Министерство образования Республики Беларусь

Министерство образования и науки Российской Федерации

ГУВПО «Белорусско-Российский университет»

Кафедра «Программное обеспечение информационных технологий»

Дисциплина «Современные системы программирования»

# Лабораторная работа 1

Введение в WPF и XAML.

Диспетчеры компоновки.

2017

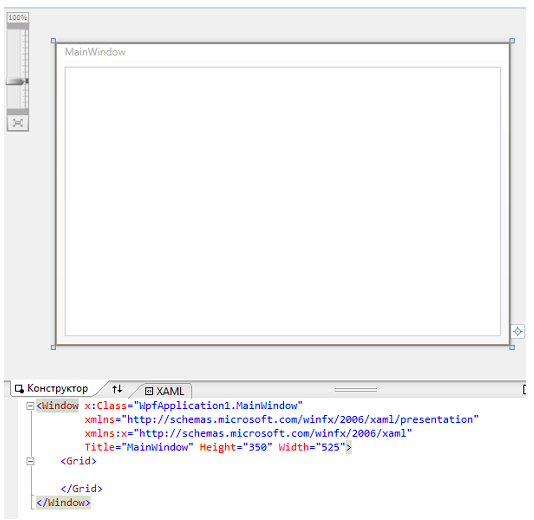
**WPF и XAML**

Графическая система Windows Presentation Foundation предназначена для создания пользовательских интерфейсов, 2D и 3D графики. Преимущества WPF заключается в том, что 2D графика строится в векторном виде, а это значит, что интерфейсы будут максимально независимы от разрешения экрана и размера окна. Они будут легко масштабироваться без потери качества и быстро работать благодаря максимальному использованию возможностей современных графических ускорителей. WPF объединяет документы, формы и мультимедийное содержание в пакет, состоящий из языка разметки и процедурного языка программирования.

Для создания и инициализации объектов в WPF используется язык разметки XAML - Extensible Application Markup Language (расширяемый язык разметки приложений). XAML является диалектом языка XML. Файл XAML содержит ровно одну корневую вершину и является деревом отображения. На вершине иерархии находится один из контейнерных объектов. Внутри этих объектов располагаются элементы управления и другие контейнеры. В XAML названия элементов чувствительны к регистру и совпадают с именами классов, доступных в кодовой части WPF.

**Задание 1:**

В среде Microsoft Visual Studio создайте проект «Приложение WPF». Среда разработки создаст заготовку, показанную на рисунке:



Среда разработки предоставляет возможность графического и дескрипторного способов разработки пользовательского интерфейса, которые являются равнозначными. Дескрипторный файл MainWindow.xaml и кодовый файл MainWindow.xaml.cs дополняют друг друга при описании одного и того же содержимого - класса MainWindow в пространстве имен WpfApplication1, совпадающим с названием проекта.

Платформа WPF проектировалась в рамках концепции отделения дизайнерской части пользовательского интерфейса от кодовой части программирования функциональности. Дизайнерская часть проектируется декларативно, чаще всего - с помощью графического дизайнера формы, который в автоматическом режиме генерирует соответствующий синтаксически правильный дескрипторный код на языке XAML.

В заготовке дескрипторного XAML-кода видно, что корнем приложения является контейнер <Window>, в который в дальнейшем будут включены дочерние элементы. Все элементы WPF существуют в двух вариантах: дескрипторном и объектном. Объектное описание размещается в пространствах имен, подключаемых в кодовую часть проекта с помощью инструкции using для видимости компилятором.

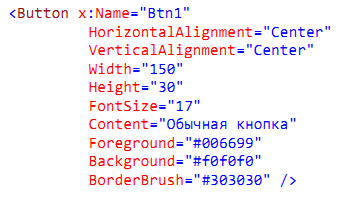
Дескрипторное описание находится в двух пространствах имен: обычном и расширенном. Эти пространства имен подключаются как значения атрибутов xmlns и xmlns:x в корневом дескрипторе проекта

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Используемые URL-адреса не указывают на какой-либо документ или содержимое на веб-сервере, а используются лишь для определения уникальных пространств имен.

Разместите в коде XAML в содержимом элемента Grid следующий код:



Запустите приложение и проверьте его поведение при изменении размеров окна.

