# Image Cropper Extended Documentation

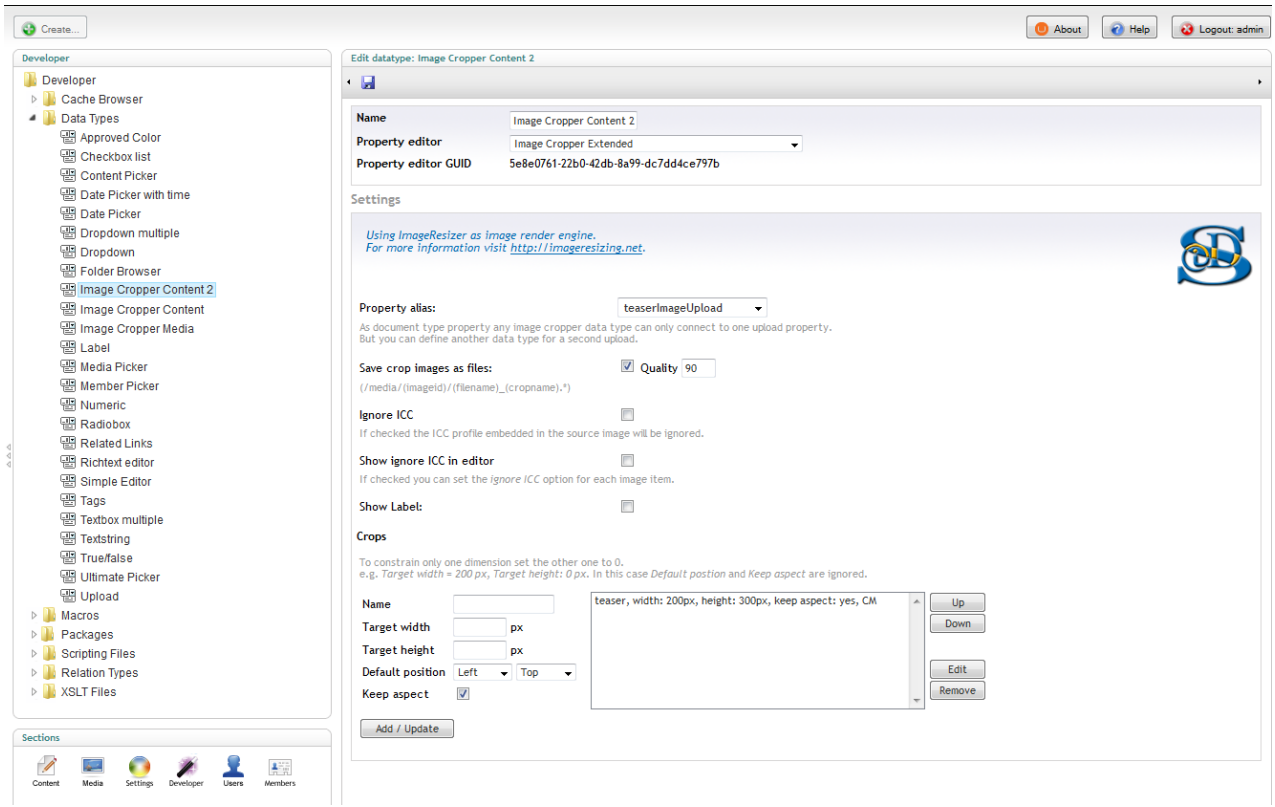
## What is the Image Cropper Extended

The Image Cropper Extended is based on the default image cropper and contains some extras and bug fixes. It is a data type that refers to an upload property and allows defining a collection of crops for a given image. The cropper can be use in three ways:

1. As a property of the image type in media types (settings section)
2. As a property of a custom document type  (settings section)
3. As a property of a custom member Type (members section)

## How to use

You can create croppers for different purposes with individual settings. For example you have a document type with multiple upload properties. Then you have to define an image cropper data type for each upload property, because any cropper can only reference a single upload property.




*Data type definition of an **Image Cropper Extended** render control*

Media images go have a default upload property called ***umbracoFile***. Thus this is the default for newly created Image Cropper Extended data types.

## The Image Cropper Settings in Detail

**Property alias:**

You connect each cropper data type to a certain upload property using its alias. Then you have to place this data type as a property on the same document, media or member type as the referenced upload field.

**Save crop images as files:**

Without this option checked the cropper just saves the rectangle data and does not generate a physically crop image file. If you checkmark this, the specified crop will be saved as a file. In this case you can also adjust the compression rate for jpeg images.

