Лабораторная работа №2

Создание простого приложения на основе событийно-ориентированного интерфейса

#### Цель работы

Получение навыков работы с основными элементами управления: создание, изменение свойств, написание простых обработчиков событий.

#### Теоретический материал

Когда вы только-только создаете проект WPF, по умолчанию в нем уже есть два компонента – окно (Window) и решетка (Grid). Решетка находится на окне, а все элементы вы будете размещать на решетке. Пока что нам достаточно будет трех видов элементов: кнопки (Button), текстового поля (TextBox) и метки (Label):

Изображение выглядит как снимок экрана, телевидение, экран, монитор

Автоматически созданное описание

1. Кнопка, текстовое поле и метка

Кнопки предназначены для того, чтобы их нажимали (логично!), и после нажатия происходили какие-то действия. Все дело в том, что интерфейс в WPF – событийно-ориентированный, то есть написанный вами код будет выполняться при наступлении какого-либо события. Простейшее событие – это нажатие кнопки.

Текстовые поля содержат текст. Кроме того, в процессе выполнения программы пользователь может изменить содержимое текстового поля. Обычно это используется для ввода какой-либо информации.

В метках может быть размещен какой-нибудь контент. Он может меняться с помощью кода, но не может меняться пользователем в процессе выполнения программы.

Каждый из этих объектов имеет свой набор свойств и свой набор возможных событий. Например, для текстового поля возможно событие TextChanged (текст в поле изменился), а для кнопки – Click (нажатие). У кнопки нет события TextChanged, так как пользователь не может менять текст кнопки в процессе работы с программой. А у текстового поля не может быть события Click, так как нажимать на текстовое поле особого смысла не имеет. Свойств и событий довольно много у каждого объекта, и вы можете с ними поэкспериментировать.

Чтобы менять какие-либо свойства или добавлять обработчики к событиям, вы должны сперва выбрать элемент.

Сейчас мною выбрана кнопка, и вы можете изменить ее свойства либо с помощью кода XAML, либо в окне свойств:

Изображение выглядит как монитор, снимок экрана, внутренний, сидит

Автоматически созданное описание

1. Свойства кнопки

Чтобы перейти к событиям, в окне свойств необходимо выбрать соответствующую вкладку:

Изображение выглядит как снимок экрана, монитор, черный, экран

Автоматически созданное описание

1. Переход на вкладку обработчиков событий

Вы попадете на вкладку обработчиков событий:

Изображение выглядит как снимок экрана, сидит, компьютер, экран

Автоматически созданное описание

1. События для кнопки

Возврат на вкладку свойств можно сделать следующим образом:

Изображение выглядит как снимок экрана, экран

Автоматически созданное описание

1. Возврат к свойствам

Чтобы была возможность обращаться к элементам управления в программе, необходимо дать им имена. Делать это можно на любой вкладке – как на вкладке свойств, так и на вкладке с обработчиками событий.

Кнопка будет называться AlarmistButton (кнопка-паникерша), текстовое поле – PokerFaceTB (поле-пофигист), метка – ReplicateLabel (метка-повторюшка). Элементам даны мнемонические имена, то есть те, которые будут отражать их суть, так будет проще в них разобраться. Если же у вас будут имена в духе Button1, Button2,…,Button217, то вероятность запутаться в них будет очень высока.

Изображение выглядит как снимок экрана

Автоматически созданное описание

1. Ввод имени объекта

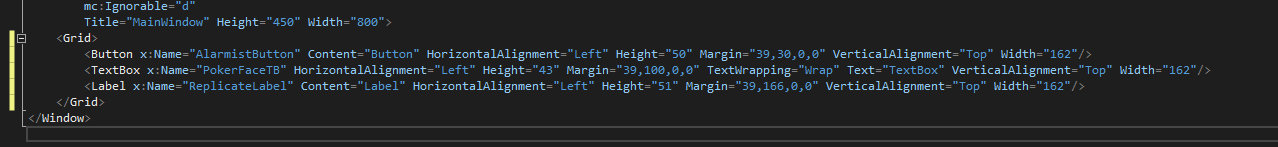
Обратите внимание на изменения в коде разметки:

Изображение выглядит как монитор, снимок экрана, экран, сидит

Автоматически созданное описание

1. Код разметки и имена объектов

Задать имя можно и в коде разметки, на рис.17 этот процесс еще не завершен (еще нет самого имени), и волнистая черта показывает наличие синтаксической ошибки. А вот на рис.18 уже все в порядке:



1. Введенные имена объектов

Теперь постепенно будем программировать обработчики событий. Будем все делать постепенно, шаг за шагом: добавлять что-то и пробовать.

