Daminion MediaProcessor Developer Guide

0.1 Draft version

Brief Introduction

Daminion can import various media formats, generate small thumbnails and view their properties, including images, vector images, CAD and 3D files, movies, audio files and various documents (PDF, Office, etc...).

Developers can add support for new media formats by writing custom MediaProcessors.

Requirements

Most of the MediaProcessors are based on some core media format processing library that is responsible for handling of specific formats. For example Daminion VideoProcessor is based on MPlayer and MediaInfo opensource libraries.
Thus please pay enough attention and time to find the right media processing library that will be reliable, up-to-date and with live active developer community. This library should be able to read/write metadata from/into the document, render thumbnails and provides affordable speed of reading metadata and rendering thumbnails.

A few examples of the well-written libraries are:

a. ExifTool (open-source) for handling almost any media format metadata.
b. FFMpeg/MPlayer (open-source) libraries for handling video files
c. Ghost (commercial) for processing PDF documents
d. ImageEN – image processing library for Windows

Selecting a right processing library is a half of the success. There are many libraries are already bundled with their .NET wrappers.

MediaProcessor development process in brief

- Add to your project references to Daminion libraries (see below for more info)
- Create a class derived from the MediaProcessor class
- Override several MediaProcessor methods.
  A few methods are declared as abstract and must be implemented. But the most methods are virtual so override them as needed.
- Compile and build your library for x86 platform and put it into the MediaProcessors folder that is located in the Daminion's installation folder.
- You welcome to invite Daminion community to test your MediaProcessor

Simple MediaProcessor

Let's create a simple PNG processor to import PNG images. Of course, Daminion ImageProcessor can handle PNG images, but this is a good sample to create our first media processor.

For simplicity, we will use Microsoft's GDI+ library as our core image processing library.

Prepare Solution

1. Create a new class library in VisualStudio (or whatever you use for .NET development) and name it as "**PNGProcessor**"
2. Add references to the following libraries:
   - **PicaJet.Daminion.MediaProcessingService.dll**
   - **PicaJet.Daminion.Plugins.Core.dll**.
   - **PicaJet.Daminion.Common.dll**

   Don't care about the library locations because they will be installed to GAC and imported from GAC but not from the location you will specify. Set the "**Copy Local**" property to **False** for all libraries.

3. Add the following "usings"
   - `using PicaJet.Daminion.Common.MediaProcessing;`
   - `using PicaJet.Daminion.MediaProcessing;`
4. Add a new class named "**PNGProcessor**" derived from MediaProcessor class (or Rename the existing default class name and class file accordingly)
5. Fill AssemblyInfo.cs file (usually located in the Properties folder of the solution folder). Add a short Processor info (Processor name, Author, Copyright and other info…
6. Create a text file named **processor.info,** write there "**PNGProcessor.dll**" and place it to your folder with your dll files. This helps to locate your processor fast without necessity to scan all dlls in your processor folder

Override the MediaProcessor methods

The only mandatory method from MediaProcessor methods that you need to override is:

```
protected abstract IDictionary<IMediaFormat, MediaProcessingInfo> GetSupportedMediaFormats()
```

The PNG media format is already declared in the MediaFormatDictionary static class from MediaProcessing.dll so we need to just get its reference.

```
protected override IDictionary<IMediaFormat, MediaProcessingInfo> GetSupportedMediaFormats()
{
   var supportedFormats = new Dictionary<IMediaFormat, MediaProcessingInfo>();
   var info = new MediaProcessingInfo(true);

   info.AddMetadataGroup(MetadataDictionary.Group_XMP.Guid, true);
   supportedFormats.Add(MediaFormat_PNG, info);

   return supportedFormats;
}
```

Now let's generate preview thumbnail for PNG images. Override the below method:

```
protected virtual Bitmap RenderPreview(MediaInfo mediaInfo, RenderInfo renderInfo, ref bool
isOrienationAdjusted)
{
   return null;
}
```

If for any reason you unable to render files you need to invoke the `UnableToRenderMediaFileException` exception. Pass the mediaInfo.MediaFormat as MediaFormat param for exception.

In this case the default thumbnail will be used for this format.