**Ignore ICC:**

If checked the ICC profile embedded in the source image will be ignored.

**Show ignore ICC in editor:**

If checked you can set the ignore ICC option for each image item.

**Show Label:**

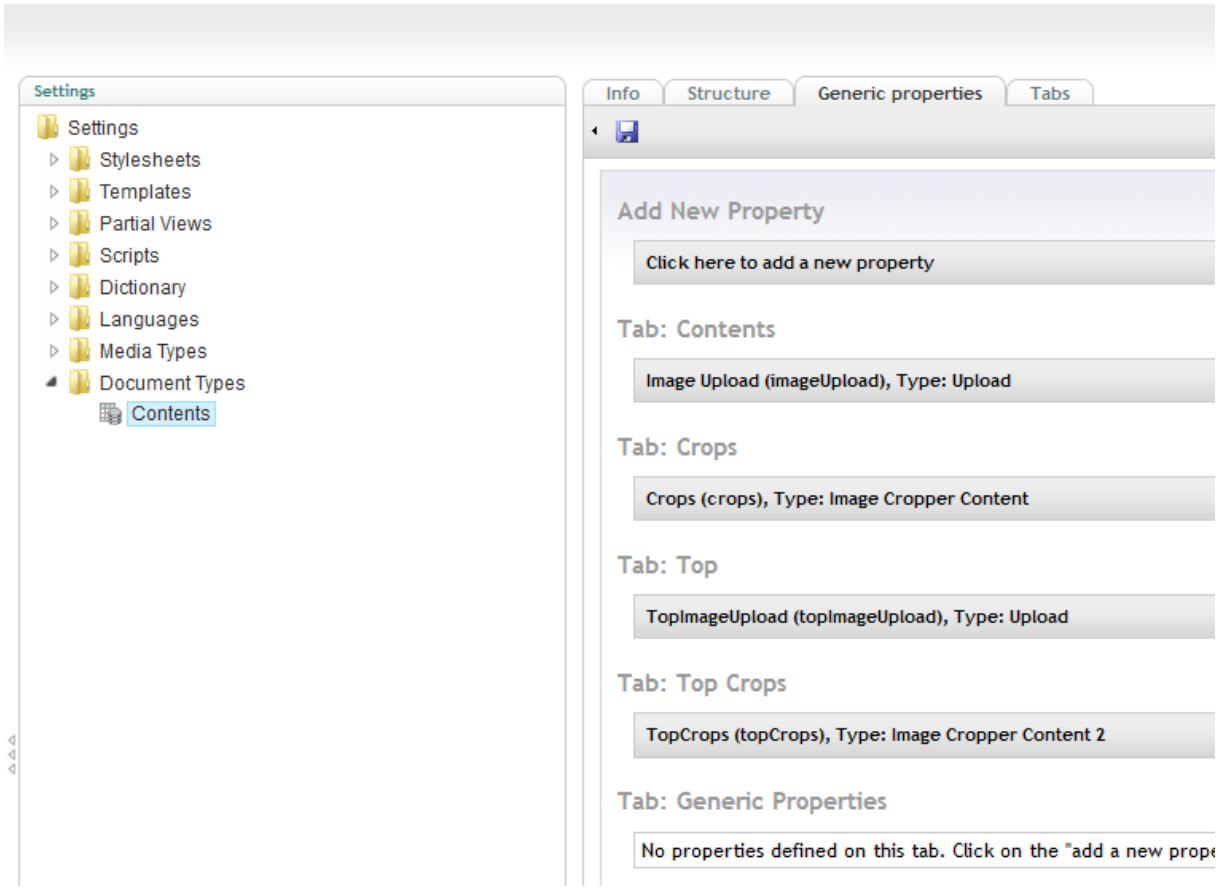
If like to display the property label to justify your backend layout, you can checkmark this. But the usable width for the cropper will be decreased.