Сперва приступим к обработчику события «нажатие кнопки». Пусть после нажатия на кнопку на ней будет написано «Ой! Меня нажали!».

Выберем кнопку с помощью клика мышью, перейдем к событиям, и дважды щелкнем мышкой напротив события Click. Секунду-другую вы сможете лицезреть название обработчика, который будет выполняться при нажатии этой кнопки, как на рис.19 (обратите также внимание на изменение в коде разметки, там тоже добавлен обработчик):

Изображение выглядит как снимок экрана, монитор, внутренний, ноутбук

Автоматически созданное описание

1. Создание обработчика событий

а затем тут же перейдете на вкладку с кодом. Пустой обработчик события уже создан, курсор помещен именно туда, где вы должны будете писать код – никаких лишних движений. Обратите внимание на название обработчика: оно состоит из названия кнопки и из события. В круглых скобках указаны параметры обработчика (их надо не трогать), фигурные скобки обозначают конец и начало блока кода.

Изображение выглядит как снимок экрана, монитор, экран, черный

Автоматически созданное описание

1. Код обработчика событий

Мы решили, что после нажатия кнопки на ней будет меняться надпись. Надпись на кнопке – это свойство Content. Поменяем его, написав в обработчике следующий оператор:

Изображение выглядит как часы, мяч, монитор, сидит

Автоматически созданное описание

1. Изменение надписи на кнопке

Запустив программу, можно посмотреть на результаты работы обработчика (слева – приложение до нажатия кнопки, справа – после).

Изображение выглядит как снимок экрана

Автоматически созданное описание

1. Результат работы программы

Это немалое достижение, но не будем на нем останавливаться. Пусть кнопка краснеет. Чтобы это происходило, надо менять фон кнопки – свойство BackGround. Делается это с помощью класса Brushes.

Примечание: когда будете набирать код, не игнорируйте контекстные подсказки, они помогут сэкономить время, особенно если вы медленно набираете на английском:

Изображение выглядит как снимок экрана, монитор, экран, черный

Автоматически созданное описание

1. Контекстная подсказка

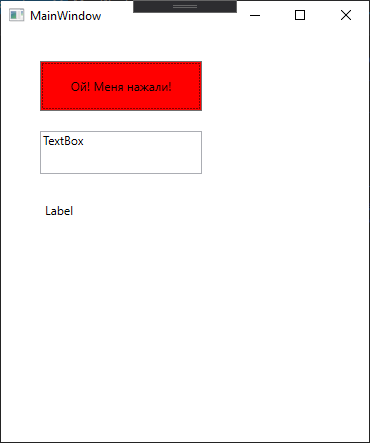
Красное подчеркивание означает, что в коде есть синтаксическая ошибка. Ничего страшного, просто код еще не дописан. Двойной щелчок мышью, и название свойства уже набрано. Получаем новый обработчик события:

Изображение выглядит как темный, сидит, оранжевый, черный

Автоматически созданное описание

1. Обработчик событий с имзенением цвета

Red – это метод, который покрасит фон объекта в красный цвет. Методов там много, каждый соответствует какому-либо цвету. Все это можно посмотреть в Приложении . Смотрим, как это будет работать:



1. Покраснение кнопки

Теперь пусть обработчик работает не только с кнопкой, но и с текстовым полем, которое будет «отвечать» кнопке «Ну и что такого?». Для этого надо будет поменять его свойство Text:

Изображение выглядит как монитор, сидит, экран, оранжевый

Автоматически созданное описание

1. Изменение текстового поля (код)

Вот как это будет работать:



