Основи на Windows Presentation Foundation

Мирослав Недялков

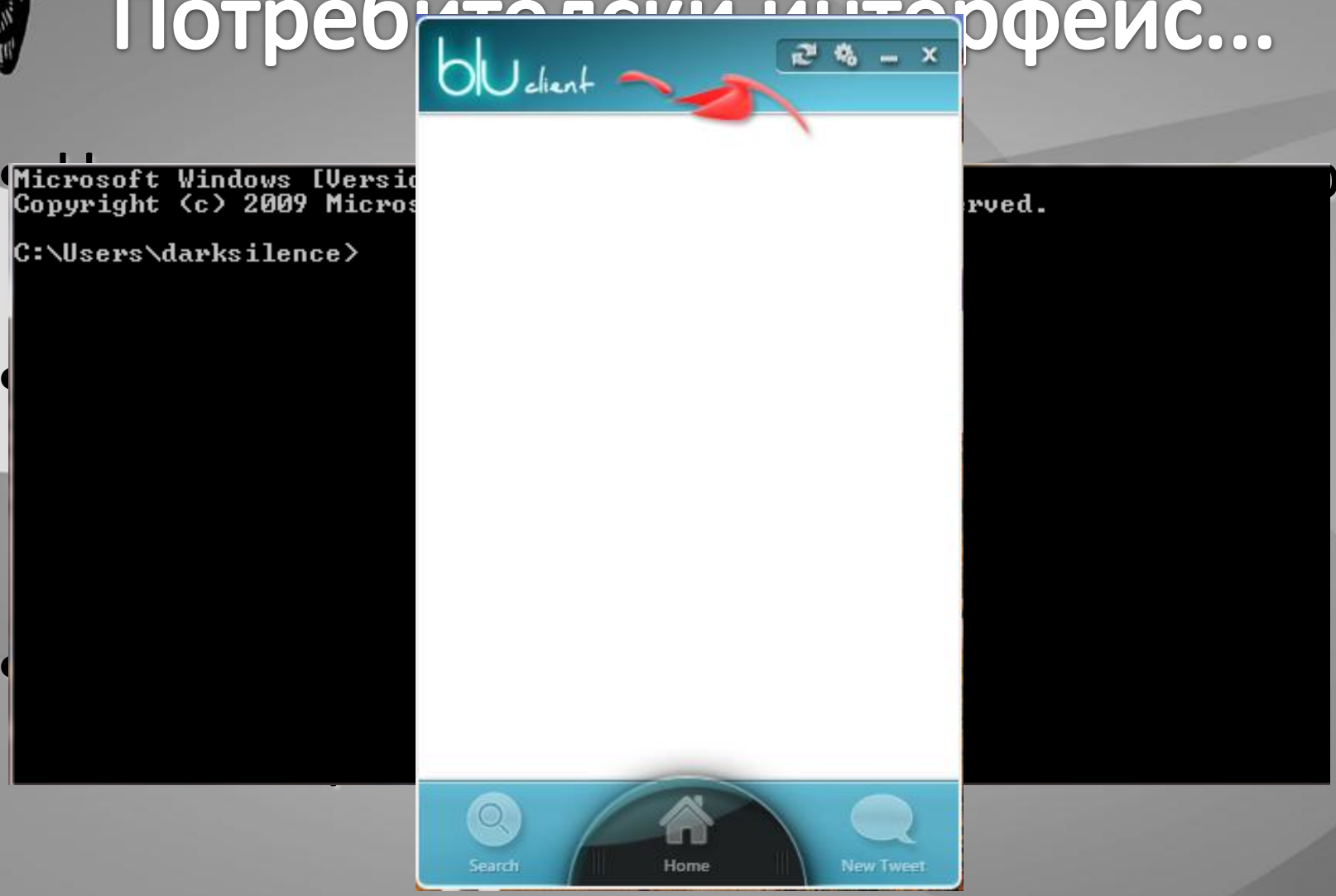
Съдържание

- Технологии за създаване на потребителски интерфейс
- WPF – какво ново на хоризонта
- Декларативен потребителски интерфейс?
- Демо – WPF приложение

Съдържание

- Езикът XAML
  - Синтаксис
  - Възможности
- Основни класове в WPF
- Основни контроли в WPF

Потребителски интерфейс...