**Crops:**

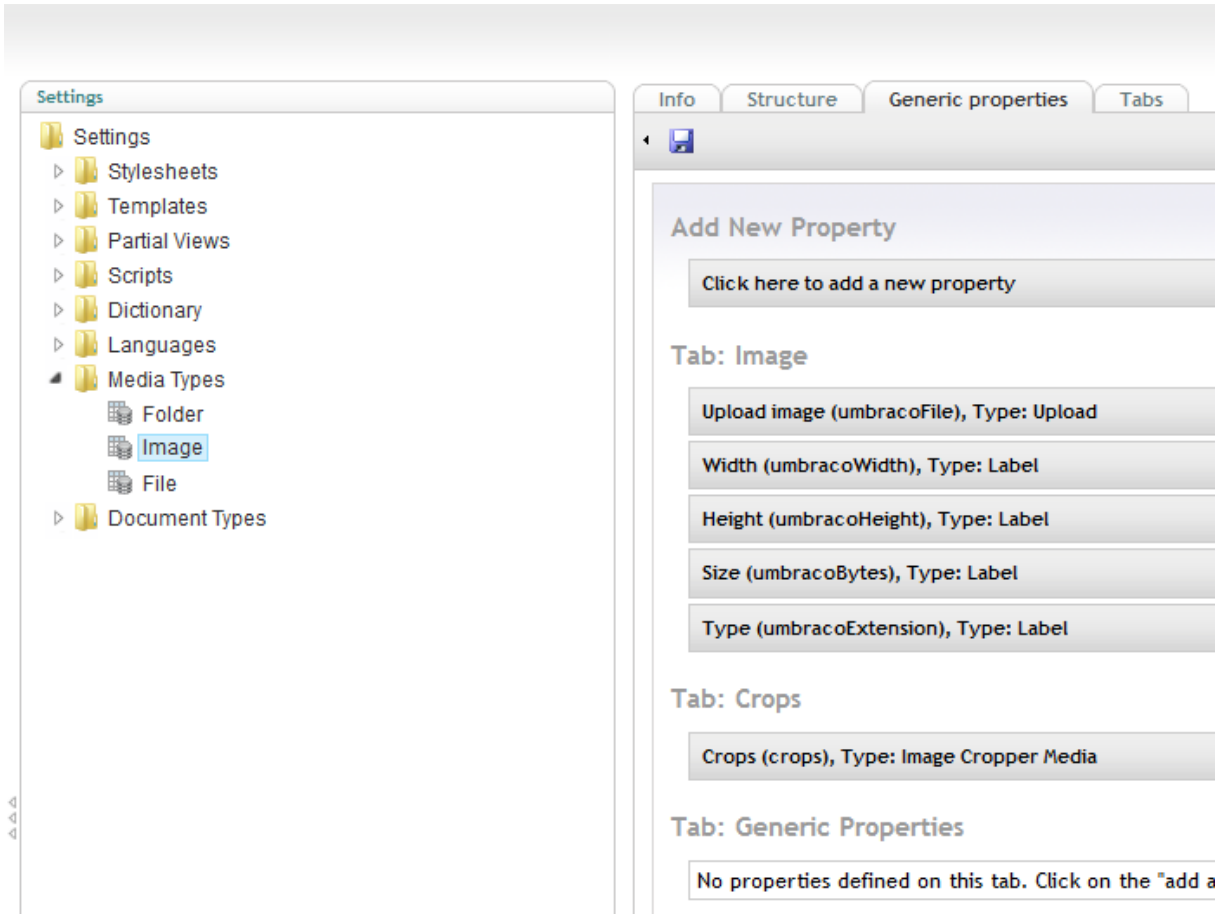
Here you define the particular crop sizes. You can define more than one for each cropper. Choose a name and define width, height, default position and aspect ratio, then add.
The Image Cropper Extended allows you to leave one target dimension blank.
This enables you for example to define crops that are fixed in width but flexible in height.

Done! Do not forget to save the newly defined or updated cropper data type.
And use it as a property.

