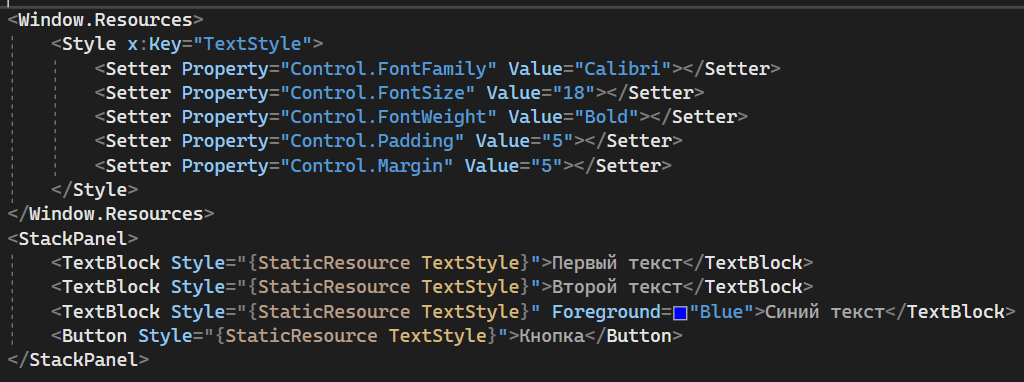
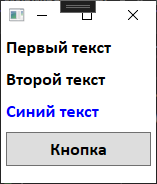
**Стили**

Стиль – это коллекция значений свойств, которые могут применяться к элементу. Система стилей WPF играет ту же роль, которую играет стандарт каскадных таблиц стилей (CSS) в HTML-разметке.

Стили позволяют определять общий набор характеристик форматирования и применять его повсюду в приложении для обеспечения согласованного вида.

Рассмотрим пример использования стилей. Предположим, нам необходимо настроить единый вид текста для всего окна:

****



В этом коде разметки создается один ресурс — объект System.Windows.Style. В этом объекте размещается коллекция Setters с пятью объектами Setter, по одному для каждого свойства, которое подлежит установке. В каждом объекте Setter указывается имя свойства, на которое он влияет, и значение, которое он должен применять к этому свойству.

Объект стиля имеет ключевое имя (свойство x:Key), по которому его можно при необходимости извлекать из коллекции. В данном случае это ключевое имя выглядит как TextStyle. (По общепринятому соглашению ключевые имена стилей обычно заканчиваются словом "Style".)

Каждый элемент WPF может использовать одиночный стиль (или не использовать его). Стиль подключается к элементу через свойство Style.

Стили задают первоначальный внешний вид элемента, но устанавливаемые ими характеристики могут быть переопределены. Например, если применить стиль TextStyle и явно установить свойство FontSize, параметр FontSize в дескрипторе кнопки переопределит этот стиль. В идеале полагаться на такое поведение не стоит — вместо этого лучше создать больше стилей, которые позволяют устанавливать максимальное количество деталей на уровне стиля. В результате обеспечивается большая гибкость при настройке пользовательского интерфейса в будущем.

Ключевые свойства, необходимые для создания стилей:

- Setters - коллекция объектов, которые устанавливают значения для свойств и присоединяют обработчики событий автоматически.

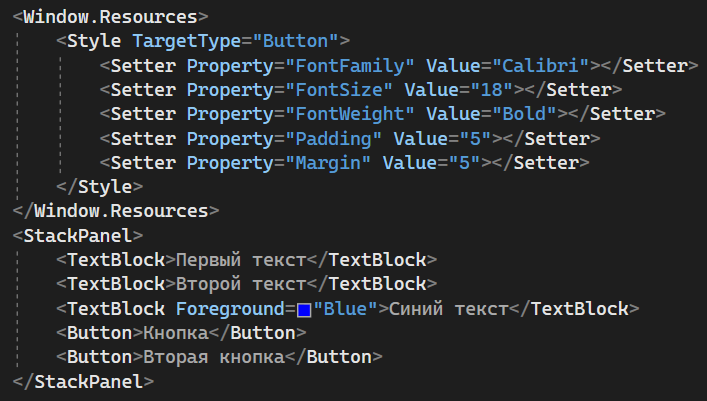
- Triggers - коллекция объектов, которые позволяют автоматически изменять настройки стиля. Настройки стиля могут модифицироваться, например, при изменении значения какого-то другого свойства или при поступлении какого-нибудь события.

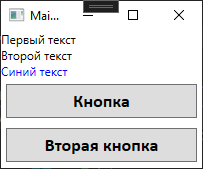
- Resources - коллекция ресурсов, которые должны использоваться со стилями. Например, может понадобиться использовать единственный объект для установки нескольких свойств. В таком случае более эффективно создать объект как ресурс и затем использовать этот ресурс в объекте Setter.

- BasedOn - свойство, которое позволяет создавать более специализированный стиль, наследующий (и дополнительно переопределяющий) параметры другого стиля

- TargetType - свойство, которое идентифицирует тип элемента, к которому применяется данный стиль. Это свойство позволяет создавать объекты Setter, влияющие только на определенные элементы.

Чаще всего стили принято определять в коллекции Resources на уровне окна. Данный вариант размещения стилей не единственный, но является самым распространенным.

