**9 лаба ответы**

1. Что такое Свойства зависимости? Для чего они нужны?

**свойства зависимостей** или **dependency property.**

Без свойств зависимостей были бы невозможны многие ключевые особенности WPF, как привязка данных, стили, анимация и т.д.

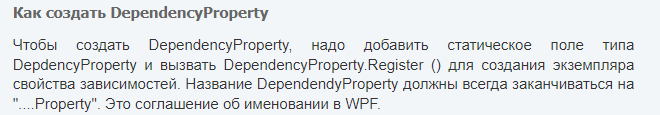
Главное отличие в том, что значения обычных .NET свойств можно читать прямо из закрытых членов вашего класса, в то время как значение DependencyProperties может динамически вызываться при вызове GetValue(), которые наследуется от DependencyObject.

Когда вы устанавливаете значение DepebdencyProperties, оно не хранится в поле вашего объекта, но хранится в словаре ваших ключей и значений, предоставленных DependencyObject базового класса. Ключ - это мя свойства, а значение - значение, которые вы хотите установить.

**Преимущества Dependency Properties:**  
Снижение памяти.  
Наследование значений.  
Уведомления об изменениях.  
Value resolution strategy

1. Как создать DependencyProperty? <https://metanit.com/sharp/wpf/13.php>

https://karpov-k.me/izuchaem-wpf/288-dependency-properties-v-wpf





WPF предоставляет два способа валидации значения свойства: **ValidateValueCallback** и **CoerceValueCallback**

3.Для чего и как используют делегат ValidateValueCallback?

**ValidateValueCallback**: делегат, который возвращает true, если значение проходит валидацию, и false - если не проходит

4.Для чего и как используют делегат CoerceValueCallback?

**CoerceValueCallback**: делегат, который может подкорректировать уже существующее значение свойства, если оно вдруг не попадает в диапазон допустимых значений

5.Какие типы маршрутизируемых событий есть в WPF (поясните каждый)?

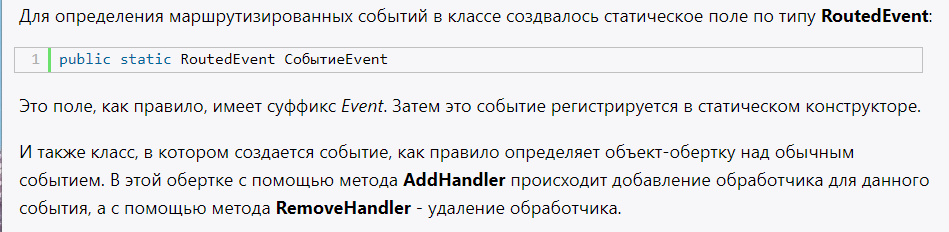
https://metanit.com/sharp/wpf/6.php

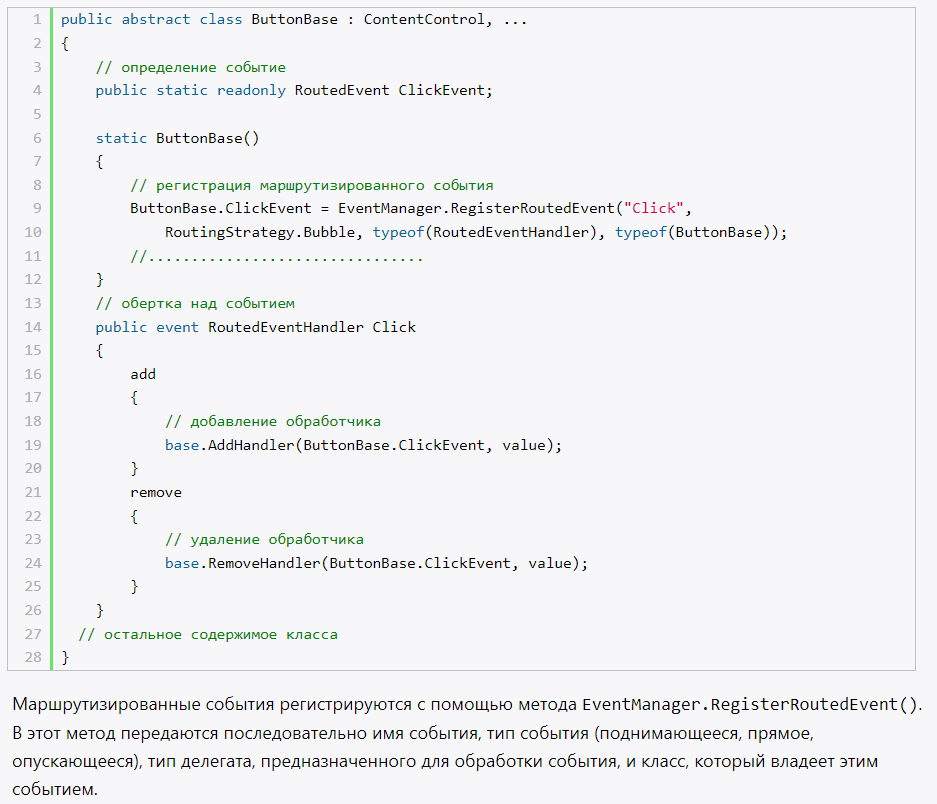
**Маршрутизируемые** **события** — это **события** с большими транспортными возможностями: они могут туннелироваться вниз и распространяться пузырьками наверх по дереву элементов и по пути запускать обработчики **событий**. **Маршрутизируемые** **события** позволяют обработать **событие** в одном элементе (например, в метке), хотя оно возникло в другом (например, в изображении внутри этой метки).

