## Replace image with your own template

Create twig filter: integrated\_content

For example {{ article.content|integrated\_content }}

That filter should then be used for every view in which you want to show customized images, or in future videos or other stuff.

If you add html\_safe to this filter then there is no need to add raw filter

This twig filter should dispatch an event: ContentEvent, with 2 properties: content and imageTemplate (videoTemplate and others could be added in future).

You can then register listener(s) that can manipulate the content, for example: ContentImageListener. The listener can read the content and if an image is found replace it with the content from the template.

## Templates

Templates should be placed in the Resources/views/themes/[themename]/objects/image

The default template should be Resources/views/themes/default/objects/image/default.html.twig

You can override the template in your own theme and should locate it with the thememanager (locateTemplate method).

## Custom templates

In future it would be nice to set an custom template.

The best way to do this would be to use a custom twig tag (just like {% form\_theme %}

For example {% integrated\_image\_template ‘my:custom:image:template.html.twig’ %}

This can be done using a token parser (see AssetTokenParser.php or <http://stackoverflow.com/questions/26170727/how-to-create-a-twig-custom-tag-that-executes-a-callback>).

## Finding images

ContentImageListener {

Public function replaceImage(ContentEvent $event) {

$pattern = '/\<img.\*?data\-integrated\-id\="(.+?)".\*?\>/';

$that = $this;

$template = $event->getImageTemplate();

$content = preg\_replace\_callback(

 $pattern,

 function ($matches) use ($that, template) {

 $storageId = $matches[1];

 return $this->renderImage($storageId, $template);

 },

 $event->getContent()

 );

 $event->setContent($content);

 }

//this should be in an abstract class so that it can be used for multiple listeners

Protected function render($storageId, $template) {

 Return $this->templating->render($template, [‘storageId’ => $storageId])

}

}

## Summary

* New twig filter: integrated\_content
* Dispatch contentEvent for manipulating content
	+ Set content
	+ Set template using thememanager locatetemplate method
* Add ImageListener
	+ Match image
	+ Replace image with content from template
* Create default template