System.Windows.Media.**Imaging** 네임스페이스에 해당하며

추가적으로 중요한 타입으로 **ImageBrush,ImageDrawing** reside in the System.Windows.Media namespace 에 있고 **Image** resides in the System.Windows.Controls namespace에 존재함

**BitmapSource**   
- 이미지의 인코딩,디코딩에 사용되는 중요한 클래스

- 여러개의 프레임으로 된 이미지에서 개개의 프레임이 될 수 있다

- BitmapFrame과 같은 주요 이미징 클래스 들의 부모이다.  
  
**BitmapFrame**  
- 임의의 이미지 포멧에서 실재의 비트맵데이타를 저장하는데 사용됨  
- 많은 이미지 포맷은 기본적으로 하나의 BitmapFrame만을 지원하고 GIF,TIFF 포맷은 하나의 이미지에 여러 프레임을 지원함  
- Frame들은 디코더(decoder)에의해 입력 데이타로 사용되어지고 인코더(endcoder)에의해 이미지를 생성하는데 사용되어진다.

**예)**BitmapSource로부터 BitmapFrame이 생성되어지고 TIFF이미지에 추가되는 예  
**BitmapSource** **image5** = BitmapSource.**Create**(  width, height,96,96,PixelFormats.Indexed1, BitmapPalettes.WebPalette,  
    pixels, stride);

**FileStream** stream5 = new FileStream("palette.tif", FileMode.Create);  
**TiffBitmapEncoder** encoder5 = new TiffBitmapEncoder();  
encoder5.Frames.Add(**BitmapFrame.Create**(**image5**));  
encoder5.Save(stream5);

**- Image Format Decoding**

이미지 포멧을 해독해서 시스템에서 사용할 수 있게 하는것, 디코딩 되고나면 출력,변환 등의 이미지 작업을 할 수 있다.

특정 디코더가 선택되지않으면 코덱(Codec)은 자동적으로 선택된다. 코덱은 뭐지?

**예)**비트맵 디코더  
// Open a Uri and decode a BMP image  
Uri myUri = new Uri("tulipfarm.bmp", UriKind.RelativeOrAbsolute);  
**BmpBitmapDecoder** decoder2 = new BmpBitmapDecoder(myUri, **BitmapCreateOptions**.PreservePixelFormat,**BitmapCacheOption**.Default);  
BitmapSource bitmapSource2 = decoder2.Frames[0];

// Draw the Image  
**Image** myImage2 = new Image();  
myImage2.Source = bitmapSource2;  
myImage2.Stretch = Stretch.None;  
myImage2.Margin = new Thickness(20);

**- Image Format Encoding**  
특정한 이미지 포맷으로 변환하는 작업,  인코딩된 이미지는 새로운 이미지 파일로 저장되는데 이용된다.

**예)**디코딩된 이미지를 BMP파일로 디코딩하여 저장하는

FileStream stream = new FileStream("new.bmp", FileMode.Create);  
**BmpBitmapEncoder** encoder = new BmpBitmapEncoder();  
TextBlock myTextBlock = new TextBlock();  
myTextBlock.Text = "Codec Author is: " + encoder.CodecInfo.Author.ToString();  
**encoder.Frames.Add(BitmapFrame.Create(image));**  
encoder.Save(stream);  
  
**- Displaying Images in WPF**

Image Control, ImageBrush, ImageDrawing등으로 출력될 수 있다.  
예)Using the Image Control  
<!-- Simple image rendering. However, rendering an image this way may not result in the best use of application memory.   
        See markup below which  creates the same end result but using less memory. -->

//단순히 Image사용하는 것은 메모리의 최적화된 이용법이 아니다..