**Маршрутизируемые события делятся на три вида:**

* **Прямые** (direct events) - они возникают и отрабытывают на одном элементе и никуда дальше не передаются. Действуют как обычные события.
* **Поднимающиеся** (bubbling events) - возникают на одном элементе, а потом передаются дальше к родителю - элементу-контейнеру и далее, пока не достигнет наивысшего родителя в дереве элементов.
* **Опускающиеся, туннельные** (tunneling events) - начинает отрабатывать в корневом элементе окна приложения и идет далее по вложенным элементам, пока не достигнет элемента, вызвавшего это событие.

6.Как создать RoutedEvent**?**





7.Поясните концепцию Command в WPF? В чем ее преимущества?

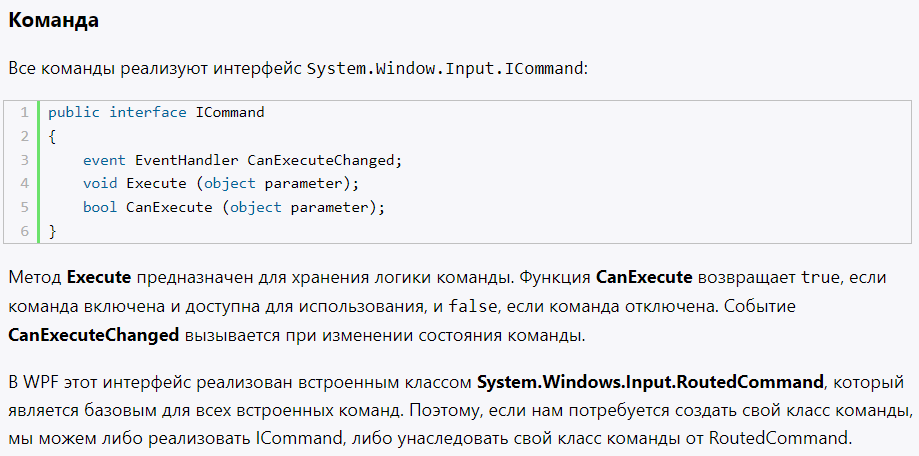
В WPF кроме обработки событий приложение может взаимодействовать с пользователем с помощью команд. **Команды** представляют механизм выполнения какой-нибудь задачи.

Использование команд помогает нам сократить объем кода и использовать одну и ту же команду для нескольких элементов управления в различных местах программы. Таким образом, команды позволяют абстрагировать набор действий от конкретных событий конкретных элементов.

Модель команд в WPF состоит из четырех аспектов:

* Сама **команда**, которая представляем выполняемую задачу
* **Привязка команд**, которая связывает команду с определенной логикой приложения
* **Источник команды** - элемент пользовательского интерфейса, который запускает команду (например, кнопка, по нажатию который выполняется команда)
* **Цель команды** - элемент интерфейса, на котором выполняется команда

1. Как используются команды?



9. Как создать RoutedUICommand?

WPF уже обладает большим набором встроенных команд. Все они представляют объекты класса **RoutedUICommand.**

семь групп встроенных команд:

