Dependency Properties

Introduction

- The dependency property is a property that can be set directly by various different property providers while having them prioritized.
- The MSDN docs defines a dependency property as a "property that is backed by the WPF property system".
- The dependency property depends on multiple property providers. Each one of them has its own level of precedence.

Introduction

- We use dependency properties just as any other property. There is no need knowing in advance that a property we work with is a dependency property.
- Some of the available silverlight features (e.g. binding) are limited to dependency properties. The attached properties are sort of dependency properties. Any property that we bind, style, template, transform or animate must be a dependency property.

Introduction

- Dependency properties are not always needed. When customizing our application most likely we will eventually end up with the need for having dependency properties.

Standard CLR Property Replacement

- The dependency properties acts as kind of wrappers around a field. The dependency properties are kind of a replacement for the field a standard property wraps.

Dependency Properties Definition

- In order to define a dependency property we should first instantiate the `System.Windows.DependencyProperty` class. This new object will represent the dependency property.
- The dependency property needs to be always available and for that reason we define it as a static field in the associated class.

Dependency Properties Definition

- The field that defines a dependency property has the name of the ordinary property plus the word `Property` at its end. This way the dependency property definition is separated from the actual property.
- The field should be defined `redonly`. This way its value can be set within the static constructor only.

Dependency Properties Definition

```
public class MyElement: UIElement
{
    public static readonly DependencyProperty TempratureProperty;
    ...
}
```

Dependency Properties Definition

- The next step should be registering the dependency property with the Silverlight platform. We should complete this registration before our using the property. Therefore, we will usually perform this required registration within the scope of the static constructor we define in the associated class.
- We create a `DependencyProperty` instance by calling the static `DependencyProperty.Register()` method.

Dependency Properties Definition

Adding 'Property' to the Property for which we create Dependency Property is another Standard to Follow

```
public class MyElement: UIElement
{
    public static readonly DependencyProperty OuterTempratureProperty;


    static MyElement()
    {
        OuterTempratureProperty = DependencyProperty.Register(
            “OuterTemprature“,
            typeof(Temprature),
            typeof(MyElement),
            null);
    }

    ...

}
```

