Easylib – The library for Stip Easy Image

Manual

Introduction

The library package which gets installed in the joomla folder ‘libraries’ consist of the following elements:

1. a form fieldtype called ‘upload’, containing the form element for Stip Easy Image
2. two classes for programmatically interacting with Stip Easy Image:
    - StipEasyImage: handles resizing and saving of uploaded files
    - StipEasyimageFormHelper

Apart from that the library makes use of the vendor package ‘Intervention/Image’ for manipulating images. In the media folder you find the assets of the library.

1. Fieldtype ‘upload’

To implement a Stip EasyImage in a form you can add the fieldtype to your xml form:

type="upload"

addfieldpath = "libraries/easylib/forms/fields" – required to link to the form field

name – required, name of the input field for storing the path in the database

ajaxurl – required, url for ajax storing of a new image. For a module or plugin it would look something like:
"index.php?option=com_ajax&amp;module =[module]&amp;format=json"
So handling of the ajax request should be done in your own module, component, plugin. The library however has a helper function which makes ajax handling very easy.

label – required, label of the form element

description – optional description

show_select – [0/1] whether or not to show the select button for selecting an image from the images/ folder. Defaults to zero.

allow_upscale – [0/1] whether or not to allow too small images to be scaled up to the selected size. Defaults to zero.

accept – Which image types to allow. Formatted for use in the accept param of an input of type ‘file’. Defaults to "image/*"

copy_to – optional pointer to the id of another input field. When set the file-path will also be set to that field on changing / uploading the Easy Image. This could come in handy if for instance you want to use the Easy Image as intro image as well as for the full article.

The field requires at least on child element of type ‘size’.
So:

```
<field type=”upload”>
	<size/>
	<size/>
	...
</field>
```

The size element can have the following parameters:

width – (int) preferred width of the image in pixels. Optional, defaults to 0.

height - (int) preferred height of the image in pixels. Optional, defaults to 0.

folder – required, path to the folder in ‘images’

name – name of this size-setting for display to the user

cancrop – [0/1] optional. Whether or not a user can crop his/her uploaded image. By default it is allowed when a width and height is set.

autopopup – [0/1] optional. Setting this to 1 the crop window will immediately pop up after uploading an image when cropping is allowed.

2a. Class StipEasyImage

With this class you can save any image file to the given size properties.

```
$easyImg = new StipEasyImage($file, “feedback name” [, ‘jpg’]);
$feedback = $easyImg->saveImage(width, height, name, path, quality);
// name = name to save the file to
// path = path to the folder to store the file in
//quality = quality of the image, the higher the bigger the filesize

if($feedback[‘succes’]){
	//this is the file
	echo $feedback[‘filename’];
} else {
	//this is the error
	echo $feedback[‘error’];
}
```

You can use any source type for $file that gets accepted by Intervention/Image:
http://image.intervention.io/api/make

Most of the times you won’t need to call this class yourself. Just use the helper method for it in the StipEasyimageFormHelper.

2b. Class StipEasyimageFormHelper

This class actually does two things:

I. manipulate an existing JForm
II. handle the ajax request from the EasyImage

ad I. Manipulating an existing JForm

This is useful within a plugin to change existing media fields in a given JForm to EasyImage fields.

An extract from the EasyImagePlugin:

```
public function onContentPrepareForm($form, $data){

      if (!($form instanceof JForm))
      {
          $this->_subject->setError('JERROR_NOT_A_FORM');
          return false;
      }

      //filter the right form
      $name = $form->getName();
      if (!in_array($name, array('com_content.article')))
      {
          return true;
      }

      //array of fieldnames to hide in the form
      $hide_fields = ['image_intro', 'spacer1', 'float_intro'];

      //array of fields to change in the form
      //fieldname => [array of new assets in the field]
      //view fieldtype ‘upload’ above for all possible features
      //you do not need to set the fieldtype of path here
      $change_fields = [
            'image_fulltext' => [
                  'label' =>
                        JText::_("PLG_CONTENT_STIPEASYIMAGE_IMAGE_LABEL"),
                  'sizes' => $sizes, //array of sizes – see chapter 1
                  'show_select' => $params->get('show_select'),
                  'allow_upscale' => (int) $params->get('allow_upscale'),
                  'copy_to' => 'jform_images_image_intro'
                ]
            ];

      $ajax_url = "index.php?option=com_ajax&plugin=stipeasyimage
                                                &group=content&format=json";

      $easyForm = new StipEasyimageFormHelper();
      $easyForm->setFormFields($form, $data, $with_select, $ajax_url,
                                          $change_fields, $hide_fields);
}
```

As you can see in $change_fields ‘sizes’ are now set using an array.
All params of <size> in the xml of chapter 1 apply here as well:

```
$sizes[] = [
      'name' => $format->naam,
      'param_key' => $key,
      'width' => $width,
      'height' => $height,
      'cancrop' => $cancrop,
      'folder' => $folder,
      'autopopup' => $autopopup
];
```

That’s all there is to it!

Ad II. handle the ajax request from the EasyImage

Ajax requests from the Stip EasyImages in your form can be handled like so:

```
function onAjaxStipeasyimage()
{
      return StipEasyimageFormHelper::processAjax((int) $this->params
                                                  ->get('quality'));
}
```

So all you have to do is call the static function processAjax() with an optional quality setting and return it and you’re done.