В приведенном выше примере для элемента Button было задано простое значение атрибута Background. Для этого был использован синтаксис

<ЭЛЕМЕНТ АТРИБУТ=”ЗНАЧЕНИЕ” />

Для задания значения атрибуты может быть использован другой синтаксис:

<ЭЛЕМЕНТ>

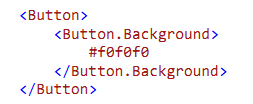
<ЭЛЕМЕНТ.АТРИБУТ>

ЗНАЧЕНИЕ\_АТРИБУТА

</ЭЛЕМЕНТ.АТРИБУТ>

</ЭЛЕМЕНТ>

Например, для задания того же значения атрибута Background можно записать:

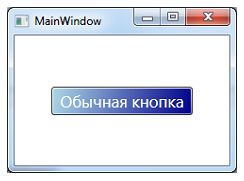


Данный синтаксис используется для задания сложных значений атрибутов в виде дерева элементов. Пример задания для фона кнопки линейной градиентной заливки:



Данное дерево элементов задает градиентную заливку с использованием двух цветов: LightBlue и DarkBlue. В атрибуте Offset указывается относительное значение от 0 до 1, соответствующее положению цвета на отрезке от начальной точки до конечной.

Внешний вид данного приложения:



**Задание 2:**

В XAML-коде для элемента Windows определите линейную градиентную заливку фона в соответствии с рисунком:



Используемые цвета: DarkBlue и LightBlue.

Необходимо указать четыре промежуточные точки со смещениями 0, 0.2, 0.8 и 1. Для задания вертикальной заливки необходимо определить атрибуты StartPoint и EndPoint для элемента LinearGradientBrush. Значения этих атрибутов указываются в формате “X,Y”, где X – относительное значение (от 0 до 1) абсциссы точки, Y – относительное значение (от 0 до 1) ординаты точки. Начало координат находится в левом верхнем углу окна. По умолчанию значения атрибутов StartPoint и EndPoint следующие: StartPoint=”0,0” EndPoint=”1,1”.

**Диспетчеры компоновки.**

Окно WPF-приложения обычно представлено корневым элементом Window. Дочерним элементом корневого элемента является диспетчер компоновки, который в свою очередь содержит любое количество элементов (в том числе, вложенных диспетчеров компоновки), определяющих пользовательский интерфейс. Диспетчер компоновки является объектом класса, унаследованного от абстрактного класса System.Windows.Controls.Panel.

Основные панели (диспетчеры компоновки, контейнерные элементы управления) WPF:

**Canvas**  Элементы остаются в точности там, где были размещены во время

проектирования

**DockPanel**  Привязывает содержимое к определенной стороне панели (Тор (верхняя),

Bottom (нижняя), Left (левая) или Right (правая))

**Grid**  Располагает содержимое внутри серии ячеек, расположенных в табличной

сетке

**StackPanel** Выводит содержимое по вертикали или горизонтали, в зависимости от

значения свойства Orientation

**WrapPanel**  Позиционирует содержимое слева направо, перенося на следующую

строку по достижении границы панели. Последовательность размещения

происходит сначала сверху вниз или сначала слева направо, в зависимости

от значения свойства Orientation

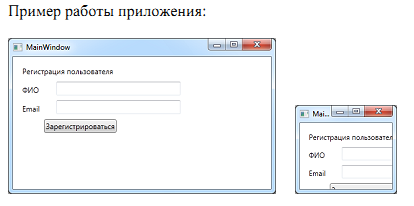
**Задание 3:**

Протестируйте работу каждого каждого диспетчера компоновки на приведенных примерах.

**Диспетчер компоновки Canvas**

Панель Canvas поддерживает абсолютное позиционирование содержимого пользовательского интерфейса. Если пользователь изменяет размер окна, делая его меньше, чем компоновка, обслуживаемая панелью Canvas, ее внутреннее содержимое становится невидимым до тех пор, пока контейнер вновь не увеличится до размера, равного или больше начального размера области Canvas.

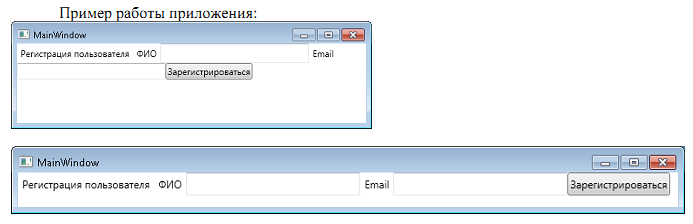
Панель Canvas обладает следующим недостатком: элементы внутри Canvas не изменяются динамически при применении стилей или шаблонов.

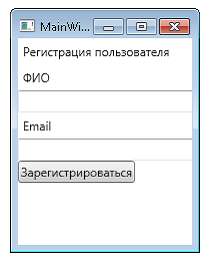


У элементов управления Label, TextBox, Button отсутствуют атрибуты Left и Top, поэтому для определения положения элементов на панели используется синтаксис присоединяемых свойств.

