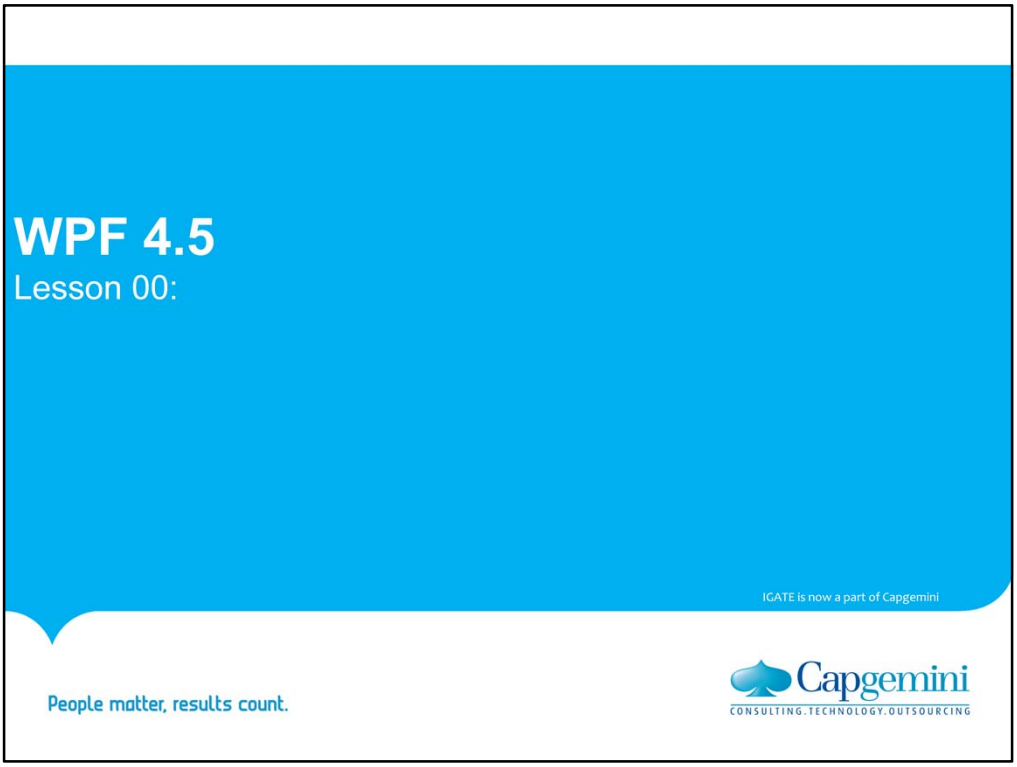


Course Goals and Non Goals

- Course Goals
  - To know what is WPF
  - To learn XAML and its features
  - To implement Styles, resources and Markup extentions
  - Work with WPF property System
  - Understand WPF Event Model
  - Work with WPF Data Binding
- Course Non Goals
  - Using tools like expression Blend

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

Pre-requisites

- .NET Framework with C# 5.0 & Winforms
- ADO.NET

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

Intended Audience

- Developers

Day Wise Schedule

- Day 1
  - Lesson 1: Overview of WPF 4.5
  - Lesson 2: Introduction to XAML
  - Lesson 3: UI Layouts and Controls
  - Lesson 4: Styles, Resources, and Shapes
- Day 2
  - Lesson 5: WPF Event Model
  - Lesson 6: Data Binding in WPF
  - Lesson 7: Live Shaping and Async in WPF

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

References

- http://msdn.microsoft.com/en-us/library/ms754130.aspx
- http://msdn.microsoft.com/en-us/library/aa970268.aspx

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Other Parallel Technology Areas

- Flex 3.0

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

Windows Presentation Foundation

Lesson 1: Overview of WPF

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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

1.1: Introduction

Windows Presentation Foundation (WPF)

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

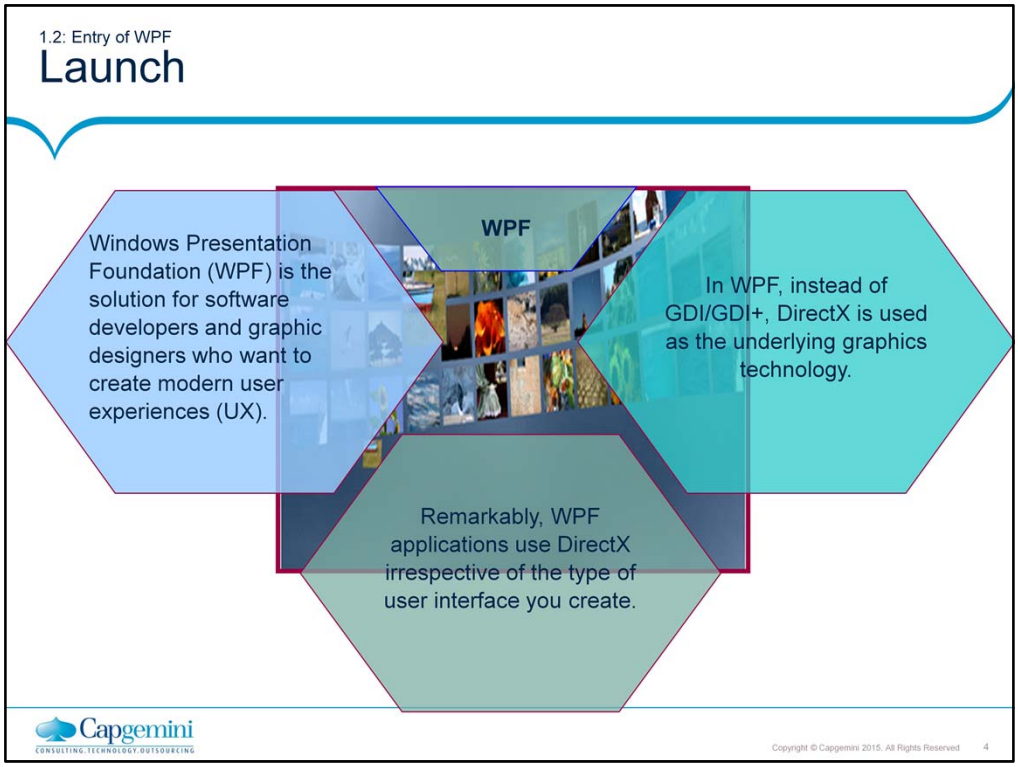
WPF is a hardware-accelerated system designed for .NET, it is influenced by modern display technologies such as HTML and Flash.

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.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

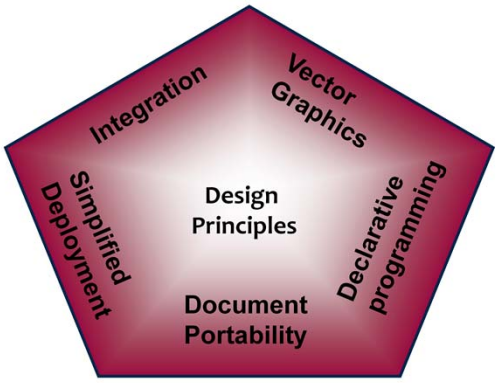


Launch of WPF:
WPF changes all the limitations of previous graphic technologies. The reasons are as follows:
In WPF, the underlying graphics technology isn't GDI/GDI+. Instead, it's DirectX.
Remarkably, WPF applications use DirectX no matter what type of user interface you create. That means that whether 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

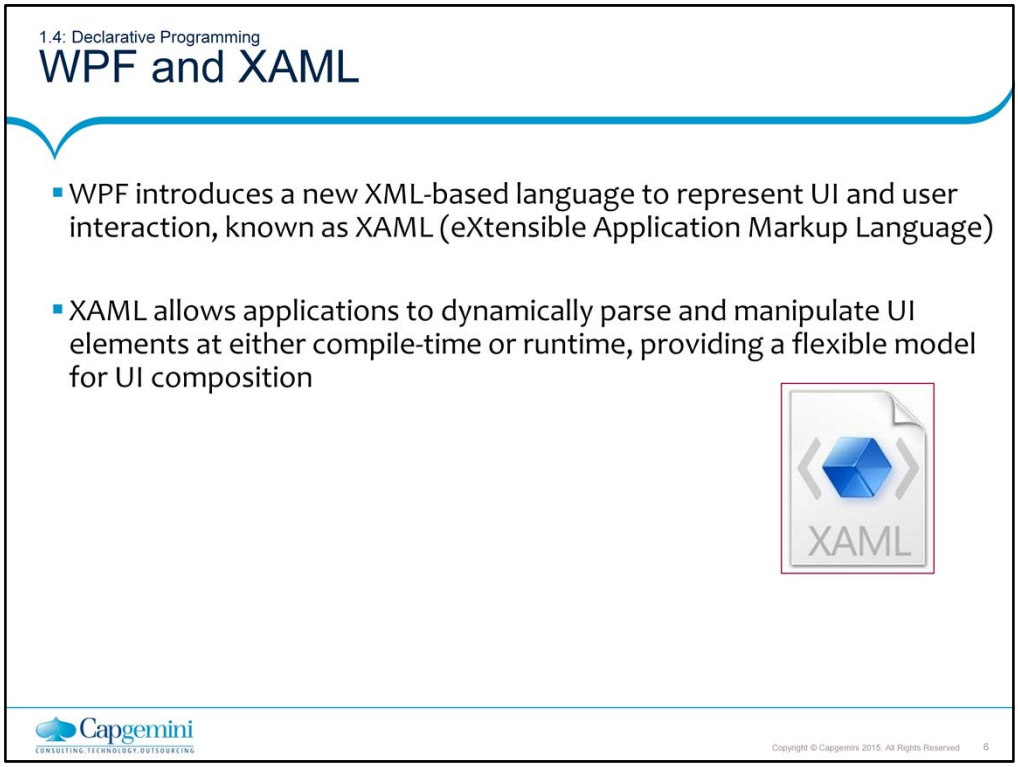
Overview

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



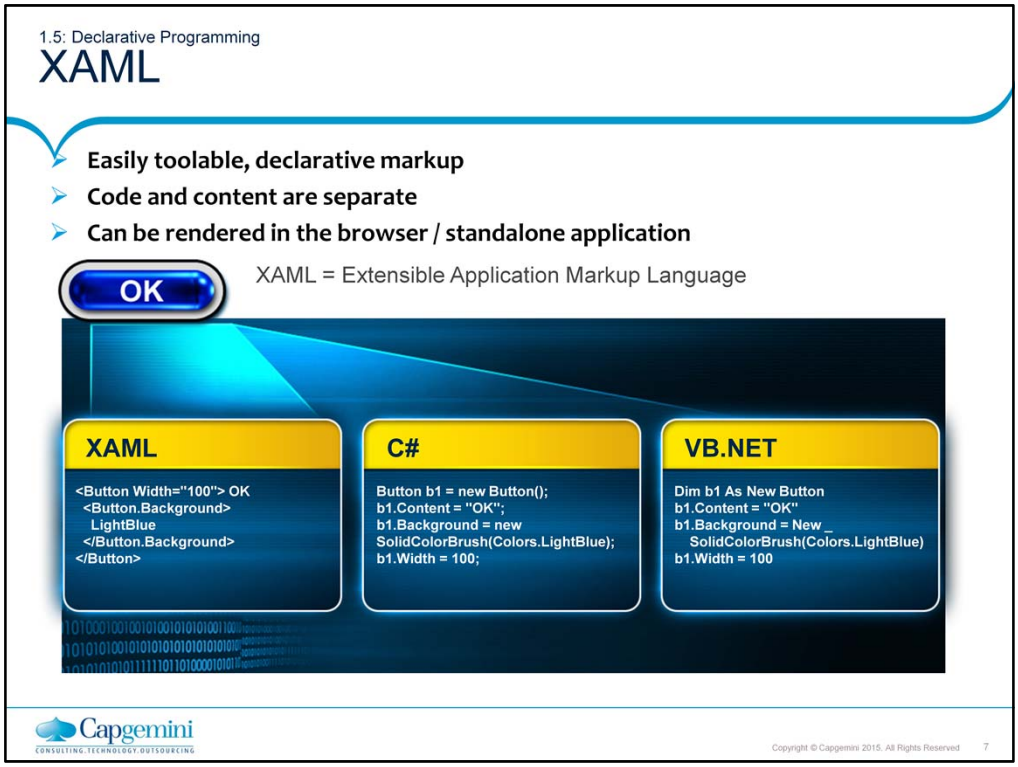
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5



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.



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

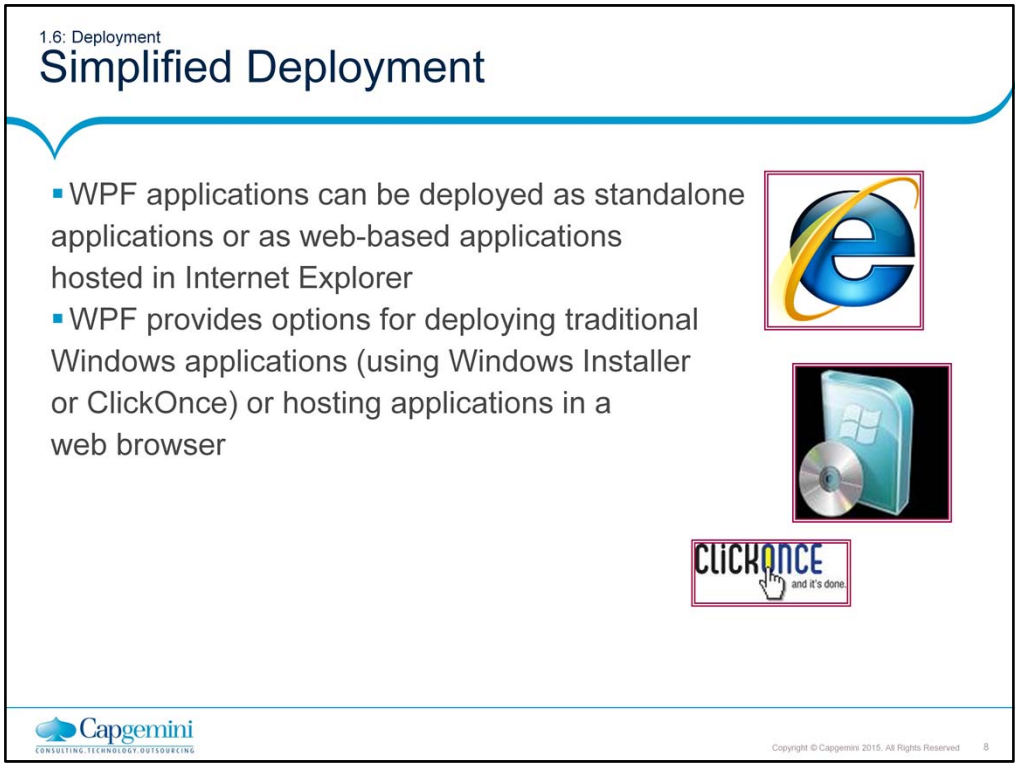
- It builds applications in simple declarative statements.
- It can be used for any CLR object hierarchy.

It's code and content are separate.

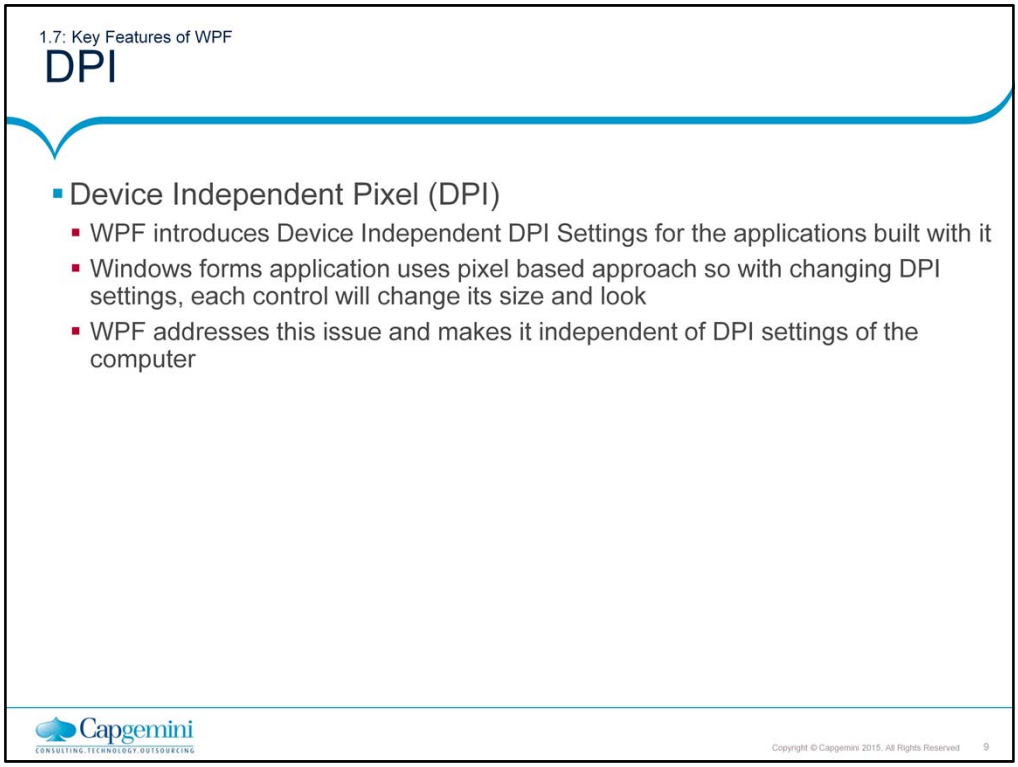
- It streamlines collaboration between designers and developers.
- Developers add business logic, while designers design discretely.

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.



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.



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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

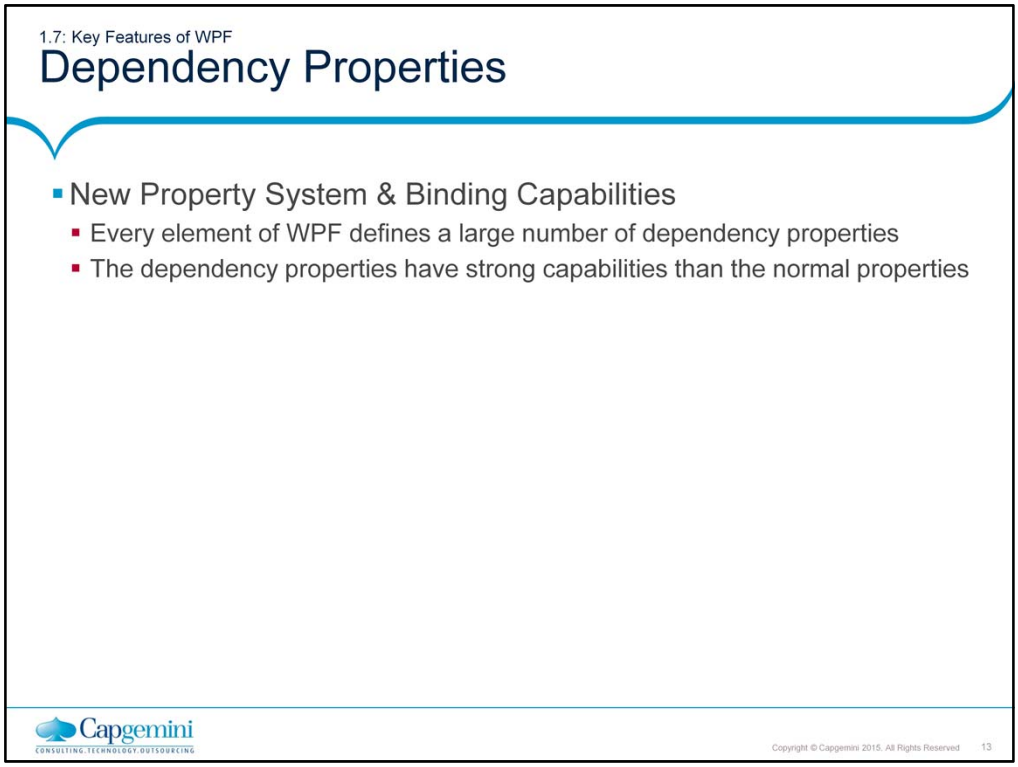
1.7: Key Features of WPF

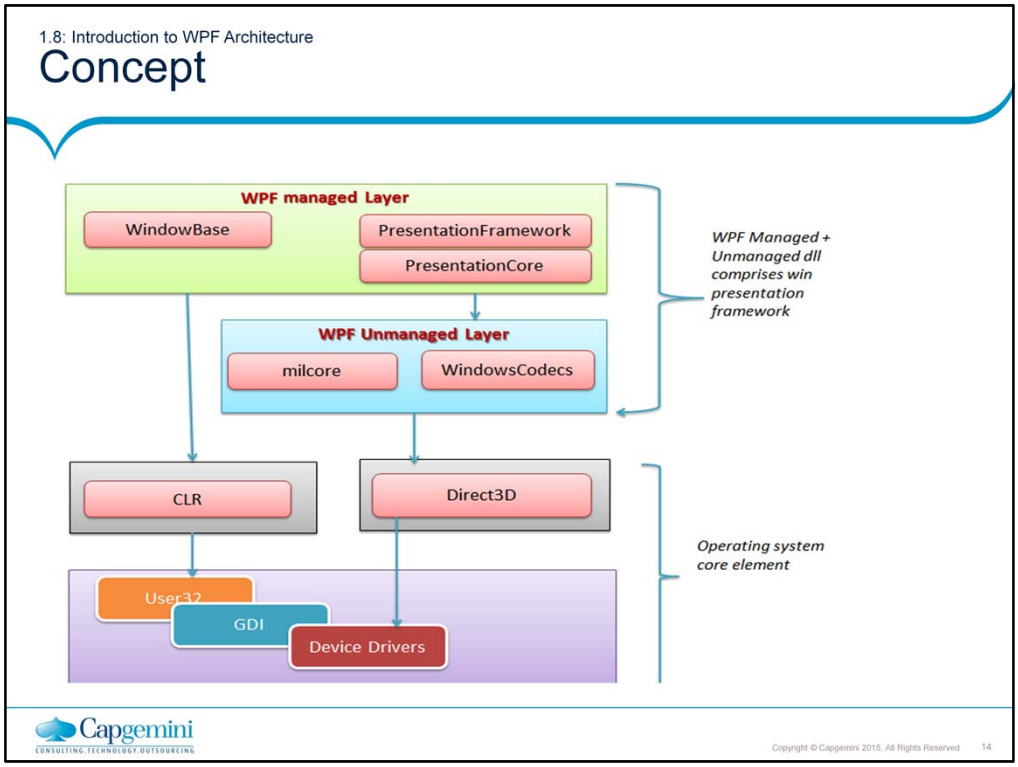
Resource based Approach

- 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.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12


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
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
13



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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

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)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

1.12: WPF Capabilities

Demo

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

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.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

Windows Presentation Foundation

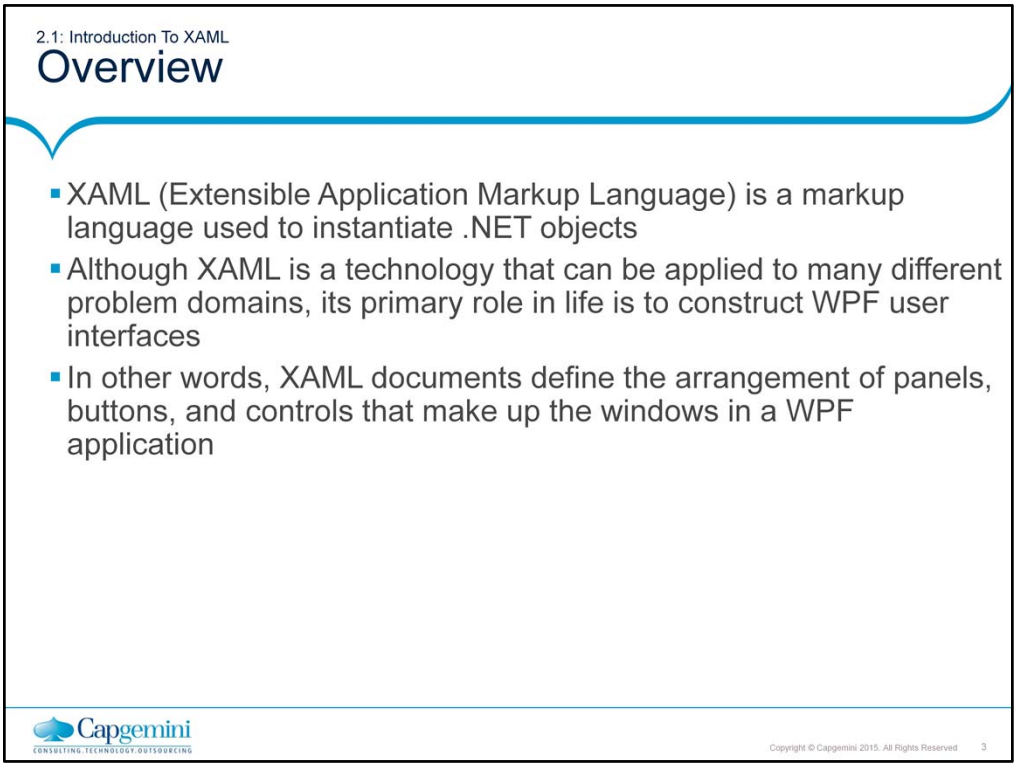
Lesson 2: Introduction To XAML And Its Features

