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.

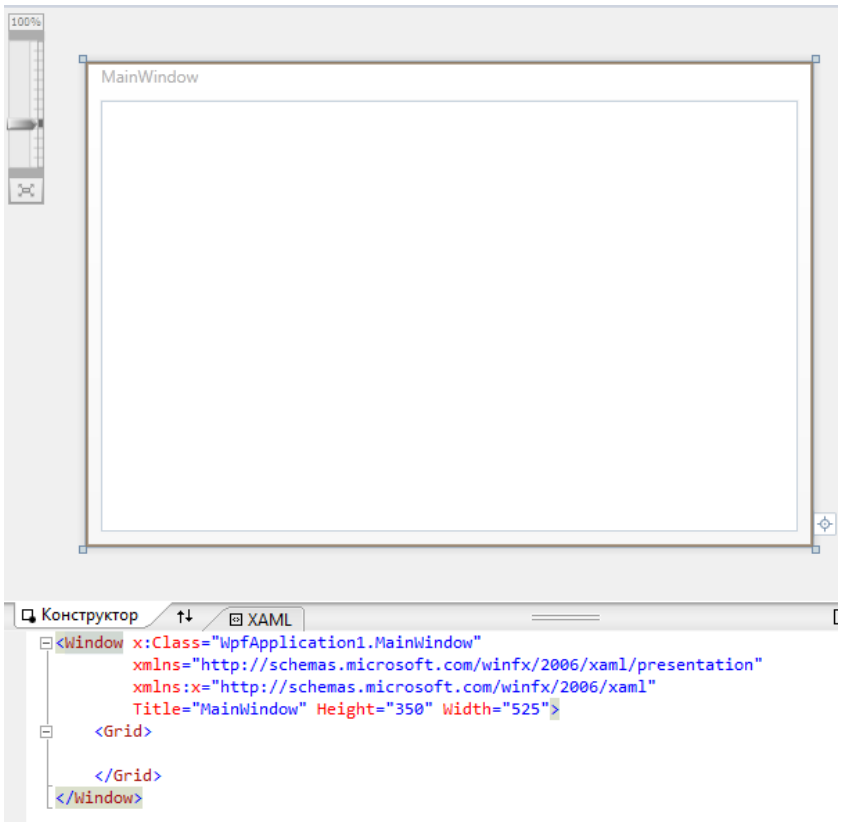
Для создания и инициализации объектов в WPF используется язык разметки XAML - Extensible Application Markup Language (расширяемый язык разметки приложений). XAML использует основные четыре категории элементов:

- панели размещения;
- элементы управления;
- элементы, связанные с документом;
- графические фигуры.

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 C# 2010 Express 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:

```
<Button x:Name="Btn1"
         HorizontalAlignment="Center"
         VerticalAlignment="Center"
         Width="150"
         Height="30"
         FontSize="17"
         Content="Обычная кнопка"
         Foreground="#006699"
         Background="#f0f0f0"
         BorderBrush="#303030" />
```

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:

```
<Button>
    <Button.Background>
        #f0f0f0
    </Button.Background>
</Button>
```

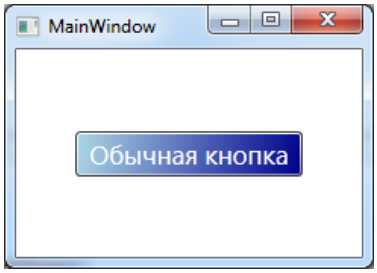
Данный синтаксис используется для задания сложных значений атрибутов в виде дерева элементов. Пример задания для фона кнопки линейной градиентной заливки:

```
<Button.Background>
         <LinearGradientBrush>
             <LinearGradientBrush.GradientStops>
                 <GradientStop Color="LightBlue" Offset="0" />
                 <GradientStop Color="DarkBlue" Offset="1" />
             </LinearGradientBrush.GradientStops>
```

```
        </LinearGradientBrush>
</Button.Background>
```

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:



Объект Applicaton

Любое приложение использует класс Application, который организует его подключение к модели событий операционной системы с помощью метода Run(). Объект Application отвечает за управление временем жизни приложения, отслеживает видимые окна, освобождает ресурсы и контролирует глобальное состояние приложения. Метод Run() запускает диспетчер среды исполнения, который начинает посылать события и сообщения компонентам приложения.

В каждый момент времени может быть активен только один объект Application и он будет работать до тех пор, пока приложение не завершится. К исполняемому объекту Application можно получить доступ из любого места приложения через статическое свойство Application.Current. Одна из основных задач объекта Application состоит в том, чтобы контролировать время жизни процесса. Конструирование объекта Application знаменует начало жизни приложения, а возврат из его метода Run() - завершение приложения.

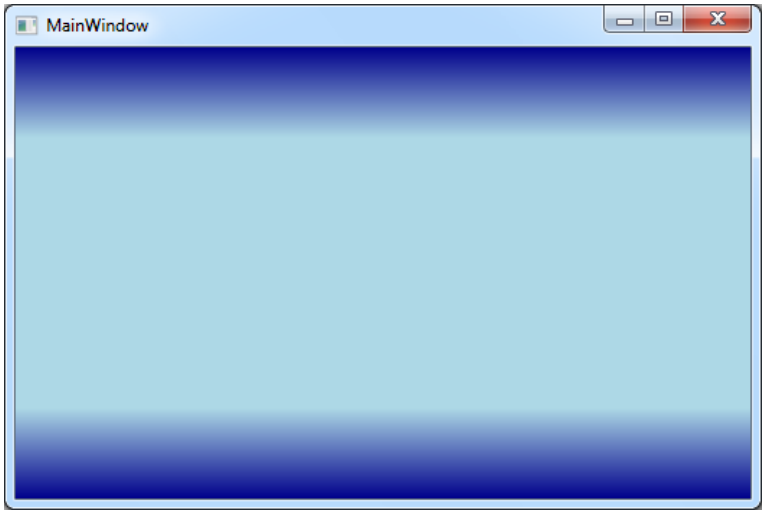
Время жизни приложения WPF и объекта Application состоит из следующих этапов:

1. Конструируется объект Application
2. Вызывается его метод Run()
3. Выполняется событие Application.Startup
4. Пользовательский код конструирует один или несколько объектов Window (или Page) и приложение выполняет работу
5. Вызывается метод Application.Shutdown()
6. Вызывается метод Application.Exit()

При создании проекта среда разработки поместила в проект два файла, связанные с объектом Application: App.xaml и App.xaml.cs. В этих файлах нет кода, создающего объекты Application и Windows и вызывающего метод Run() – это происходит неявно. В файле App.xaml для элемента Window задается атрибут StartupUri, в котором определяется имя XAML-файла с окном, которое открывается при запуске 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”.