Tell to Daminion customers who you are

Override the MediaProcessor.GetPluginInfo (don't forget to generate your own GUID)

```
public override PluginInfo GetPluginInfo()
{
       if (pluginInfo == null)
       {
            pluginInfo = new PluginInfo(
                "My PNG Processor",
                "Process PNG image fast and accurate",
                "My Company ",
                DateTime.Now.Date,
                "0.0.1.0",
                new Guid("36944260-491C-463B-A9D8-3D5E2AD718FA")
                );
        }
        return pluginInfo;
}
```

Register new media Formats

If your MediaProcessor supports new MediaFormats which are not supported by Daminion you can ask to MediaProcessingService to add them to the MediaFormatDictionary by overriding the MediaProcessor.GetNewMediaFormatSpecifications

```
protected virtual IList<MediaFormat> RegisterNewMediaFormatSpecifications()
protected virtual IList<MediaFormatGroup> RegisterNewMediaFormatSpecificationGroups()
```

Some tips:

- before creating a new MediaFormat make sure that this format wasn't already declared in the MediaFormatDictionary class.
- MediaFormat name will be displayed in the CatalogTags under the Media Format group. Avoid using long names like Monkey's Audio APE Files, Canon Camera RAW Files, etc...
- Make sure that you generated a Unique GUID for each media format
- Make sure that you specify the right MediaFormatGroup for each processor. Daminion has several predefined media format groups so check existing groups at MediaFormatDictionary.Group_* before creating a new one.
- provide the list of the extension in the following manner: ".jpg; .bmp; .gif" or just ".png"
- If your MediaProcessor can detect media format based on a file content (by reading file header, special markers, etc...) you can override the **MediaProcessor.DetectMediaFormat** method. This method should returns MediaFormat according to the file content rather than a file extension.

If your MediaProcessor can't provide such functionality don not override this method. Media format will be detected according to the file extension by MediaProcessingService.

Compose a list of media formats that can be processed by PNGProcessor

```
protected abstract IDictionary<IMediaFormat, MediaProcessingInfo> GetSupportedMediaFormats();
```

Define Metadata Specifications supported by media processor

Override the below method to tell about metadata specifications which are supported by your Processor. You can specify already Daminion's specifications (from MetadataDictionary) or your custom ones.

```
protected abstract IList<MetaTagSpecification> GetSupportedMetaTagSpecifications();
```

Register a new Tag Specifications

You can see a lot of Tags in the Daminion's Catalog tags tree. These are Catalog Tags. They are stored in the Daminion's database called catalog. Each **Catalog Tag** is mapped to a set of Metadata Tags, that is describe by MetaTagSpecification class in the Daminion.

Example of MetaTags are EXIF metadata, IPTC, XMP, IDv3 etc... For example the catalog tag **Title** is mapped to IPTC Object Name and XMP dc:title.

So when an image file will be imported to Daminion the title from IPTC ObjectName or XMP title will be copied to Title catalog tag so you can see it in the Tags tree after importing an image. And visa-versa, when you will change the Title of an image it will be recorded to IPTC:ObjectName and XMP:dc:title metadata fields after synchronization.

So the next step is to Register new CatalogTag and Metadata tag specifications:

```
protected virtual IList<MetaTagSpecification> RegisterNewMetaTagSpecifications()
protected virtual IList<MetaTagSpecificationGroup> RegisterNewMetaTagSpecificationGroups()

protected virtual IList<CatalogTagSpecificationGroup>RegisterNewCatalogTagSpecificationGroups()
protected virtual IList<CatalogTagSpecification> RegisterNewCatalogTagSpecifications()
```

Tag specification structures is limited by 2 levels (subgroups and groups) to keep things simple.

- Don't forget to specify default values for numeric metadata fields

Relationship between CatalogTags and MetaTags

Mapping is the relationship between them.

Generating media file preview

```
protected virtual Bitmap RenderPreview(MediaInfo mediaInfo, RenderInfo renderInfo)
protected virtual Bitmap GetEmbeddedThumbnail(MediaInfo mediaInfo, int thumbWidth, int
thumbHeight)
protected virtual Bitmap GetDefaultMediaFormatThumbnail(IMediaFormat mediaFormat)
```

- For some media formats (like music) you can return your own predefined thumbnail if it's not exists in the file. You can read it from resources. Don't forget to cache it, so to not read it from resources each time.
- Resize the resulted preview images using GDIPlusLibrary

Formatting Tags

Use CustomVocabulary class

Reading the metadata from media files

```
protected virtual void GetMetaTags(MediaInfo mediaInfo, IEnumerable<MetaTag> collection)
```

Fill in the collection of MetaTags passed as the argument to this method. If your MediaProcessor implemented a lot of Metadata specifications it will be useful to include some helper classes or methods to associate MetaTag collection with your core library values

Writing the metadata from media files

XMP Metadata

MediaProcessingService offers an interface to work with Adobe XMPToolKit. So you might use this built in capability to update XMP metadata in a range of media formats that supported by the XMPToolkit.

How media files will be displayed in the Daminion

```
protected abstract IList<IMediaFormatPresentation> GetMediaFormatPresentation();
```

This method defines how the media formats supported by your Processor will look in Daminion. In the Tags tree, Properties panel and Tray.