Lesson Objectives

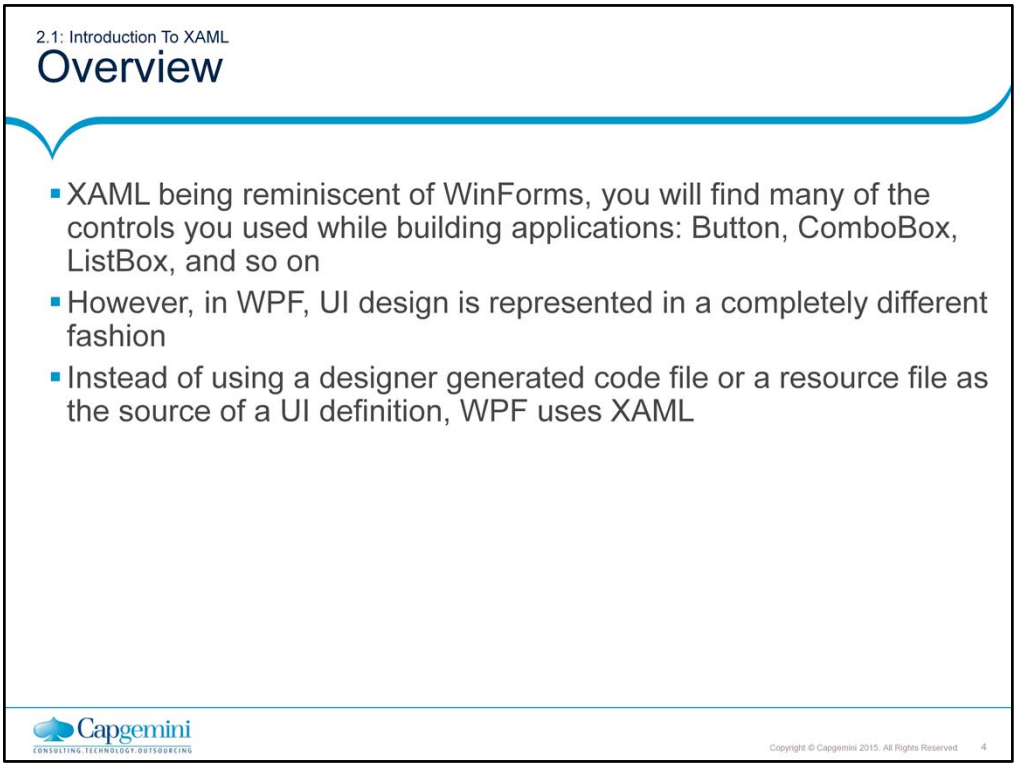
- In this lesson, you will learn:
  - Introduction to XAML
  - WPF Property System

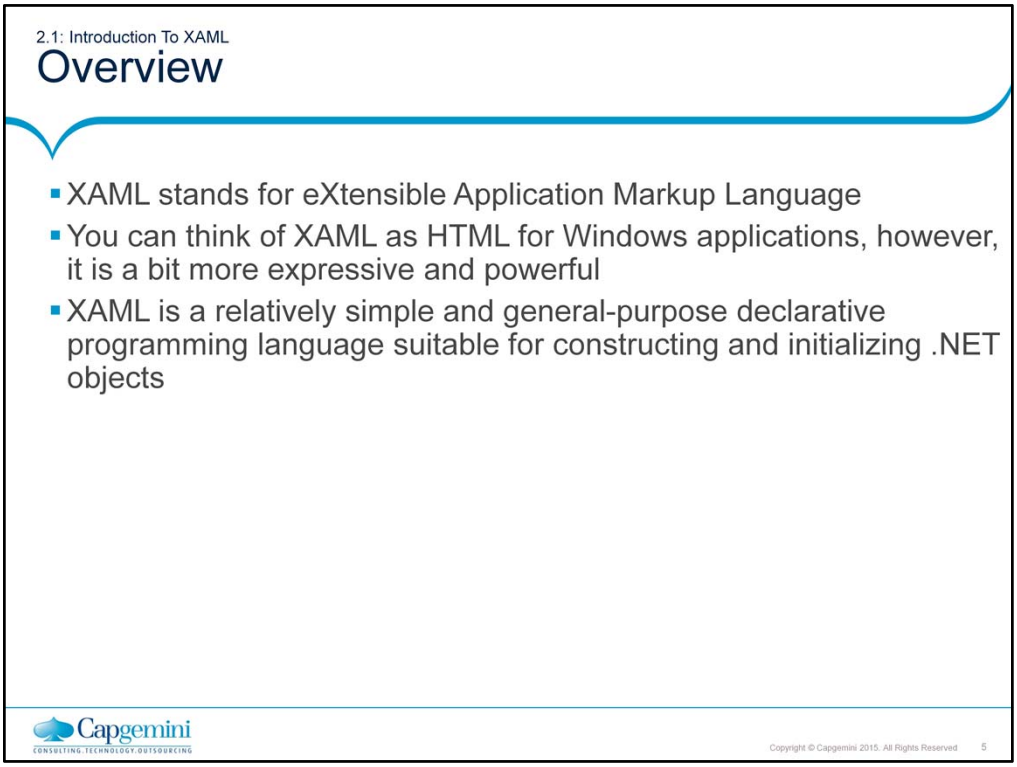
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

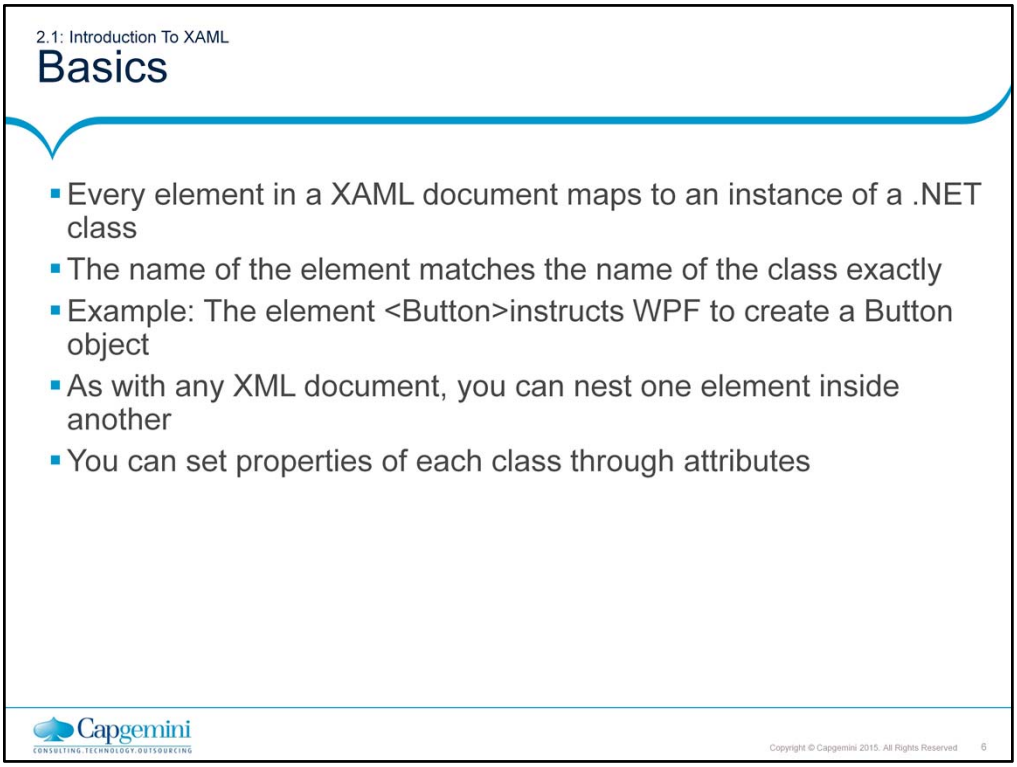


XAML:
XAML is a relatively simple and general-purpose declarative programming language suitable for constructing and initializing .NET objects.
The .NET Framework 3.0 includes a compiler and run-time parser for XAML, as well as a plug-in that enables you to view standalone WPF-based XAML files (sometimes called loose XAML pages) inside Internet Explorer.


2.1: Introduction To XAML
Overview
XAML being reminiscent of WinForms, you will find many of the controls you used while building applications: Button, ComboBox, ListBox, and so on
However, in WPF, UI design is represented in a completely different fashion
Instead of using a designer generated code file or a resource file as the source of a UI definition, WPF uses XAML
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
4



XAML:
XAML stands for eXtensible Application Markup Language.
It is an all-purpose XML-based language used for declaring object graphs, which are instantiated at runtime. XAML is used by WPF developers to declare the layout of a user interface (UI), and the resources used in that UI.



XAML:
The XAML standard is quite straightforward once you understand a few ground rules:

Every element in a XAML document maps to an instance of a .NET class.
The name of the element matches the name of the class exactly.

Example: The element <Button>instructs WPF to create a Button object.

As with any XML document, you can nest one element inside another. As you will see, XAML gives every class the flexibility to decide how it handles this situation. However, nesting is usually a way to express containment — in other words, if you find a Button element inside a Grid element, your user interface probably includes a Grid that contains a Button inside.

You can set the properties of each class through attributes. However, in some situations an attribute is not powerful enough to handle the job. In these cases, you will use nested tags with a special syntax.

2.1: Introduction To XAML

Basic Elements And Attributes

- The .NET Framework 3.0 includes a compiler and run-time parser for XAML. It also includes a plug-in that enables you to view standalone WPF-based XAML files (sometimes called loose XAML pages) inside Internet Explorer
- The XAML specification defines rules that map .NET namespaces, types, properties, and events into XML namespaces, elements, and attributes

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

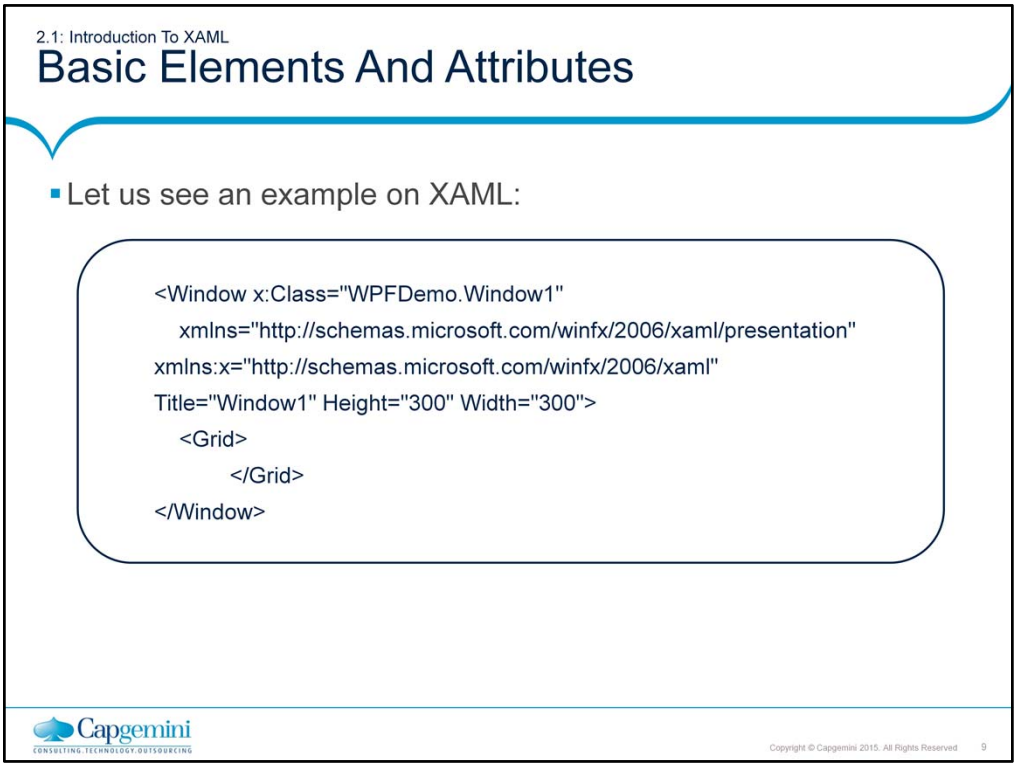
2.1: Introduction To XAML

Basic Elements And Attributes

- When you compile an application that contains XAML files, the markup gets converted into BAML
- BAML is a tokenized, binary representation of XAML
- This binary representation is then stored inside the application's resources and loaded as needed by WPF during the execution of your program
- The main advantage of this approach is that you get a faster UI load time by reading a binary stream than through parsing XML

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8



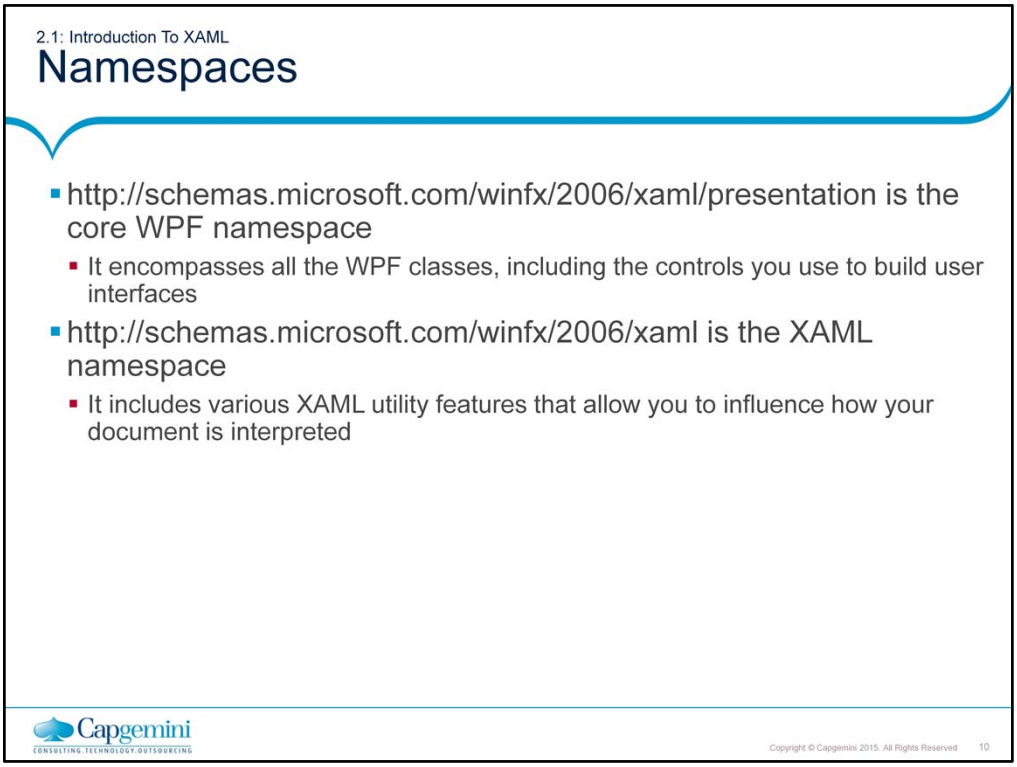
XAML:
Example of XAML:
This document includes only two elements:

- the top-level Window element, which represents the entire window,
- the Grid, in which you can place all your controls

Although you can use any top-level element, WPF applications rely on just a few:

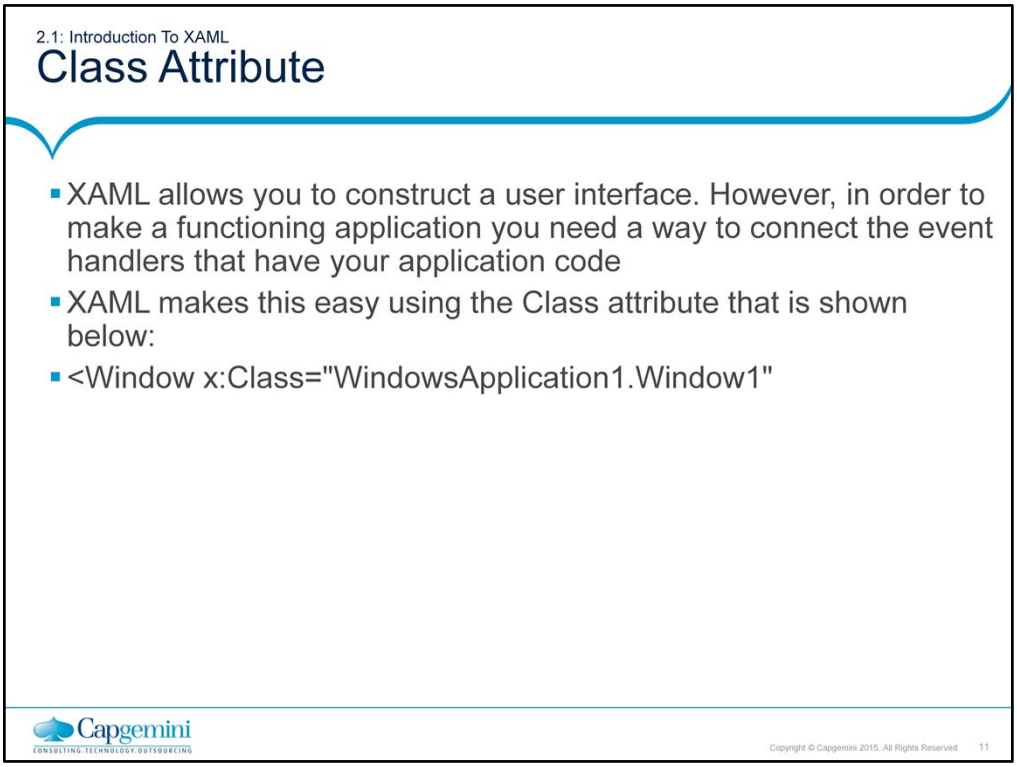
- Window
- Page (which is similar to Window, but used for navigable applications)
- Application (which defines application resources and startup settings)

As in all XML documents, there can only be one top-level element. In the previous example, as soon as you close the Window element with the </Window> tag, you end the document. No more content can follow.

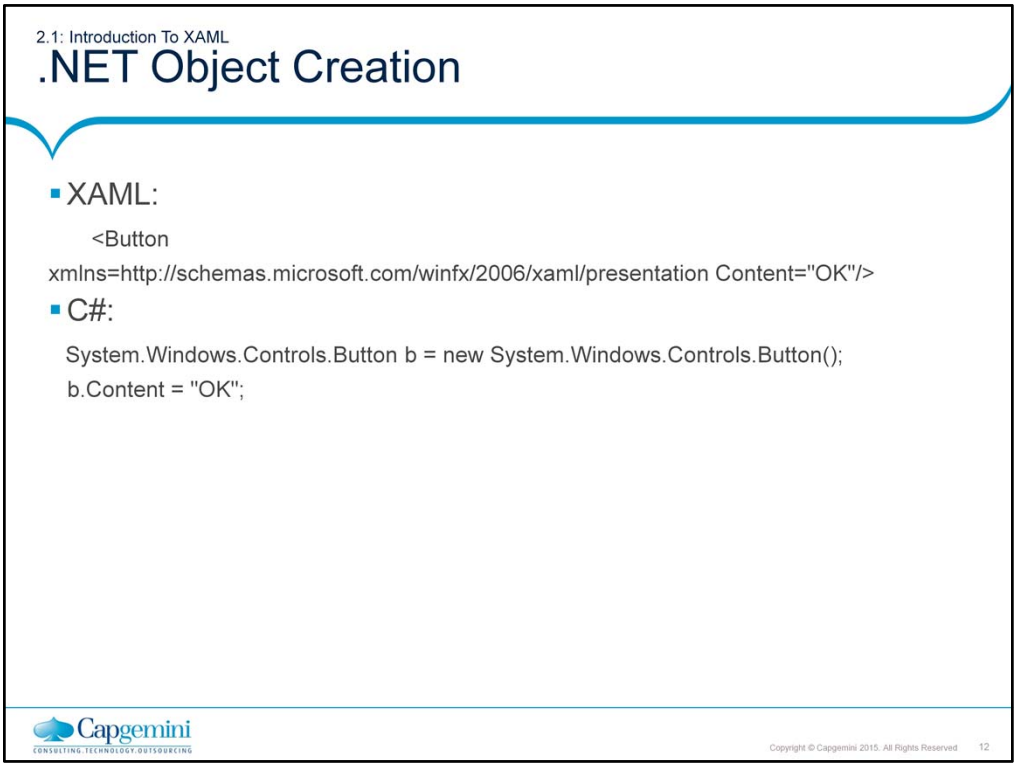


XAML:
XAML Namespaces:
The most mysterious part about comparing the previous XAML example with the equivalent C# examples is how the XML namespace http://schemas.microsoft.com/winfx/2006/xaml/presentation maps to the .NET namespace System.Windows.Controls. Resultantly the mapping to XML and other WPF namespaces is hardcoded inside the WPF assemblies with several instances of an XmlnsDefinitionAttribute custom attribute.
The root object element in a XAML file must specify at least one XML namespace that is used to qualify itself and any child elements. You can declare additional XML namespaces (on the root or on children). However, each one must be given a distinct prefix to be used on any identifiers from that namespace.
Example: WPF XAML files typically use a second namespace with the prefix x (denoted by using xmlns:x instead of just xmlns):

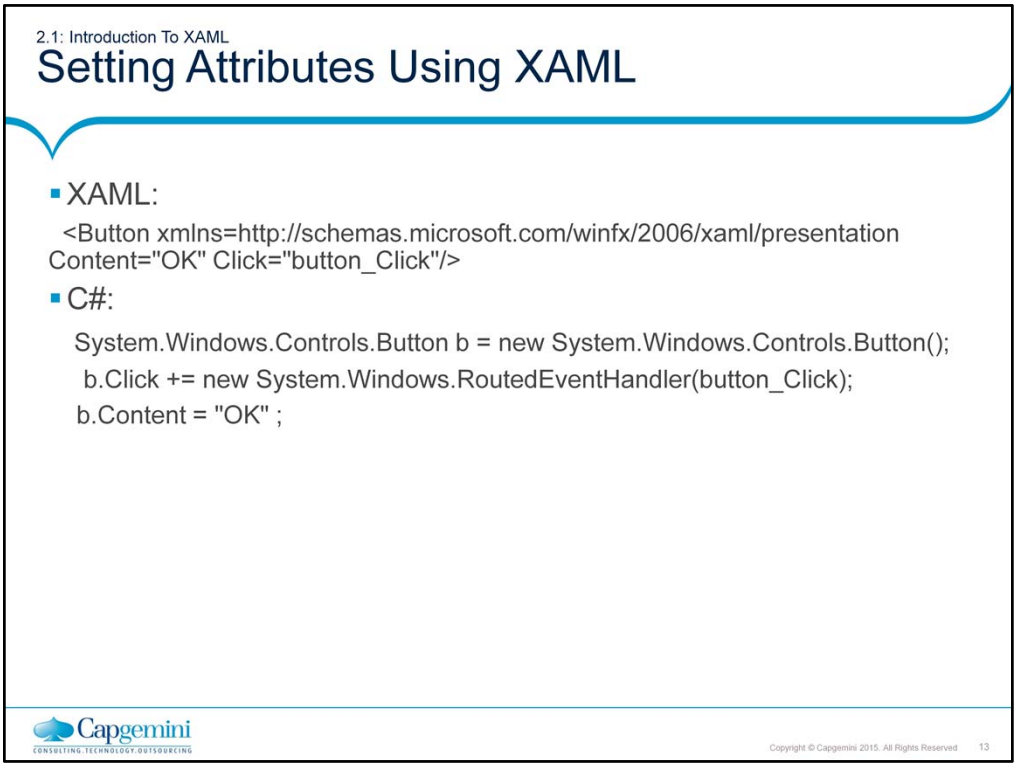
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
This is the XAML language namespace, which maps to types in the System.Windows.Markup namespace but also defines some special directives for the XAML compiler or parser.



XAML:
The Code-Behind Class:
The x namespace prefix places the Class attribute in the XAML namespace. This is a more general part of the XAML language. In fact, the Class attribute tells the XAML parser to generate a new class with the specified name. That class derives from the class that is named by the XML element.
In other words, this example creates a new class named WindowsApplication1.Window1, which derives from the base Window class.



XAML:
.NET Object Creation:
Declaring an XML element in XAML (known as an object element) is equivalent to instantiating the corresponding .NET object (always via a default constructor).



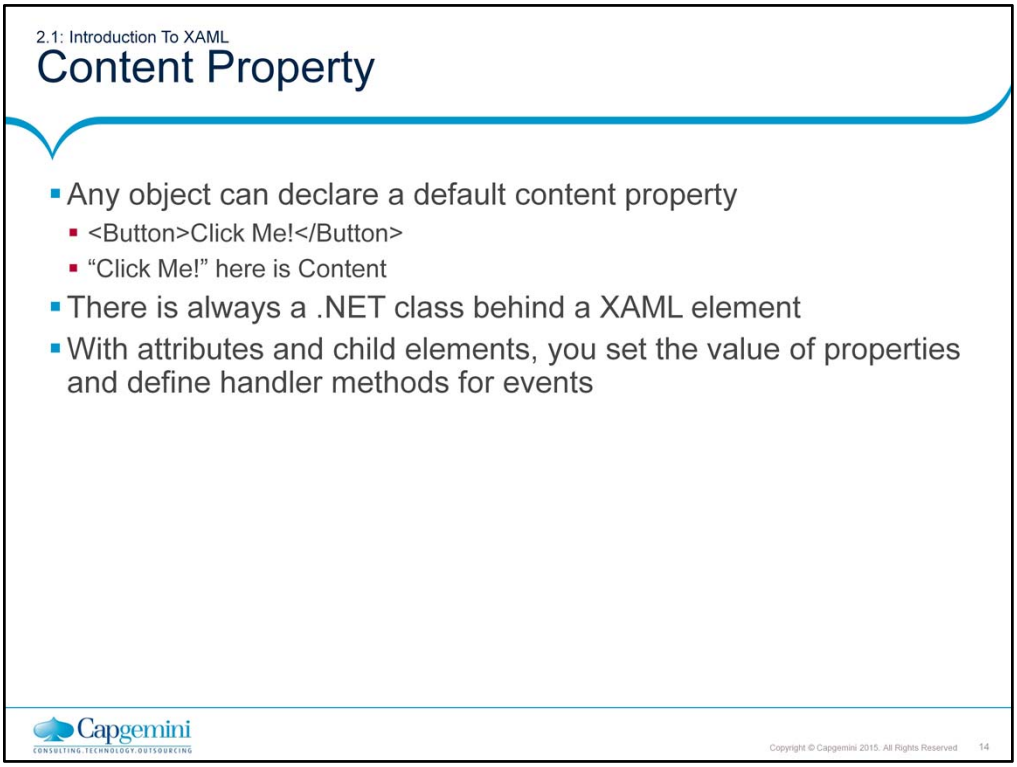
XAML:

Setting Attributes using XAML:

Setting an attribute on the object element is equivalent to setting a property of the same name (called a property attribute) or hooking up a handler for an event of the same name (called an event attribute).

