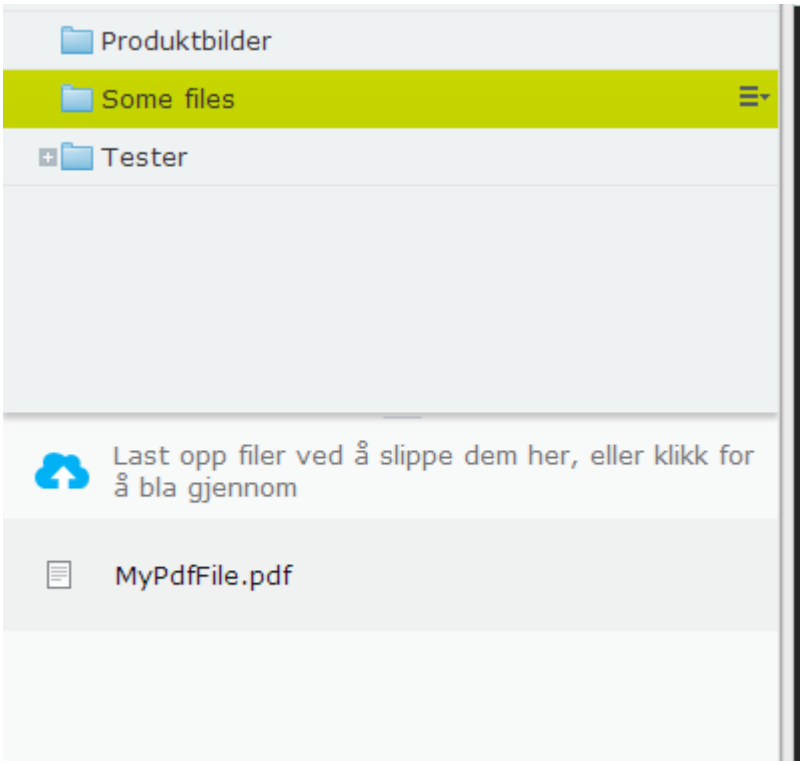
epinova.no

Custom icons for content types

by: Arve Systad 05 May 2014

Usually when you're uploading files to the new media system, you will get thumbnails for images and video, and just a default icon for other file types. This can be somewhat confusing for editors that has a lot of different files (Word-files, Excel spreadsheets, PDFs or the like). It will typically look like this:



In CMS7 and onwards, one can use a `UIDescriptor` to describe the edit mode presentation of a content type. For example, one can set the `IconClass` property, which is what we're going to do here. However, I didn't want to make one implementation of the logic for each and every media type I have. I'd rather rely on a super simple convention. My Media classes will be named `<Extension>File`,

for instance `PdfFile`. This makes it super simple to add more icons when needed.

So first, my generic UIDescriptor:

```
public class FileIconDescriptor<T> : UIDescriptor<T> where T :
ContentData
{
public FileIconDescriptor()
{
Type type = GetType();
string fileTypeName = type.BaseType.GetGenericArguments()
[0].Name;
IconClass = fileTypeName.Replace("File", "").ToLower() + "Icon";
}
}
```

Well, that was boring. Nothing really happened. But at least everything's ready to use. You now have to add an actual descriptor for each of your types - and I will be using our beloved PDF files as an example. So I added this (I keep it in the same file as the generic type to keep reduce the amount of near empty files):

```
[UIDescriptorRegistration]
public class PdfFileDescriptor : FileIconDescriptor<PdfFile> { }
```

Along with this, of course, is a super simple media class called `PdfFile` like shown below. If you now inspect the markup in the media browser, you might see that your files has gotten a `span` with a class `pdfIcon` on it.

```
[ContentType(GUID = "34D69D83-83DF-
```

```
4739-9132-8805BBC82196")]
[MediaDescriptor(ExtensionString = "pdf")]
public class PdfFile : MediaData
{
}
```

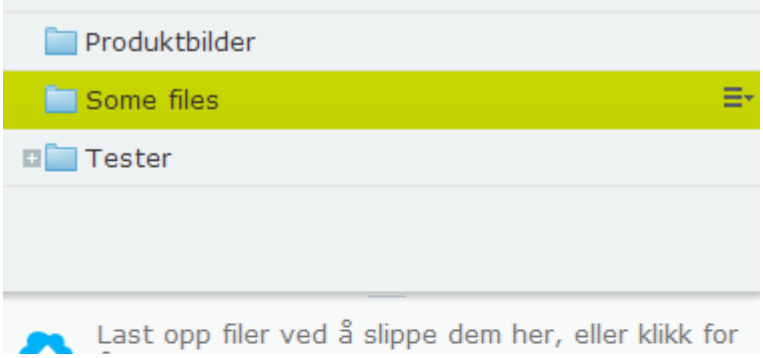
Now, to tie it all togheter, you need to add some custom styles to make an actual icon show up. To do this, add a file `module.config` to the root of your web application (or modify your existing one), and fill it with the following:

```
<?xml version="1.0" encoding="utf-8" ?>
<module>
<clientResources>
<add name="epi-cms.widgets.base" path="/Styles
/EpiserverUIExtensions.css" resourceType="Style"
isMinified="false" />
</clientResources>
</module>
```

...and then add a stylesheet on the correct path in which you can add some styling. Like this:

```
.pdfIcon { background-image: url(/UI/Icons/pdfIcon.png);width:
32px;height: 32px;margin: 5px!important; }
```

And now, it should look like something like this:



Neat. That looks quite a bit better. And hopefully your editors will enjoy it too.

So, for those of you who did not pay attention, copypaste the code above, and follow these conventions:

- Add CSS-classes called `<extension>Icon` that applies an actual icon. For example `docxIcon` and `pptxIcon`.
- Add `UIDescriptor` classes for the media types you want to give icons, just like my `PdfFileDescriptor` above. Examples are `DocxFile` and `PptxFile`.
- As my `UIDescriptor` needs `T where T : ContentData`, you should be able to add custom icons to your page tree with this too.

Note: This is partially based on stuff from the following articles. My aim was simply to have the entire step-by-step-guide for such a practical thing in one place:

- Customizing the look and behavior in the UI for your content types
- Adding a new device to the view resolution drop-down

- Changes to the module.config for EPiServer 7.1