1. Работа программы с изменением текстового поля

Метка пока осталась незадействованной, что не есть хорошо. Во-первых, это несправедливо. А во-вторых, должны же вы посмотреть, как можно изменить метку. Пусть в ней повторяются все изменения, которые будут происходить с текстовым полем. Для этого напишем новый обработчик, который будет вызываться при изменении текстового поля, в нем мы будем переписывать содержимое поля в метку. Кликнем мышкой на текстовое поле, найдем событие TextChanged и добавим его обработчик:

Изображение выглядит как снимок экрана

Автоматически созданное описание

1. Добавление обработчика изменения текстового поля

Теперь у нас уже два обработчика событий:

Изображение выглядит как снимок экрана

Автоматически созданное описание

1. Код нового обработчика событий

Новый обработчик будет содержать всего лишь одну строчку:

Изображение выглядит как объект, часы, оранжевый, темный

Автоматически созданное описание

1. Копирование содержимого текстового поля в метку

Убедившись, что синтаксических ошибок нет (нет красного подчеркивания), запускаем код, и:

Изображение выглядит как снимок экрана, телефон

Автоматически созданное описание

1. Результат работы программы. Точнее, его отсутствие

Программа отказывается работать. В чем же дело? Синтаксических ошибок не было, программа была успешно скомпилирована и начала работать, но во время ее работы что-то пошло не так: возникло исключение. В тексте исключения написано, что ReplicateLabel (наша метка) была null, т.е., еще не существовала. Дело в том, что под изменением текста в текстовом поле подразумевается **любое** изменение, в том числе и начальная инициализация всех объектов при запуске программы. Эта инициализация описана в коде, который был вставлен по умолчанию:

Изображение выглядит как снимок экрана, монитор, ноутбук, компьютер

Автоматически созданное описание

1. Инициализация компонент

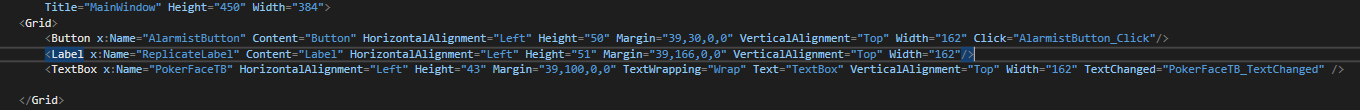
Остановим выполнение программы с помощью красной кнопки:

Изображение выглядит как снимок экрана, монитор, телефон, экран

Автоматически созданное описание

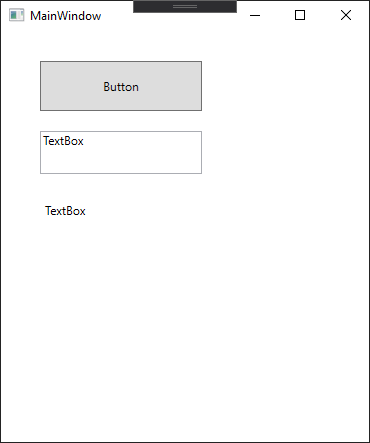
1. Остновка работы программы

А теперь попробуем решить проблему. Инициализация компонент происходит в том порядке, в котором мы добавляли их в окно: сперва появляется кнопка, затем появляется текстовое поле, в которое в процессе появления записывается его содержимое. А пока оно записывается, поле меняется, и вызывается обработчик, который должен менять метку. Но метка-то еще не инициализирована! Чтобы выкрутиться из этой ситуации, нам просто надо, чтобы метка была инициализирована раньше. А для этого перенесем ее в коде XAML повыше (просто вырезав в буфер и скопировав из него):



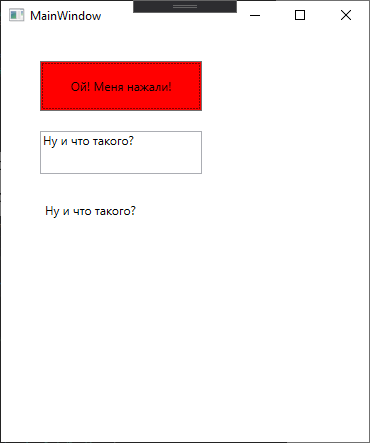
1. Изменения в коде разметки

Запускаем программу и любуемся результатами ее работы:



1. Работа программы после изменений

Обратите внимание, что в метку уже скопировано содержимое текстового поля, хотя мы туда еще ничего не вводили. Копирование содержимого произошло в момент инициализации компонент. Нажмем на кнопку и убедимся, что метка меняется вместе с текстовым полем:



1. Изменение метки при нажатии на кнопку

Теперь попробуем изменить текстовое поле во время работы программы вручную. В метке будет воспроизводиться каждое изменение:

Изображение выглядит как снимок экрана

Автоматически созданное описание

1. Изменение метки вместе с редактированием текстового поля

Обратите внимание, что код на C# мы не меняли, только код разметки на XAML. Да и то, меняли только порядок объектов в нем. Тот самый случай, когда от перестановки слагаемых результат меняется.

На самом деле, этот порядок довольно важен. Тот объект, который инициализируется позже (расположен в коде ниже), будет перекрывать все предыдущие объекты, если они будут расположены хотя бы частично в одной области. Сделаем так, чтобы текстовое поле немного «заезжало» на кнопку:

Изображение выглядит как снимок экрана, монитор, телевидение, экран

Автоматически созданное описание

1. Перемещение кнопки и текстового поля

А теперь сделаем в коде разметки кнопку последней (самой нижней) и посмотрим, как изменился внешний вид окна:

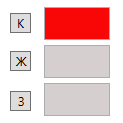
Изображение выглядит как внутренний, монитор, сидит, ноутбук

Автоматически созданное описание

1. Изменение кода разметки

#### Выполните самостоятельно

1. Создайте многооконное приложение с главным окном, где можно будет нажимать на кнопки и переходить в другие окна. Также требуется обеспечить возврат из этих окон обратно (по кнопкам). В каждом окне должен присутствовать свой функционал. Учтите, что интерфейс должен быть понятен пользователю.
2. Окно 1. «Кнопки детям не игрушка»
   1. Разместите в окне текстовое поле с надписью «Добрый день!»
   2. Разместите чуть ниже красную кнопку с надписью «Не нажимать!»
   3. Напишите обработку события нажатия кнопки. Текст в поле должен поменяться на «Написано же – не нажимать!».
   4. Сделайте так, чтобы после нажатия кнопка исчезала.
3. Окно 2. «Светофор»
   1. Расположите друг под другом три текстовых поля – это светофор (лучше, если вместо прямоугольных полей будут круги, при этом круги не обязаны быть текстовыми полями). Верхний сигнал – красный, два остальных изначально – серые. Рядом разместите три кнопки. Один из возможных вариантов изображен на рисунке, но творчество не ограничено ничем, кроме времени и правил приличия:



* 1. Реализуйте светофор следующим образом. Каждая из кнопок отвечает за свой сигнал светофора. Когда ее нажимают, загорается тот цвет, что напротив. Следует иметь в виду, что гореть должен только один сигнал светофора, остальные гаснут.

1. Окно 3. «Убегающая кнопка»
   1. В окне пользователь должен видеть кнопку с надписью «Нажми меня!».
   2. При нажатии на кнопку она должна переместиться в другое место. Кнопка не должна быть то на одном месте, то на другом. Сделайте так, чтобы разных положений было хотя бы пять или перемещайте кнопку случайным образом.

#### Контрольные вопросы

1. Что такое событийно-ориентированный интерфейс?
2. Чем отличаются текстовое поле и метка?
3. Что означает событие Click? Для какого элемента управления оно может быть?
4. Что означает событие TextChanged? Для какого элемента управления оно может быть?
5. Какое свойство метки нужно изменить, чтобы поменять ее текст?
6. Какое свойство кнопки нужно изменить, чтобы поменять текст на ней?
7. Как влияет на компоненты порядок их следования в коде разметки XAML?
8. Что будет, если свойство Opacity установить в 50?
9. Как разместить в текстовом поле текст в две строчки?
10. Как реализовать исчезновение кнопки после ее нажатия?
11. Какой элемент управления можно использовать, чтобы сделать сигналы светофора круглыми?
12. Как сделать так, чтобы горел только один сигнал светофора?
13. Как поменять цвет объекта?
14. Обработчик какого события надо было написать, чтобы поменялась надпись на кнопке?

## Цвета и их обозначения

