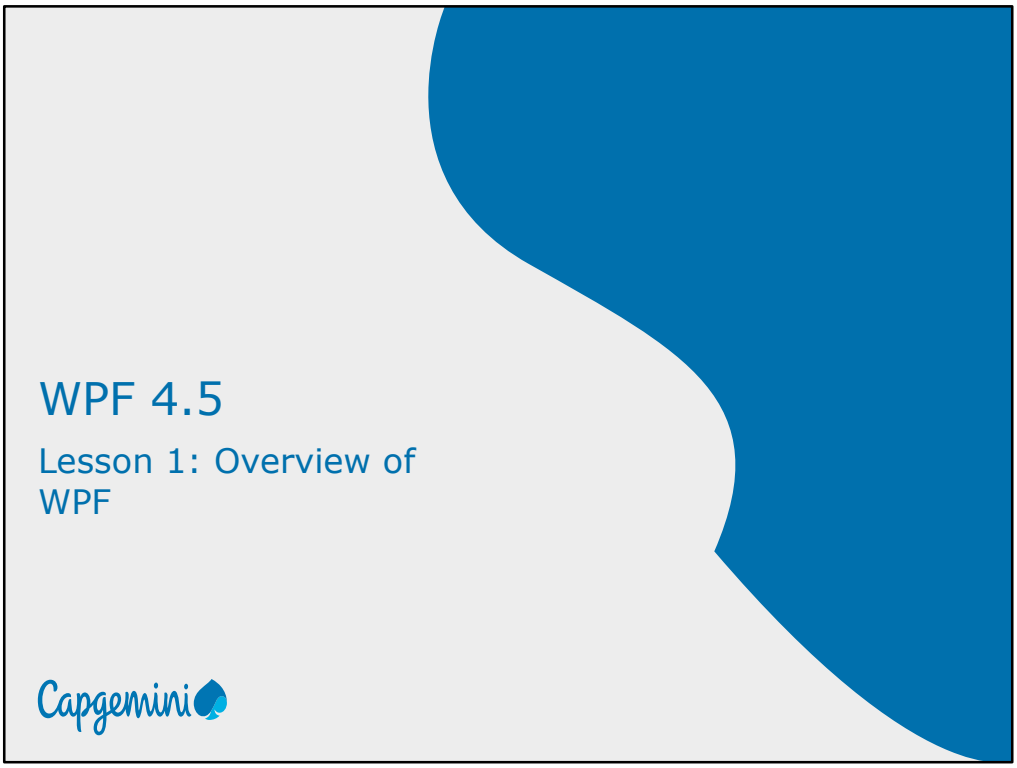

WPF 4.5
Lesson 1: Overview of WPF
Capgemini

Lesson Objectives

- On completion of this lesson, you will be able to:
  - Define WPF
  - Describe the world before WPF
  - State significance of WPF
  - Describe WPF design principles
  - Key features of WPF
  - Architecture of WPF

1.1: Introduction

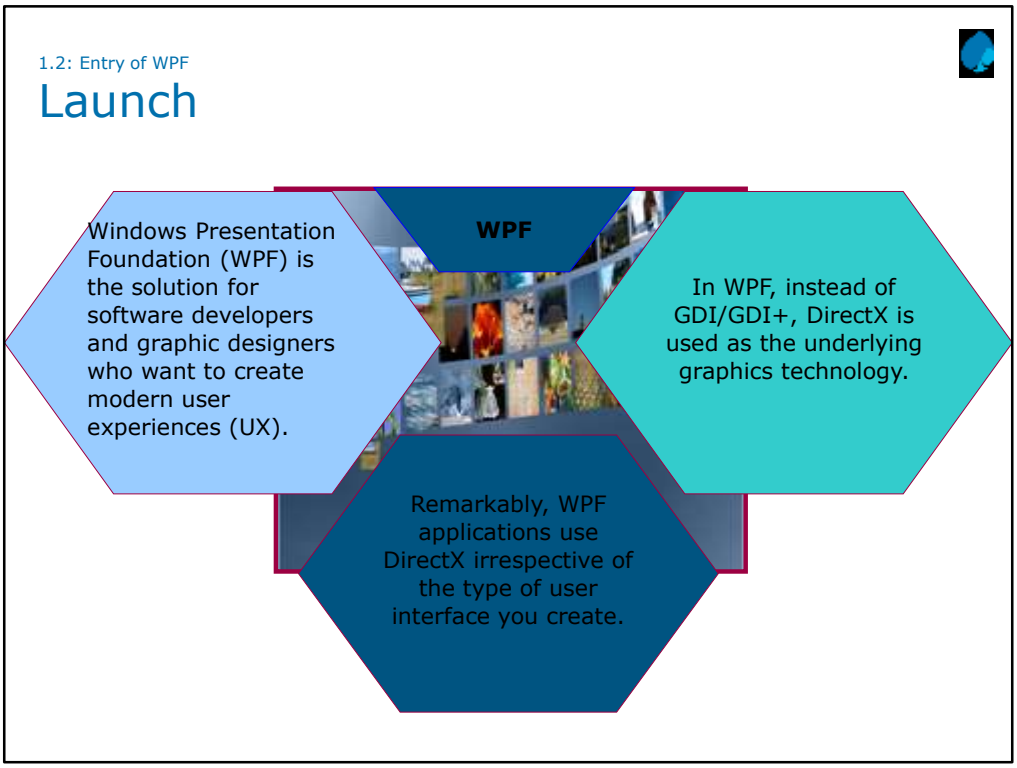
Windows Presentation Foundation (WPF)