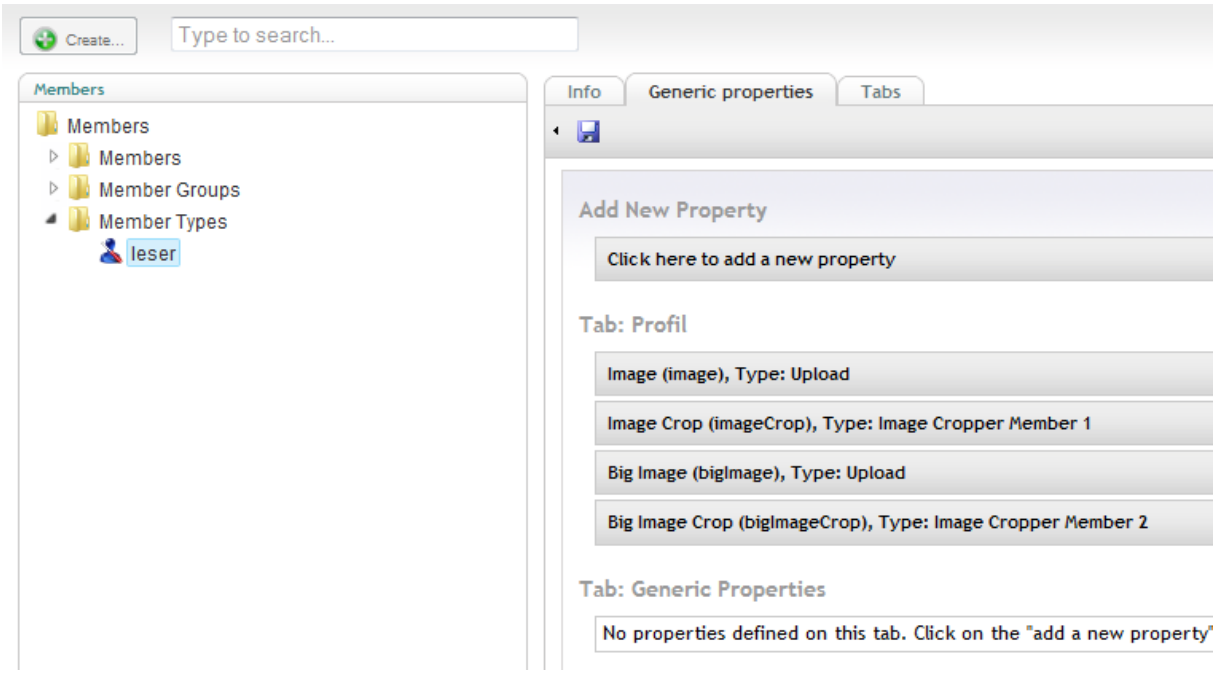
## Define a Property




*As a property of a document type*



*As a property of the image media type*




*As a property of a member type*

## Use the Cropper on a document type

The following screenshot shows an example of the image cropper in combination with an upload field property on a document type in the content section of the backend.

Beneath the image with the crop selection rectangle you can see the defined crop info areas. You toggle between the crops by clicking on these areas. They also indicate that the chosen section will lead to an up scaled crop image. In this case the area background is red. Actually the cropper does not prevent you from using up scaled crops, you have to consider on your own...

**Attention!**
The old image cropper does not support preview / publishing features well. There the new image cropper save an additional attribute ***newurl***. The original attribute ***url*** is kept for compatibility.

With XSLT use something like the following snippet:

```
...
<xsl:param name="currentPage"/>
<xsl:template match="/">
	<xsl:variable name="imageUrl"
		select="$currentPage/crops/crops/* [@name = 'teaser']/@newurl"/>	<xsl:if
test="$imageUrl != ''">
		<img style="float:left">
			<xsl:attribute name="src">
				<xsl:value-of select="$imageUrl "/>
			</xsl:attribute>
		</img>
	</xsl:if>
</xsl:template>
...
```