Подобным образом можно задать стиль для какого-то отдельного элемента (заменив свойство x:Key на свойство TargetType): 



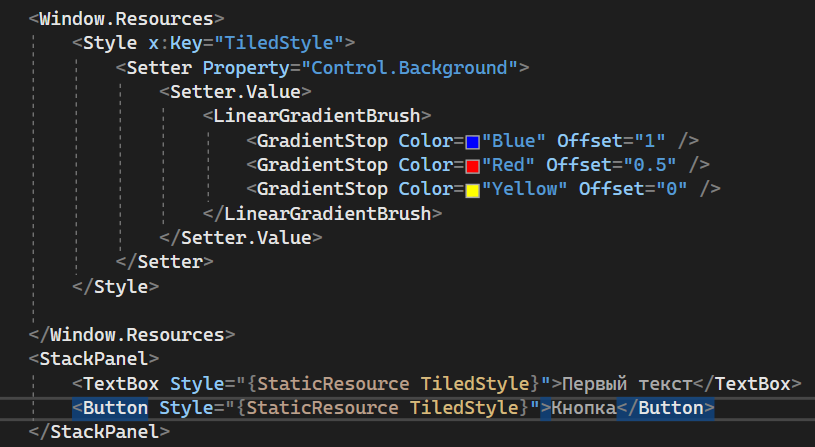
Объект Style умещает в себе коллекцию объектов Setter. Каждый объект Setter устанавливает одно свойство в элементе.

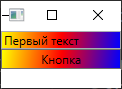
Объекты Setter имеют два свойства:

- Property – указывает на свойство, к которому будет применяться данный сеттер.

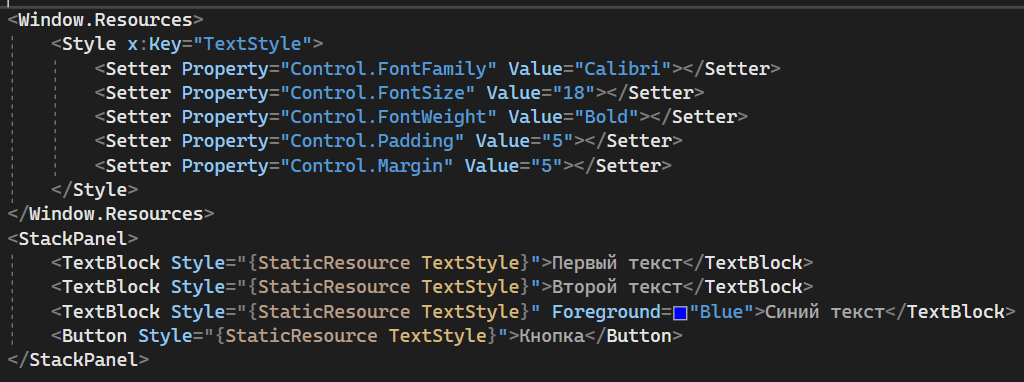
- Value – устанавливает значение.

В некоторых случаях значение свойства не может устанавливаться с использованием простой строки атрибута. Например, объект ImageBrush нельзя создать с помощью простой строки. В такой ситуации должен применяться уже знакомый трюк с XAML-разметкой, который заключается в замещении атрибута вложенным элементом, например:

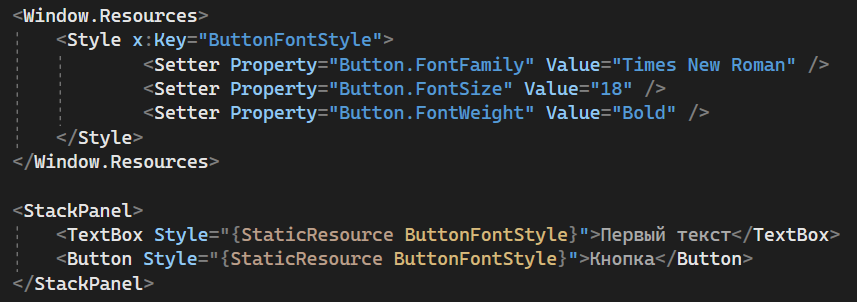


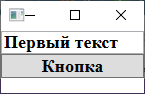


Также обратите внимание, чтобы идентифицировать свойство, которое должно устанавливаться, необходимо предоставить имя класса (или элемента) и имя свойство. В данном коде, например, именем класса выступает Control (так как все свойства для работы со шрифтами общие и объявлены в классе Control):

****

Имя класса, однако, не обязательно должно представлять тот класс, в котором данное свойство определено. Это может быть имя производного класса, который наследует свойство. Например, взгляните на стиль, в котором ссылки на класс Control заменяются ссылками на класс Button (так как данные свойства также доступны в классе Button):





При этом, стили будут корректно отображаться и не с кнопками несмотря на то, что свойства были взяты из класса Button. Это работает, так как, по сути, мы обращаемся к одному и тому же свойству зависимости несмотря на то, что происходит это разными путями. Однако, при таком способе задания свойств могут возникать различные непредвиденные ошибки, поэтому предпочтительным является указание базового класса для данного свойства.

Стиль приложения

Также WPF поддерживает создание стилей не для одного окна, а для всего приложения. Данная возможность реализуется в файле App.xaml. В таком варианте стили размещаются внутри файла App.xaml в свойстве Application.Resources.