Подходи

- **GDI** (Graphics Device Interface) –въведена с Windows 1.0
- **GDI+** - въведено с Windows XP

Други подходи

- **OpenGL** – платформа за разбработка на 2D и 3D GUI и игри
- **DirectX**
  - въведен с Windows 95
  - API за създаване на 2D и 3D GUI и игри за Windows
  - Xbox е базиран изцяло на DirectX.
- **А да смесим?**

WPF

Платформа за създаване на богат потребителски интерфейс

WPF

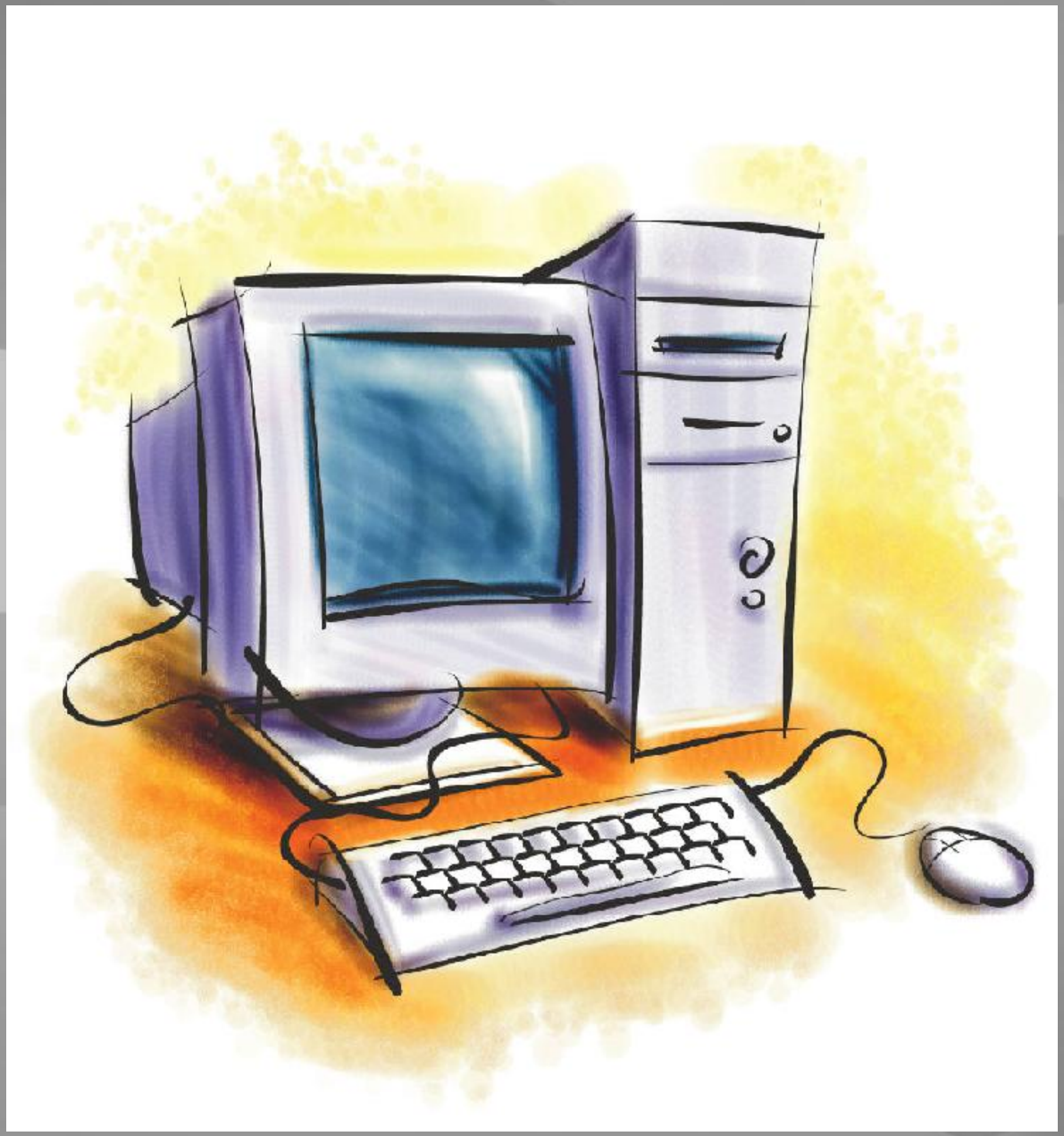
- **Нова рендираща система**, базирана на DirectX
  - Осигурява поддръжка на хардуерно ускорение
  - Поддръжка на ефекти
  - Вградена поддръжка на 3D
- **Добра интеграция** на 2D и 3D UI