The Windows Presentation Foundation (WPF) is an entirely new graphical display system for Windows.

WPF is a hardware-accelerated system designed for .NET, it is influenced by modern display technologies such as HTML and Flash.

WPF provides a holistic means for combining user interface, 2D and 3D graphics, documents, and digital media.

It is the most radical change to hit Windows user interfaces since Windows 95.

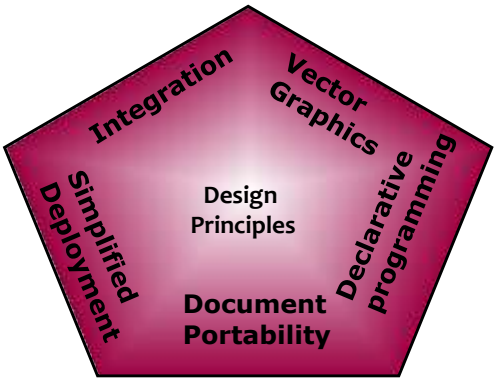


Launch of WPF:
WPF changes all the limitations of previous graphic technologies. The reasons are as follows:
In WPF, the underlying graphics technology isn't GDI/GDI+. Instead, it's DirectX.
Remarkably, WPF applications use DirectX no matter what type of user interface you create. That means that whether you're designing complex three-dimensional graphics or just drawing buttons and plain text, all the drawing work travels through the DirectX pipeline. As a result, even the most mundane business applications can use rich effects such as transparency and antialiasing.
You also benefit from hardware acceleration, which simply means DirectX hands off as much work as possible to the GPU (graphics processing unit), which is the dedicated processor on the video card.

1.3: WPF Design Principles

Overview

- To enhance UX and empower both designers and developers, WPF has been built under the following design principles



1.4: Declarative Programming

WPF and XAML

- WPF introduces a new XML-based language to represent UI and user interaction, known as XAML (eXtensible Application Markup Language)
- XAML allows applications to dynamically parse and manipulate UI elements at either compile-time or runtime, providing a flexible model for UI composition

Combination of WPF and XAML:
The combination of WPF and XAML is similar to using HTML to define a user interface, but with an incredible range of expressiveness. This expressiveness even extends beyond the boundaries of user interfaces. WPF uses XAML as a document format, a representation of 3D models, and more. The result is, graphic designers are empowered to contribute directly to the look and feel of applications, as well as some behavior for which you would typically expect to have to write code.