**Диспетчер компоновки WrapPanel**

Панель WrapPanel выводит дочерние элементы последовательно слева направо либо сверху вниз и при достижении границы окна переходит на новую строку (столбец).

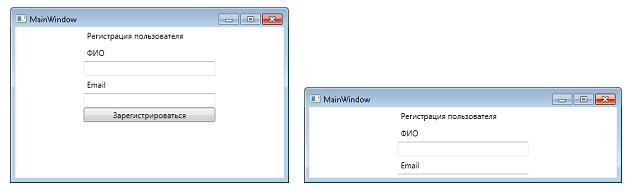




**Диспетчер компоновки StackPanel**

Панель StackPanel располагает содержащиеся в нем элементы управления либо в вертикальном столбце, либо в горизонтальной строке. Если в панель StackPanel добавлено больше элементов управления, чем может быть отображено по ширине/высоте StackPanel, лишние элементы обрезаются и не отображаются.

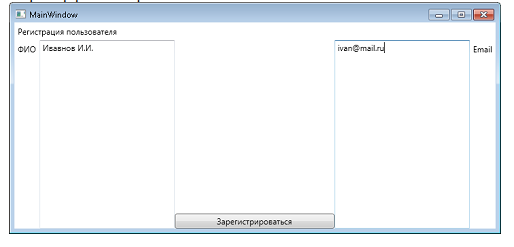
Пример работы приложения:



**Диспетчер компоновки DockPanel**

Панель DockPanel пристыковывает дочерние элементы к различным сторонам панели.

Пример работы приложения:



**Диспетчер компоновки Grid**

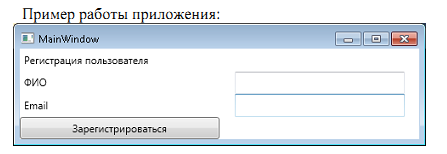
Подобно HTML-таблице, панель Grid может состоять из набора ячеек, каждая из которых имеет свое содержимое. При определении панели Grid выполняются следующие шаги:

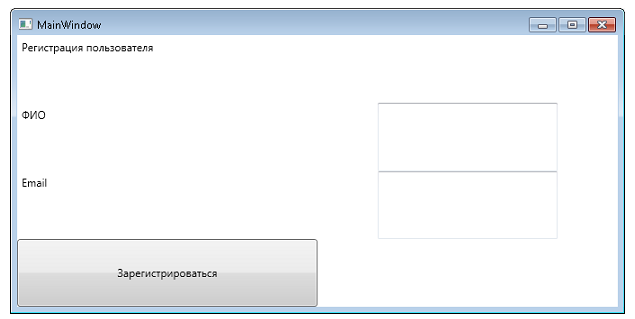
1. Определение и конфигурирование каждого столбца.

2. Определение и конфигурирование каждой строки.

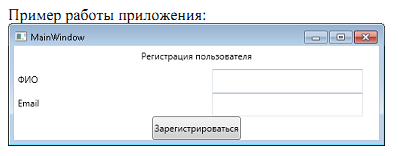
3. Назначение содержимого каждой ячейке сетки с использованием синтаксиса присоединяемых свойств.

Если не определить никаких строк и столбцов, то по умолчанию панель Grid будет состоять из одной ячейки, занимающей всю поверхность окна. Кроме того, если не указать ячейку для дочернего элемента, то он разместится в столбце 0 и строке 0.



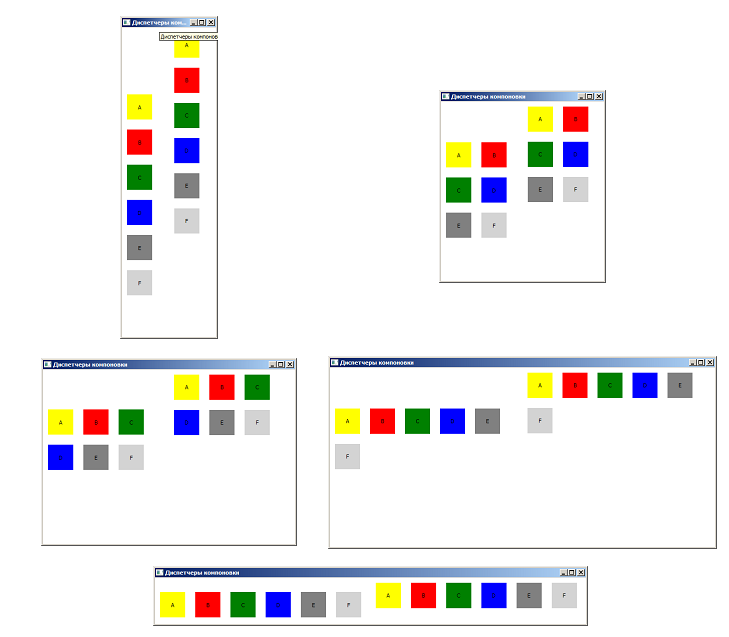


Расположите кнопку «Зарегистрироваться» посередине строки:



**Задание 4.**

1. Разработать приложение WPF со следующим графическим интерфейсом и с аналогичным поведением при изменении размеров окна.