WPF

- Независим от резолюцията!
- Декларативно програмиране – XAML
- Добри инструменти за разрабтване на GUI – Blend
- Стилове и теми
- Вградени анимации

WPF

- Композиране на елементи
- Разделяне на данните (**Data**) от поведението (**Behavior**)
- Лесно разпространение
  - ClickOnce
  - Browser (XBAP)

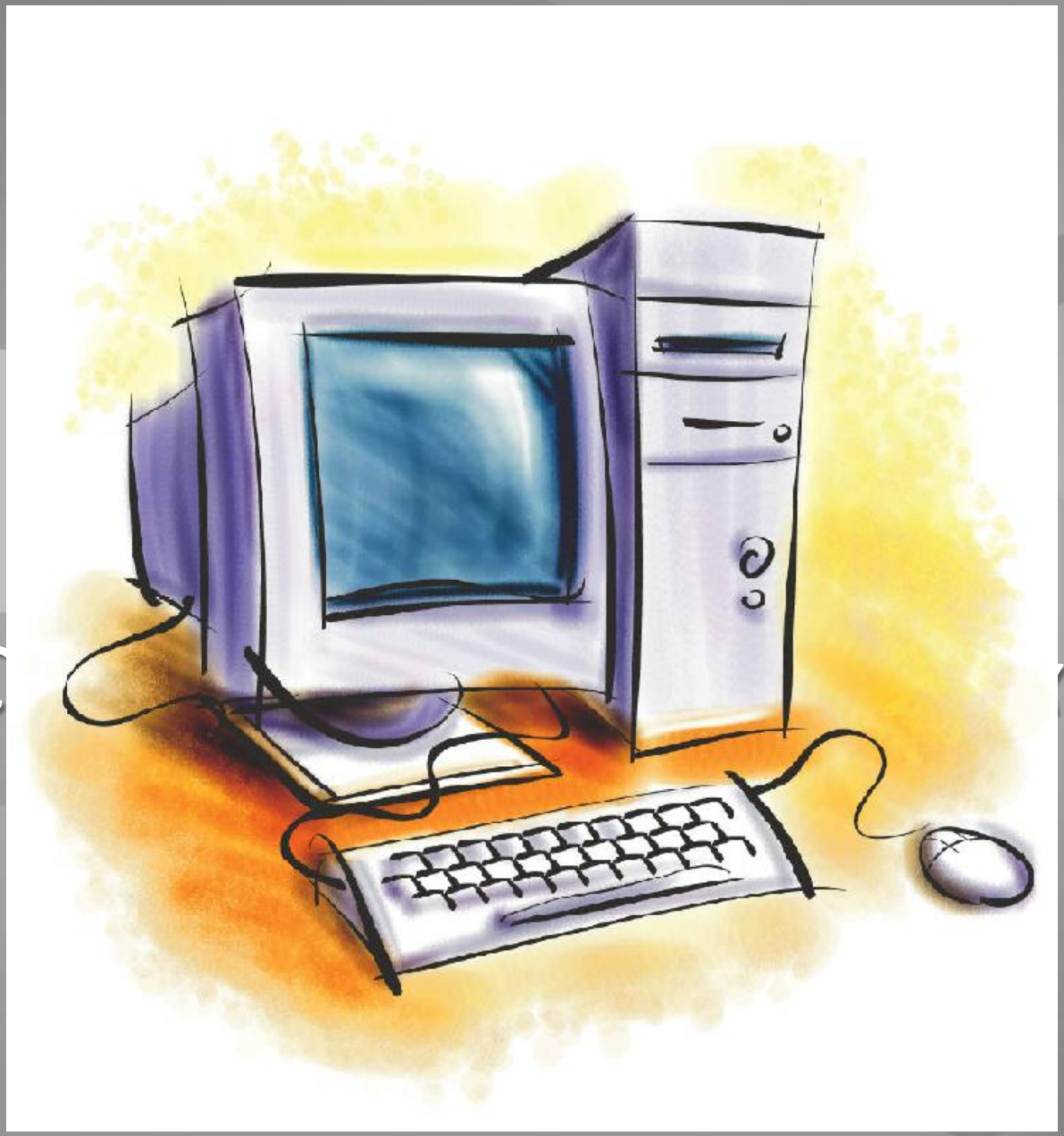


XAML

- XML базиран език => тагове и атрибути
- Декларативен
  - Разделение на описание от поведение
- Описва .NET обекти
- Използва се за описване на потребителски интерфейс – работи с класовете от WPF платформата

Защо ни е декларативен език?

- Четимост и разбираемост
- Улеснява поддръжката
- Тулируемост – възможност за обработка с инзтруенти (като Blend)



How to XAML?

Създаване на обекти

- Как да създадем бутон:

```
<!--XAML-->

<Button Content="OK" />
```

Атрибут – променя стойност на свойство

Таг – класа на обекта

Еквивалентно

```
//C#

Button b = new Button() {Content = "Ok"  };
```

Property Елементи

- Не създават нови обекти
- Присвоява стойност на свойство

```
<Button>
    <Button.Content>
        <Rectangle Height="40"
                   Width="40"
                   Fill="Black" />
    </Button.Content>
</Button>
```

Таг – класа, собственик на свойството и името на свойството

Background…

- XAML:

```
<Button Background="Black"/>
```

- Нормално да си мислим, че на C# е:

```
Button b = new Button();
b.Background = “Black”; //Wrong
```

