YAG - Yet Another Gallery

Extension Key: yag

Language: en

Keywords: gallery, forEditors, forAdmins, forDevelopers

Copyright 2000-2010, Daniel Lienert, Michael Knoll, <info@yag-gallery.de>

For further information see http://www.yag-gallery.de

Table of Contents

Some Credits

Who made this?

This extension was developed by Daniel Lienert and Michael Knoll with a lot of help from many different people. We spent a lot of our free time, implementing all the stuff. As we could use quite some OpenSource software like TYPO3, Extbase Extension Framework, FLUID, JQuery and pt_extlist, we would like to give something back to the community which we do by uploading this extension to TER.

Special thanks goes to the Extbase team – Jochen and Sebastian and now Bastian – you did and do a great job on this one! We both work at punkt.de in Karlsruhe and could use quite some software that was developed there and now supports YAG. So another big thank you goes to Jürgen Egeling for letting us use punkt.de's extensions to make things a lot easier.

We hope that you enjoy our extension and would appreciate your feedback. We need you to find all the bugs in there! Please let us know, if you have any questions or problems: info@yag-gallery.de

How can you help?

There are several ways to contribute: First of all, you can install the extension and give us your feedback. If you like it and use it on your website, it would be great, if you would let us know the link for our reference page on www.yag-gallery.de. If you would like to help us with development, we can set up a git account for you and you can easily join the team. If YAG gallery helps you on a commercial website or you want to donate – feel free to contact us on info@yag-gallery.de

If you need any features to be implemented – yes we also work for money :-)

Introduction

What does it do?

YAG is a gallery extension for Typo3. It offers many great features like gallery, album and image management as well as a full-featured backend-module for administration. See the screenshots below to get an impression on how it looks like.

Screenshots

Frontend

You can set up a list of galleries and use the as folders for your albums. The galleries can be sorted and paged as a list:

Within each gallery you can have as many albums as you like. These albums can be displayed as a list in the frontend:
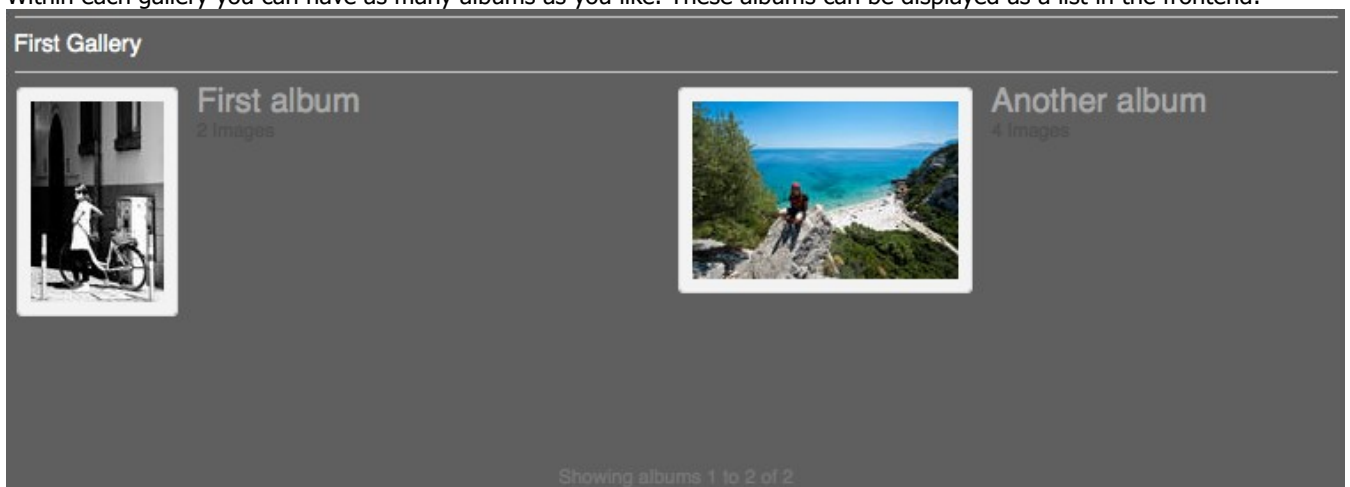
Each album can be displayed as a list of images contained by this album:
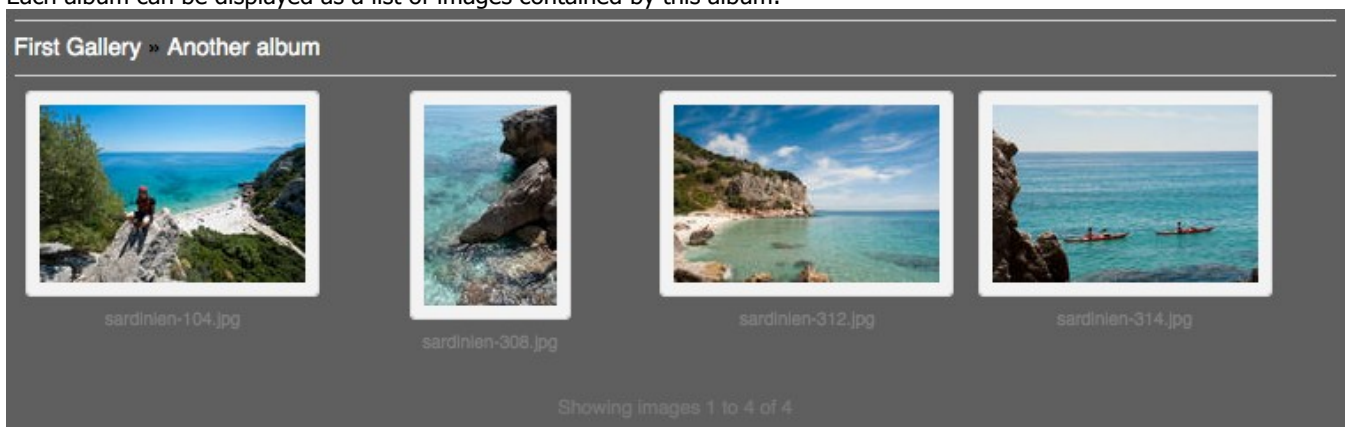
An image can be presented in many different ways, using different templates and themes. Here is the standard template – showing the image with some useful information:

