Filters describe image operations that can be applied on images saved as page attachments. These operations may include actions such as crop, resize, watermark, etc. The filters then serve as building blocks when creating image variant definitions.

By default, no filters are included in the system. To add a new filter, you need to create a class in the Kentico project and implement the filter in code.

Creating a crop filter

The following example describes how to create a basic crop image filter that crops images based on the provided width. The crop is applied on the center of the image. The example uses the built-in *ImageHelper* library to perform the image operation.

1. Open your project in Visual Studio (using the **WebSite.sln** or **WebApp.sln** file).
2. Create a new class in the **App_Code** folder (or *Old_App_Code* folder if the project is installed as a web application). You can name it, for example, **CropImageFilter.cs**.

   Using the class in your own project

   If you want to use the code file in a separate library or external application, you need to allow the system to detect the class in a custom assembly. In your project, navigate to the *Properties/AssemblyInfo.cs* file and add the *Assembly Discoverable* assembly attribute:

   ```
   using CMS;

   [assembly: AssemblyDiscoverable]
   ```

3. Add *using* statements for the following namespaces:

   ```
   using System;
   using System.IO;

   using CMS.Core;
   using CMS.Helpers;
   using CMS.ResponsiveImages;
   ```

4. Set the class to implement the **IImageFilter** interface.
5. Provide a property for setting the image width and use the property in the filter class constructor.

   ```
   /// <summary>
   /// Sample filter for cropping images.
   /// </summary>
   public class CropImageFilter : IImageFilter
   {
       private int Width { get; set; }

       public CropImageFilter(int width)
       {
           Width = width;
       }

       ...
   }
   ```

6. In the class, implement the **ApplyFilter** method required by the interface. The *ApplyFilter* method defines the actions applied on the image data.

ApplyFilter return values

Return values of the following types in the *ApplyFilter* method:

- **ImageContainer** – when the filter is applied successfully, return a new *ImageContainer* object.
- **null** – if the filter was not applied, return *null* to signify that the image data was not modified. Returning a *null* value means that the filter returns unmodified input image data.
- **ImageFilterException** – if the application of the filter fails or the filter cannot be applied, throw an *ImageFilterException* exception with an appropriate error message.

```
public ImageContainer ApplyFilter(ImageContainer container)
{
    try
    {
        using (Stream imageStream = container.OpenReadStream())
        {
            ImageHelper imageHelper = new ImageHelper(BinaryData.
GetByteArrayFromStream(imageStream));

            // Calculates correct image height to maintain aspect ratio
            int[] dimensions = imageHelper.EnsureImageDimensions(Width, 0, 0);
            int croppedWidth = dimensions[0];
            int croppedHeight = dimensions[1];

                        // Crops the center of image to the specified size
            byte[] trimmedImage = imageHelper.GetTrimmedImageData(croppedWidth,
croppedHeight, ImageHelper.ImageTrimAreaEnum.MiddleCenter);

            using (MemoryStream croppedImageData = new MemoryStream(trimmedImage))
            {
                // Updates image metadata to reflect new image size, maintains MIME
type and extension
                                ImageMetadata croppedImageMetadata = new
ImageMetadata(croppedWidth, croppedHeight, container.Metadata.MimeType, container.
Metadata.Extension);

                // Returns the modified image data
                return new ImageContainer(croppedImageData , croppedImageMetadata);
            }
        }
    }
    catch (ArgumentException ex)
    {
        throw new ImageFilterException("Failed to crop the image.", ex);
    }
}
```

7. On web application projects, build the solution.

With the crop filter prepared, you can now use it to create an image variant definition.