- Типа на Background свойството е **SolidColorBrush**! Как WPF разбира какво означава низа Black?

TypeConverter

- **Низ от XAML атрибут -> .NET тип**
- Метаданни на свойство/тип
- Вградените типове и свойства имат
  - SolidColorBrush
  - Thickness
  - Enum
  - Color
  - GridLength (ще разберете по-късно ;))

Markup Extensions

- За сложни стойности на атрибути
- Оградени са с {}
- Предоставят съкратен синтаксис
- Пример:

```
<Button Background="{StaticResource backGround}"/>
```

- Наследяват класа MarkupExtension

Класа StaticResource и параметър ResourceKey със стойност “backGround”

Влагане на Елементи

- Content Property

```
<Button>
    <Button.Content>
        <Rectangle Height="40"
                   Width="40"
                   Fill="Black" />
    </Button.Content>
</Button>
```

```
<Button>
        <Rectangle Height="40"
                   Width="40"
                   Fill="Black" />
</Button>
```

Магията...

- ContentPropertyAttribute

```
[ContentProperty("Content")]
public class Button
{
        ...
}
```

- Вложените елементи също set-ват свойства!
- Типове
  - Стойност (1 вложен таг)
  - Колекция (много вложени тагове)

Attached Property

```
<Button Grid.Row="2"
        Grid.Column="1"/>
```

- Grid.Row и Grid.Column не са свойства на Button
- Те са закачени/прикачени свойства (attached properties) на Grid
- Закачат се на определени обекти
- Синтаксис – <име на клас>.<име на свойство>