XAML is no more than a special type of CLR object serialization. It was intentionally architected to serialize CLR object graphs into an XML representation that is both verbose and human readable.

This makes it possible for developers to easily edit the markup by hand and enable the creation of powerful graphical tools that could generate markup for you behind the scenes.



WPF Design Principles:
Properties:
Consider the Text attribute on the Window element:

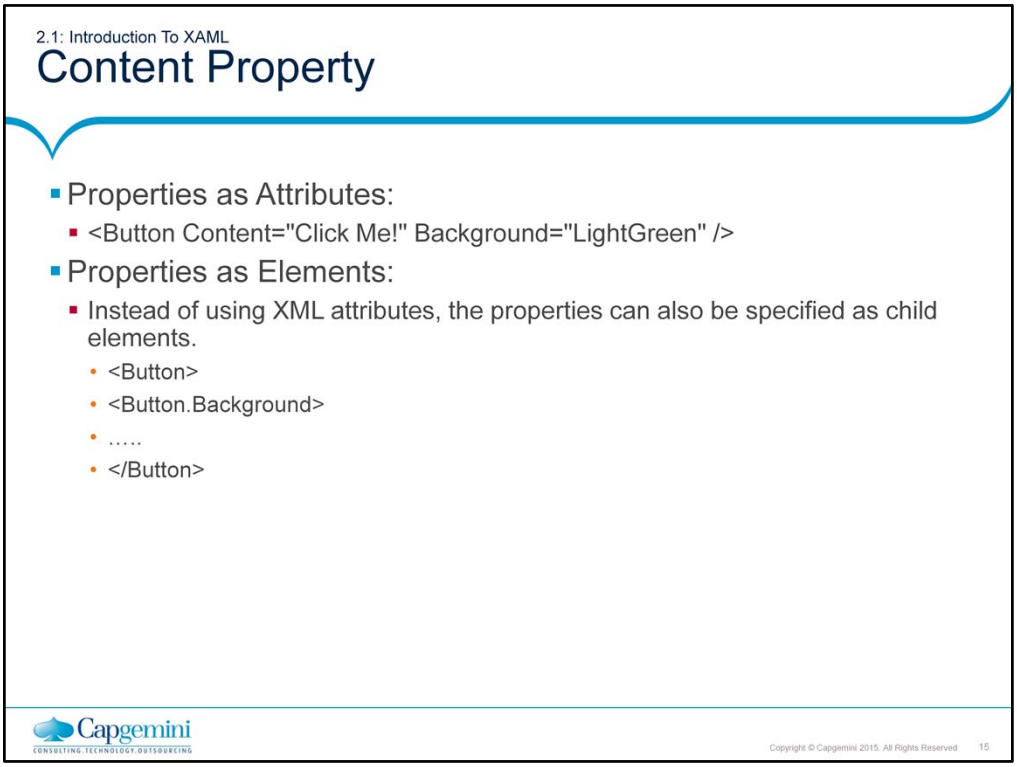
```
<Window xmlns="http://schemas.microsoft.com/winfx/avalon/2005"
xmlns:x="http://schemas.microsoft.com/winfx/xaml/2005"
x:Class="XamlProj.Window1" Text="Main Window">
```

Text: This attribute has no namespace qualifier. In XAML, unqualified attributes usually correspond to properties on the .NET object to which the element refers. (They can also refer to events, as we will see later). The Text attribute indicates that when an instance of this generated XamlProj.Window1 class is constructed, it should set its own Text property to "Main Window". This is equivalent to the following code:

```
myWindow.Text = "Main Window";
```

Before working with XAML, you need to know important characteristics of the XAML syntax. You can use XML attributes to specify the properties of classes. The following example shows the setting of the Content and Background properties of the Button class:

```
<Button Content="Click Me!" Background="LightGreen" />
```

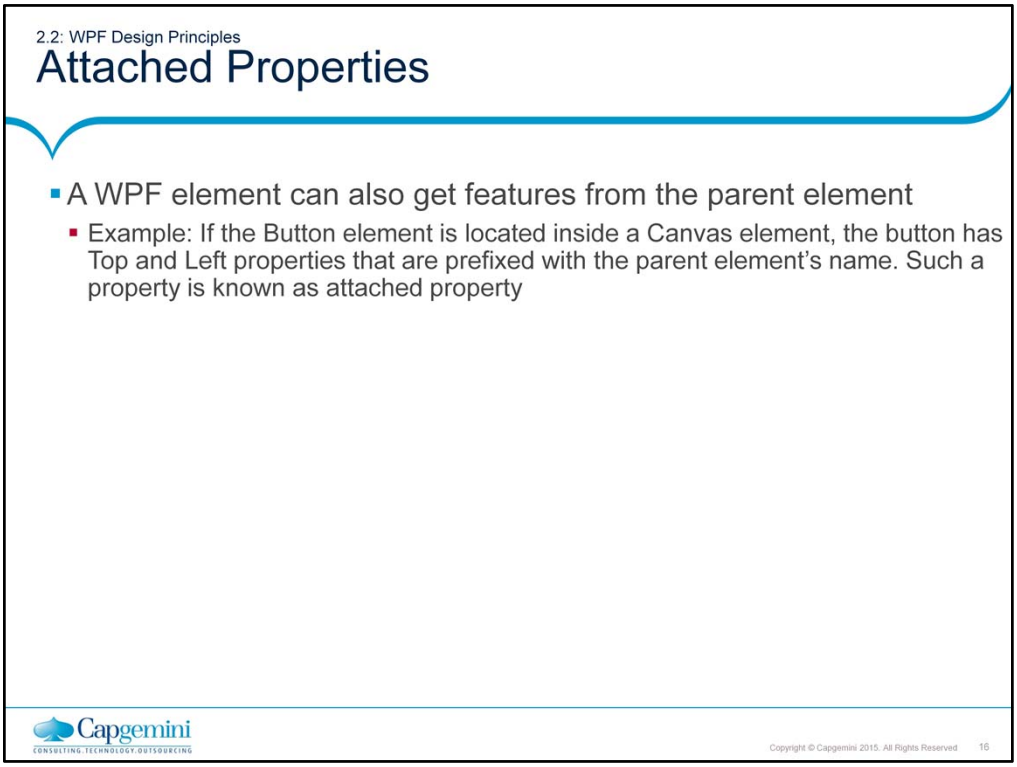


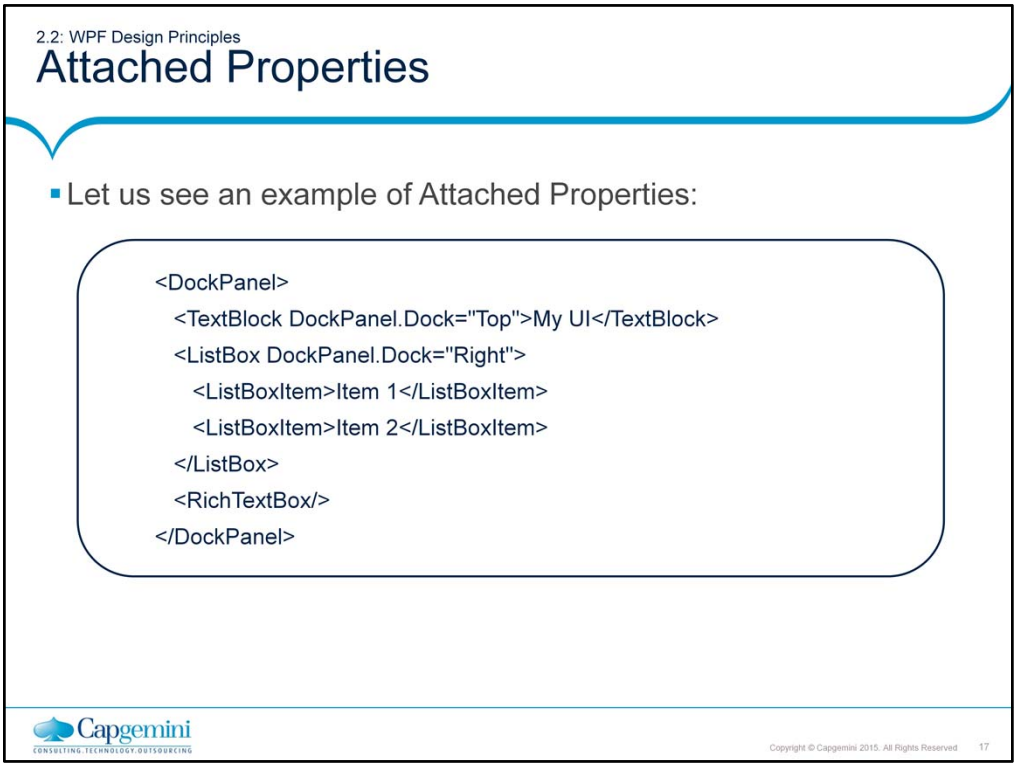
WPF Design Principles:
Properties as Elements:
Instead of using XML attributes, the properties can also be specified as child elements. The value for the content can be directly set by specifying the child elements of the Button element. For all other properties of the Button, the name of the child element is defined by the name of the outer element, followed by the property name:

```
<Button>
  <Button.Background> LightGreen </Button.Background> Click
Me!
</Button>
```

In the previous example, it is not necessary to use child elements. By using XML attributes, the same result was achieved. However, using attributes is no longer possible if the value is more complex than a string. For example: Not only the background can be set to a simple color but also to a brush, for example, a linear gradient brush:

```
<Button>
   <Button.Background>
      <LinearGradientBrush StartPoint="0,0"
EndPoint="1,1">
         <GradientStop Color="Yellow" Offset="0.0" />
         <GradientStop Color="Orange" Offset="0.25" />
          <GradientStop Color="Red" Offset="0.75" />
          <GradientStop Color="Violet" Offset="1.0" />
      </LinearGradientBrush>
   </Button.Background>
   Click Me!
</Button>
```


2.2: WPF Design Principles
Attached Properties
A WPF element can also get features from the parent element
Example: If the Button element is located inside a Canvas element, the button has Top and Left properties that are prefixed with the parent element's name. Such a property is known as attached property
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
16



WPF Design Principles:
Attached Properties:
Both TextBlock and ListBox have a DockPanel.Dock attribute on them.
If you examine these classes, you will find that neither has a property that looks like this. DockPanel declares a DependencyProperty called DockProperty. This special type of DependencyProperty, called an Attached Property, allows a parent control to store information with its children.
In this case, the DockPanel stores information it needs for layout with its child controls. In XAML this manifests itself as an attribute on the child element in the form: ParentName.AttachedPropertyName.

2.2: WPF Design Principles

Attached Properties

- Both TextBlock and ListBox have a DockPanel.Dock attribute on them
- If you examine these classes, you will find that neither has a property that looks like this
- DockPanel declares a DependencyProperty called DockProperty
- This special type of DependencyProperty, called an Attached Property, allows a parent control to store information with its children

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

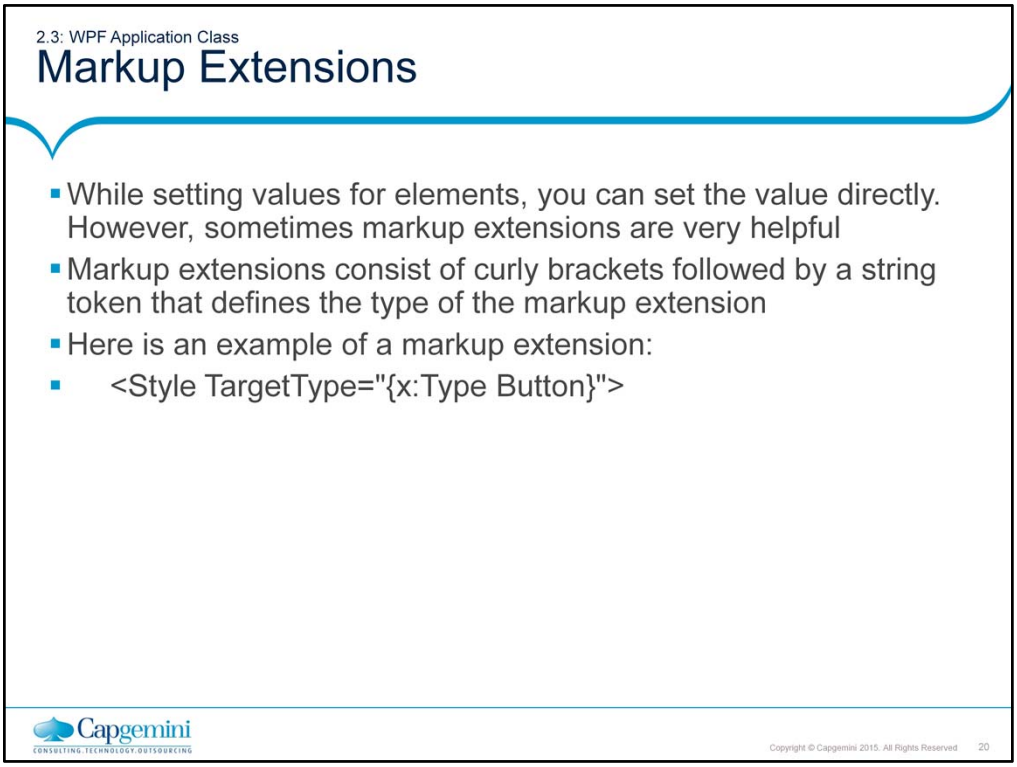
2.2: WPF Design Principles

Attached Properties

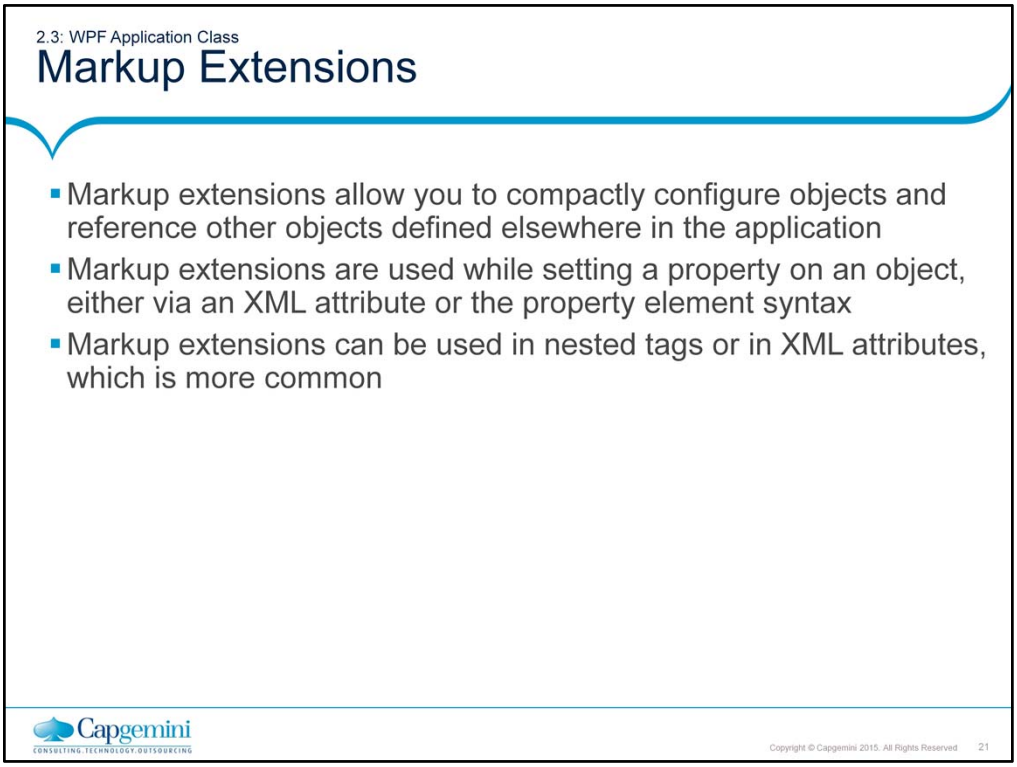
- In this case, the DockPanel is storing information it needs for layout with its child controls
- In XAML this manifests itself as an attribute on the child element in the form: ParentName.AttachedPropertyName

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19



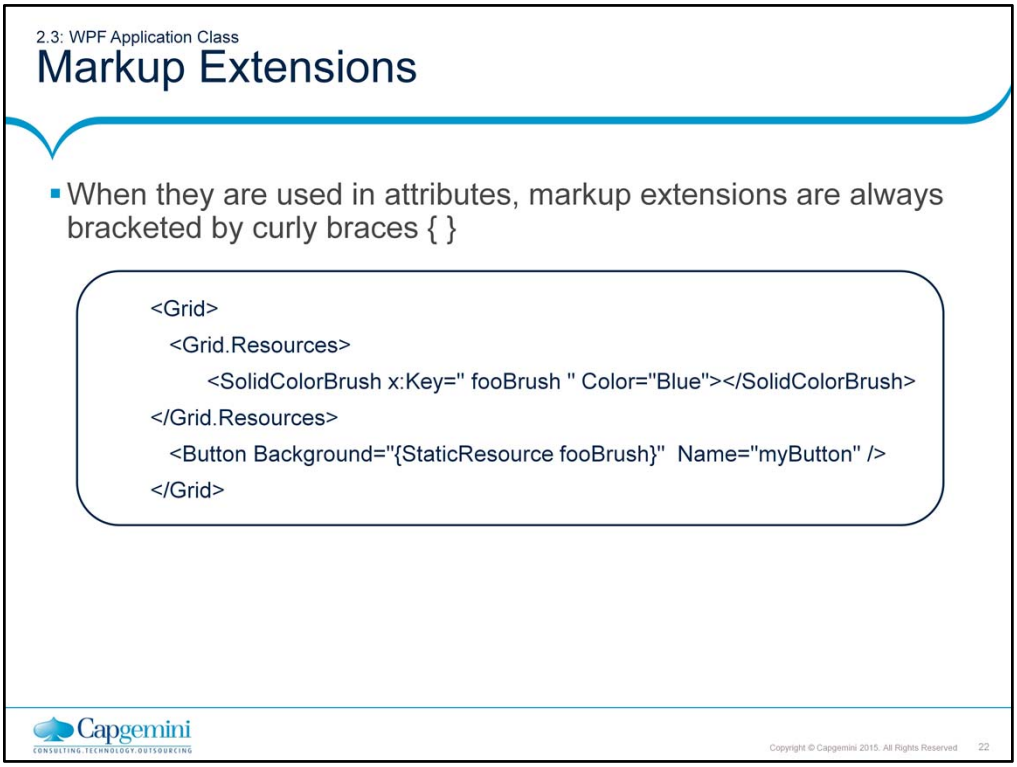
WPF Design Principles:
Markup Extensions:
Type converters and property elements allows you to initialize most properties to constant values or fixed structures. But there are certain situations where a little more flexibility is required. For example, you might want to set a property to be equal to the value of some particular static property. However, we do not know at compile time what that value will be (for example, setting properties representing user-configured colors). XAML provides a powerful solution in the form of markup extensions. A markup extension is a class that decides what the value of a property should be.
The TargetType property of the Style has a value enclosed in braces. This indicates to the XAML compiler that a markup extension is being used. The first string inside the braces is the name of the markup extension class, and the remaining contents are passed to the markup extension during initialization.
We are using x:Type in the above example. There is no class called Type in any of the .NET namespaces represented by the XAML namespace. However, when the XAML compiler fails to find the markup-extension class, it tries appending Extension to the name. There is a TypeExtension class, so the XAML compiler will use that, passing the string Button to its constructor. Then it will call the extension's ProvideValue method, to obtain the real value to be used for the property. The TypeExtension will return the Type object for the Button class.



WPF Design Principles:
Markup Extensions:
For most properties, the XAML property syntax works perfectly well.
However, in some cases, it just is not possible to hard-code the property value.
For example,

> You may want to set a property value to an object that already exists.
> Alternatively, you may want to set a property value dynamically, by binding it to a property in another control.

In both these cases, you need to use a markup extension — specialized syntax.



WPF Design Principles:
Markup Extensions (contd.):
Whenever an attribute value is enclosed in curly braces ({}), the XAML compiler / parser treats it as a markup extension value rather than a literal string (or something that needs to be type-converted).
StaticResource returns the value of the specified resource. Note that StaticResource does not need to be qualified with an x: prefix. This is because resource management is a WPF feature, rather than a generic XAML feature. Hence the resource markup extensions are in the WPF namespace.
In the given example, the use of StaticResource in this markup is effectively equivalent to the following code:

```
myButton.Background = (Brush)
myButton.FindResource("fooBrush");
```

This is a one-shot resource lookup. The property value will be set to the resource value during initialization and then never changed. If the value associated with the resource name changes, then the property will not be updated automatically.

2.3: WPF Application Class

Demo

- Demo on how to create a WPF Application and use Properties and Markup Extensions

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

Windows Presentation Foundation

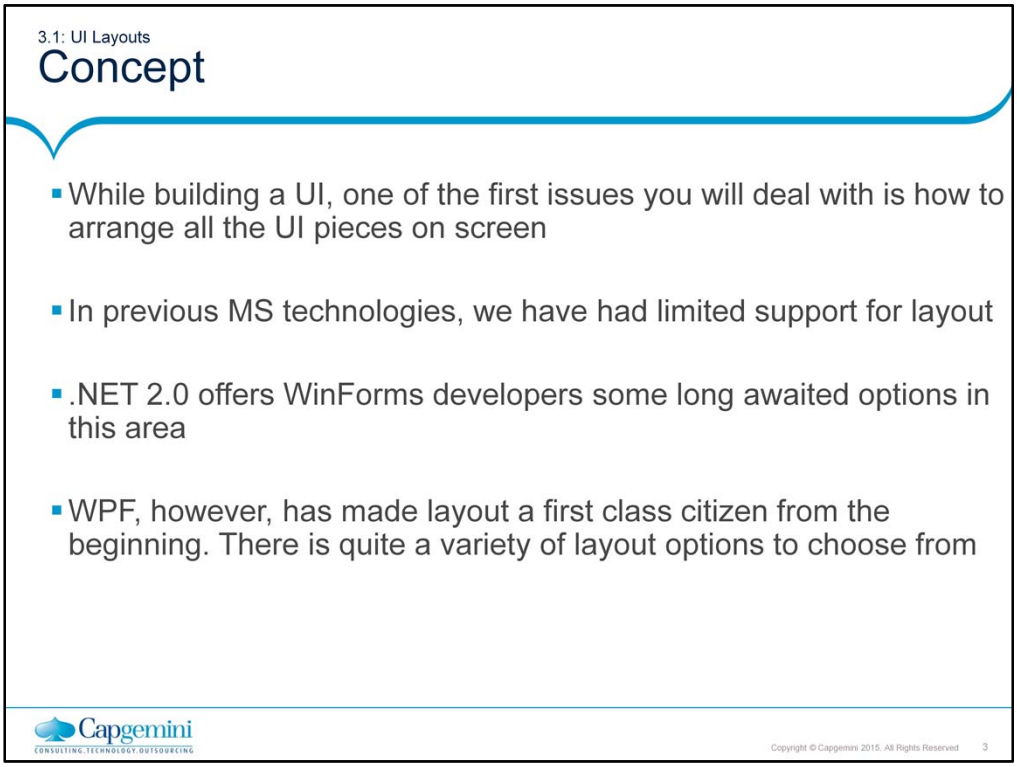
Lesson 3: UI Layouts & Controls

Lesson Objectives

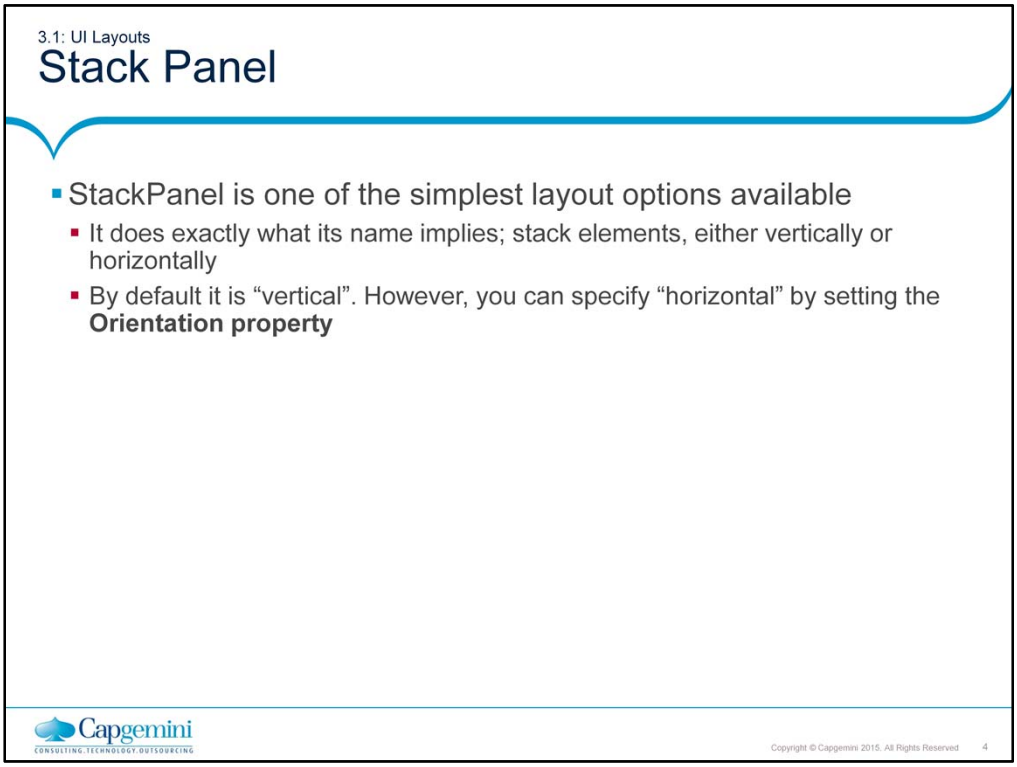
- In this lesson, you will learn:
  - UI Layouts
  - WPF Controls
  - Dependency Properties

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



WPF Design Principles: Shapes, Controls, and Layouts:
UI Layouts:
All applications need to present information to users. For this information to be conveyed effectively, it should be arranged onscreen in a clear and logical way. WPF provides a powerful and flexible array of tools for controlling the layout of the user interface.
WPF provides a set of panels – elements that handle layout. Each individual panel type offers a straightforward and easily understood layout mechanism. As with all WPF elements, layout objects can be composed in any number of different ways, so while each individual element type is fairly simple, the flexible way in which they can be combined makes for a very powerful layout system.



