDefinition Height="20" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>

        <TextBlock>Введите пароль:</TextBlock>
        <TextBox Name="passwordBox" Grid.Row="1" MinWidth="250">Пароль</TextBox>

        <WrapPanel Grid.Row="2" HorizontalAlignment="Right" Margin="0,15,0,0">
            <Button IsDefault="True" Click="Accept_Click" MinWidth="60"
Margin="0,0,10,0">OK</Button>
            <Button IsCancel="True" MinWidth="60">Отмена</Button>
        </WrapPanel>
    </Grid>
```

Диалоговые окна

Здесь определено текстовое поле для ввода пароля и две кнопки. Вторая кнопка с атрибутом IsCancel="True" будет выполнять роль отмены. А первая кнопка будет подтверждать ввод.

Для подтверждения ввода и успешного выхода из модального окна определим в файле кода PasswordWindow определим обработчик первой кнопки Accept_Click

```
        private void Accept_Click(object sender,
RoutedEventArgs e)
        {
            this.DialogResult = true;
        }

        public string Password
        {
            get { return passwordBox.Text; }
        }
```

Диалоговые окна

Для успешного выхода из модального диалогового окна нам надо для свойства **DialogResult** установить значение true. Для второй кнопки необязательно определять обработчик, так как у нее установлен атрибут IsCancel="True", следовательно, ее нажатие будет эквивалентно результату this.DialogResult = false;. Этот же результат будет при закрытии диалогового окна на крестик.

Кроме того, здесь определяется свойство Password, через которое мы можем извне получить введенный пароль.

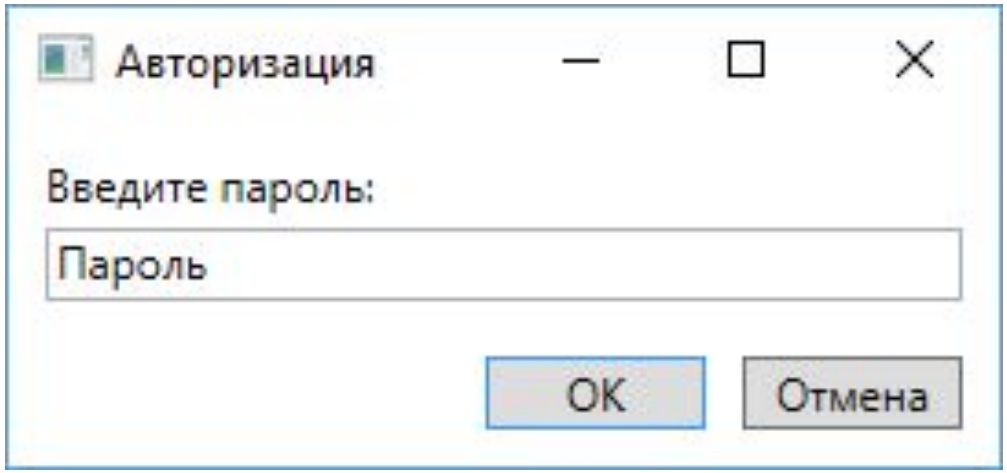
И изменим главную форму MainWindow, чтобы из нее запускать диалоговое окно.

Диалоговые окна

```
<Grid>
        <Button Width="100" Height="30" Content="
Авторизация" Click="Login_Click" />
</Grid>
```

```
private void Login_Click(object sender, RoutedEventArgs e)
{
    PasswordWindow passwordWindow = new PasswordWindow();

    if(passwordWindow.ShowDialog()==true)
    {
        if(passwordWindow.Password=="12345678")
            MessageBox.Show("Авторизация пройдена");
        else
            MessageBox.Show("Неверный пароль");
    }
    else
    {
        MessageBox.Show("Авторизация не пройдена");
    }
}
```


Авторизация
Введите пароль:
Пароль
OK
Отмена

Выводы

Все виды меню имеют похожее описание

Диалоговые окна используются как модальные окна и могут быть использованы, как вспомогательные для работы меню