First Gallery » Another album » sardinien-314.jpg

« previous   4 of 4   next »

Title:	sardinien-314.jpg		Camera Model:	NIKON CORPORATION - NiKON D700
Description:			Lens:	24.0-70.0 mm f/2.8
Artist:			Focal Length:	
Keywords:	Sardinien 2010		Iso:	200
			Shutter Speed:	9965784/1000000
			Aperture:	
			Flash:	0

If you like, you can also use a lightbox like here:
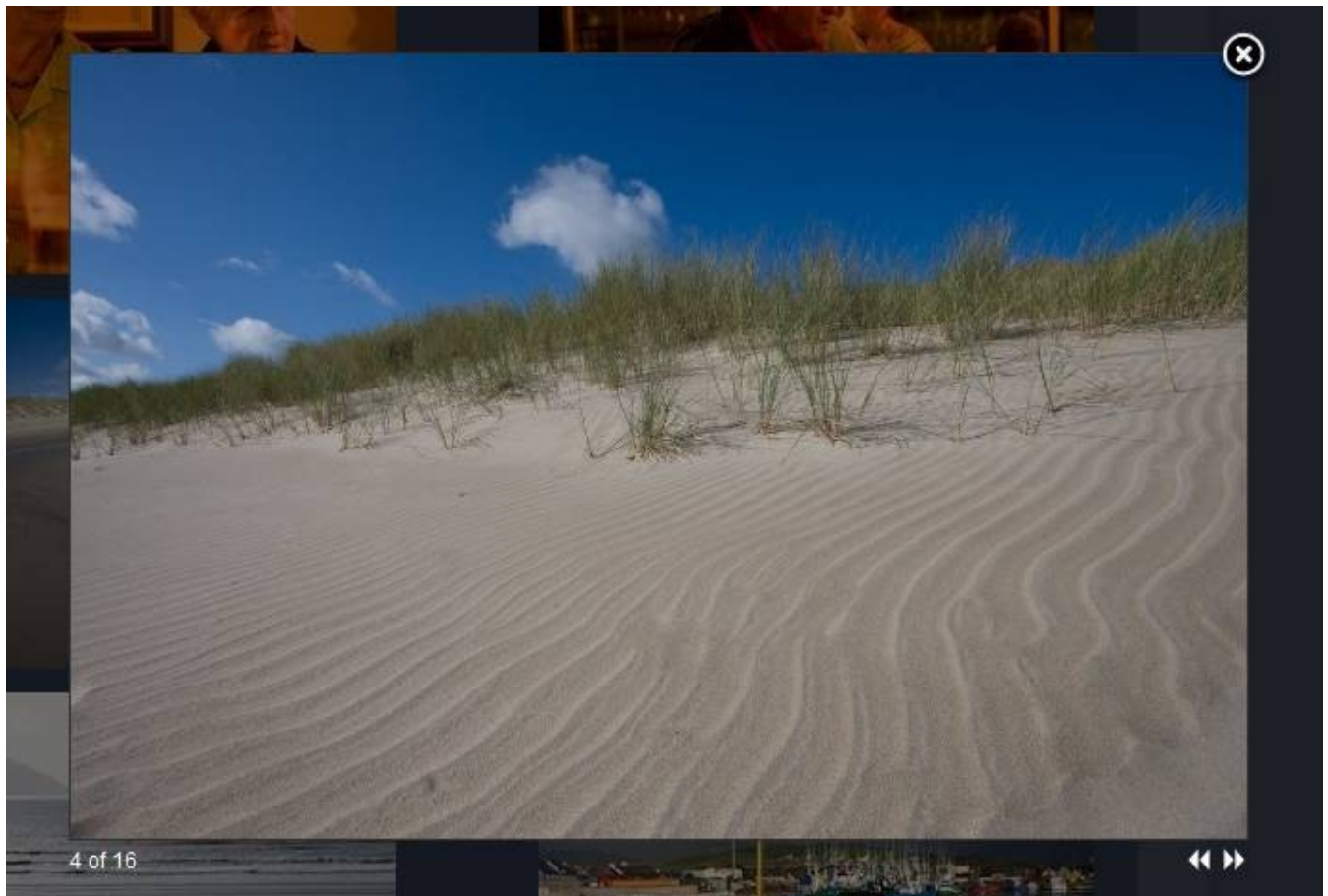
You can use many different templates and plugins (like JavaScript Lightboxes or Flash albums) to actually render your albums on your site. It is one goal of YAG to enable you write your own templates easily. Here is another example of a rendered album using SimpleViewer:

Backend

YAG ships with a full-featured backend module that lets you administrate your gallery easily. After installing YAG, you get a new icon in the menu bar:
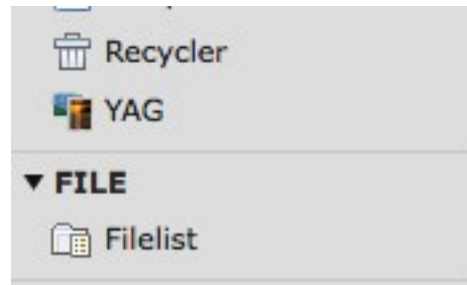
This will show an administration backend:
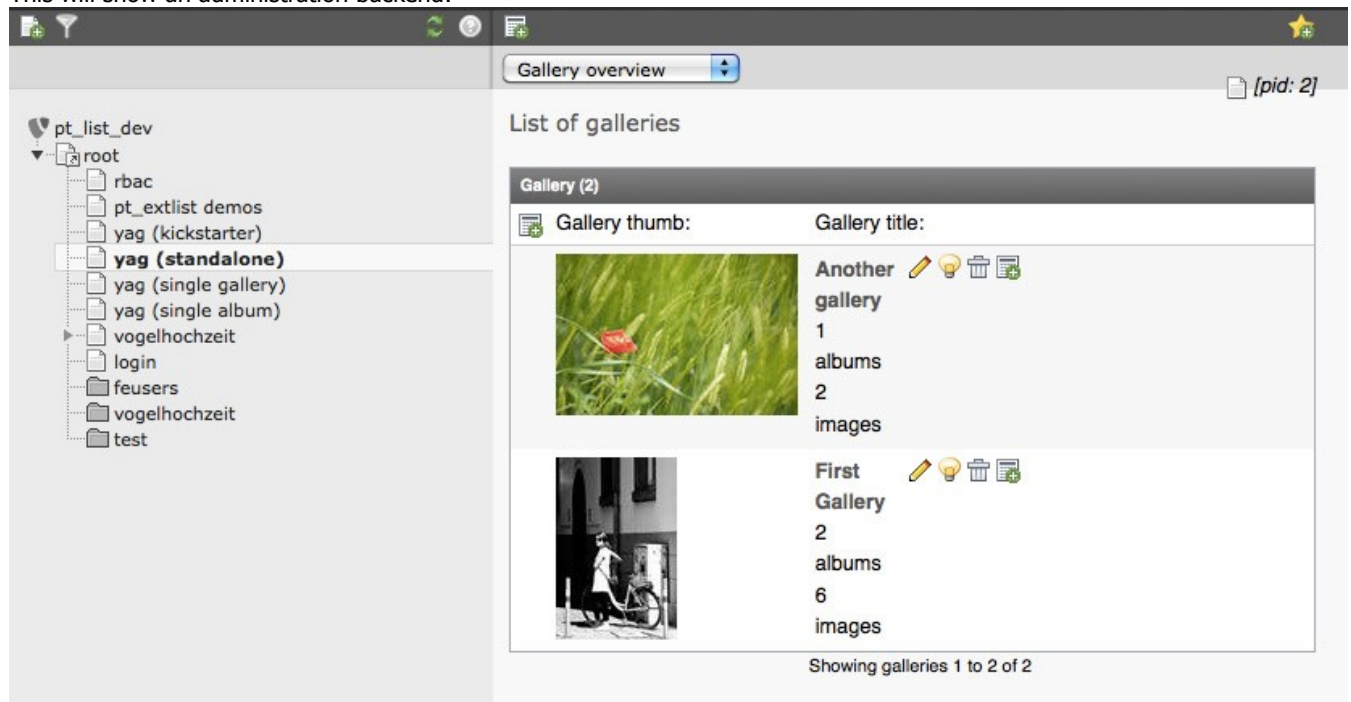