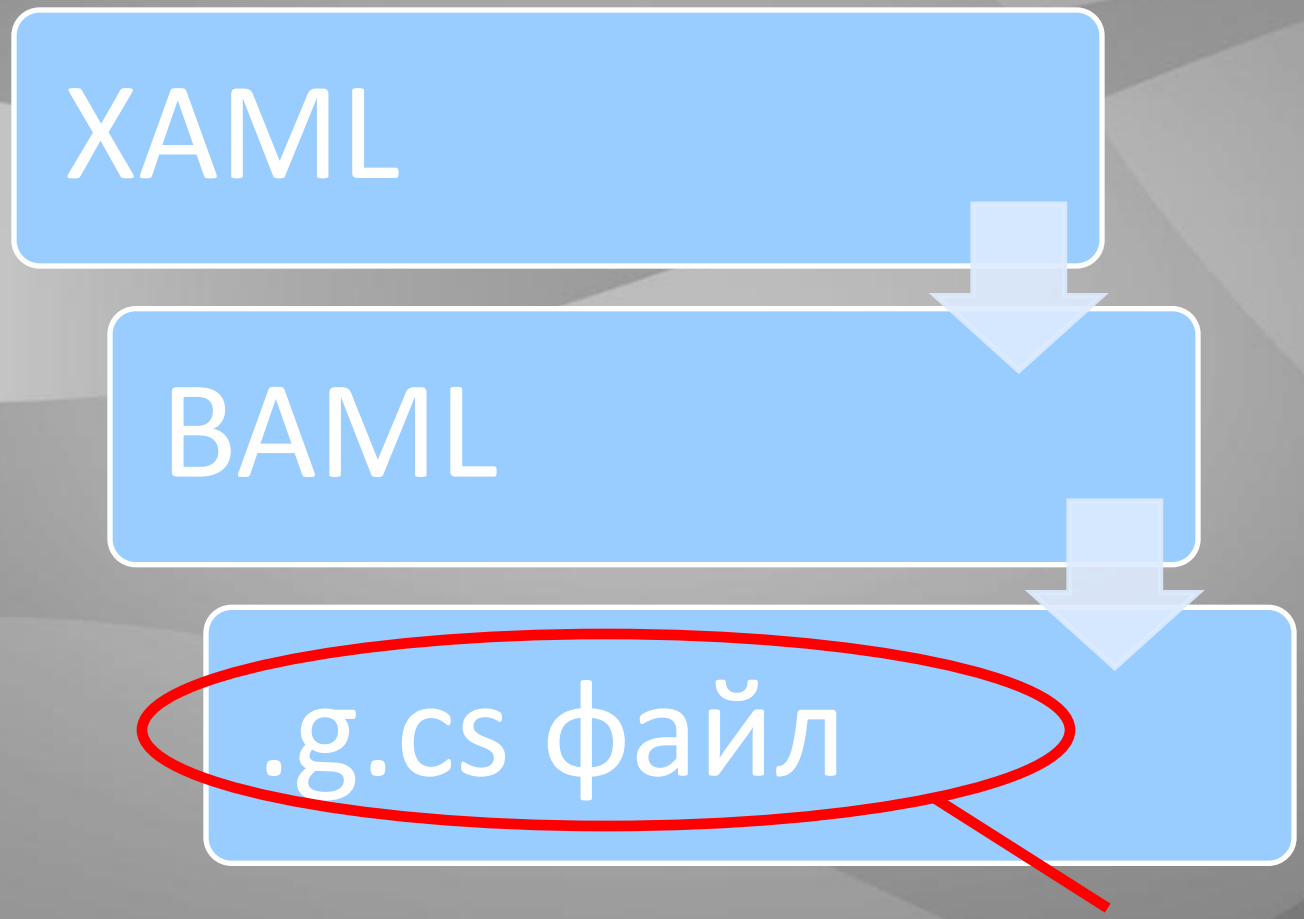
Attached Property

- Могат да се използват като атрибути и елементи
- Разширяват функционалността на съществуващи елементи
- Използват се за закачане на допълнителни данни или маркиране
- Те са **DependencyProerty**-та

Code-Behind file

- “Закачен” за XAML файл.
- Класът му наследява root елемента в XAML
- Използва се най-често за обработка на събитията и инициализация на UI от XAML
- **EventHandlers**
- **x:Name**

Compilation



Основни класове на WPF

- DispatcherObject
- DependencyObject
- Visual
- UIElement
- FrameworkElement и Control
- Shapes и Text, ContentPresenter
- Control, ContentControl, UserControl
- Window