This is a standard Pattern for Defining Dependency Properties

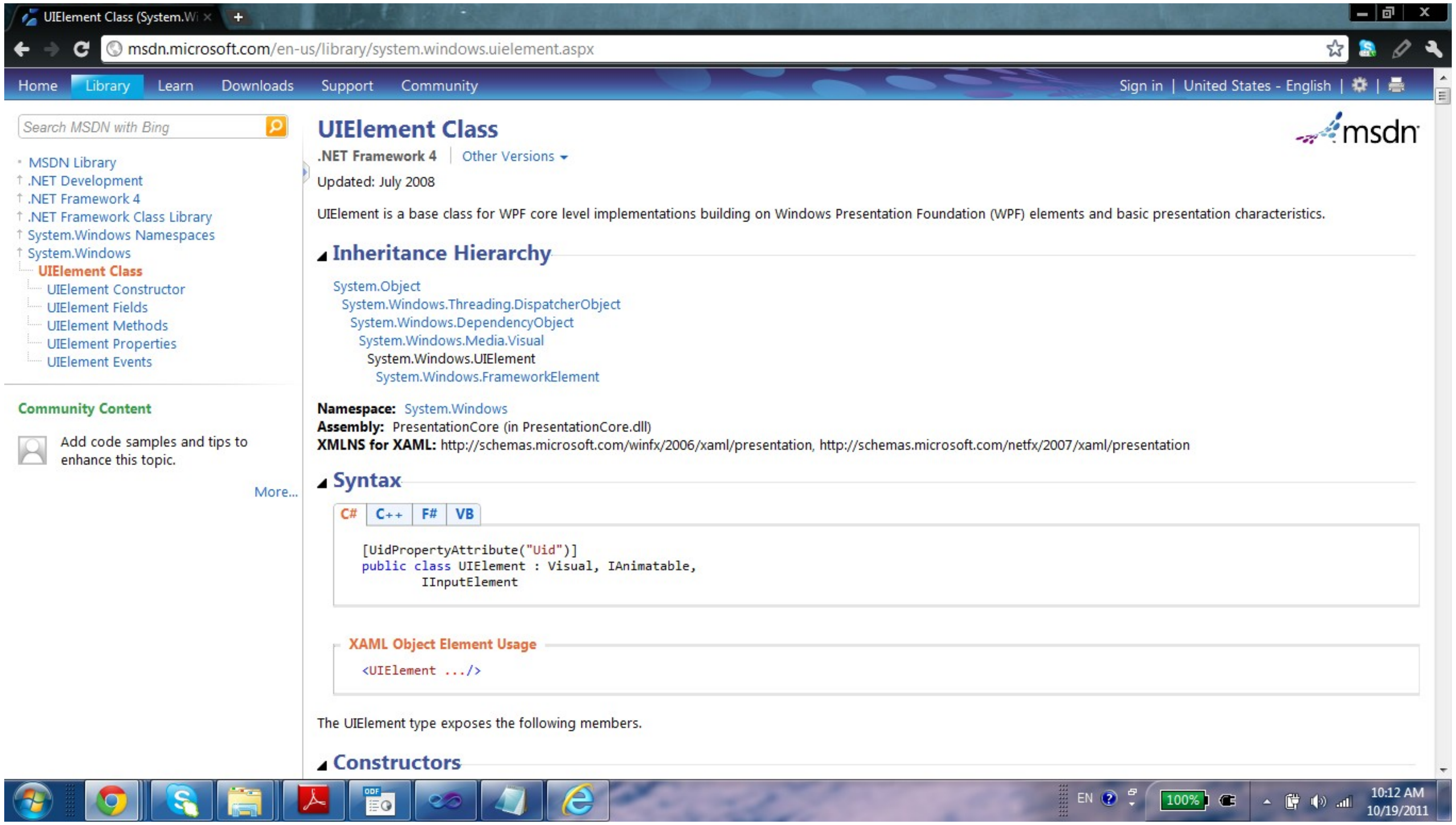
Dependency Properties Definition

- The actual storage of the dependency property value is automatically taken care of, deep inside the WPF property system.

Dependency Properties Definition

- The class that contains a dependency properties must derive from `DependencyObject`.
- Defining a class that extends `UIElement` we fulfill this requirement.

Dependency Properties Definition



UIElement Class

.NET Framework 4 | Other Versions

Updated: July 2008

UIElement is a base class for WPF core level implementations building on Windows Presentation Foundation (WPF) elements and basic presentation characteristics.

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      System.Windows.Media.Visual
        System.Windows.UIElement
          System.Windows.FrameworkElement

Namespace: System.Windows
Assembly: PresentationCore (in PresentationCore.dll)
XMLNS for XAML: http://schemas.microsoft.com/winfx/2006/xaml/presentation, http://schemas.microsoft.com/netfx/2007/xaml/presentation

Syntax

```
[UidPropertyAttribute("Uid")]
public class UIElement : Visual, IAnimatable,
        IInputElement
```

XAML Object Element Usage

```
<UIElement .../>
```

The UIElement type exposes the following members.

Constructors

Dependency Properties Definition

```
public class MyElement: UIElement
{
    public static readonly DependencyProperty OuterTempratureProperty;


    static MyElement()
    {
        OuterTempratureProperty = DependencyProperty.Register(
            “OuterTemprature“,
            typeof(Temprature),
            typeof(MyElement),
            null);
    }

    public Temprature OuterTemprature
    {
        get {return (Temprature)GetValue(OuterTempratureProperty);}
        set {SetValue(OuterTempratureProperty, value);}
    }
    ...
}
```

Dependency Properties Definition

- Calling the `DependencyProperty.Register` function we can pass over a `PropertyMetaData` object through which we can specify the function we want to be called when a dependency property changes its value.

Dependency Properties Definition

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;

namespace ConsoleApplication14
{
    public class Program
    {
        static void Main(string[] args)
        {
            Book book = new Book();
            book.Title = "Core Python";
            Console.WriteLine("### "+book.Title+" ###");
            book.Title = "abc";
            Console.WriteLine("### " + book.Title + " ###");
            book.Title = "Core PHP";
            Console.WriteLine("### " + book.Title + " ###");
            book.Title = "a";
            Console.WriteLine("### " + book.Title + " ###");
        }
    }
```

Dependency Properties Definition

```
public class Book : DependencyObject
{
    public static readonly DependencyProperty TitleProperty =
        DependencyProperty.Register(
            "Title",
            typeof(string),
            typeof(Book),
            new PropertyMetadata(
                "No Name", TitleChangedCallback, TitleCoerceCallback),
            TitleValidateCallback);

    private static void TitleChangedCallback(
        DependencyObject obj, DependencyPropertyChangedEventArgs e)
    {
        Console.WriteLine("Log Message: within TitleChangedCallback");
        Console.WriteLine(e.OldValue + " " + e.NewValue);
    }
```

Dependency Properties Definition