You can get a list of galleries, and for each gallery a list of albums inside this gallery:
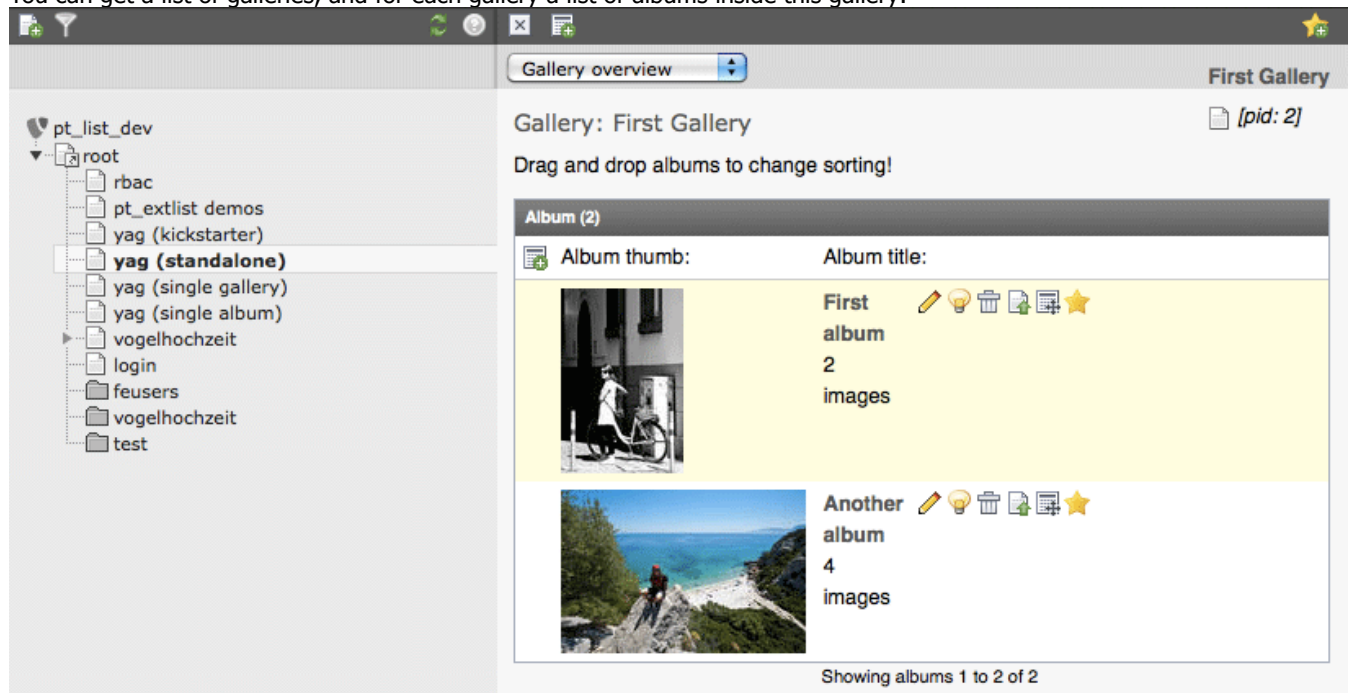
There is a administration view that lets you easily manipulate your images in your albums. You can drag and drop images for