WPF Design Principles: Shapes, Controls, and Layouts:
StackPanel:
StackPanel is a very simple panel. It arranges its child elements in a row or a column.
You will rarely use StackPanel to lay out your whole user interface. It is at its most useful for small-scale layout.
You use DockPanel or Grid to define the overall structure of your user interface. Subsequently, you use StackPanel to manage the details.

3.1: UI Layouts

Stack Panel (Contd…)

- Let us see an example of Stack Panel:

```
<StackPanel>
  <TextBlock>My UI</TextBlock>
  <ListBox>
    <ListBoxItem>Item 1</ListBoxItem>
    <ListBoxItem>Item 2</ListBoxItem>
  </ListBox>
  <RichTextBox/>
</StackPanel>
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

3.1: UI Layouts

Wrap Panel

- Wrap Panel positions the children from left to right, one after the other as long as they fit into the line Subsequently, it continues with the next line

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

3.1: UI Layouts

DockPanel

- DockPanel allows you to dock elements to the top, bottom, left, or right of the container
- The last element will, by default, fill the remaining space

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

3.1: UI Layouts

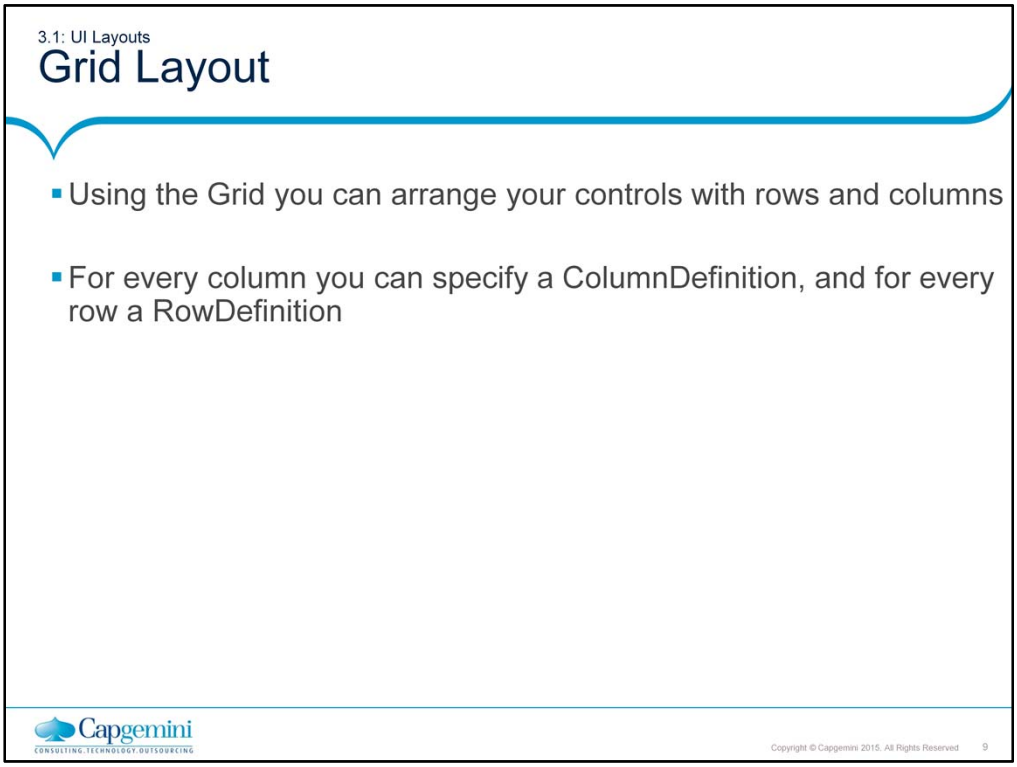
DockPanel (Contd…)

- Let us see an example of DockPanel:

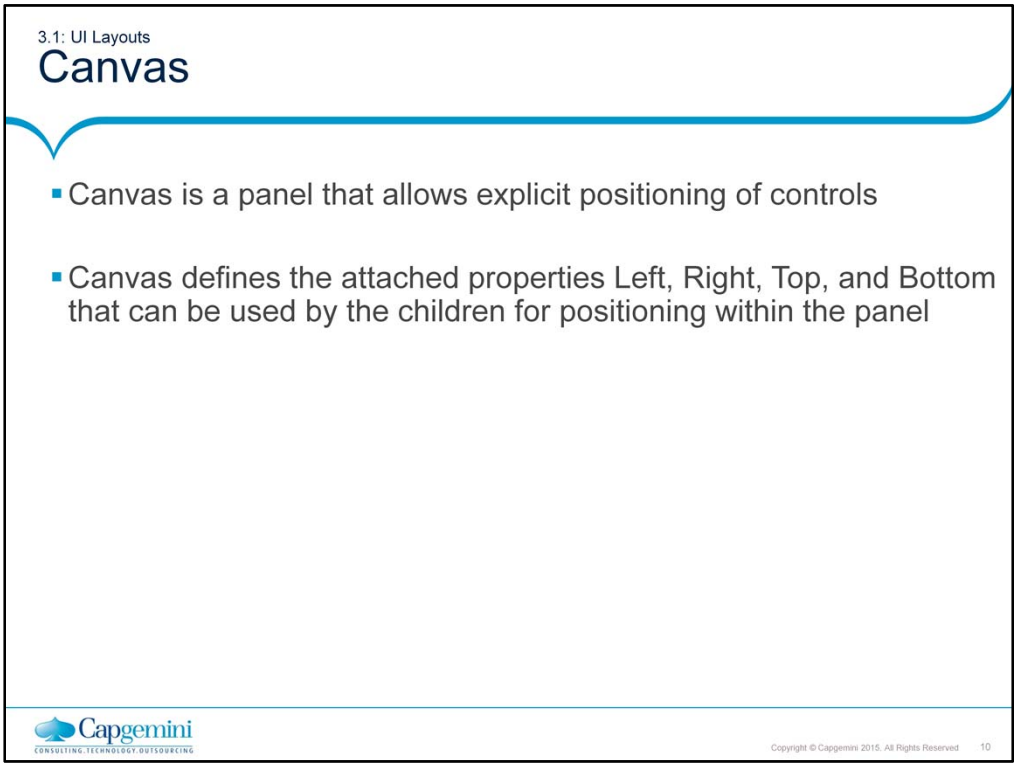
```
<DockPanel>
  <TextBlock DockPanel.Dock="Top">My UI</TextBlock>
  <ListBox DockPanel.Dock="Right">
    <ListBoxItem>Item 1</ListBoxItem>
    <ListBoxItem>Item 2</ListBoxItem>
  </ListBox>
  <RichTextBox/>
</DockPanel>
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8



WPF Design Principles: Shapes, Controls, and Layouts:
Grid Layout:
The Grid needs to know how many columns and rows we require, and we indicate this by specifying a series of ColumnDefinition and RowDefinition elements at the start. This may seem rather verbose.
A simple pair of properties on the Grid itself might seem like a simpler solution. However, you will typically want to control the characteristics of each column and row independently. So in practice, it makes sense to have elements representing them.



WPF Design Principles: Shapes, Controls, and Layouts:
Canvas:
Occasionally, the automatic layout offered by DockPanel, StackPanel, or Grid will not enable the look you require. Thus it will be necessary to take complete control of the precise positioning of every element.
For example: When you want to build an image out of graphical elements, the positioning of the elements is dictated by the picture you are creating, not by any set of automated layout rules. For these scenarios, you will want to use the Canvas.
The Canvas is the simplest of the panels. It allows the location of child elements to be specified precisely relative to the edges of the canvas. The Canvas does not really do any layout at all. It simply puts things where you tell it to.
While using a Canvas, you must specify the location of each child element. If you do not do so, all your elements will end up at the top left-hand corner. Canvas defines four attached properties for setting the position of child elements. Vertical position is set with either the Top or Bottom property, and horizontal position is determined by either the Left or Right property.

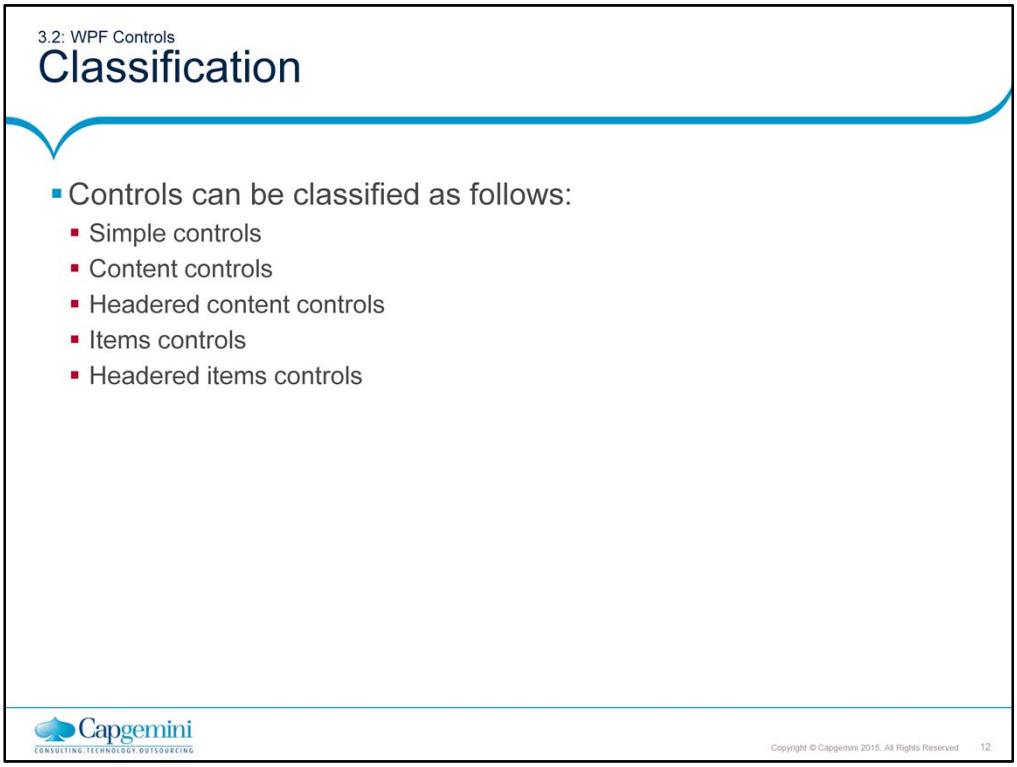
3.1: UI Layouts

Demo

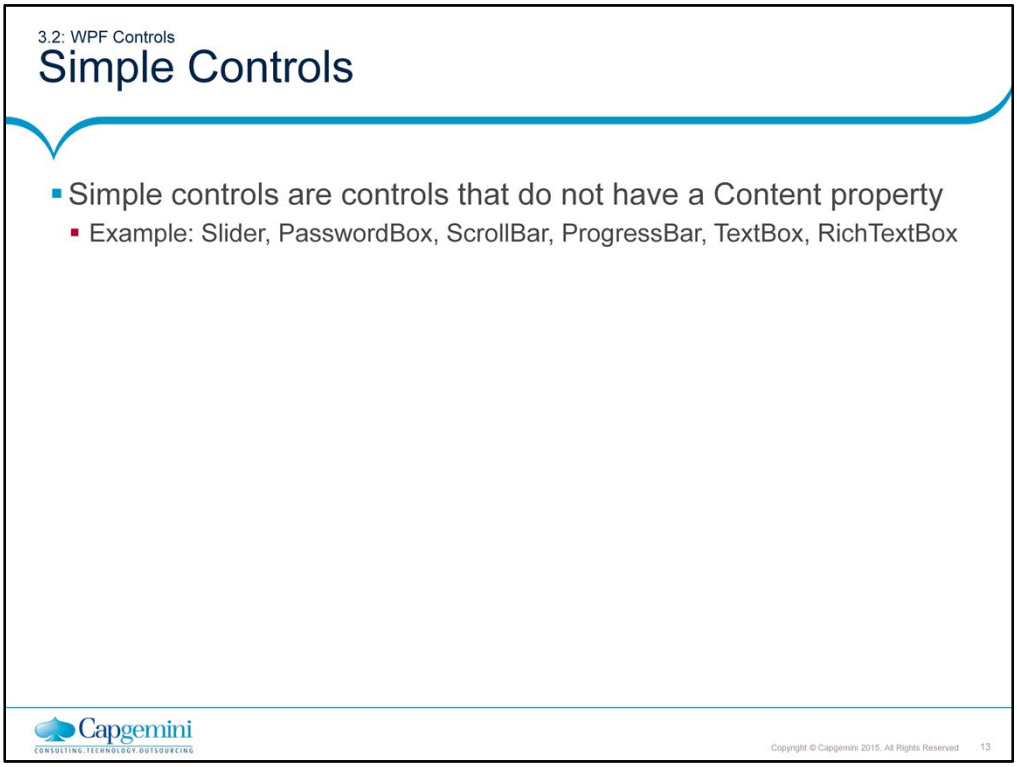
- Demo of various Layouts

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

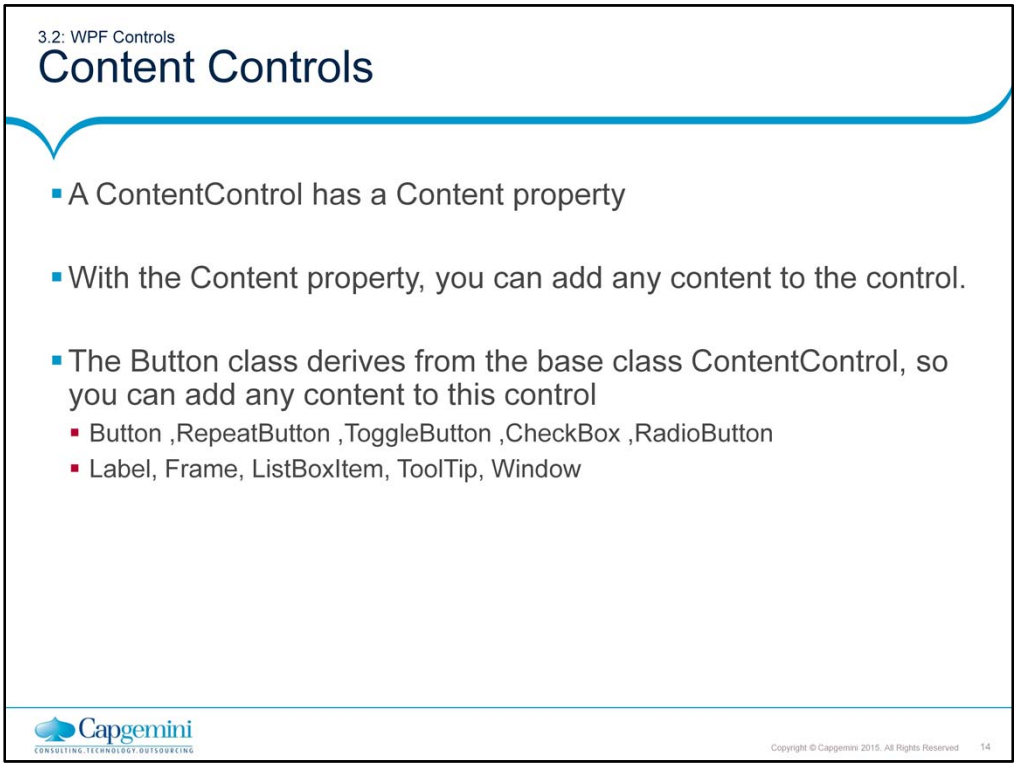
 11



WPF Design Principles: Shapes, Controls, and Layouts:
Controls:
Controls are the building blocks of an application's user interface. They are interactive features such as text boxes, buttons, or listboxes. You may be familiar with similar constructs from other user-interface technologies most UI frameworks offer an abstraction similar to a control.
However, WPF is somewhat unusual, in that controls are typically not directly responsible for their own appearance. Many GUI frameworks require you to write a custom control when customizing a control's appearance. In WPF, this is not necessary. Nested content, and templates offer powerful yet simpler solutions. You only need to write a custom control if you need behavior that is different from any of the built-in controls.



WPF Design Principles: Shapes, Controls, and Layouts:
Controls – Simple Controls:
Slider and Scroll Controls:
WPF provides controls that allow a value to be selected from a range. They all offer a similar appearance and usage. They show a track, indicating the range, and a “thumb” with which the value can be adjusted.
There are two slider controls, HorizontalSlider and VerticalSlider. There are also two scrollbar controls, HorizontalScrollBar and VerticalScrollBar. The main difference is one of convention rather than functionality. The scrollbar controls are commonly used in conjunction with some scrolling viewable area, while the sliders are used to adjust values.



WPF Design Principles: Shapes, Controls, and Layouts:
Controls – Content Control:
Buttons are controls that a user can click.
The result of the click is up to the application developer. However, there are common expectations, depending on the type of button.
For example: Clicking on a CheckBox or RadioButton is used to express a choice and does not normally have any immediate effect beyond visually reflecting that choice. By contrast, clicking on a normal Button usually has some immediate effect.

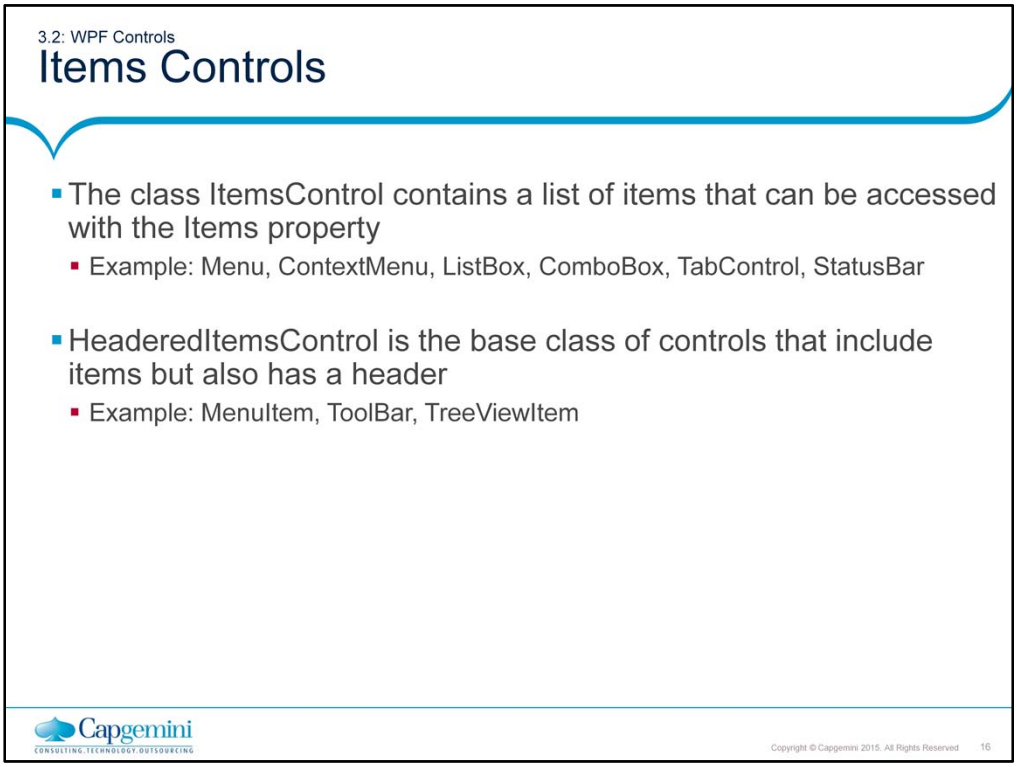
3.2: WPF Controls

Headered Content Controls

- Content controls with a header are derived from the base class HeaderedContentControl
  - HeaderedContentControl itself is derived from the base class ContentControl
  - The class HeaderedContentControl has a property Header to define the content of the header and HeaderTemplate for complete customization of the header
  - Example: Expander, GroupBox, TabItem

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15



WPF Design Principles: Shapes, Controls, and Layouts:
Controls – Items and Headered Items Controls:
Many Windows applications provide access to their functionality through a hierarchy of menus. These are typically presented as either a main menu at the top of the window or as a pop-up “context” menu. WPF provides two menu controls. Menu is for permanently visible menus (such as a main menu), and ContextMenu is for context menus.
Most Windows applications offer toolbars as well as menus. Toolbars provide faster access for frequently used operations. This is because the user does not need to navigate through the menu system; the toolbar is always visible onscreen.

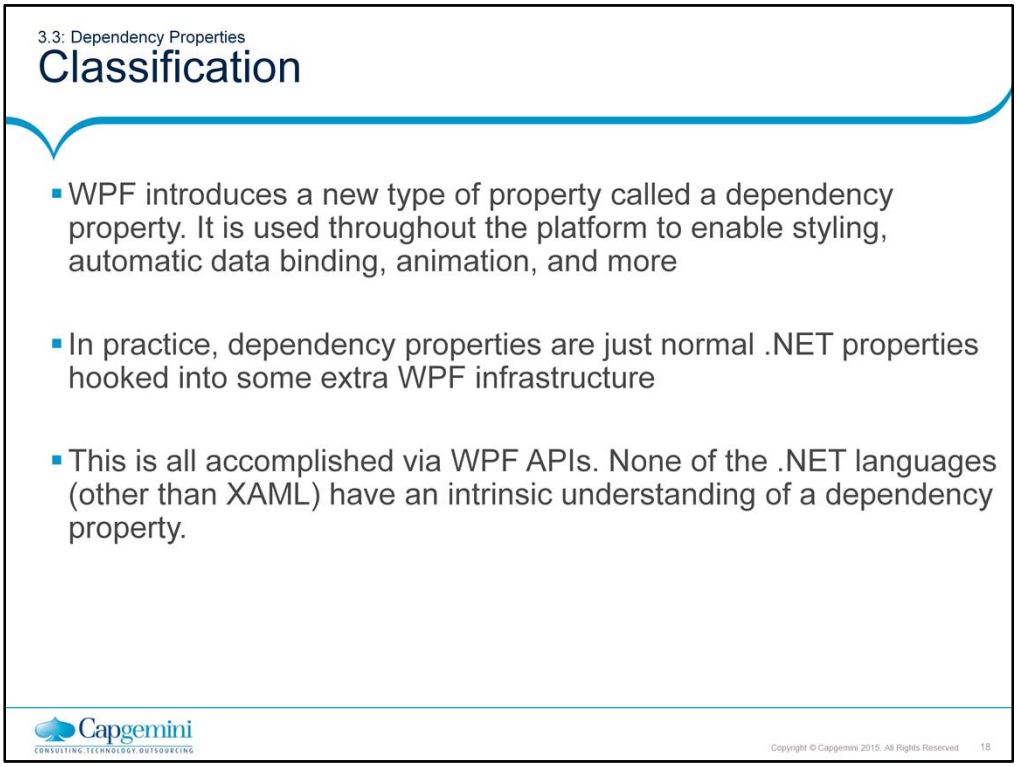
3.2: WPF Controls

Demo

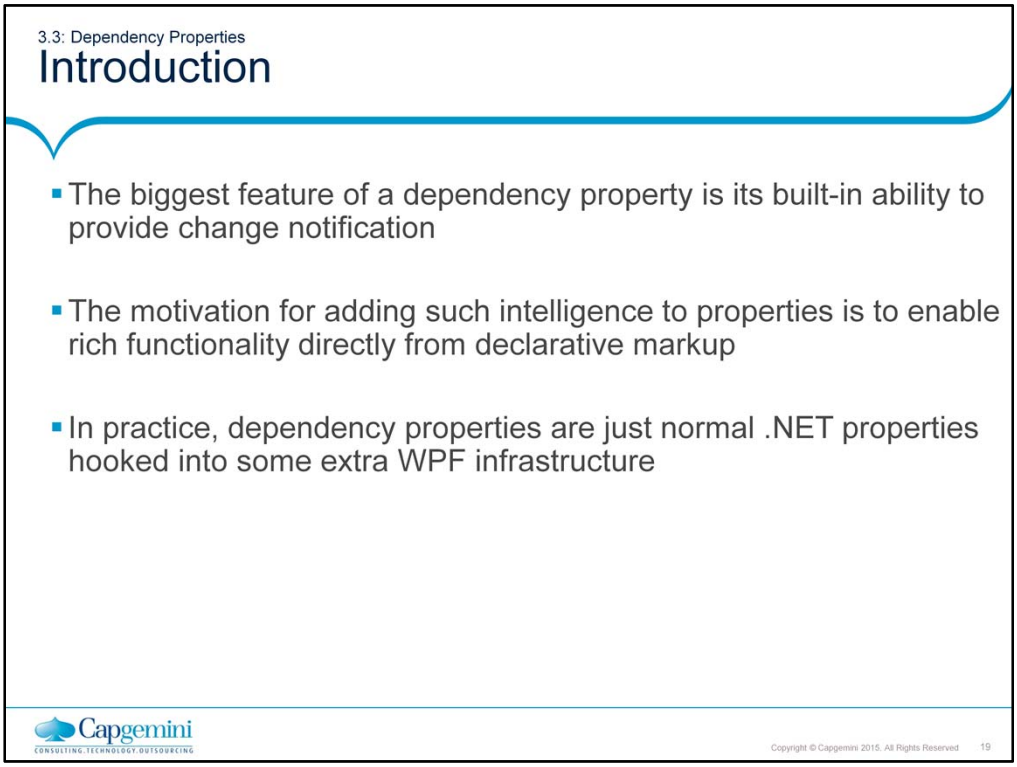
- Using Controls

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

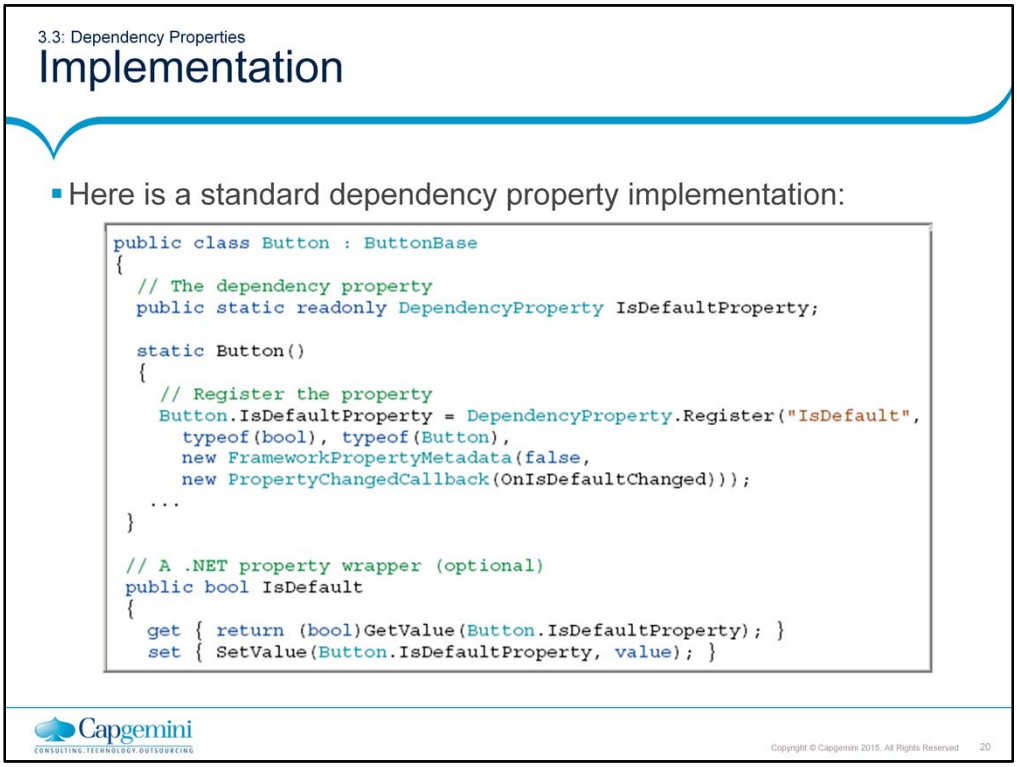
 17



WPF Design Principles:
Dependency Properties:
A dependency property depends on multiple providers for determining its value at any point in time.
These providers can be an animation continuously changing its value, a parent element whose property value trickles down to its children, and so on.
Arguably the biggest feature of a dependency property is its built-in ability to provide change notification.