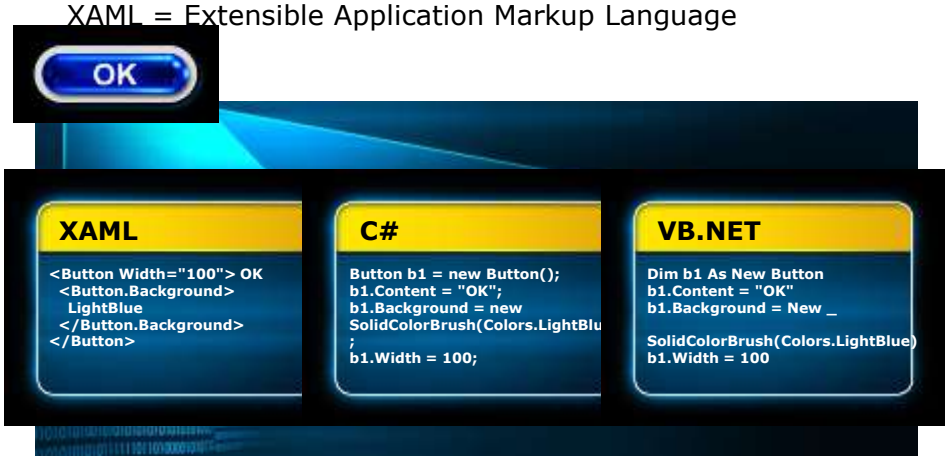
Following the success of ASP.NET, XAML follows the code-behind model, allowing designers and developers to work in parallel and seamlessly combine their work to create a compelling UX. With the aid of design-time tools such as the Visual Designer for Windows Presentation Foundation add-in for Visual Studio 2005 or Visual studio 2008 (along with express edition), the experience of developing XAML-based applications resembles that of WinForms development.
Moreover, designers accustomed to visual tools such as Macromedia Flash 8 Professional can quickly ramp-up to building XAML-based solutions using visual design tools such as Microsoft Expression Blend.

1.5: Declarative Programming

XAML

- Easily toolable, declarative markup
- Code and content are separate
- Can be rendered in the browser / standalone application

XAML = Extensible Application Markup Language



Declarative Programming Through XAML:
Extensive Application Markup Language has the following advantages:
It is an easily toolable declarative markup.

It builds applications in simple declarative statements.

It can be used for any CLR object hierarchy.

It's code and content are separate.

It streamlines collaboration between designers and developers.

Developers add business logic, while designers design discretely.

It can be rendered in the browser (as part of a web page) or as a standalone application.

XAML:
WPF introduces a new role to the rich client software development team – that of professional designer. Monotonous buttons and poorly designed applications are out of trend. With the declarative programming model enabled by XAML, you can split off presentation and logic in the same way as with a web application. XAML is a markup language that is inherently toolable, allowing for designers and developers to use independent tools.

1.6: Deployment

Simplified Deployment

- WPF applications can be deployed as standalone applications or as web-based applications hosted in Internet Explorer
- WPF provides options for deploying traditional Windows applications (using Windows Installer or ClickOnce) or hosting applications in a web browser

Simplified Deployment:
WPF applications can be deployed as standalone applications or as web-based applications hosted in Internet Explorer. As with smart client applications, web-based WPF applications operate in a partialtrust sandbox, which protects the client computer against applications with malicious purpose.
Furthermore, WPF applications hosted in Internet Explorer can exploit the capabilities of local client hardware, providing a rich web experience with 3D, digital media, and more, which is the best argument for web-based applications available today.

1.7: Key Features of WPF

DPI

- Device Independent Pixel (DPI)
  - WPF introduces Device Independent DPI Settings for the applications built with it
  - Windows forms application uses pixel based approach so with changing DPI settings, each control will change its size and look
  - WPF addresses this issue and makes it independent of DPI settings of the computer

Let say you have drawn a box, which is 1 inch long in 96 dpi screen. Now if you see the same application in 120 dpi settings the box will appear smaller. This is because the things that we see on the screen are totally dependent on dpi settings.

In case of WPF, this is modified to density based approach. That means when the density of pixel is modified, the elements will adjust them accordingly and hence the pixel of WPF application is Device Independent Pixel. The size of the control remains same in case of WPF application and it takes more pixel in case of 120 DPI application to adjust the size properly.

