MODULE 05

XAML

MODULE TOPICS

Declarative Approach to UI Design

XAML Namespaces

Elements and Attributes

Property Elements

Type Converters

Markup Extensions

Nested Elements

BAML and Generated Code

Code-Behind Files

DECLARATIVE APPROACH TO UI DESIGN

- In WPF, Extensible Application Markup Language (XAML and pronounced zammel) is used to describe user interfaces
- Allows for separation of concerns
- Declarative way to construct and initialize objects

DECLARATIVE APPROACH TO UI DESIGN

- WPF and XAML can be used independently
  - Everything done in XAML can be done in code
  - WPF, XPS, Silverlight, Xarmin Forms all use XAML
- Usually a tool is used to write XAML, such as Blend or Visual Studio
  - XAML Studio is another tool available

GRAPHICAL USER INTERFACES BEFORE WPF

- Windows Forms were developed visually, but the designer simply added code to the app
- Graphic designers did not have any tools to work with Windows Forms
- Used wireframing and mock ups instead

XML NAMESPACES

- XAML uses XML namespaces to specify what .NET namespace the class is located in

  ```
  http://schemas.microsoft.com/winfx/2006/xaml/presentation
  ```

  - Core WPF namespace, includes all WPF classes
  - Usually the default namespace

  ```
  http://schemas.microsoft.com/winfx/2006/xaml
  ```

  - XAML namespace, includes XAML utility features
  - Usually the namespace is map to the prefix x

ELEMENTS AND ATTRIBUTES

```
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        Content="OK"/>
```

```
System.Windows.Controls.Button b = new System.Windows.Controls.Button();
b.Content = "OK";
```

PROPERTY ELEMENTS

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

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

TYPE CONVERTERS

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

- XAML attributes are always strings, but need to map to any .NET type for class properties
- Type Converters convert the attribute strings to .NET types

MARKUP EXTENSIONS

```
<TextBox Text="{Binding Path=LastName}"/>
```

- Markup extensions enable you to extend the expressiveness of XAML
- Whenever an attribute value is enclosed in curly braces, the XAML compiler treats it as a markup extension

NESTED CONTENT

```
<TextBox Width="250">Text Property</TextBox>
```

3 mechanisms are used evaluated in this order

1. If parent implements IList, the parser calls IList.Add() and passes the content
2. If the parent implements IDictionary, the parser calls IDictionary.Add() and passes the content
3. If the parent is decorated with [ContentProperty], the parser uses content to set that property

BAML

- When a WPF app is compiled, the XAML files are converted into BAML which is embedded as a resource into the assembly
- BAML is tokenized, so longer XAML is replaced with shorter tokens
- Also optimized for faster parsing and runtime
- Can also use XAML without compiling it, probably for Just In Time design

CODE BEHIND FILE

```
<Window x:Class="WindowsApplication1.Window1"
```

XAML uses the class attribute to connect it to a code-behind file

WALKTHRU - XAML

ANY QUESTIONS?