**<Image Width="200"  Source="C:\Water Lilies.jpg"/>**

// DecodePixelWidth or DecodePixelHeight 을 명시하면 명시한 크기대로 최적화된 메모리를 사용.  
<Image Width="200">  
  <Image.**Source**>  
    <!-- To save significant application memory, set the **DecodePixelWidth or DecodePixelHeight** of the BitmapImage value   
           of the image source to the desired  height and width of the rendered image.   
             If you don't do this, the application will cache the image as though it were rendered as its normal size rather   
             then just the size that is displayed. -->  
    <!-- Note: In order to preserve aspect ratio, only set either DecodePixelWidth or DecodePixelHeight but not both. -->  
    **<BitmapImage** **DecodePixelWidth="200"**    UriSource="C:\Water Lilies.jpg" /> //**DecodePixelWidth :** 비율이 유지되게  
  </Image.Source>  
</Image>

예) C#버젼

// Create Image Element  
**Image myImage = new Image();**myImage.Width = 200;

// Create source  
**BitmapImage myBitmapImage = new BitmapImage();**

// BitmapImage.UriSource must be in a BeginInit/EndInit block  
**myBitmapImage.BeginInit();**  
myBitmapImage.UriSource = new Uri(@"C:\Water Lilies.jpg");

**myBitmapImage.DecodePixelWidth = 200;**  
**myBitmapImage.EndInit();**  
//set image source  
myImage.Source = myBitmapImage;  
  
**- Rotating, Converting, and Cropping Images**  
BitmapImage속성 또는 추가적인 BitmapSource 객체들( such as CroppedBitmap or FormatConvertedBitmap)을 이용하여 scale, rotata,픽셀포맷변경,crop등의   이미지 변형을 기능을 제공한다.

예) BitmapImage의 Rotate속성을 이용하여 90도 회전시키기

<Image Width="150" Margin="5" Grid.Column="0" Grid.Row="1">  
  <Image.Source>  
    <TransformedBitmap Source="/sampleImages/watermelon.jpg" >  
      <TransformedBitmap.Transform>  
        <RotateTransform Angle="90"/>  
      </TransformedBitmap.Transform>  
    </TransformedBitmap>  
  </Image.Source>  
</Image>

예) C#버젼

// Create Image element.  
**Image rotated90 = new Image();  
rotated90.Width = 150;**

// Create the TransformedBitmap to use as the Image source.  
**TransformedBitmap** tb = new TransformedBitmap();

// Create the source to use as the tb source.  
**BitmapImage** bi = new BitmapImage();  
**bi.BeginInit();**  
bi.UriSource = new Uri(@"sampleImages/watermelon.jpg", UriKind.RelativeOrAbsolute);  
**bi.EndInit();**

// Properties must be set between BeginInit and EndInit calls.  
**tb.BeginInit();**  
tb.Source = bi;  
// Set image rotation.  
**RotateTransform** transform = new RotateTransform(90);  
**tb.Transform = transform;  
tb.EndInit();**  
// Set the Image source.  
**rotated90.Source = tb;**

예)**FormatConvertedBitmap**을이용하여 grayscale과 같은 pixel format으로 변환하기

<!-- Grayscale XAML Image -->  
<Image Width="200" Grid.Column="0" Grid.Row="1">  
   <Image.Source>  
      <FormatConvertedBitmap Source="/sampleImages/rocks.jpg"  **DestinationFormat="Gray4"** />  
   </Image.Source>  
</Image>

예) C#버전

//Create Image Element  
Image grayImage = new Image();  
grayImage.Width = 200;  
grayImage.Margin = new Thickness(5);

//Create source using xaml defined resource.  
**FormatConvertedBitmap** fcb = new **FormatConvertedBitmap**(  
   (BitmapImage)this.Resources["masterImage"],PixelFormats.Gray4,null,0);  
//set image source  
grayImage.Source = fcb;

예) Image또는 CroppedBitmat등의 Clip속성을 이용하여 이미지 크롭 시키기  
      **저장을 위한 인코딩된 이미지가 필요하다면 CroppedBitmap을 사용해야 한다.**