sorting, change titles and descriptions and set thumbnails for albums:
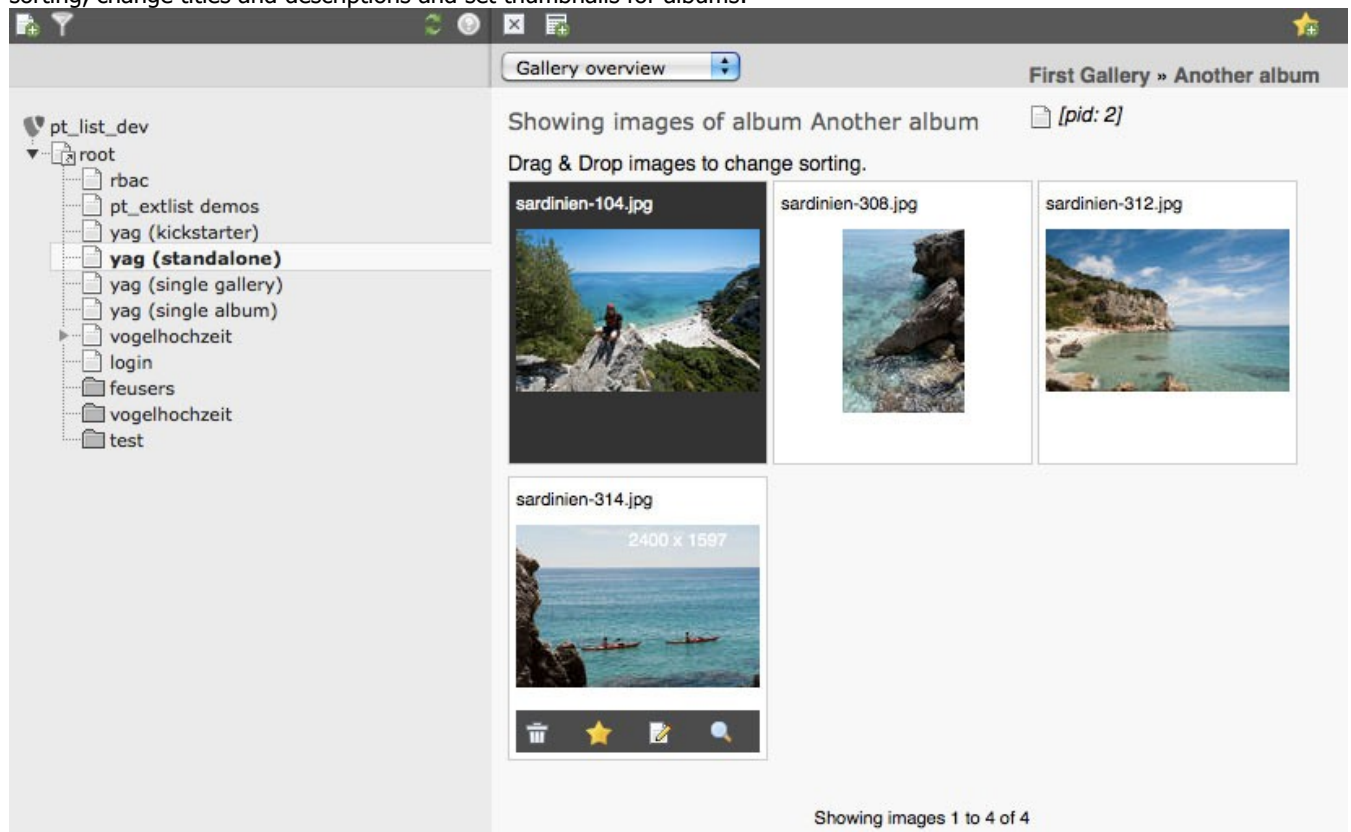
There is also a preview in backend, using a lightbox:
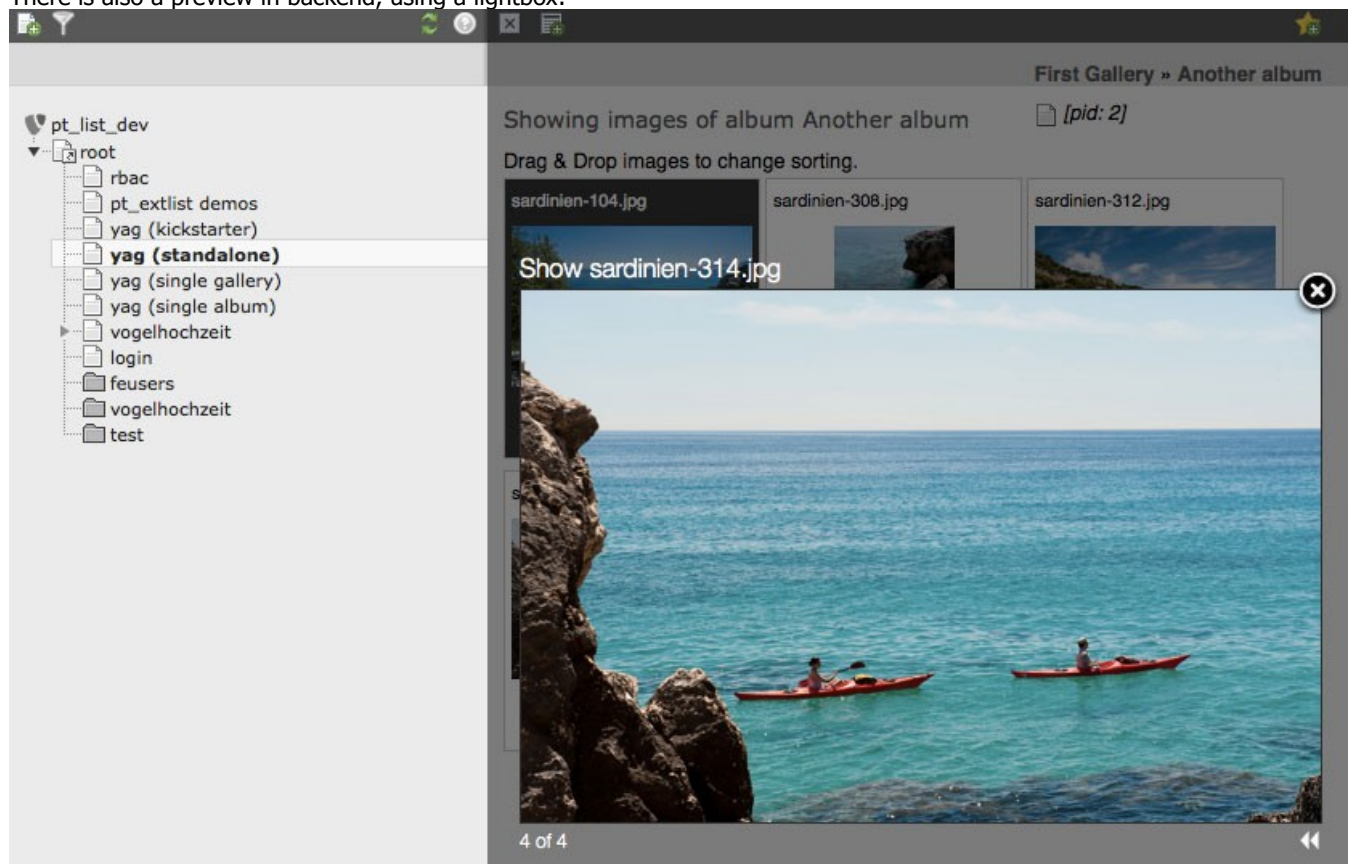
Several upload possibilities including Flash-Multifile-Uploader and ZIP uploader make it easy to add images to your album:
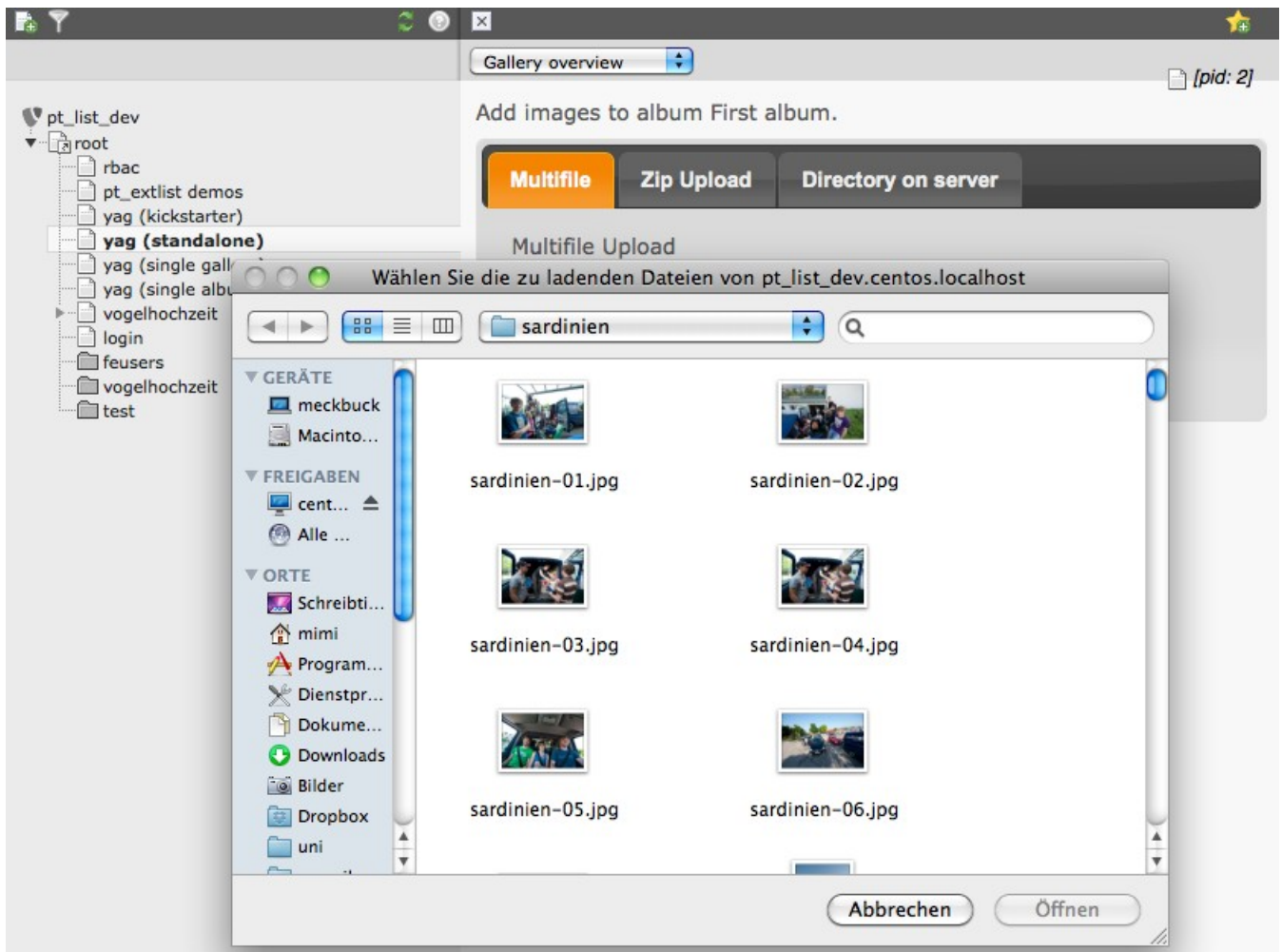
Finally you get some statistics on your gallery and can rebuild the image cache:
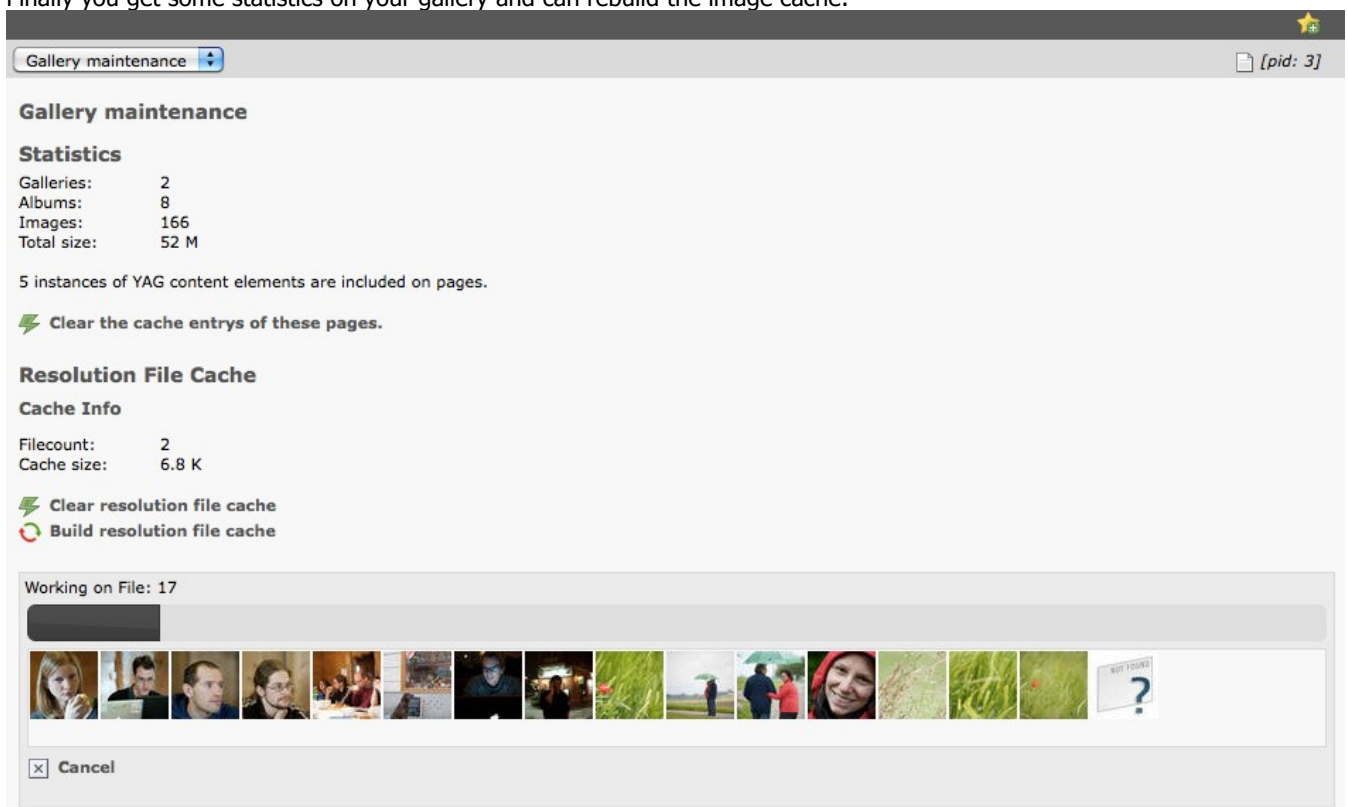
Inserting plugin via FlexForm