WPF Design Principles:
Dependency Properties (contd.):
Properties can be easily set in XAML (directly or by a design tool) without any procedural code.
However, without the extra plumbing in dependency properties, it will be hard for the simple action of setting properties to get the desired results without writing additional code.
This is all accomplished via WPF APIs. None of the .NET languages (other than XAML) have an intrinsic understanding of a dependency property.



WPF Design Principles:
Dependency Property Implementation:
The static IsDefaultProperty field is the actual dependency property, represented by the System.Windows.DependencyProperty class. By convention all DependencyProperty fields are public, static, and have a Property suffix. Dependency properties are usually created by calling the static DependencyProperty.Register method, which requires a name (IsDefault), a property type (bool), and the type of the class claiming to own the property (Button). Optionally (via different overloads of Register), you can pass metadata that customizes how the property is treated by WPF, as well as callbacks for handling property value changes, coercing values, and validating values. Button calls an overload of Register in its static constructor to give the dependency property a default value of false and to attach a delegate for change notifications. Finally, the traditional .NET property called IsDefault implements its accessors by calling GetValue and SetValue methods inherited from System.Windows.DependencyObject, a low-level base class from which all classes with dependency properties must derive.

GetValue returns the last value passed to SetValue, or if SetValue has never been called, the default value registered with the property. The IsDefault .NET property (sometimes called a property wrapper in this context) is not strictly necessary. Consumers of Button can always directly call the GetValue/SetValue methods because they are exposed publicly. However, the .NET property makes programmatic reading and writing of the property much more natural for consumers, and it enables the property to be set via XAML.

3.3: Dependency Properties

Implementation (Contd…)

- The static IsDefaultProperty field is the actual dependency property, represented by the System.Windows.DependencyProperty class
- By convention all DependencyProperty fields are public, static, and have a Property suffix
- Dependency properties are usually created by calling the static DependencyProperty.Register method, which requires a name (IsDefault), a property type (bool), and the type of the class claiming to own the property (Button)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

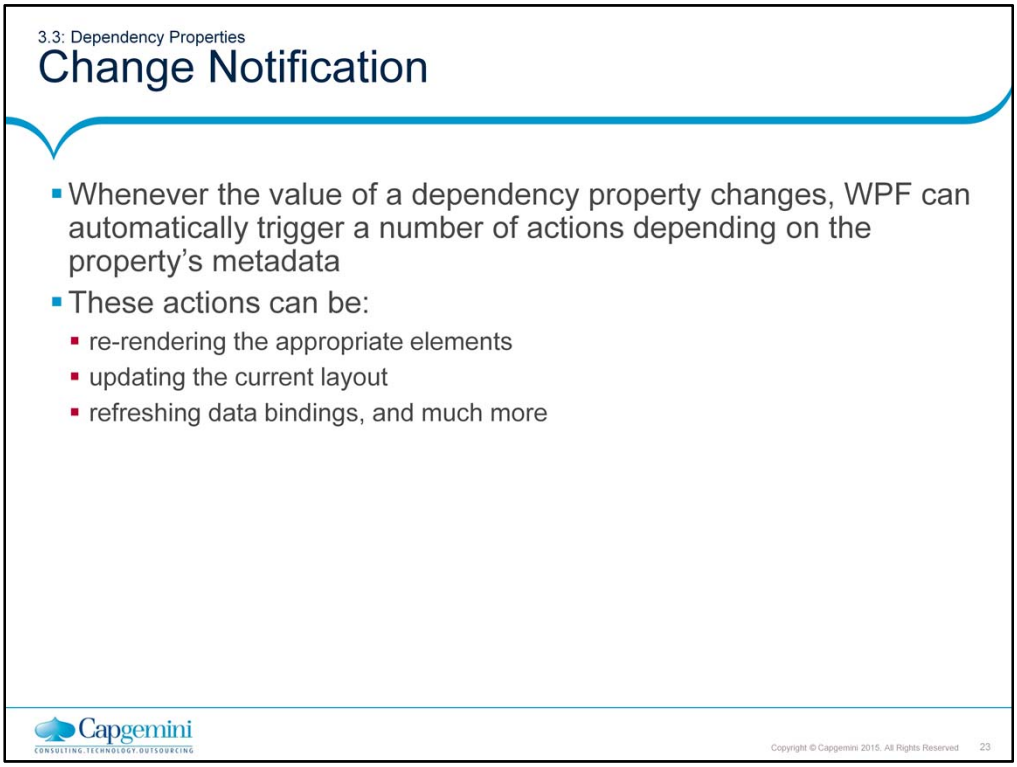
3.3: Dependency Properties

Implementation (Contd…)

- Finally, the traditional .NET property called IsDefault implements its accessors by calling GetValue and SetValue methods inherited from System.Windows.DependencyObject, a low-level base class from which all classes with dependency properties must derive

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22



WPF Design Principles:
Change Notification:
One of the most interesting features enabled by this built-in change notification is property triggers, which enable you to perform your own custom actions when a property value changes without writing any procedural code.
For example: You want the text in each Button on the Window to turn blue when the mouse pointer hovers over it.

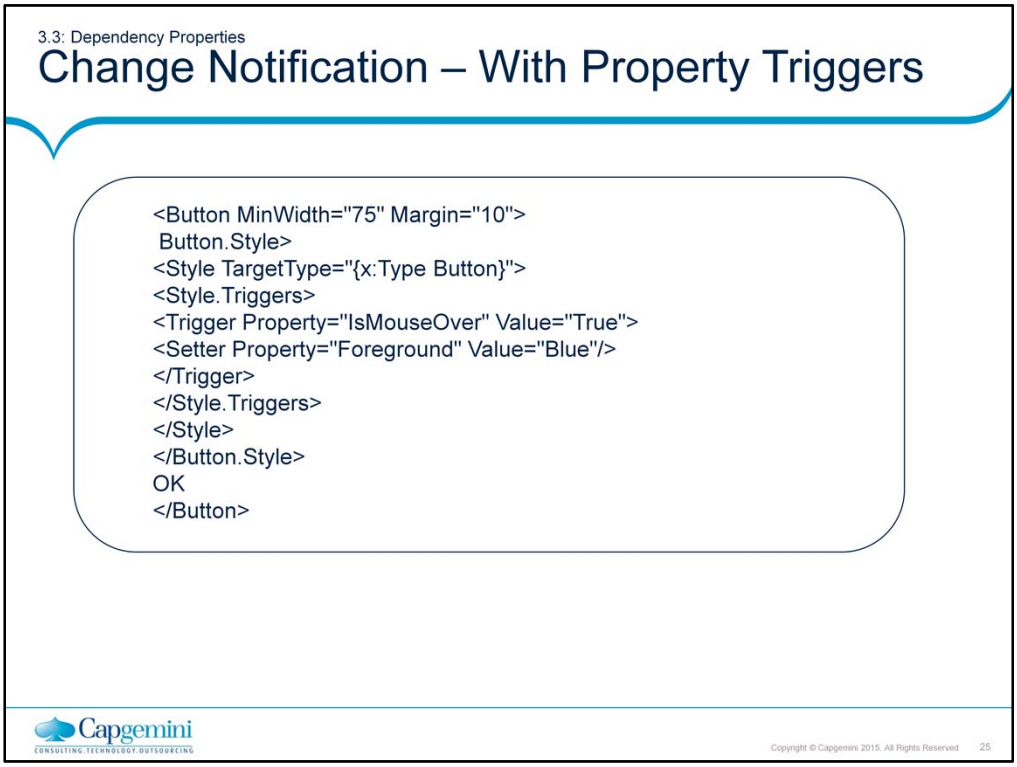
3.3: Dependency Properties

Change Notification (Contd…)

- One of the most interesting features enabled by this built-in change notification is property triggers, which enable you to perform your own custom actions when a property value changes without writing any procedural code
- For example, if you want the text in each Button on the Window to turn blue when the mouse pointer hovers over it

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24



WPF Design Principles:
Change Notification – With Property Triggers:
Without property triggers, you can attach two event handlers to each Button, one for its MouseEnter event and one for its MouseLeave event.
These two handlers can be implemented in a C# code-behind file with a property trigger. However, you can accomplish this same behavior purely in XAML.
This trigger can act upon Button's IsMouseOver property, which becomes true at the same time the MouseEnter event is raised and false at the same time the MouseLeave event is raised.
You do not have to worry about reverting Foreground to black when IsMouseOver changes to false. This is automatically done by WPF!

3.3: Dependency Properties

Change Notification (Contd…)

- This trigger can act upon Button's IsMouseOver property, which becomes true at the same time the MouseEnter event is raised and false at the same time the MouseLeave event is raised
- You don't have to worry about reverting Foreground to black when IsMouseOver changes to false
- This is automatically done by WPF!

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26

3.3: Dependency Properties

Demo

- Implementing Dependency Properties
- Use of property Triggers

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 27

Summary

- WPF has made layout a first class citizen from the beginning. There is quite a variety of layout options to choose from
- The various Layouts like StackPanel, Wrap Panel, Grid etc.
- The various Controls -
  - Simple controls,Content controls, Headered content controls, Items controls, Headered items controls
- Working with Dependency Properties

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 28

Windows Presentation Foundation

Lesson 4: Styles, Resources & Shapes

Lesson Objectives

- In this lesson, you will learn:
  - Pages and Navigation
  - Styles & Resources
  - Shapes
  - Brushes

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

4.1: Pages and Navigation

Description

- Most traditional Windows applications are arranged around a window that contains toolbars and menus
- The toolbars and menus drive the application—as the user clicks them, actions happen, and other windows appear
- In a bid to give desktop developers the ability to build web-like desktop applications, WPF includes its own page-based navigation system

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

4.1: Pages and Navigation

Description

- To create a page-based application in WPF, you need to stop using the Window class as your top-level container for user interfaces
- Instead, WPF provides a Page class available under System.Windows.Controls.Page
- You can add a page to any WPF project
- Just choose Project ➤ Add Page in Visual Studio

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4

4.1: Pages and Navigation

Description

- Although pages are the top-level user interface ingredient when you are designing your application, they are not the top-level container when you run your application
- Instead, your pages are hosted in another container
- You can use one of several different containers
  - The NavigationWindow, which is a slightly tweaked version of the Window class
  - A Frame that'is inside another window or Page

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

4.1: Pages and Navigation

Code

```
<Page x:Class="NavigationApplication.Page1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
WindowTitle="Page1"
>
</Page>
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

4.1: Pages and Navigation

Description

- When you run this application, WPF is intelligent enough to realize that you are pointing it to a page rather than a window.
- It automatically creates a new NavigationWindow object to serve as a container and shows your page inside of it
- The NavigationWindow looks more or less like an ordinary window, aside from the back and forward navigation buttons that appear in the bar at the top

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

4.1: Pages and Navigation

Hyperlink

```
<TextBlock Margin="3" TextWrapping="Wrap">
This is a simple page.
Click <Hyperlink NavigateUri="Page2.xaml">here</Hyperlink> to go to Page2.
</TextBlock>
```

- The easiest way to allow the user to move from one page to another is using hyperlinks

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

4.1: Pages and Navigation

The Navigation Service

- To allow programmatic navigation, the most flexible and powerful approach is to use the WPF navigation service
- You can access the navigation service through the static NavigationService.GetNavigationService() method

```
NavigationService nav;
nav = NavigationService.GetNavigationService(this);
Page2 nextPage = new Page2();
nav.Navigate(nextPage);
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

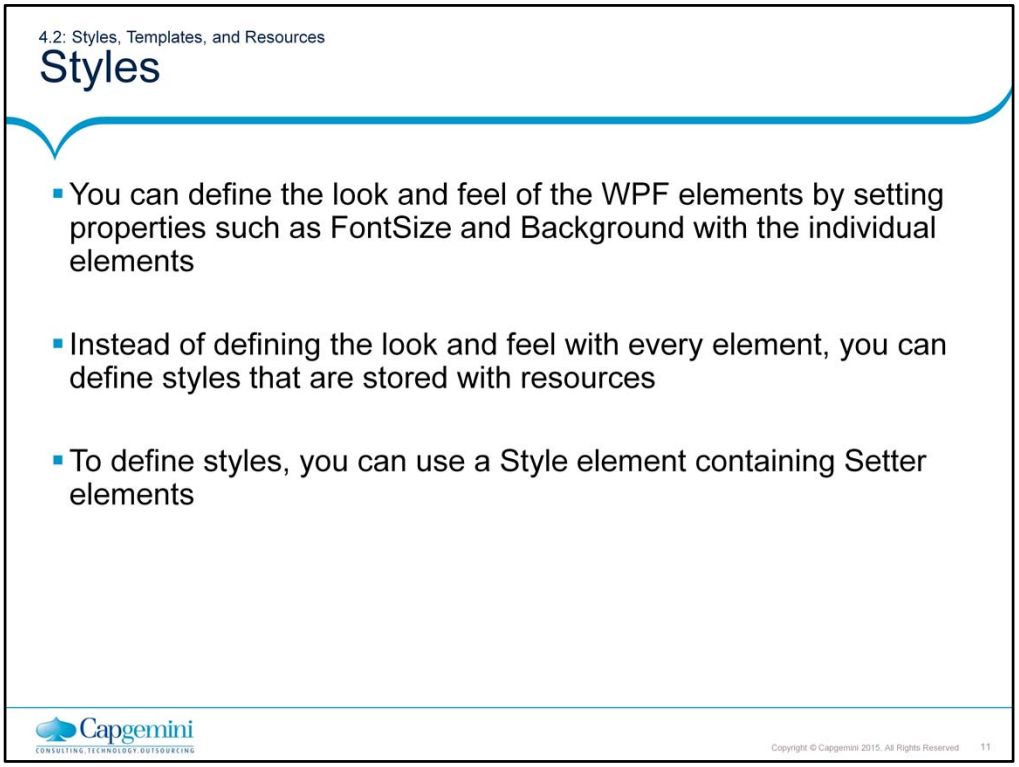
4.1: Pages and Navigation

Demo

- Demo of Page based Navigation

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

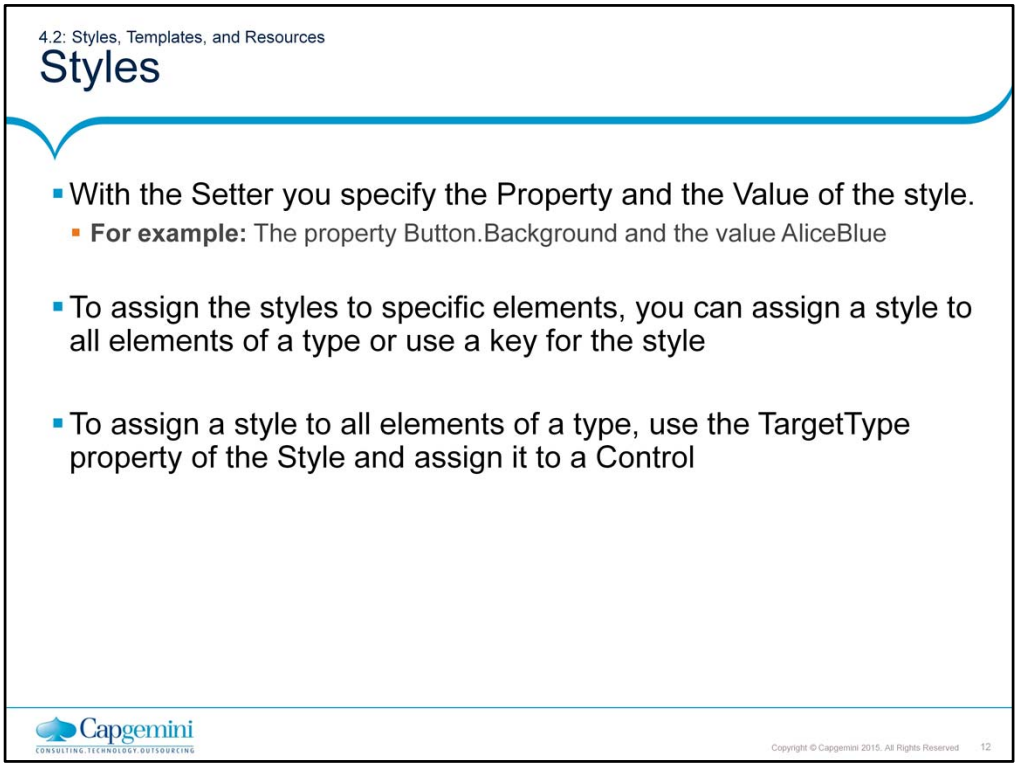


WPF Design Principles: Styles, Templates, and Resources:
Styles:
In WPF, a style is also a set of properties applied to content used for visual rendering.

A style can be used to set properties on an existing visual element, such as setting the font weight of a Button control.
It can also be used to define the way an object looks, such as showing the name and age from a Person object.

In addition to the features in word processing styles, WPF styles have specific features for building applications, including the following:

The ability to associate different visual effects based on user events
Provide entirely new looks for existing controls
Designate rendering behavior for non-visual objects



WPF Design Principles: Styles, Templates, and Resources:
Styles:
Let us see an example of setting a named style:

```
<Window ...>
 <Window.Resources>
   <Style x:Key="CellTextStyle">
        <Setter Property="Control.FontSize" Value="32" />
        <Setter Property="Control.FontWeight" Value="Bold" />
   </Style>
  </Window.Resources>
   ...
  <Button Style="{StaticResource CellTextStyle}" ...
x:Name="cell00" /> ... </Window>
```

4.2: Styles, Templates, and Resources

Demo

- Demo of styles

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

4.2: Styles, Templates, and Resources

Styles

- Usually styles are stored within resources
- You can define any element within a resource
  - In the previous example, the resources were defined with the **Window** element or **StackPanel** element.
- The base class FrameworkElement defines the property Resources of type ResourceDictionary
- Hence, resources can be defined with every class that is derived from the FrameworkElement – any WPF element

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

4.2: Styles, Templates, and Resources

Resources

- If you need the same style for more than one Window, then you can define the style with the application
- In a Visual Studio WPF project, the file App.xaml is created for defining global resources of the application
- The application styles are valid for every window of the application. Every element can access resources that are defined with the application
- If resources are not found with the parent window, then the search for resources continues with the Application

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

4.2: Styles, Templates, and Resources

Dynamic Resources

- With the StaticResource markup extension, resources are searched at load time.
- If the resource changes while the program is running, then you should use the DynamicResource markup extension instead.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

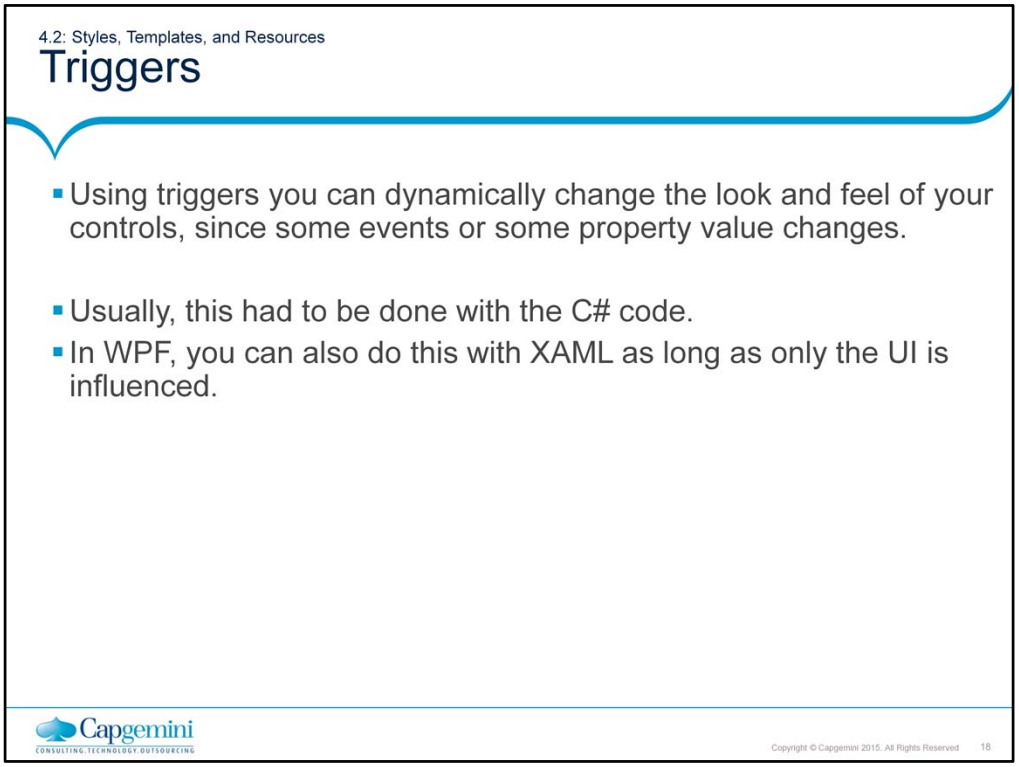
4.2: Styles, Templates, and Resources

Demo

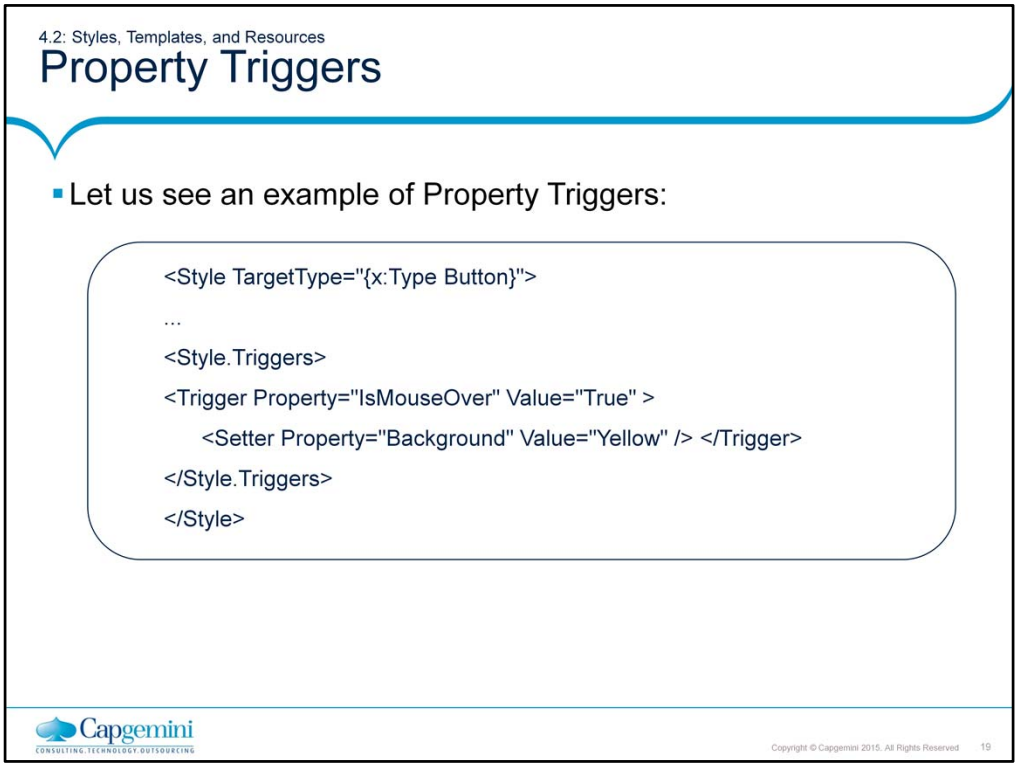
- Demo of Resources

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17



WPF Design Principles: Styles, Templates, and Resources:
Triggers:
So far, we have seen styles as a collection of Setter elements. When a style is applied, the settings described in the Setter elements are applied unconditionally (unless overridden by per-instance settings).
On the other hand, triggers are a way to wrap one or more Setter elements in a condition so that, if the condition is true, the corresponding Setter elements are executed. Furthermore, when the condition becomes false, the property value reverts to its pre-trigger value.