***Crops*** *in this example is the alias name of the document property.*

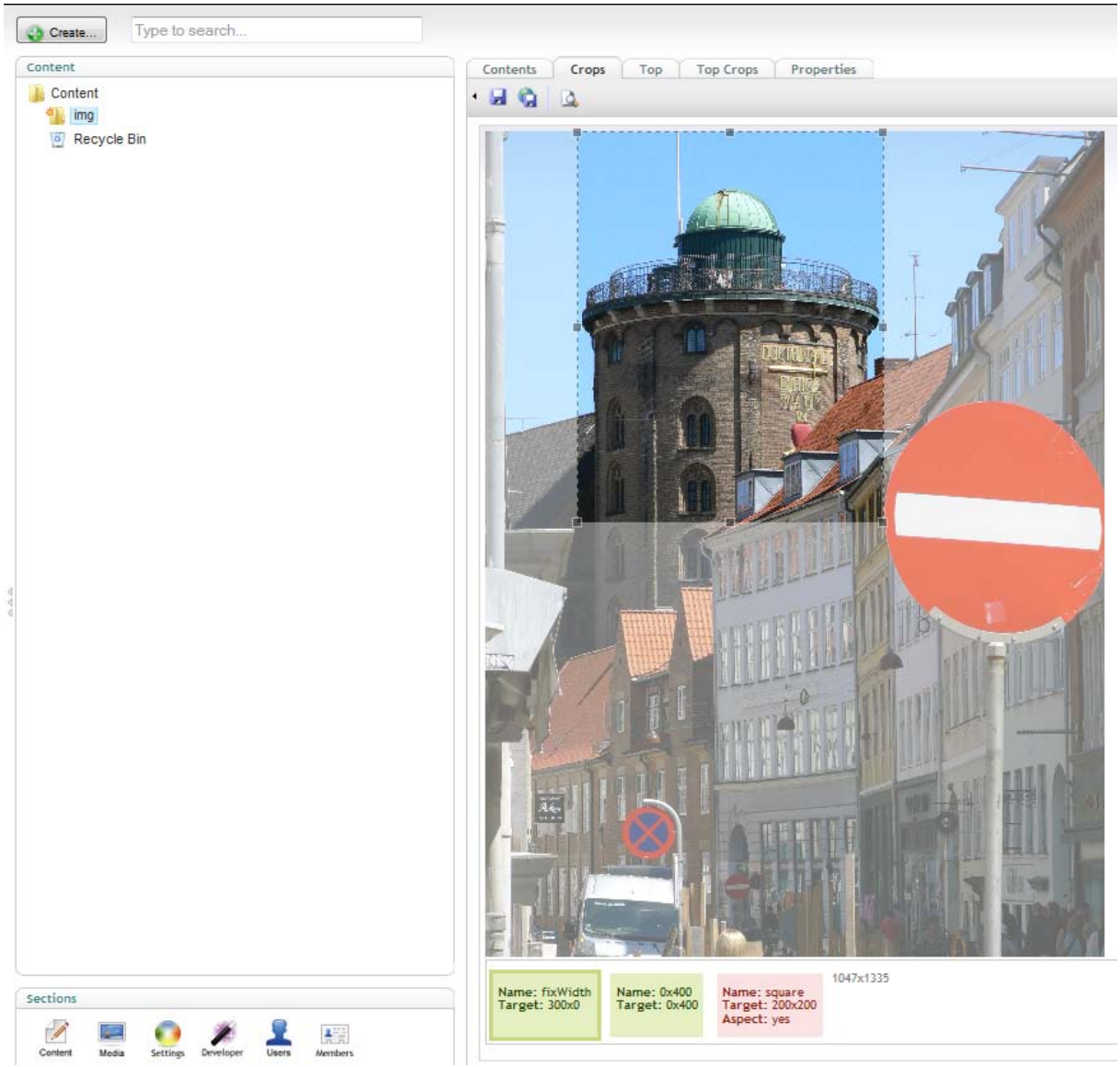



*Image cropper within the content section*

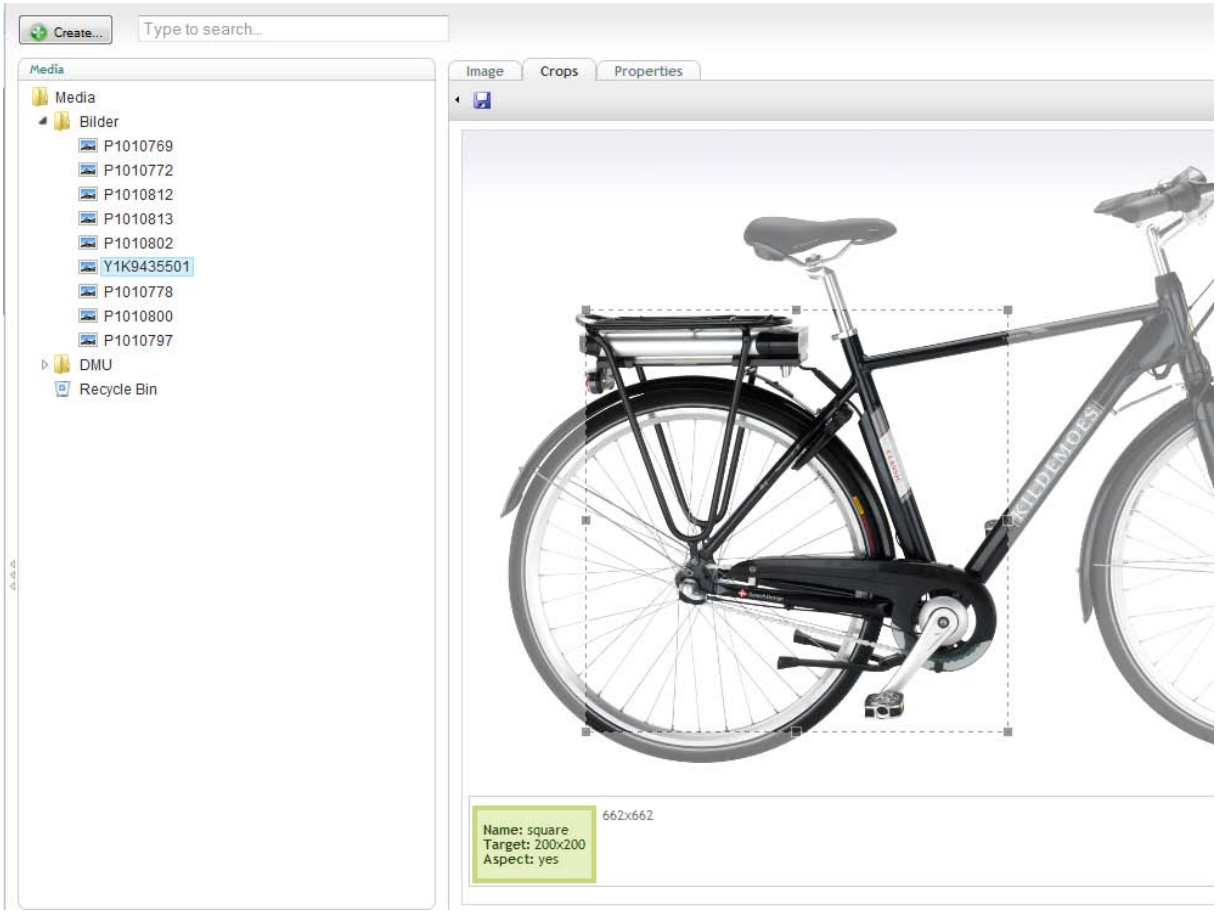
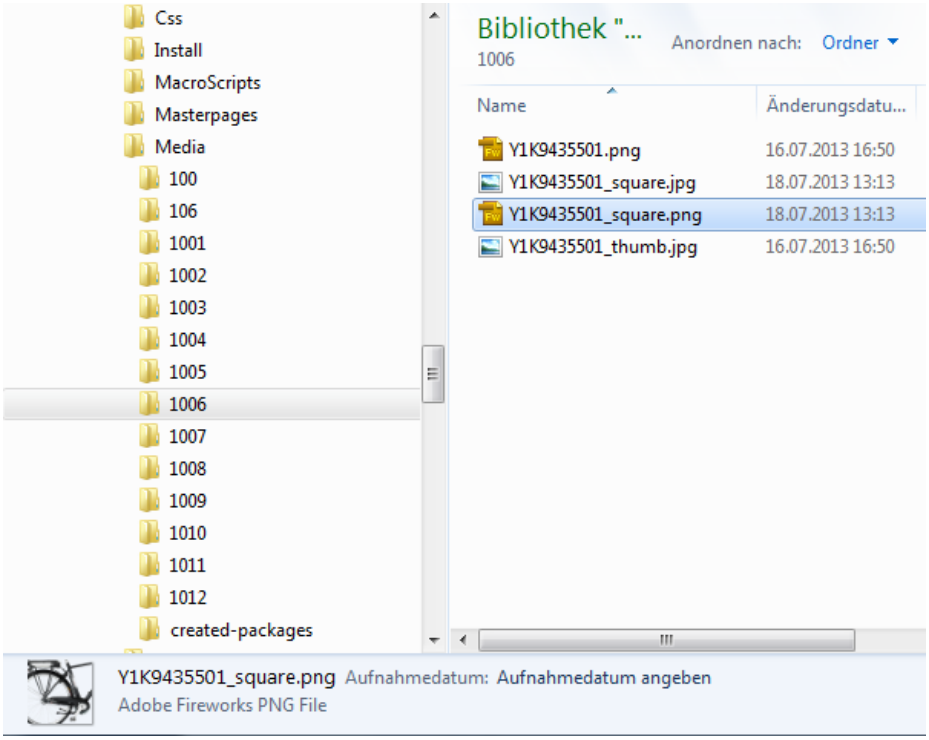