1.7: Key Features of WPF

Supports Graphics and Animation

- Built-In Support for Graphics and Animation
  - WPF applications as being rendered within DirectX environment, it has major support of graphics and animation capabilities
  - A separate sets of classes are there which specifically deals with animation effects and graphics
  - The graphics that you draw over the screen is also Vector based and are object oriented

1.7: Key Features of WPF

Flexibility in Defining Styles

- Redefine Styles and Control Templates
  - In addition to graphics and animation capabilities, WPF also comes with a huge flexibility to define the styles and ControlTemplates
  - Style based technique (like CSS) are a set of definitions which defines how the controls will look like when it is rendered on the screen
  - In case of WPF, Styles are completely separated from the UIElement
  - Once you define a style, you can change the look and feel of any control by just putting the style on the element

1.7: Key Features of WPF

Resource based Approach

- Resource based Approach for every control
  - In WPF, you can store styles, controls, animations, and even any object as resource.
  - Thus each resource will be declared once when the form loads itself, and you may associate them to the controls.
  - You can maintain a full hierarchy of styles in separate file called ResourceDictionary, from which styles for the whole application will be applied.
  - Thus, WPF application could be themed very easily.

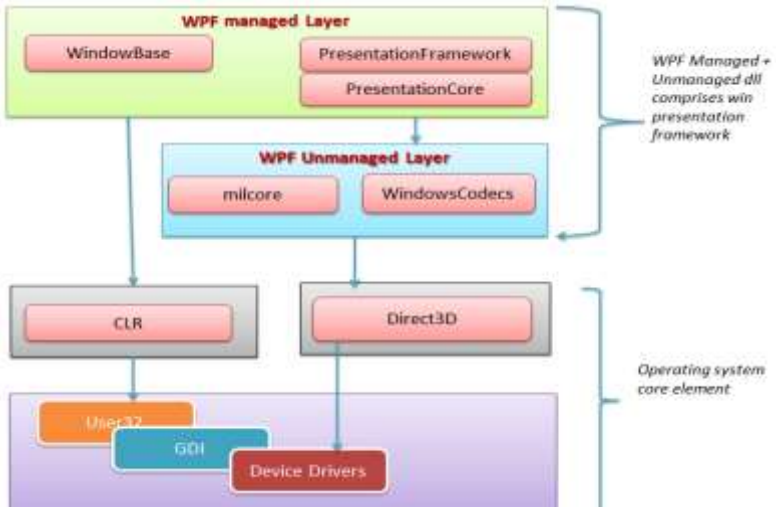
1.7: Key Features of WPF

Dependency Properties

- New Property System & Binding Capabilities
  - Every element of WPF defines a large number of dependency properties
  - The dependency properties have strong capabilities than the normal properties

1.8: Introduction to WPF Architecture

Concept



WPF is actually a set of assemblies that build up the entire framework.
These assemblies can be categorized as

- Managed Layer
- UnManaged Layer
- Core API

1.9: Basics of WPF Architecture

Overview

- Managed Layer :
  - Managed layer of WPF is built using a number of assemblies
  - These assemblies build up the WPF framework, communicates with lower level unmanaged API to render its content

1.10: Components of WPF

Features

- PresentationFramework.dll :
  - Creates the top level elements like layout panels, controls, windows, styles etc.
- PresentationCore.dll :
  - It holds base types such as UIElement, Visual from which all shapes and controls are Derived in PresentationFramework.dll
- WindowsBase.dll :
  - They hold even more basic elements which are capable to be used outside the WPF environment like Dispatcher object, Dependency Objects

1.11: WPF Architecture

Features (Contd…)

- Unmanaged Layer (milcore.dll):
  - The unmanaged layer of WPF is called milcore or Media Integration Library Core
  - It basically translates the WPF higher level objects like layout panels, buttons, animation etc into textures that Direct3D expects
  - It is the main rendering engine of WPF

1.11: WPF Architecture

Features (Contd…)

- WindowsCodecs.dll :
  - This is another low level API which is used for imaging support in WPF applications
  - WindowsCodecs.dll comprises of a number of codecs which encodes / decodes images into vector graphics that would be rendered into WPF screen

Summary

- In this lesson, you have learnt the following:
  - WPF aims to combine the best attributes of systems such as DirectX (3D and hardware acceleration), Windows Forms (developer productivity), Adobe Flash (powerful animation support), and HTML (declarative markup and easy deployment)

1.12: WPF Capabilities

Demo

Review Questions

- In WPF, instead of GDI/GDI+, _____ is used as the underlying graphics technology
- WPF is actually a set of assemblies that build up the entire framework. These assemblies can be categorized as?
- Which dll is used to creates the top level elements like layout panels, controls, windows, styles etc.