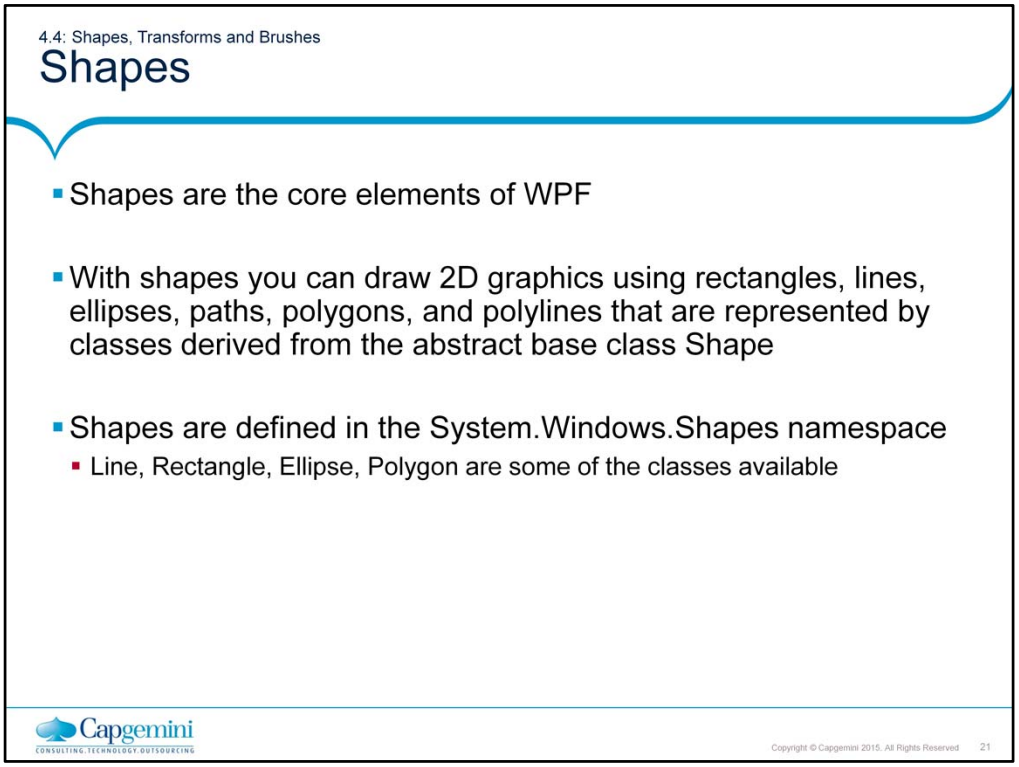
WPF Design Principles: Styles, Templates, and Resources:
Property Triggers:
The simplest form of a trigger is a property trigger. It watches for a dependency property to have a certain value. For example, suppose we want to light up a button in yellow as the user moves the mouse over it. Then we can do so by watching for the IsMouseOver property to have a value of True, as shown in the example in the above slide.
Triggers are grouped together under the Style.Triggers element. In this case, we have added a Trigger element to the button style. When the IsMouseOver property of our button is true, the Background value of the button will be set to yellow.

4.2: Styles, Templates, and Resources

Demo

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20



WPF Design Principles: Shapes, Controls, and Layouts:
Shapes:
Shapes are drawing primitives, represented as elements in the user interface tree. WPF supports a variety of different shapes and provides element types for each of them.
All of the elements listed in this slide derive from a common abstract base class, that is Shape. Although you cannot use this type directly, it is useful to know about it, because it defines a common set of features that you can use on all shapes. These common properties are all concerned with the way in which the interior and outline of the shape are painted.
The Fill property specifies the Brush that will be used to paint the interior. The Line and Polyline classes do not have interiors, so they will ignore this property. (This was simpler than complicating the inheritance hierarchy by having separate Shape and FilledShape base classes.) The Stroke property specifies the Brush that will be used to paint the outline of the shape.

4.4: Shapes, Transforms and Brushes

Shapes

- Shapes draw themselves
  - You do not need to manage the invalidation and painting process.
  - For example, you do not need to manually repaint a shape when content moves, the window is resized, or the shape's properties change
- Shapes are organized in the same way as other elements
  - You can place a shape in any of the layout containers

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

4.4: Shapes, Transforms and Brushes

The Shape Class

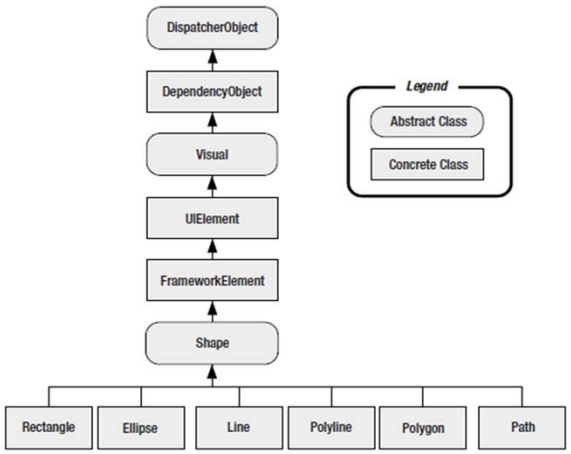
- Every shape derives from the abstract System.Windows.Shapes.Shape class
- Shape Class Properties
  - Fill: Sets the brush object that paints the surface of the shape (everything inside its borders)
  - Stroke: Sets the brush object that paints the edge of the shape (its border)
  - StrokeThickness: Sets the thickness of the border, in device-independent units

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

4.4: Shapes, Transforms and Brushes

The Shape Class



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

4.4: Shapes, Transforms and Brushes

Rectangle and Ellipse

- The Rectangle and Ellipse are the two simplest shapes
- To create either one, set the Height and Width properties to define the size of your shape, and then set the Fill or Stroke property (or both) to make the shape visible

```
<Ellipse Fill="Yellow" Stroke="Blue"
Height="50" Width="100" Margin="5" HorizontalAlignment="Left"></Ellipse>
<Rectangle Fill="Yellow" Stroke="Blue"
Height="50" Width="100" Margin="5" HorizontalAlignment="Left"></Rectangle>
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25

4.4: Shapes, Transforms and Brushes

Line

- The Line shape represents a straight line that connects one point to another
- The starting and ending points are set by four properties: X1 and Y1 (for the first point) and X2 and Y2 (for the second)

```
<Line Stroke="Blue" X1="0" Y1="0" X2="10" Y2="100"></Line>
```

  - The Fill property has no effect for a line. You must set the Stroke property

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26

4.4: Shapes, Transforms and Brushes

Polyline

- The Polyline class allows you to draw a sequence of connected straight lines
- You just supply a list of X and Y coordinates using the Points property

```
<Polyline Stroke="Blue" Points="5,100 15,200"></Polyline>

<Polyline Stroke="Blue" StrokeThickness="5" Points="10,150 30,140 50,160 70,130
  90,170 110,120 130,180 150,110 170,190 190,100 210,240" >
</Polyline>
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 27

4.4: Shapes, Transforms and Brushes

Transform

- A great deal of drawing tasks can be made simpler with the use of a transform—an object that alters the way a shape or element is drawn by secretly shifting the coordinate system it uses
- In WPF, transforms are represented by classes that derive from the abstract System.Windows.Media.Transform class

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 28

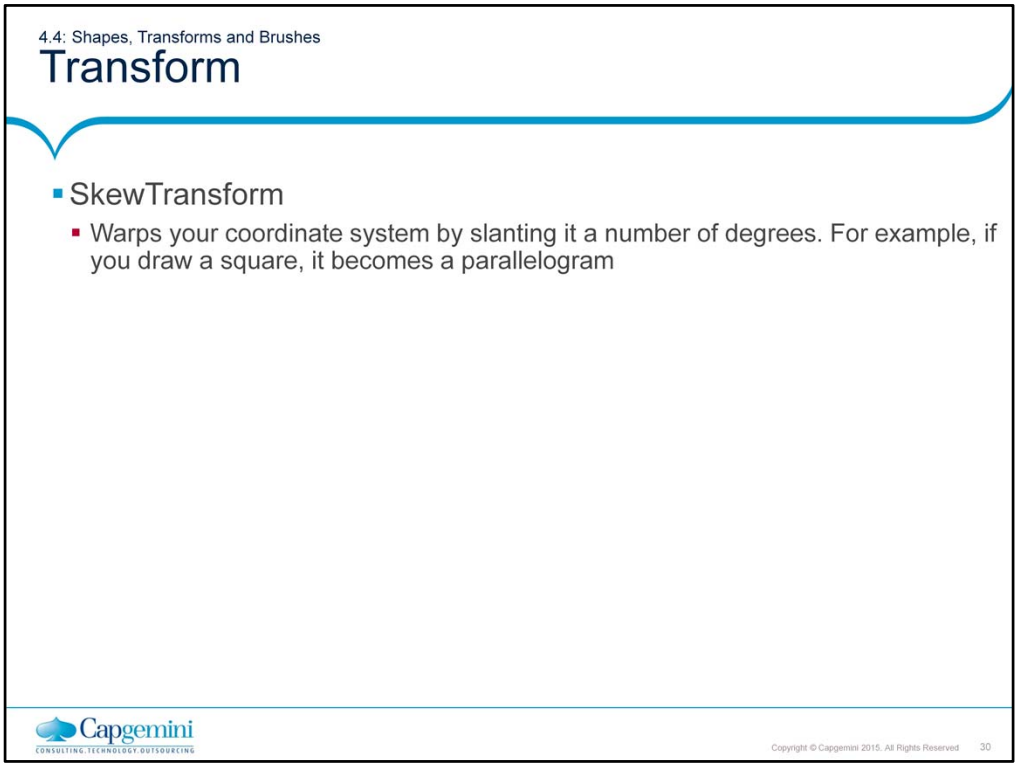
4.4: Shapes, Transforms and Brushes

Transform

- TranslateTransform
  - Displays your coordinate system by some amount. This transform is useful if you want to draw the same shape in different places
- RotateTransform
  - Rotates your coordinate system. The shapes you draw normally are turned around a center point you choose
- ScaleTransform
  - Scales your coordinate system up or down, so that your shapes are drawn smaller or larger

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 29


4.4: Shapes, Transforms and Brushes
Transform
SkewTransform
Warps your coordinate system by slanting it a number of degrees. For example, if you draw a square, it becomes a parallelogram
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
30

4.4: Shapes, Transforms and Brushes

Sample Code

```
<Button Background="Orange">
  <TextBlock RenderTransformOrigin="0.5,0.5">
  <TextBlock.RenderTransform>
    <RotateTransform Angle="45"/>
  </TextBlock.RenderTransform>
    45°
  </TextBlock>
</Button>
```

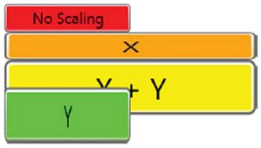
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 31

4.4: Shapes, Transforms and Brushes

Sample Code

```
<StackPanel Width="100">
  <Button Background="Red">No Scaling</Button>
  <Button Background="Orange">
  <Button.RenderTransform>
    <ScaleTransform ScaleX="2"/>
  </Button.RenderTransform>
    X</Button>
  <Button Background="Yellow">
  <Button.RenderTransform>
    <ScaleTransform ScaleX="2" ScaleY="2"/>
  </Button.RenderTransform>
    X + Y</Button>
  <Button Background="Lime">
  <Button.RenderTransform>
    <ScaleTransform ScaleY="2"/>
  </Button.RenderTransform>
    Y</Button>
</StackPanel>
```



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 32

4.4: Shapes, Transforms and Brushes

Sample Code

```
<Button>
<Button.RenderTransform>
  <TransformGroup>
    <RotateTransform Angle="45"/>
    <ScaleTransform ScaleX="5" ScaleY="1"/>
    <SkewTransform AngleX="30"/>
  </TransformGroup>
</Button.RenderTransform>
  OK
</Button>
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 33

4.4: Shapes, Transforms and Brushes

Brushes

- Brushes fill an area, whether it is the background, foreground, or border of an element, or the fill or stroke of a shape
- Brush Classes:
- LinearGradientBrush
  - Paints an area using a gradient fill, a gradually shaded fill that changes from one color to another (and, optionally, to another and then another, and so on)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 34

4.4: Shapes, Transforms and Brushes

Brushes

- RadialGradientBrush
  - Paints an area using a radial gradient fill, which is similar to a linear gradient except it radiates out in a circular pattern starting from a center point
- ImageBrush
  - Paints an area using an image that can be stretched, scaled, or tiled

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 35

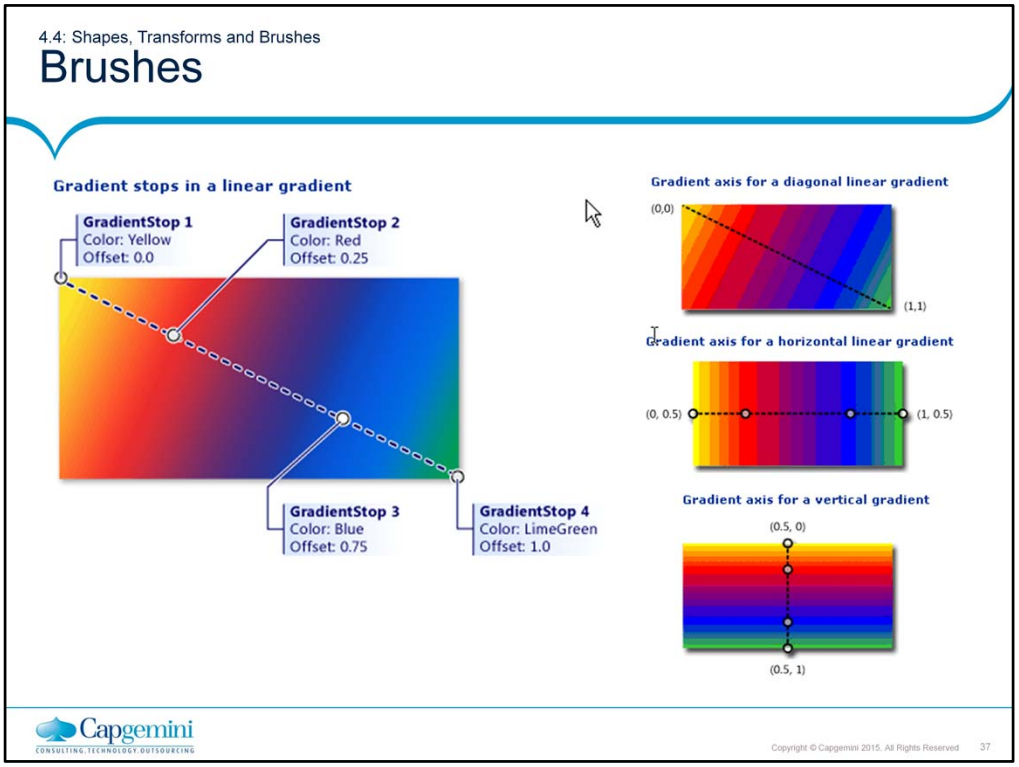
4.4: Shapes, Transforms and Brushes

Brushes

```
<Button>
  <Button.Background>
   <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
     <GradientStop Color="Yellow" Offset="0.0" />
     <GradientStop Color="Red" Offset="0.25" />
     <GradientStop Color="Blue" Offset="0.75" />
     <GradientStop Color="LimeGreen" Offset="1.0" />
   </LinearGradientBrush>
  </Button.Background>   Click Me!</Button>
```

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 36


4.4: Shapes, Transforms and Brushes
Brushes
Gradient stops in a linear gradient
GradientStop 1
Color: Yellow
Offset: 0.0
GradientStop 2
Color: Red
Offset: 0.25
GradientStop 3
Color: Blue
Offset: 0.75
GradientStop 4
Color: LimeGreen
Offset: 1.0
Gradient axis for a diagonal linear gradient
(0,0)
(1,1)
Gradient axis for a horizontal linear gradient
(0, 0.5)
(1, 0.5)
Gradient axis for a vertical gradient
(0.5, 0)
(0.5, 1)
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
37

4.4: Shapes, Transforms and Brushes

Brushes

- The GradientStop is the basic building block of a gradient brush. A gradient stop specifies a Color at an Offset along the gradient axis:
  - The gradient stop's Color property specifies the color of the gradient stop. You may set the color by using a predefined color or by specifying ScRGB or hexadecimal ARGB values
  - The gradient stop's Offset property specifies the position of the gradient stop's color on the gradient axis
  - The offset is a Double that ranges from 0 to 1
  - The closer a gradient stop's offset value is to 0, the closer the color is to the start of the gradient
  - The closer the gradient's offset value is to 1, the closer the color is to the end of the gradient

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 38

4.4: Shapes, Transforms and Brushes

Demo

- Drawing various shapes, Transforms, Brushes

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 39

Summary

- In this lesson, we had a look at:
  - Working with Pages and Navigation
  - Using Styles, Resources, Brushes, Shapes

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 40

Windows Presentation Foundation

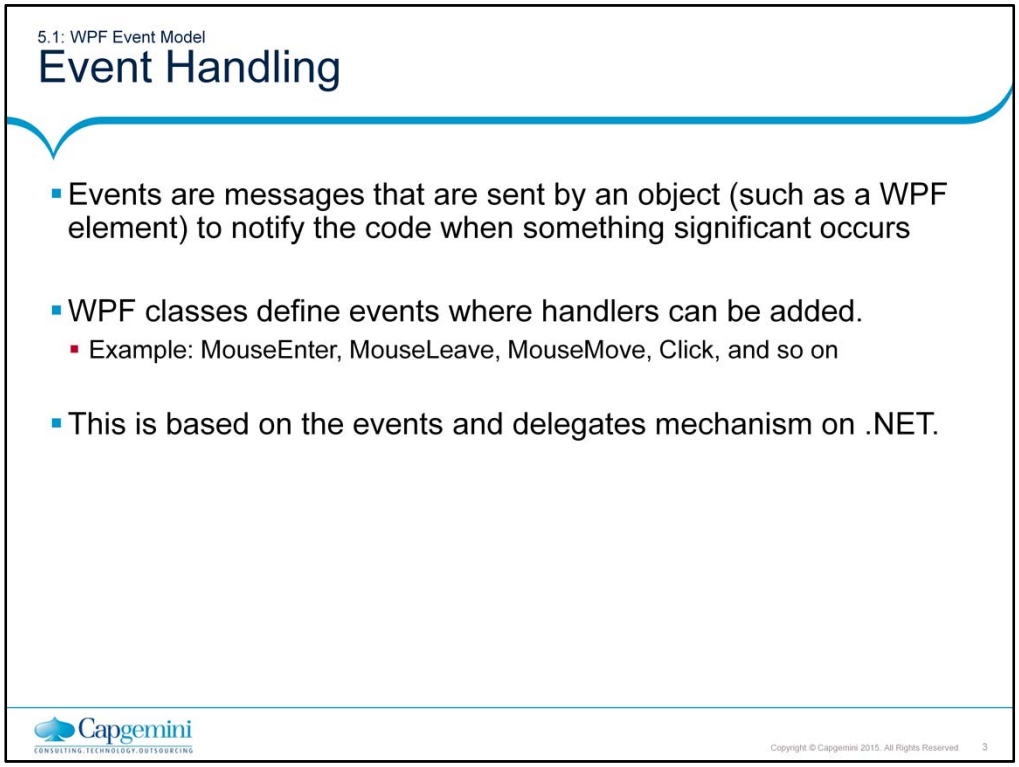
Lesson 5: WPF Event Model

Lesson Objectives

- In this lesson, you will learn:
  - WPF Event Model
    - Event Routing, Bubbling & Tunneling

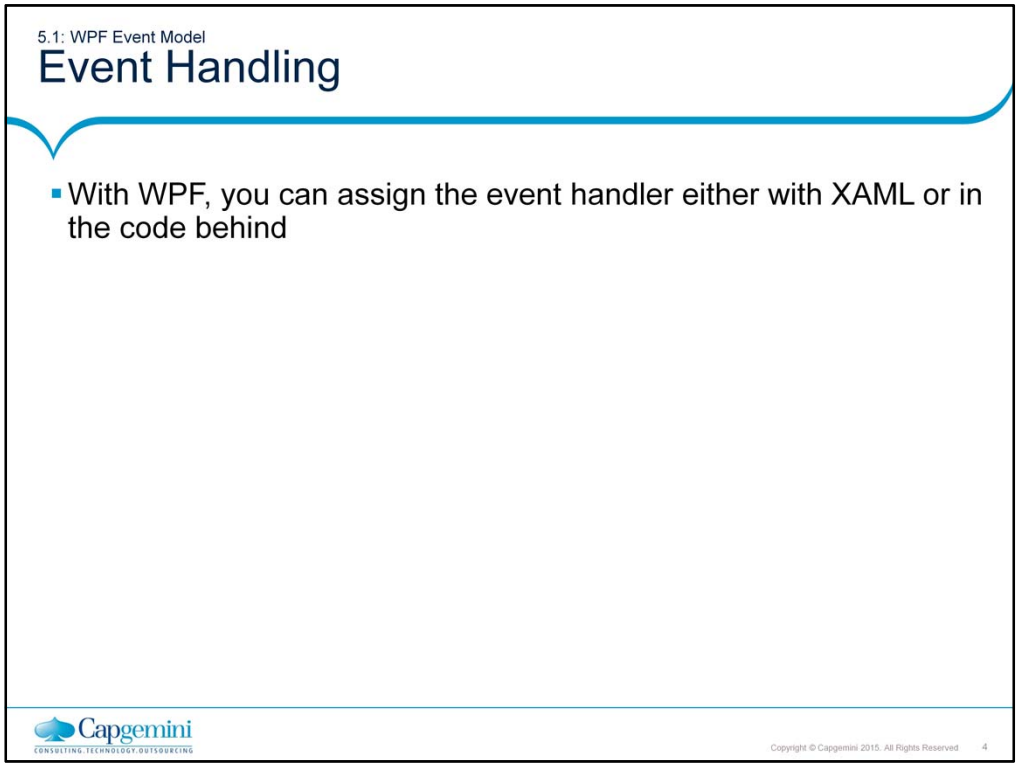
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

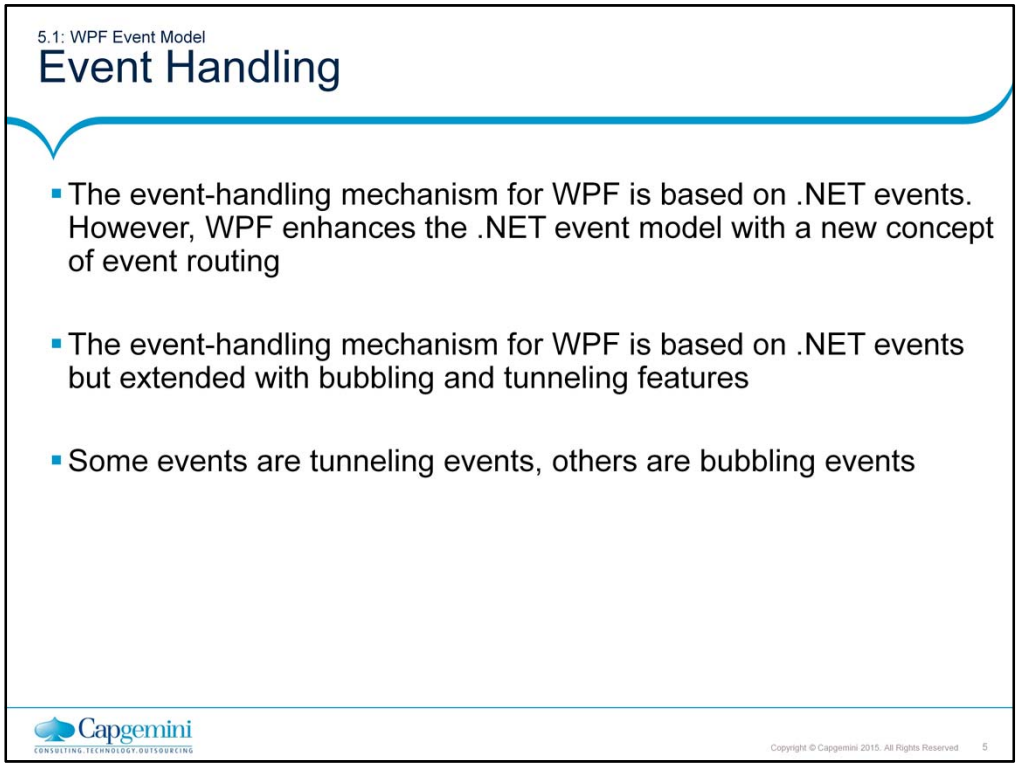
 2



WPF Design Principles: Event Handling in WPF:
Event Handling:
As you have already learnt, a Button can contain graphics, list boxes, another button, and so on.
What happens if a CheckBox is contained inside a Button, and you click the CheckBox? Where should the event arrive?

The answer is that the event is bubbled. First, the Click event arrives with the CheckBox, and it then bubbles up to the Button. This way you can handle the Click event for all elements that are inside the Button with the Button.


5.1: WPF Event Model
Event Handling
With WPF, you can assign the event handler either with XAML or in the code behind
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
4



WPF Design Principles: Event Handling in WPF:
Event Handling (contd.):
As you have already learnt, a Button can contain graphics, list boxes, another button, and so on. What happens if a CheckBox is contained inside a Button and you click the CheckBox? Where should the event arrive?
The answer is that the event is bubbled. First, the Click event arrives with the CheckBox, and it then bubbles up to the Button. In this way, you can handle the Click event for all elements that are inside the Button with the Button.

5.1: WPF Event Model

Types of Routed Events

- Bubbling events are events that travel up the containment hierarchy
  - For example, MouseDown is a bubbling event
- It is raised first by the element that is clicked
- Next, it is raised by that element's parent, and then by that element's parent, and so on, until WPF reaches the top of the element tree

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

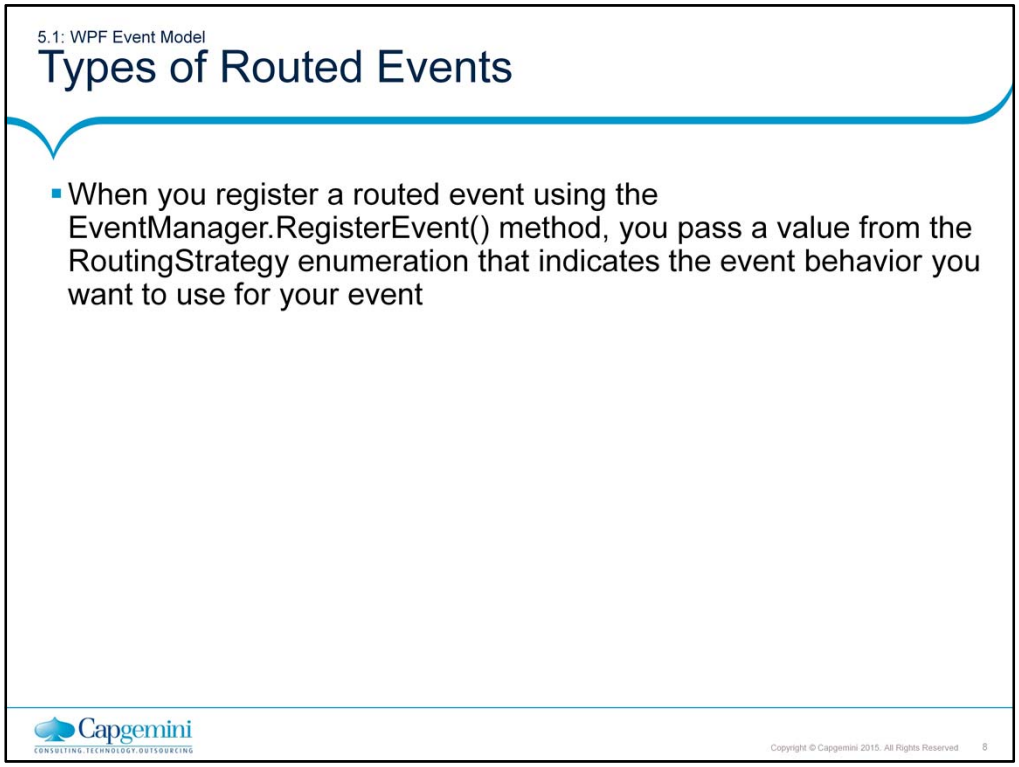
5.1: WPF Event Model

Types of Routed Events

- Tunneling events are events that travel down the containment hierarchy
- They give you the chance to preview (and possibly stop) an event before it reaches the appropriate control
  - For example, PreviewKeyDown allows you to intercept a key press
- First at the window level, and then in increasingly more specific containers until you reach the element that had focus when the key was pressed

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7


5.1: WPF Event Model
Types of Routed Events
When you register a routed event using the EventManager.RegisterEvent() method, you pass a value from the RoutingStrategy enumeration that indicates the event behavior you want to use for your event
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
8

5.1: WPF Event Model

Event Handling

- Event routing allows an event to originate in one element but be raised by another one
  - For example, event routing allows a click that begins in a toolbar button to rise up to the toolbar and then to the containing window before it is handled by your code
- A tunneling event first arrives with the outer element and tunnels to the inner elements
- Bubbling events start with the inner element and bubble to the outer elements

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

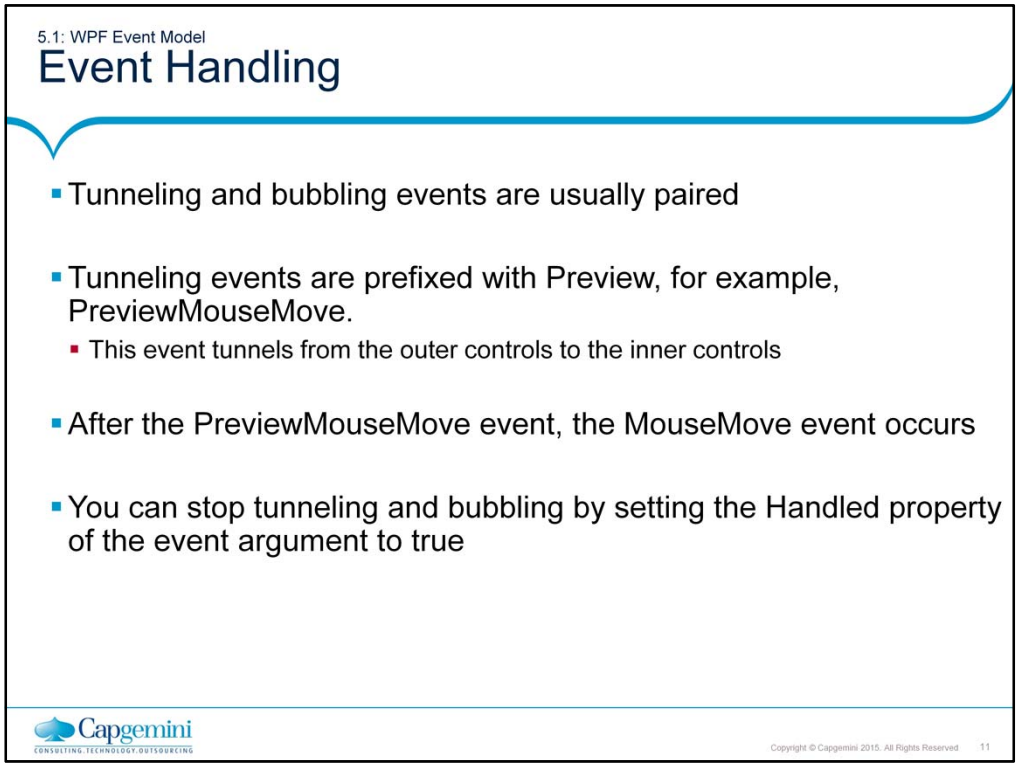
5.1: WPF Event Model

Routed Events

- Just as WPF adds more infrastructure on top of the simple notion of .NET properties, it also adds more infrastructure on top of the simple notion of .NET events
- Routed events are events that are designed to work well with a tree of elements
- When a routed event is raised, it can travel up or down the visual and logical tree, getting raised on each element in a simple and consistent fashion, without the need for any custom code

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10



WPF Design Principles: Event Handling in WPF:
Event Handling (contd.):
You can stop tunneling and bubbling by setting the Handled property of the event argument to true.
The Handled property is a member of the RoutedEventArgs class. All event handlers that participate with the tunneling and bubbling facility have an event argument of type RoutedEventArgs or a type that derives from RoutedEventArgs.

5.1: WPF Event Model

Routed Events