```
private static object TitleCoerceCallback(
    DependencyObject obj, object o)
{
    Console.WriteLine("Log Message: within TitleCoerceCallback");
    string str = o as string;
    //here we can validate the title and change it if needed
    if(str.Length>0)
    {
        Console.WriteLine("new title is OK");
    }
    else
    {
        Console.WriteLine("new title is not OK");
        Console.WriteLine("will set 'no name' instead");
        str = "no name";
    }
    return str;
}
```

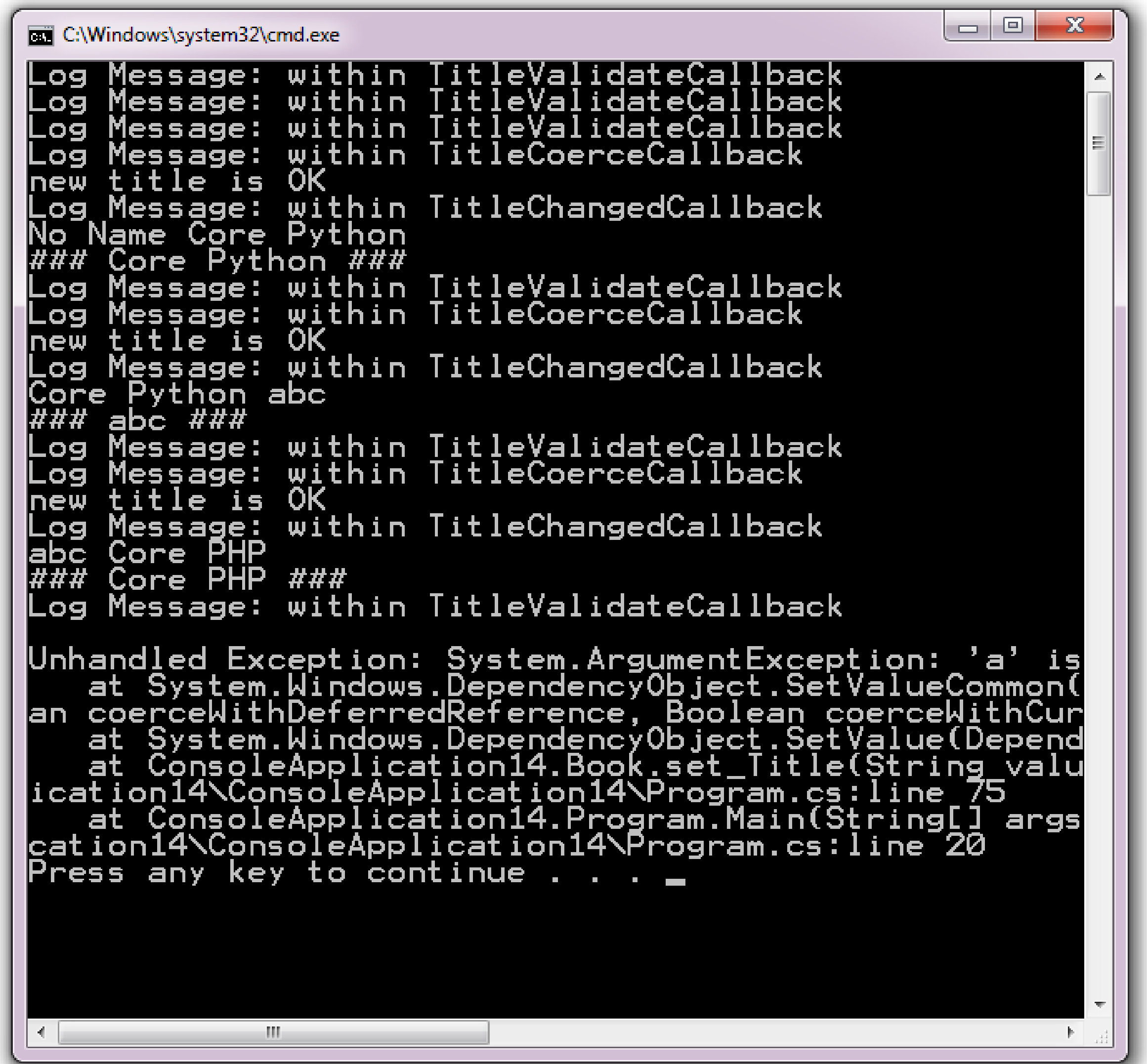
Dependency Properties Definition

```
        private static bool TitleValidateCallback(object value)
        {
            Console.WriteLine("Log Message: within TitleValidateCallback");
            //return true if there is a place to call the validation method
            return value != null && ((string) value).Length > 2;
        }

        public string Title
        {
            get
            {
                return (string)GetValue(TitleProperty);
            }
            set
            {
                SetValue(TitleProperty, value);
            }
        }
    }

}
```

Dependency Properties Definition



```
Log Message: within TitleValidateCallback
Log Message: within TitleValidateCallback
Log Message: within TitleValidateCallback
Log Message: within TitleCoerceCallback
new title is OK
Log Message: within TitleChangedCallback
No Name Core Python
### Core Python ###
Log Message: within TitleValidateCallback
Log Message: within TitleCoerceCallback
new title is OK
Log Message: within TitleChangedCallback
Core Python abc
### abc ###
Log Message: within TitleValidateCallback
Log Message: within TitleCoerceCallback
new title is OK
Log Message: within TitleChangedCallback
abc Core PHP
### Core PHP ###
Log Message: within TitleValidateCallback

Unhandled Exception: System.ArgumentException: 'a' is
   at System.Windows.DependencyObject.SetValueCommon(
an coerceWithDeferredReference, Boolean coerceWithCur
   at System.Windows.DependencyObject.SetValue(Depend
   at ConsoleApplication14.Book.set_Title(String valu
ication14\ConsoleApplication14\Program.cs:line 75
   at ConsoleApplication14.Program.Main(String[] args
cation14\ConsoleApplication14\Program.cs:line 20
Press any key to continue . . . _
```