Основни класове на WPF

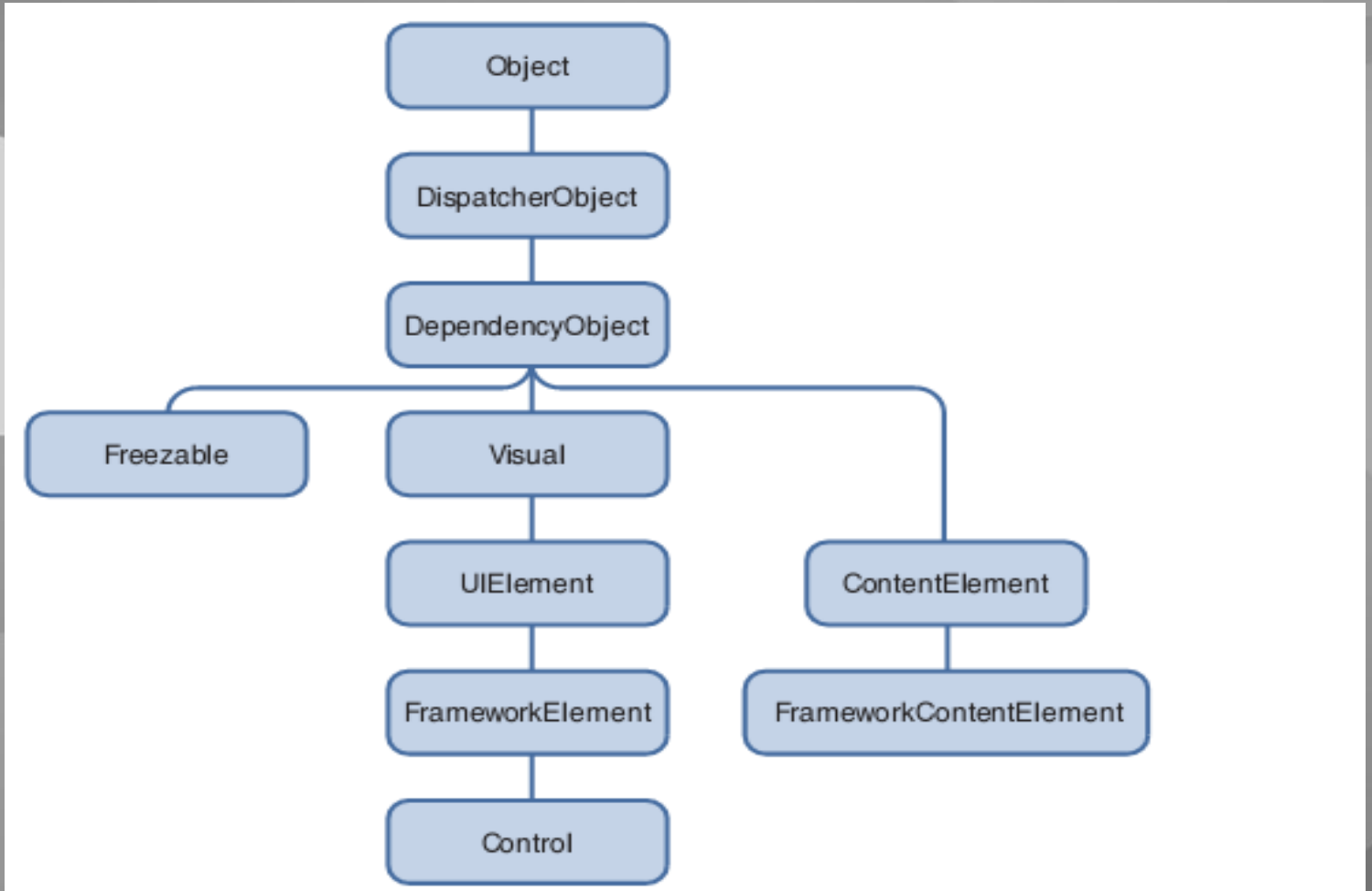


FIGURE 3.9 The core classes in the WPF Presentation Framework.