```
public abstract class ButtonBase : ContentControl, ...
{
// The event definition.
public static readonly RoutedEvent ClickEvent;
// The event registration.
static ButtonBase()
{
ButtonBase.ClickEvent = EventManager.RegisterRoutedEvent(
"Click", RoutingStrategy.Bubble,
typeof(RoutedEventHandler), typeof(ButtonBase));
...
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

5.1: WPF Event Model

Routed Events

```
// The traditional event wrapper.
public event RoutedEventHandler Click
{
add
{
base.AddHandler(ButtonBase.ClickEvent, value);
}
remove
{
base.RemoveHandler(ButtonBase.ClickEvent, value);
}
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

5.1: WPF Event Model

Demo

- Demo of Event Handling

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Summary

- In this lesson, we had a look at:
  - WPF Event Model
    - What is Event Routing, Event Bubbling & Tunneling

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

Windows Presentation Foundation

Lesson 6: WPF Data Binding

Lesson Objectives

- In this lesson, you will learn:
  - WPF Data Binding model
  - Working with INotifyPropertyChanged interface
  - Using Observable Collections

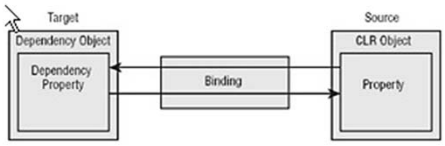
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

6.1: Introduction

Concept Of Data Binding

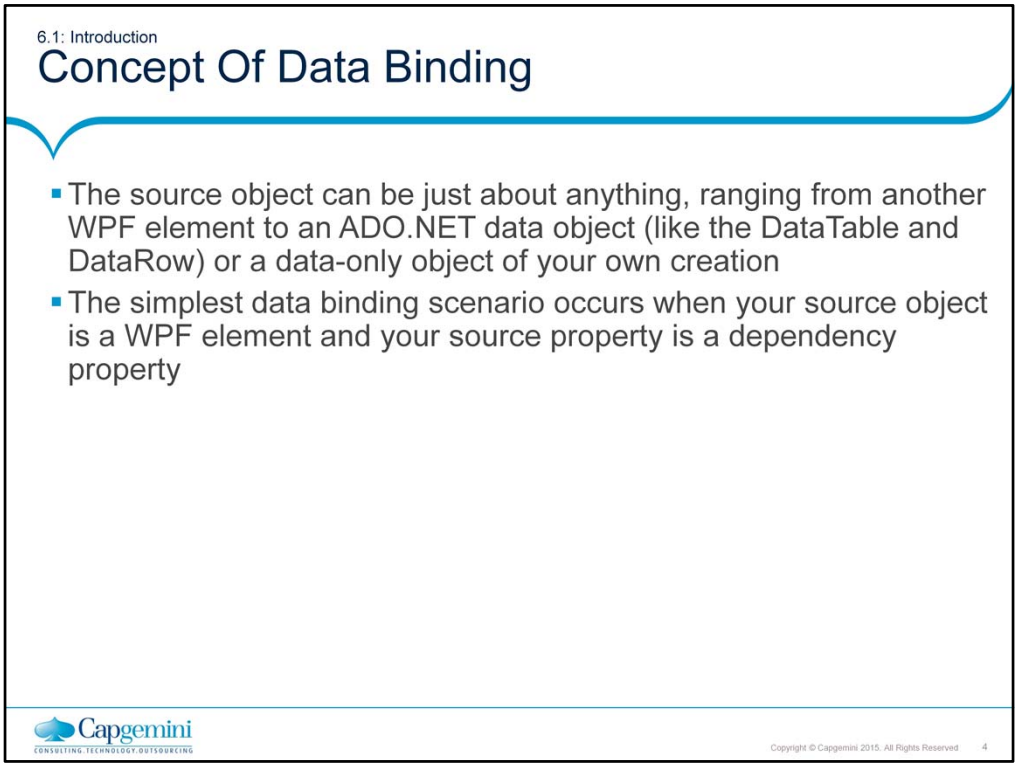
- Data binding is a relationship that conveys to WPF to extract some information from a source object and use it to set a property in a target object
- The target property is always a dependency property, and it is usually in a WPF element
- The ultimate goal of WPF data binding is to display some information in the user interface



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

Data Binding:
Concept :
At its simplest, data binding is a relationship that conveys to WPF to extract some information from a source object and use it to set a property in a target object. The target property is always a dependency property, and it is usually in a WPF element — after all, the ultimate goal of WPF data binding is to display some information in your user interface. However, the source object can be just about anything, ranging from another WPF element to an ADO.NET data object (like the DataTable and DataRow) or a data-only object of your own creation.



Data Binding:
Concept (contd.):
The simplest data binding scenario occurs when your source object is a WPF element and your source property is a dependency property.
That is because dependency properties have built-in support for change notification. As a result, when you change the value of the dependency property in the source object, the bound property in the target object is immediately updated. This is exactly what you want, and it happens without requiring you to build any additional infrastructure.

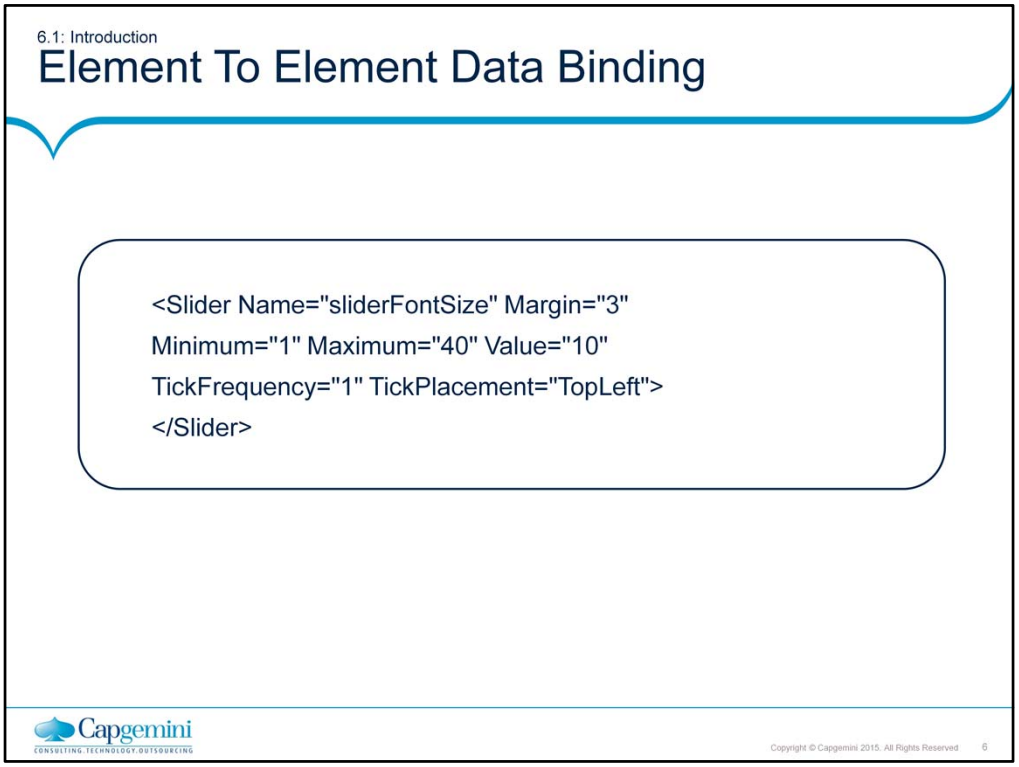
6.1: Introduction

Concept Of Data Binding

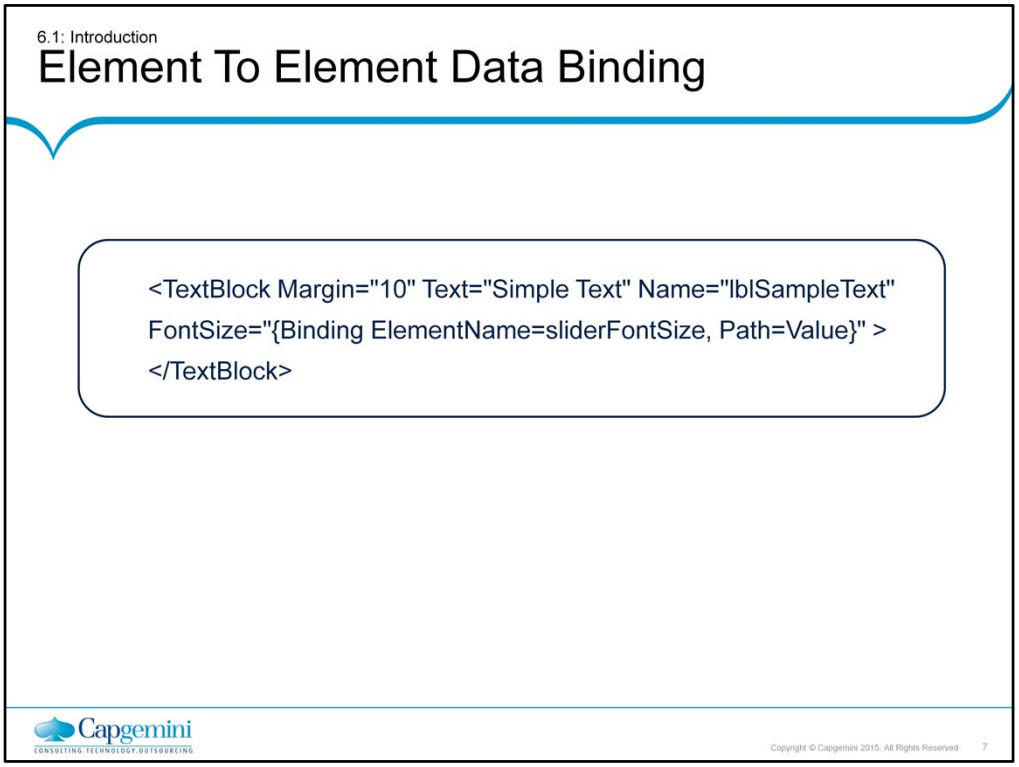
- This is because dependency properties have built-in support for change notification
- As a result, when you change the value of the dependency property in the source object, the bound property in the target object is updated immediately

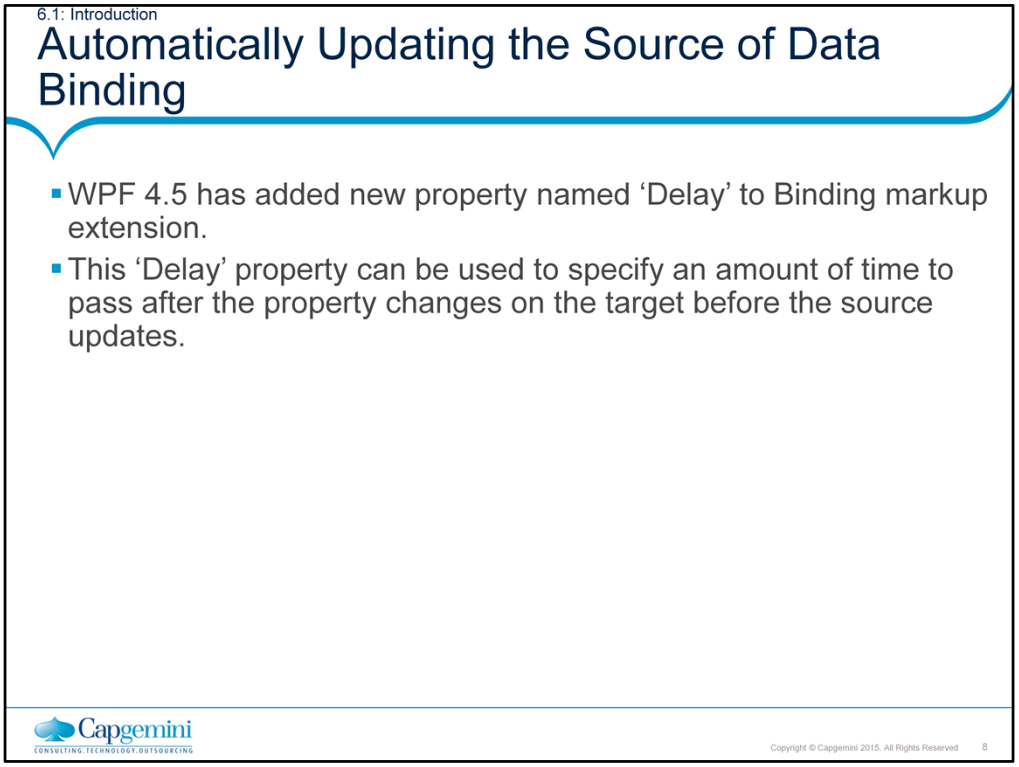
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5



Data binding expressions use a XAML markup extension (and hence have curly braces). You begin with the word Binding, because you're creating an instance of the System.Windows.Data.Binding class. Although you can configure a Binding object in several ways, in this situation you need to set just two properties: the ElementName that indicates the source element and a Path that indicates the property in the source element.


6.1: Introduction
Element To Element Data Binding
<TextBlock Margin="10" Text="Simple Text" Name="lblSampleText"
FontSize="{Binding ElementName=sliderFontSize, Path=Value}" >
</TextBlock>
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved. 7

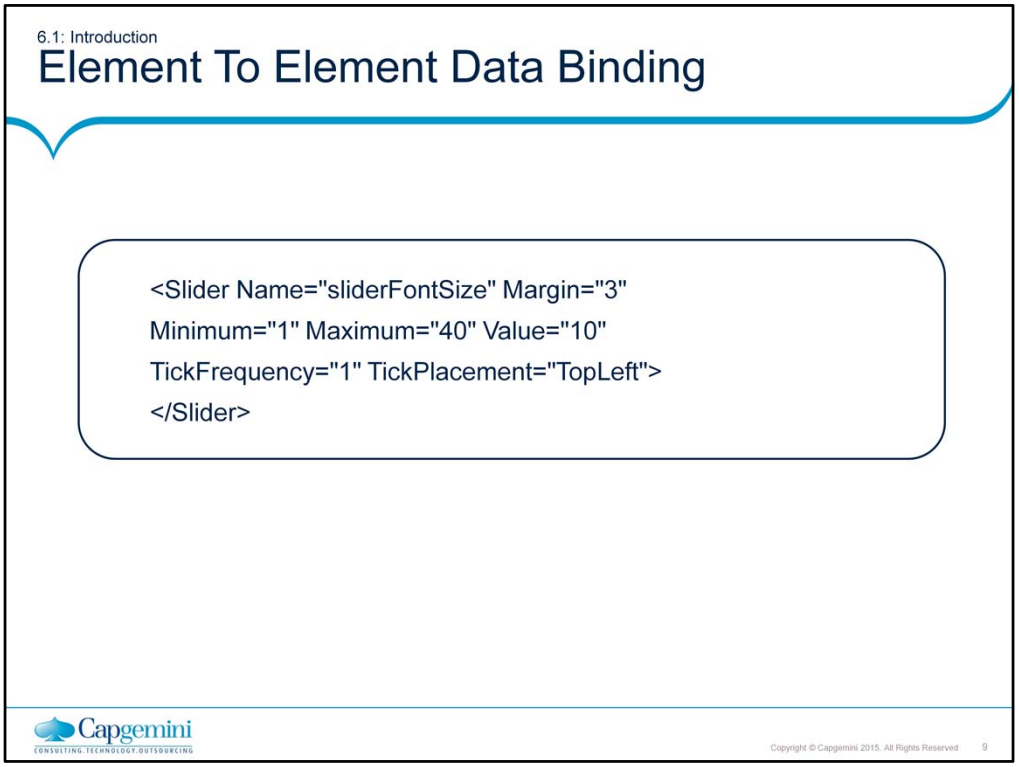


Automatically Updating the Source of Data Binding:

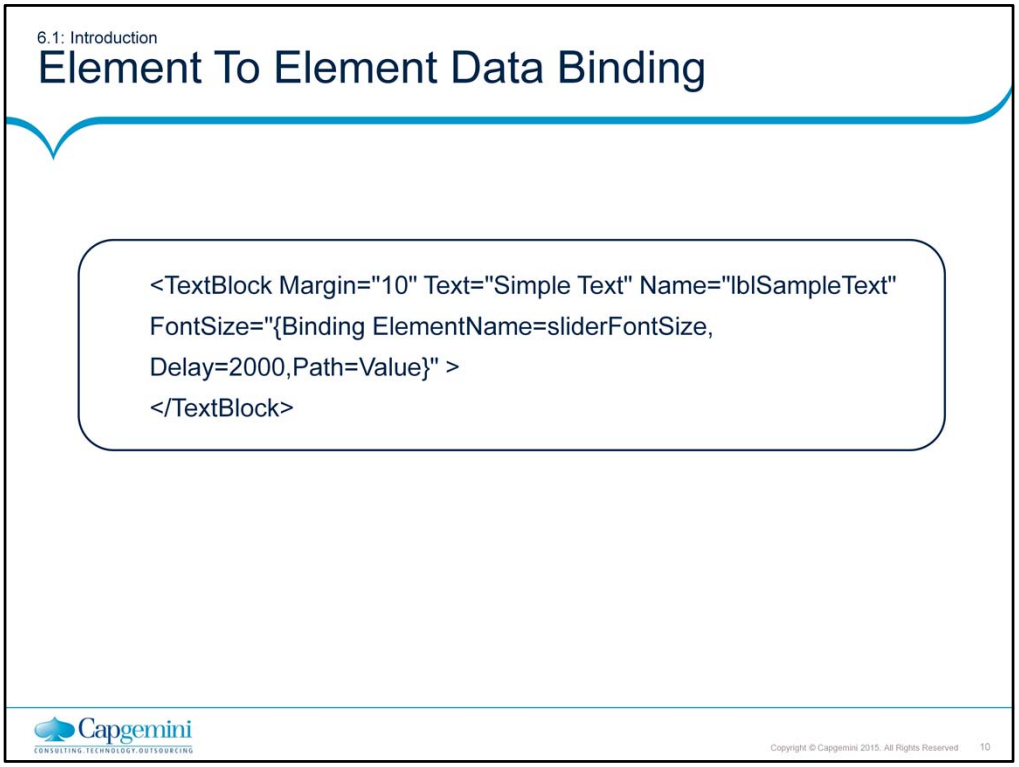
A binding is created between two objects namely the source (from where the data comes) and the target (where the data is propagated).
A Binding is nothing but synchronization between two properties. This synchronization takes place immediately i.e. each change of the target value even a small one will update the source.
WPF 4.5 has added new property named ‘Delay’ to Binding markup extension. This ‘Delay’ property can be used to specify an amount of time to pass after the property changes on the target before the source updates.

This feature can be very useful:

When working with controls like the Slider for which there is no need to update a source value for each pixel that the Slider moves.
To display the typed text in TextBox by “block” instead displaying one letter at a time.



Data binding expressions use a XAML markup extension (and hence have curly braces). You begin with the word Binding, because you're creating an instance of the System.Windows.Data.Binding class. Although you can configure a Binding object in several ways, in this situation you need to set just two properties: the ElementName that indicates the source element and a Path that indicates the property in the source element.


6.1: Introduction
Element To Element Data Binding
<TextBlock Margin="10" Text="Simple Text" Name="lblSampleText"
FontSize="{Binding ElementName=sliderFontSize,
Delay=2000,Path=Value}" >
</TextBlock>
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved. 10

Concept Of Data Binding

- Databinding and DataTemplates are the most powerful features of WPF
- Support for Databinding is built into WPF from its core
- Almost every graphics/UI object that you will work with in WPF inherits from DependencyObject
- The functionality supported by this base is what powers animation, styling, and databinding

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

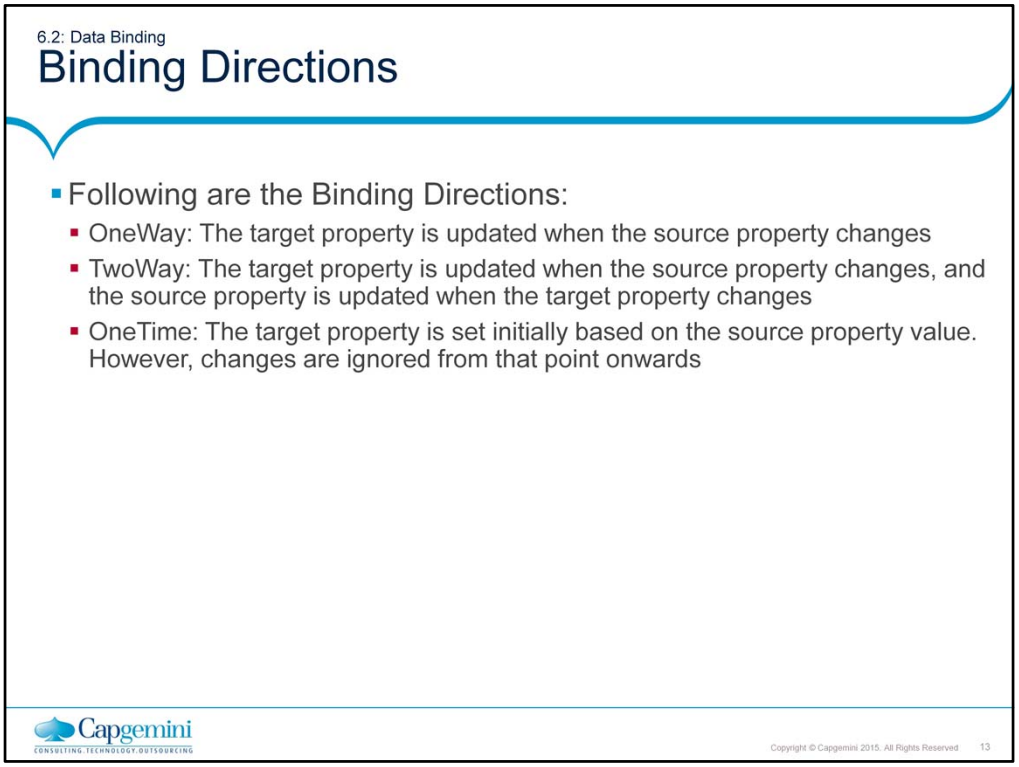
6.1: Introduction

Concept Of Data Binding

- Objects that inherit from DependencyObject support a special type of property called a DependencyProperty
- Most of the properties you will work with are DependencyProperties
- Example: Text, Content, Width, Height
- Any DependencyProperty can be animated, styled, and databound

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12



Data Binding:
Binding Directions:
Following are the binding directions:

One-time:

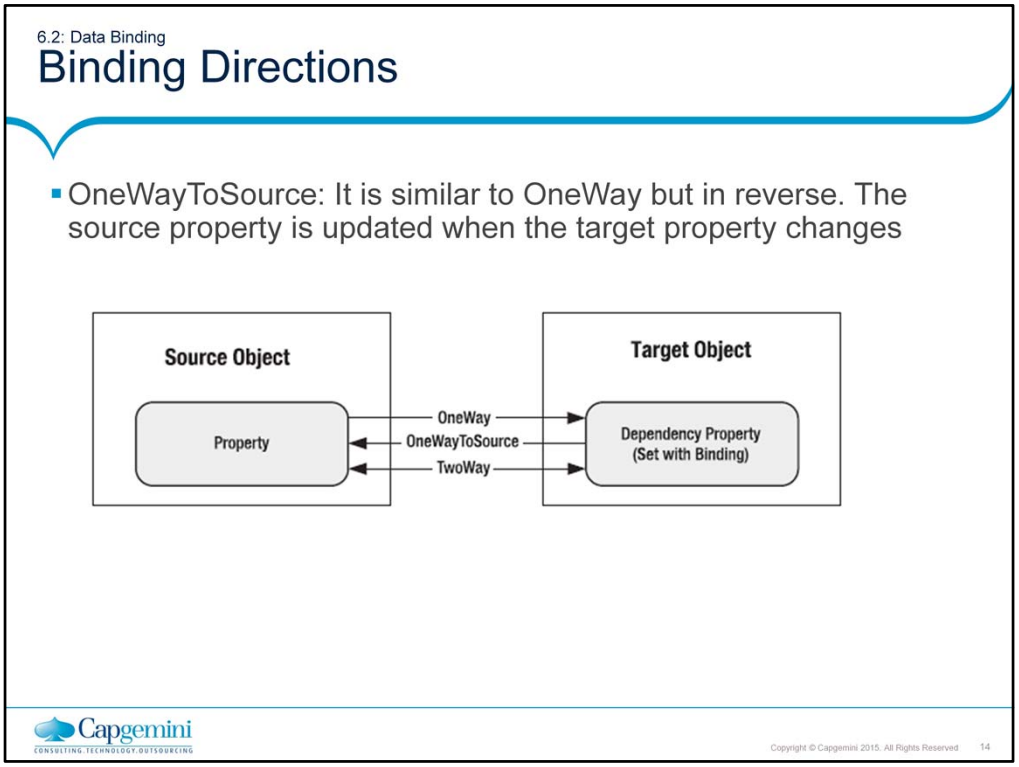
Binding goes from the source to the target and occurs only once when the application is started or the data context changes. Here, you get a snapshot of the data.

One-way:

Binding goes from the source to the target. This is useful for read-only data, as it is not possible to change the data from the user interface. To get updates to the user interface, the source must implement the interface INotifyPropertyChanged.

Two-way:

With a two-way binding, the user can make changes to the data from the UI. Binding occurs in both directions - from the source to the target and from the target to the source.



Data Binding:
Binding Directions (contd.):

One-way-to-source:

With one-way-to-source binding, if the target property changes, then the source object gets updated.

6.2: Data Binding

Binding To Objects

- When binding to an object that is not an element, you need to give up the Binding.Element-Name property and use one of the following properties instead:
- Source. This is a reference that points to the source object,in other words, the object that's supplying the data
- DataContext. If you do not specify a source using the Source property, WPF searches up the element tree starting at the current element

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

6.2: Data Binding

Binding To Objects

- It examines the DataContext property of each element and uses the first one that is not null
- The DataContext property is extremely useful if you need to bind several properties of the same object to different elements, because you can set the DataContext property of a higher-level container object rather than directly on the target element

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

6.2: Data Binding

Demo

- Demo on Data Binding custom objects

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

6.2: Data Binding

Change Notification

- When the WPF element property changes, it notifies the target about the change
- This will not happen for custom Objects, as they will not provide Change Notification
- You can use three approaches to solve this problem
  - You can make each property in the Custom class a dependency property
  - You can raise an event for each property

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

6.2: Data Binding

Change Notification

- You can implement the System.ComponentModel.INotifyPropertyChanged interface, which requires a single event named PropertyChanged
- The first approach relies on the WPF dependency property infrastructure, while both the second and the third rely on events
- Usually, when creating a data object, you will use the third approach

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

6.2: Data Binding

INotifyPropertyChanged