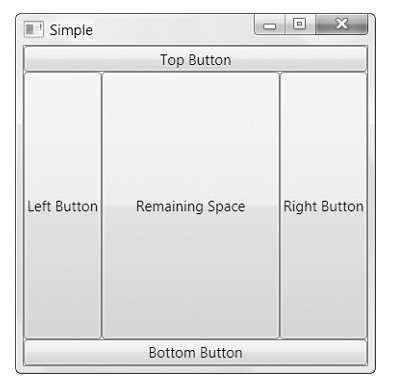


**Задание 5. (по вариантам)**

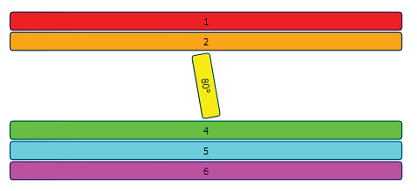
1. Создайте WPF-приложение с пятью кнопками следующего вида и с аналогичным поведением при изменении размеров окна.



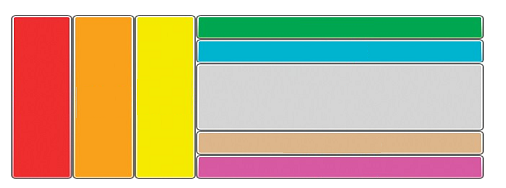
1. Создайте WPF-приложение с пятью кнопками следующего вида.



1. Создайте WPF-приложение с шестью кнопками следующего вида.



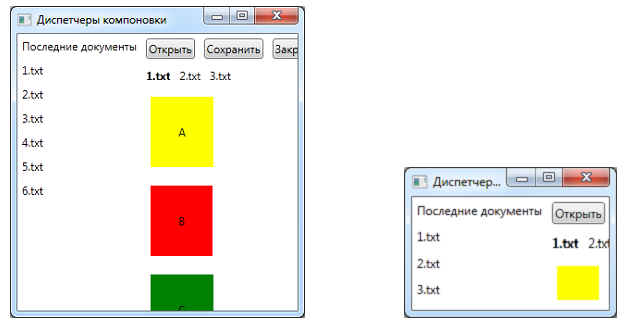
1. Создайте WPF-приложение следующего вида.



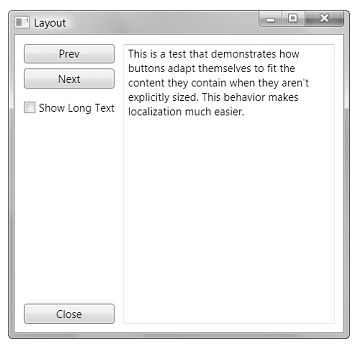
**Задание 6. (по вариантам)**

1. Разработать приложение WPF со следующим графическим интерфейсом:

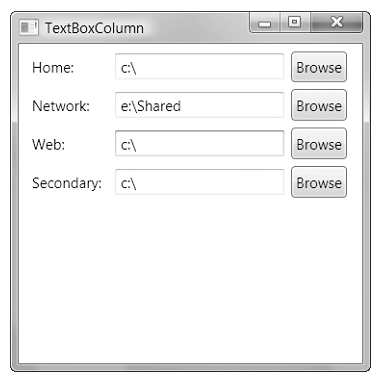




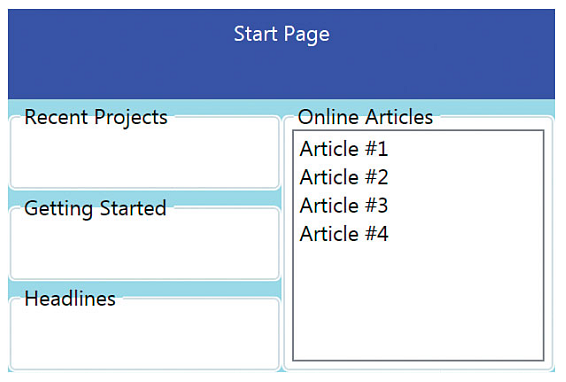
1. Создайте WPF-приложение следующего вида.



1. Создайте WPF-приложение с кнопками следующего вида.



1. Создайте WPF-приложение следующего вида. Используйте элемент GroupBox.



1. Создайте WPF-приложение следующего вида.