Контроли в WPF

- Content Controls
  - Buttons
    - Button
    - RepeatButton
    - ToggleButton
    - CheckBox
    - RadioButton

Контроли в WPF

- Simple Containers
  - Label
  - ToolTip
  - Frame
- Header Containers
  - GroupBox
  - Expander

Контроли в WPF

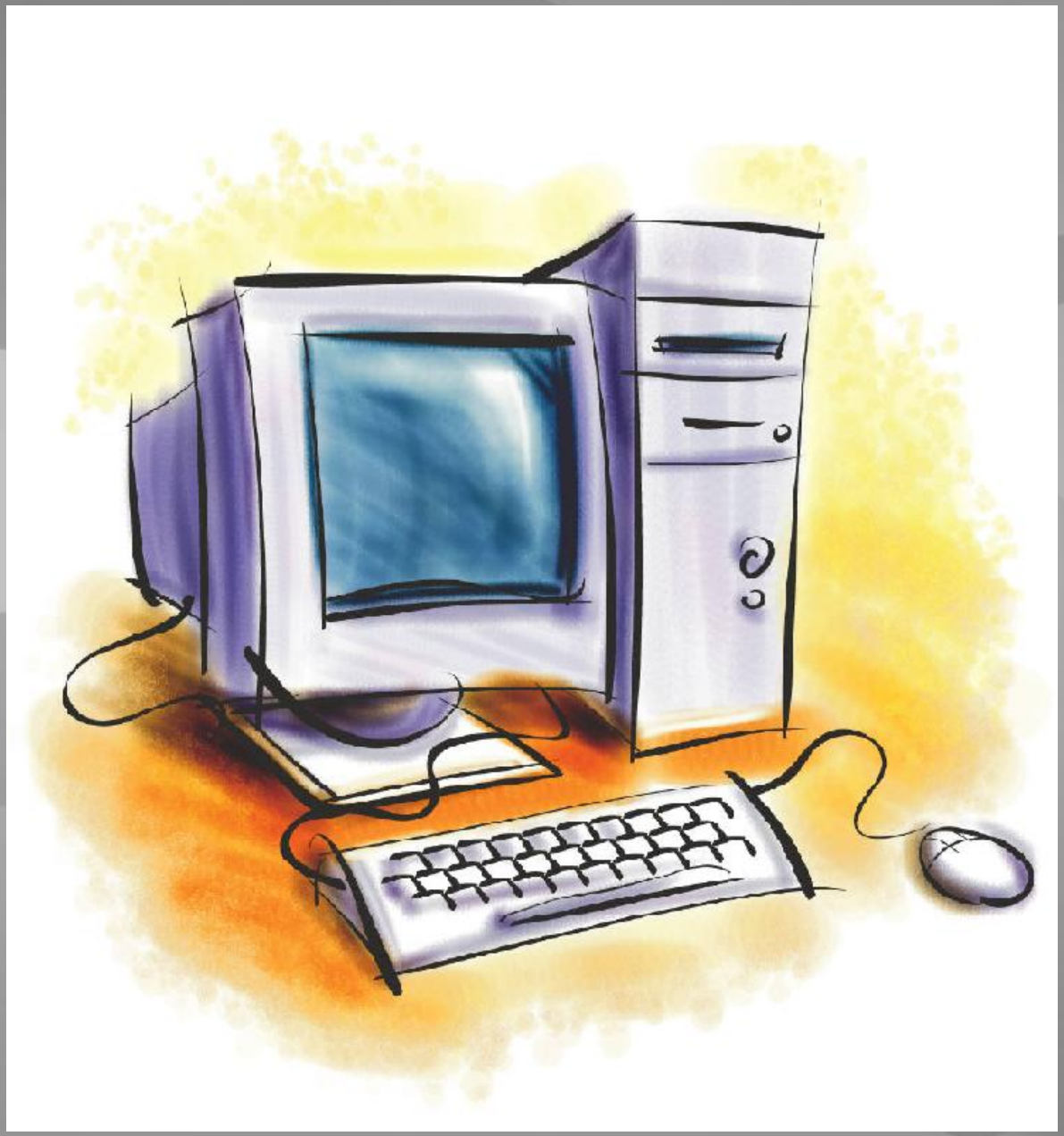
- Items Controls
  - ItemsControl
  - ListBox
  - ListView
  - ComboBox
  - Menus
    - Menu
    - ContextMenu
  - TreeView
  - ToolBar
  - StatusBar

Контроли в WPF

- Range Controls
  - ProgressBar
  - Slider
  - Text Controls

Контроли в WPF

- За показване/редактиране на текст
  - TextBox
  - TextBlock
  - RichTextBox
  - PasswordBox



Въпроси

?