```
public class Product : INotifyPropertyChanged
{
public event PropertyChangedEventHandler
PropertyChanged;
public void OnPropertyChanged(PropertyChangedEventArgs e)
{
if (PropertyChanged != null)

Continued on next slide..
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

6.2: Data Binding

INotifyPropertyChanged

```
PropertyChanged(this, e);{private decimal unitCost;
public decimal UnitCost
{
get { return unitCost; }
set {
unitCost = value;
OnPropertyChanged(new
PropertyChangedEventArgs("UnitCost"));
} } }
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

6.2: Data Binding

List Binding

- Binding to a list is more frequently done than binding to simple objects
- Binding to a list is very similar to binding to a simple object
- You can assign the complete list to the DataContext from code behind, or you can use an ObjectDataProvider that accesses an object factory that returns a list

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

6.2: Data Binding

List Binding

- With elements that support binding to a list (for example, a ListBox), the complete list is bound
- With elements that just support binding to one object (for example, a TextBox), the current item is bound

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

6.2: Data Binding

Binding To XML

- WPF data binding has special support for binding to XML data
- You can use XmlDataProvider as a data source and bind the elements by using XPath expressions
- For a hierarchical display, you can use the TreeView control and create the view for the items by using the HierarchicalDataTemplate

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

6.2: Data Binding

Demo

- Demo on Data Binding and List Binding

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25

Summary

- In this lesson, we had a look at:
  - What is Data Binding
  - Working with the following interfaces
    - INotifyPropertyChanged
  - Using Observable Collection

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26

Windows Presentation Foundation

Lesson 7: New Features in WPF 4.5

Lesson Objectives

- In this lesson, you will learn:
  - New Features in WPF 4.5
  - Working with ICustomTypeProvider interface

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

7.1: Introduction

New Features in WPF 4.5

- Just like any other product or technology that undergoes a version update, WPF also brings up a number of new/enhanced features to the table.
- The new features in WPF 4.5 are
  - Binding to types that implement ICustomTypeProvider
  - Repositioning the data as the data's value change (Live shaping)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

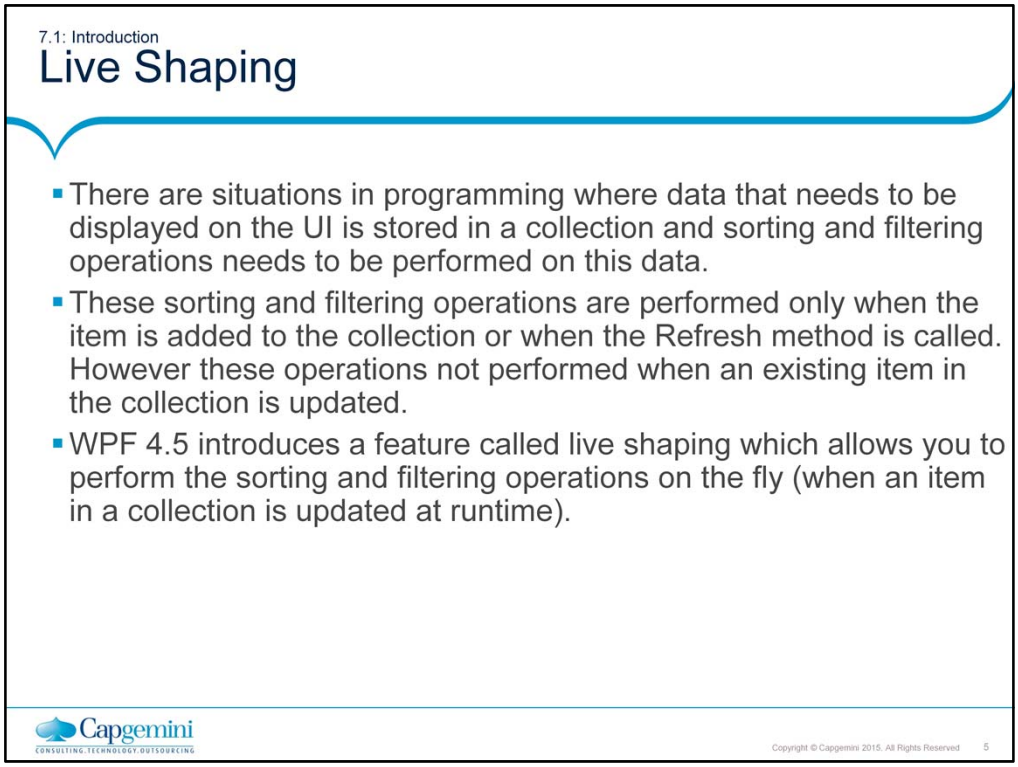
7.1: Introduction

Binding to Types that Implement CustomTypeProvider

- WPF 4.5 adds data-binding support for the interface ICustomTypeProvider. This interface enables data binding to objects the structure of which is not known until runtime.
- With ICustomTypeProvider you can add properties to objects on the fly and then data bind to these newly created objects. For example, you can use this interface to generate bindable objects from XML data with a schema that is known only at run time.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4



CollectionView class represents a view of the specified collection. One of new features introduced in WPF 4.5 is ICollectionViewLiveShaping interface. This interface provides properties which enable sorting and filtering on the CollectionView in real time. Using Live Shaping feature in WPF 4.5 changes made to the specific property of the collection can be monitored and if the property value has changed then the UI is refreshed to reflect the changes.

When to use Live Shaping:

Suppose you have developed an application that lists stocks in the stock market. The stocks are sorted by value. If live sorting is enabled on the stock's CollectionView, a stock's position in the DataGrid moves when the value of the stock becomes greater or less than another stock's value.

6.2: Data Binding

Demo

- Demo on New Features in WPF 4.5

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

Summary

- In this lesson, we had a look at:
  - New Features available in WPF 4.5
  - Working with the following interfaces
    - ICustomTypeProvider

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

Review Question

- Question 1: The ___ property added to the Binding markup extension defines the timespan after which source is updated.
- Question 2: The _____ interface can be used to bind to objects with dynamically generated properties.
- Question 3: _____ feature of WPF 4.5 shapes the collection view in live.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

WINDOWS PRESENTATION FOUNDATION

Lab Book

Document Revision History

Date	Revision No.	Author	Summary of Changes
21-Oct-09	1.0	Ganesh Desai	Initial Document
21-Oct-09	1.0	Ummeaiman Diwanji	Quality Review, Transfer to new template.
15-July-2011	2.0	Ganesh Desai	Changes made as per integration process
28-Aug-2012	3.0	Abishek Radhakrishnan	Revamp of Lab Book.

Table of Contents

Getting Started

Overview

This lab book is an unguided tour for learning WPF 4.5. It comprises of a use case diagram which will have to be realized in a sequence of exercises. Screen snap shots are provided wherever necessary.

Setup Checklist for WPF

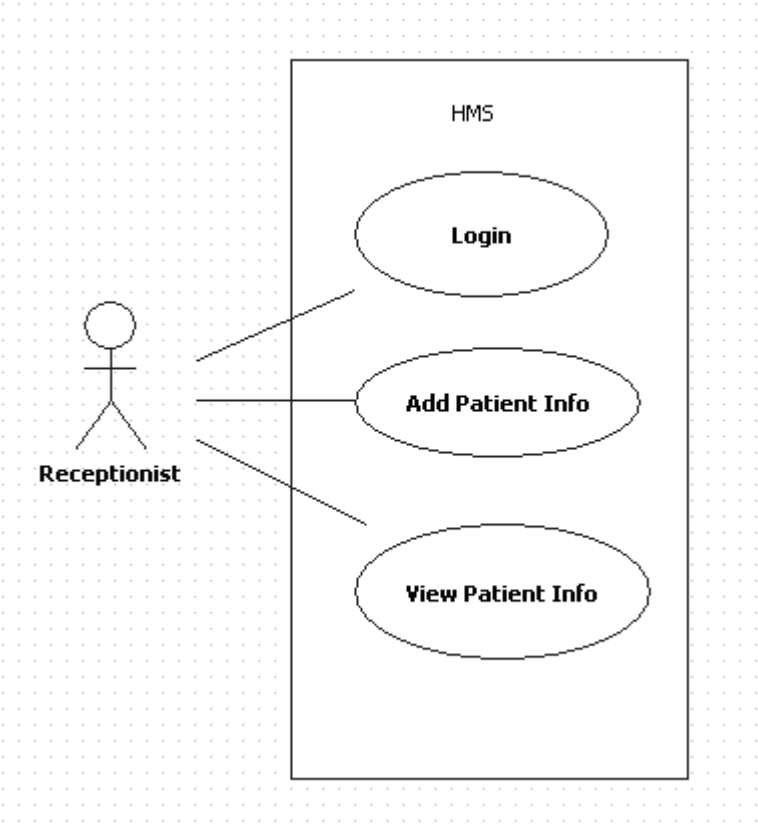
Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

- *Hardware:* Networked PCs with minimum 1 GB RAM and 60 MB HDD.
- Software: Microsoft Visual C# 2012 Express Edition or Visual Studio.NET 2012.

Problem Statement/ Case Study

HelpLine Hospitals needs a Hospital Management System (HMS) to manage their daily work flow. You have been given a part of this project to implement. The following use cases have to be realized.



Lab 1.WPF Layouts And Controls

Goals	• Working with a Window. • Using Container controls.
Time	1.5 hours.

1.1: Create a Login form as shown in the sample screen below. The user name and password can be “admin” and “admin”. Use a StackPanel,DockPanel and Grid to implement the design.

Note: complete questions 1 to 4 from the “Extended Practice” section.

.

CONCLUSION POINTS:

a. **think about scenarios where a WrapPanel can be used.**
b. **Which container control is the most flexible for a developer to work with? What would be the limitation of the same?**

Lab 2.Styles and Resources

Goals	• Working with styles and resources. • Using Gradient colours and shapes.
Time	2 hours.

2.1: Modify the window completed in exercise 1.1 to meet the following requirements:

- the font of the labels must be bold, Arial and size 12.
- the button must have an elliptical shape.
- the login button's opacity property changes from 0.5 to 1 when the mouse focus is on it. Use triggers in XAML to achieve this.
- The window must have a gradient back ground colour of your choice. You must have a minimum of 3 colours.
- All the above requirements must be fulfilled using styles or resources in the App.xaml.

Note: complete questions 5 to 7 from the "Extended Practice" section.

CONCLUSION POINTS:

a. **Where else can styles or resources be applied? What would be the level of reusability in each case?**
b. **Can the above requirements be implemented using C# code? If so, what is the advantage of using XAML ?**

Lab 3.Event Handling and Data Binding

Goals	• Working with Events. • Binding Data and WPF controls.
Time	4 hours.

3.1: The TextBox used for entering the user name must not allow the user to enter any numbers. Implement this functionality using the "PreviewTextInput" event of the TextBox. Is this event a Tunneled event or a Bubbled event ? Discuss.

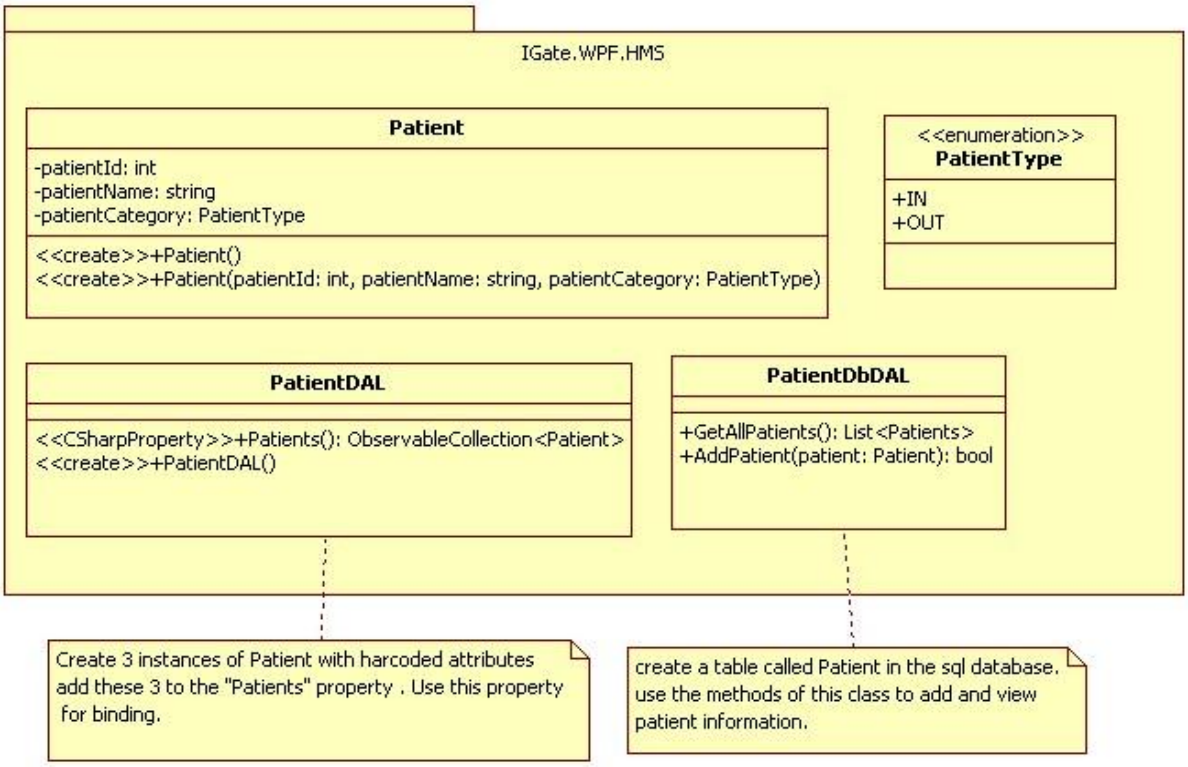
3.2 : Add Functionality to the Login and Reset Button.

enter user name as "admin", password as "admin", click on Login Button	Open a new window as shown in the screen shot
invalid attempt of more than 3	the application exits
clicking on Reset Button.	Clears the fields and brings the focus on the user name TextBox

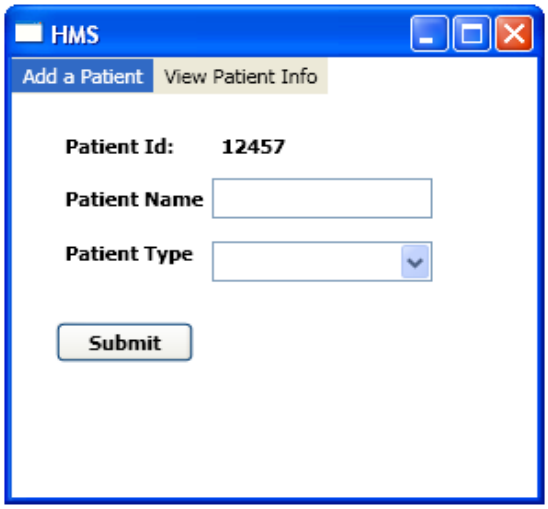
Note : implement the above functionality using bubbled events. [The container of the button controls will handle the click event].

3.3 : Write code to realize the following screens:

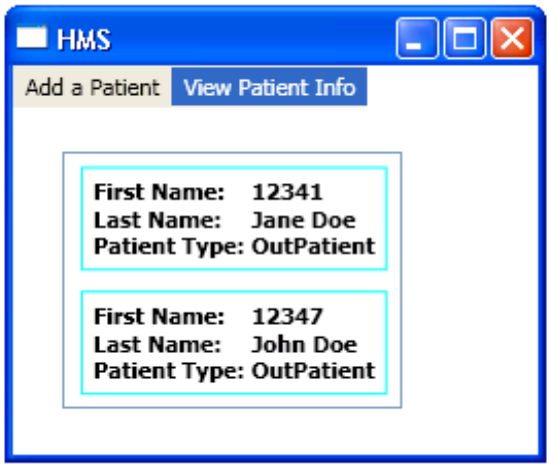
Note: You will have to implement this question with two data sources:
a. PatientDbDAL
b. PatientDAL.
Refer to the class diagram for the DAL structure.



The sample screen shots for the window to be displayed after successful login are given in the following page:



The highlighted menu item exposes the UI to add a patient. Generate the patient Id automatically.



The highlighted menu item exposes the UI to view patient details.

CONCLUSION POINTS:

a. **What is the special feature of an Observable Collection?**
b. **What is the specialty of Routed Events in WPF?**
c. **What scenarios can you think for using the various modes in data binding?**

EXTENDED PRACTICE:
TIME: 1 hour.

1. Explore the use of the Expander and Tab Control.
2. What is the use of the “Stretch” property in an Image control?
3. Clip an Image to make it circular or elliptical in shape.
4. A Button’s content needs to have an Image and it’s description in a textblock. Button, being a Content control,can have its content property set only once. How will you achieve the above requirement?
5. Apply a Linear Gradient to a control using C# code.
6. Create 2 pages, Page1.xaml and Page2.xaml. use a hyperlink to navigate from Page1 to Page2.
7. Transform a rectangle using the Render, Translate, Scale and Skew Transforms

CODE DEBUGGING /ENHANCEMENTS
TIME: 1 hour.

Your instructor will provide you with a code which has some errors. Identify the errors, and make the changes necessary to make the code compile and execute