*Image cropper in the media section*

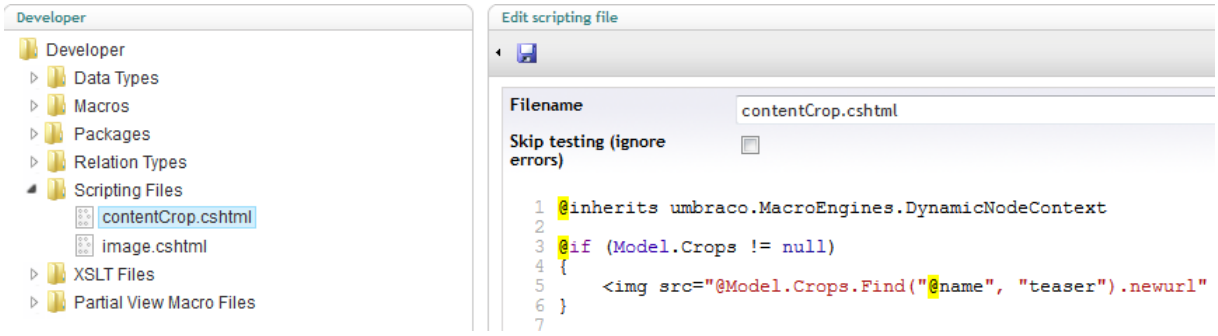



*The image files as they are save into the media folder (explorer view Windows7, outside Umbraco)*

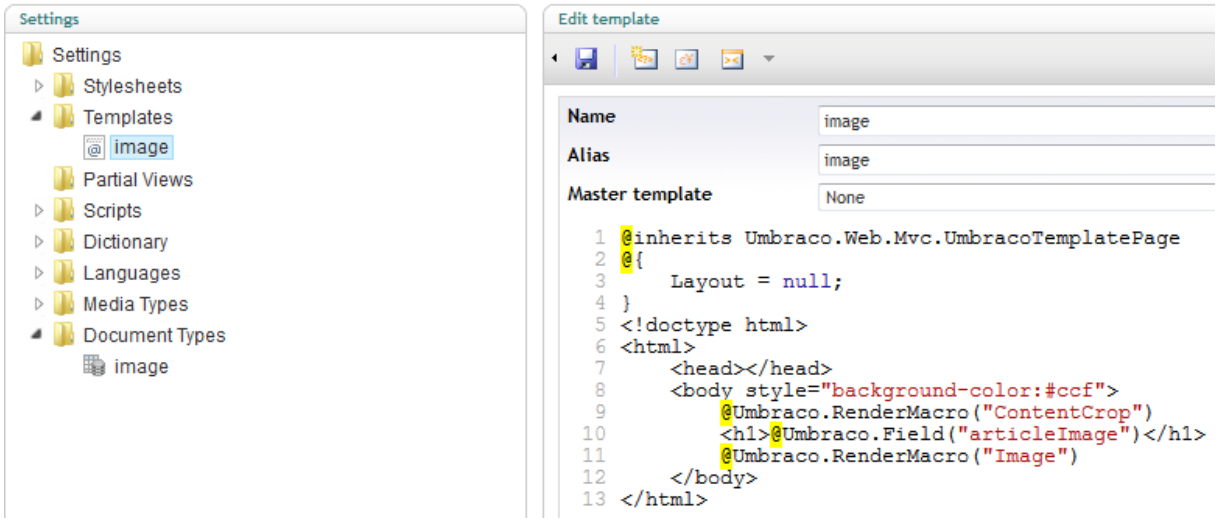
## Razor Snippets

Define an "Article Image" property of type Media Picker on a document type. For the Image media type define an image cropper data type with a crop named "article". Then you can display the "article" crop with this Razor script snippet:


The second snippet is for an image cropper attached to upload property on a document type. This kind of croppers supports a nice feature for crop versioning. Thus you will have an extra version of a changed crop for preview mode. Just use the newly defined **newurl** attribute. Define another image cropper data type associated to the document types upload property with a "teaser" named crop. Then you can display the "teaser" crop with such a Razor script snippet:




And here you see an example for a MVC template with both macros in place:



## MVC

An PropertyEditorValueConverter (PEVC) is defined for MVC support. Here is an example for a partial view displaying a crop:

```
@using idseefeld.de.imagecropper.PropertyEditorValueConverter;
@inherits Umbraco.Web.Mvc.UmbracoTemplatePage
@{
	var typesImageCropperExtended = Model.Content.GetPropertyValue<CropList>("topCrops");
}
@if (typesImageCropperExtended.Any())
{
	foreach(var crop in typesImageCropperExtended){
		<h4>@crop.Name</h4>
		<img src="@crop.NewUrl" alt="@crop.Name" width="@crop.Width" height="@crop.Height" /> <br />
	}
}

@{
	// select a single crop by name
	var teaserCrop = typesImageCropperExtended.Find(n => n.Name.Equals("teaser"));
}
<h3>selected crop</h3>
<h4>teaser</h4>
<img src="@teaserCrop.NewUrl" alt="@teaserCrop.Name" width="@teaserCrop.Width" height="@teaserCrop.Height" />
```

