特效(托管)

# 概述:

用户应该能够在两张照片之间添加特效.目前,我们只打算提供两个简单的特效:淡出特效和飞入特效.将来我们可能会扩展特效库.因此,我们需要设计特效功能的接口.

此文档仅包含托管部分. 参见非托管部分接口设计的规格文档.

# 期望的结果:

* 为应用程序创建和特效库交互的接口.

# 不期望的结果:

* 创建 plug-in 模型.
* 调用实际特效库.

# 设计:

我们提供一个接口和一个包含了一些接口的默认实现的抽象基类.是否要删除接口仅使用抽象基类可以再辩论.无论如何,接口命名为ITransition,设计如下.

    public interface ITransition

    {

        string Name { get; }

        TimeSpan TransitionDuration { get; set; }

        bool ImageZIndexModified { get; }

        // Foreground/BackgroundElement可以是图片或者MediaElement.

        // 在PreviewPage设定这些属性.

        FrameworkElement ForegroundElement { get; set; }

        FrameworkElement BackgroundElement { get; set; }

        event EventHandler TransitionCompleted;

        ITransition Clone();

        // 开始/终止特效.

        void Begin();

        void Stop();

        // 序列化/反序列化特效.

        void Save(XElement transitionElement);

    }

## 常用属性/事件/方法

一个特效应该有一个唯一的名称.用户可以指定每张照片上的特效花了多少时间.目前,我们要求每个特效执行到的两张照片上:一个背景的照片(照片过境)和一个前台的照片(照片目前正在显示).预览页面是负责传递正确的背景/前景图像给特效.特效完成后,TransitionCompleted事件必须触发通知预览页面. TransitionBase类有触发此事件的默认实现.最后,合成页面要求每个特效支持深层克隆.每个特效可以实现Clone方法,并选择一个(或数个)需要被克隆的属性.

## Z-索引

预览页面需要相关背景/前景图片的z-索引信息.一些特效,比如淡出特效,不改变z-索引.其他特效,如飞入特效,改变z-索引.例如,在飞入特效需要前景图片覆盖背景图片.在这种情况下,特效必须设置ImageZIndexModified为true,使预览页面知道如何处理它们.

## 开始/终止特效

预览页面需要开始/终止特效.它将这个任务拖管给特效.每个特效必须实现开始/终止的方法.我们不在TransitionBase中提供默认实现.如果特效使用storyboard,可以简单地开始/终止的storyboard.

## 序列化/反序列化

所有的特效必须支持序列化/反序列化.我们在TransitionBase类提供一些常用的持久性逻辑,同时允许子类重写它. TransitionBase实现Save方法.它还包含一个静态的Load方法.

public static ITransition Load(XElement transitionElement);

Load方法应该是静态的,因为它创建一个XElement ITransition. 此设计仍存在一些问题,一个看似更好的设计是创建一个空ITransition,然后调用实例的Load方法.实例的Load方法应该是通过ITransition接口暴露出来.但由于时间所限.我们当前维持现状.我们将在下一版本的设计时考虑这一点.  
TransitionBase还包含一个给派生类使用的LoadChild方法进行进一步的反序列化.