<!-- Cropping an Image using Clip -->  
<Image Width="200" Grid.Column="0" Grid.Row="5" Margin="5"  
   Source="/sampleImages/gecko.jpg">  
  <Image.Clip>  
    <EllipseGeometry Center="75,50" RadiusX="50" RadiusY="25" />  
  </Image.Clip>  
</Image>

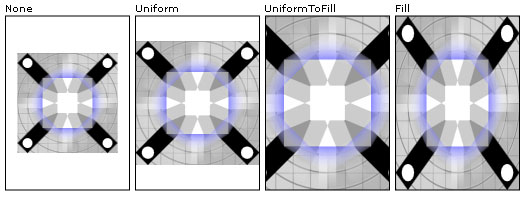
예) C#버전

//Create the image for clipping  
**Image clipImage = new Image();**  
clipImage.Width = 200;  
clipImage.Margin = new Thickness(5);

//Create & Set source  
**BitmapImage** bi = new BitmapImage();  
//BitmapImage.UriSource must be in a BeginInit/EndInit block  
bi.BeginInit();  
bi.UriSource = new Uri("pack://application:,,/sampleImages/gecko.jpg");  
bi.EndInit();  
clipImage.Source = bi;

//Clip the using an EllipseGeometry  
EllipseGeometry clipGeometry = new **EllipseGeometry**(new Point(75, 50), 50, 25);  
clipImage.Clip = clipGeometry;

**- Stretching Images ( none,Fill,Uniform,UniformToFill 등의 모드존재)**이미지를 포함하는 컨테이너에 어떻게 맞춰져서 출력되는가 결정하는 Stretch속성의 값을 지정

  
**- Painting with Images**  
ImageBrush를 이용하여 다른 UI객체등의 그리기에 참여  
예) 버튼의 배경을 이미지로 채우기

<!-- Sets the button's Background property with an ImageBrush. The resulting  
     button has an image as its background. -->  
<Button Grid.Row="3" Grid.Column="2"   
 Height="75" Width="100" Foreground="White" FontWeight="Bold"  
 HorizontalAlignment="Left">  
  A Button  
  <Button.Background>  
    <ImageBrush ImageSource="sampleImages\blueberries.jpg" />  
  </Button.Background>  
</Button>

**- Image Methdata**

어떤 이미지 파일은 파일의 특성와 내용을 설명하는 메타데이타를 소유, 예를들면 디지털카메라고 촬영한 이미지는 카메라의 제조사와 모델정보등을 가진다.   
BitmapSource의 Metadata속성으로 접근제공  
 WPF Imaging supports the following image metadata schemas: Exchangeable image file (Exif), tEXt (PNG Textual Data), image file directory (IFD), International Press Telecommunications Council (IPTC), and Extensible Metadata Platform (XMP).  
  
메타데이타에 단순한 접근을 제공하는 BitmapMetadata를 이용하면 Author,Title,CameraModel정보등을 간단히 얻을 수 있다.  
추가적인 메타데이터의 접근을 원할때는 Metadata Query Reader를 이용

예)C#

// Add the metadata of the bitmap image to the text block.  
TextBlock myTextBlock = new TextBlock();  
myTextBlock.Text = "The Description metadata of this image is: " + pngInplace.GetQuery("/Text/Description").ToString();

**예) SetQuery로 메타에이타 쓰기**  
Stream pngStream = new System.IO.FileStream("smiley.png", FileMode.Open, FileAccess.ReadWrite, FileShare.ReadWrite);  
PngBitmapDecoder pngDecoder = new PngBitmapDecoder(pngStream, BitmapCreateOptions.PreservePixelFormat, BitmapCacheOption.Default);  
BitmapFrame pngFrame = pngDecoder.Frames[0];  
InPlaceBitmapMetadataWriter pngInplace = pngFrame.CreateInPlaceBitmapMetadataWriter();  
**if (pngInplace.TrySave() == true)  
{ pngInplace.SetQuery("/Text/Description", "Have a nice day."); }**  
pngStream.Close();