Until umbraco version 6.2.0 you may use the following code for media type images select by a media picker property:

```
@using idseefeld.de.imagecropper.PropertyEditorValueConverter;
@inherits Umbraco.Web.Mvc.UmbracoTemplatePage
@{
	idseefeld.de.imagecropper.Model.CropModel crop = null;
	idseefeld.de.imagecropper.Model.ImageCropperModel crops = null;
	CropList cropList = null;
	var mediaItem = Umbraco.TypedMedia(Model.Content.GetPropertyValue<int>("myMediaPicker"));
	string cropValue = String.Empty;
	if(mediaItem!=null){
		//Image Cropper Extended property with alias name "crops" assigned to Media Type Image
		cropList  = mediaItem.GetPropertyValue<CropList>("crops");
		cropValue = mediaItem.GetPropertyValue<string>("crops");
		if(cropList==null && !String.IsNullOrEmpty(cropValue)){
			crops = new idseefeld.de.imagecropper.Model.ImageCropperModel(cropValue);
		}
	}
	if(crops!=null){
		//Image Cropper Extended data type contains a crop definition with name "square"
		crop = crops.Find("square");
	}
	if(cropList!=null){
		crop = cropList.Find("square");
	}
}
<h3>Named Crop of Media Picker Image</h3>
@if (crop != null){
	<img src="@crop.NewUrl" width="@crop.Width" height="@crop.Height" alt="@crop.Name" />
}
```

Umbraco version 6.2.0 will fix an issue and you can shorten the above example to:

```
@using idseefeld.de.imagecropper.PropertyEditorValueConverter;
@inherits Umbraco.Web.Mvc.UmbracoTemplatePage
@{
	idseefeld.de.imagecropper.Model.CropModel crop = null;
	CropList cropList = null;
	var mediaItem = Umbraco.TypedMedia(Model.Content.GetPropertyValue<int>("myMediaPicker"));
	if(mediaItem!=null){
		//Image Cropper Extended property with alias name "crops" assigned to Media Type Image
		cropList  = mediaItem.GetPropertyValue<CropList>("crops");
	}
	if(cropList!=null){
		crop = cropList.Find("square");
	}
}
<h3>Named Crop of Media Picker Image</h3>
@if (crop != null)
{
	<img src="@crop.NewUrl" width="@crop.Width" height="@crop.Height" alt="@crop.Name" />
}
```

## New ImageProvider "imageCropperProvider"

For the new Umbraco server control for images is an ImageProvider implemented. This is mostly the same as the build in provider, but supports PNG images as the Image Cropper Extended save the correct extension. On your master pages you can simply use:

```
<umbraco:Image runat="server"
	field="theImage"
	Parameters="crop=square"
	Provider="imageCropperExtended" />
```

theImage is the alias of a property of type media picker.

Read more on Morten Bock Sørensens blog: Introducing The umbraco:image Control

## ImageResizer settings

The image render engine is based on the free ImageResizer.dll. For more information see: http://imageresizing.net/

The ImageResizer engine limits by default the maximum size of the crops to 3200x3200 pixels. In most cases this should be fine. But if you need bigger crops, add the following line to the web.config:

```
<configuration>
	<configSections>
		...

		<section name="resizer" type="ImageResizer.ResizerSection,ImageResizer"
requirePermission="false" />
	</configSections>
	<resizer>
		<sizelimits imageWidth="0" imageHeight="0" totalWidth="3200"
totalHeight="3200" totalBehavior="throwexception" />
	</resizer>
	...
</configuration>
```

Just increase totalWidth and totalHeight. Further details can be found on: http://imageresizing.net/docs/configuration-all

## Upgrade Umbracos Default Image Cropper

Define a crop for an *Image Cropper Extended* data type with the same name and settings like the default cropper that you want to upgrade. Then change type of the properties using the default cropper to the new *Image Cropper Extended* data type. That's all.

In the new *Image Cropper Extended* data type you can also define new crops. The order of the defined crops does not matter.

## Project Links

Source code is available on GitHub: https://github.com/idseefeld/imagecropper4umbraco

For bug reports use: https://github.com/idseefeld/imagecropper4umbraco/issues/

Umbraco package: http://our.umbraco.org/projects/backoffice-extensions/image-cropper-extended