YAG comes with a very comfortable FlexForm that lets you select albums, galleries and images easily:
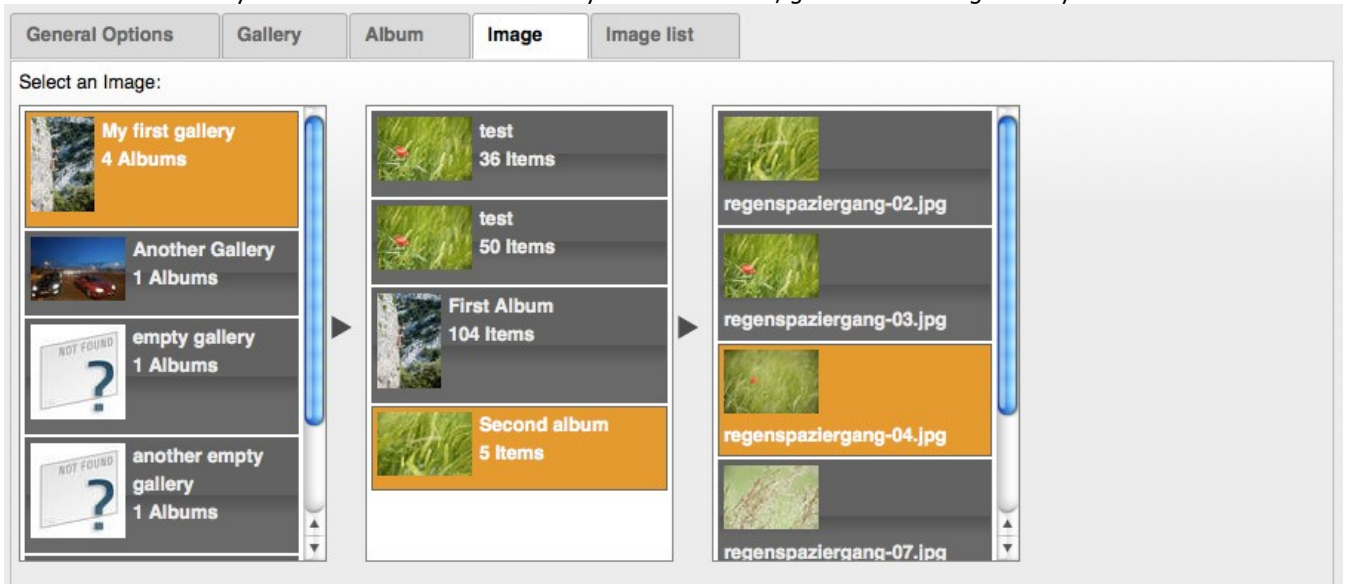
After you inserted the plugin, you get some information about your plugin in the page-module:
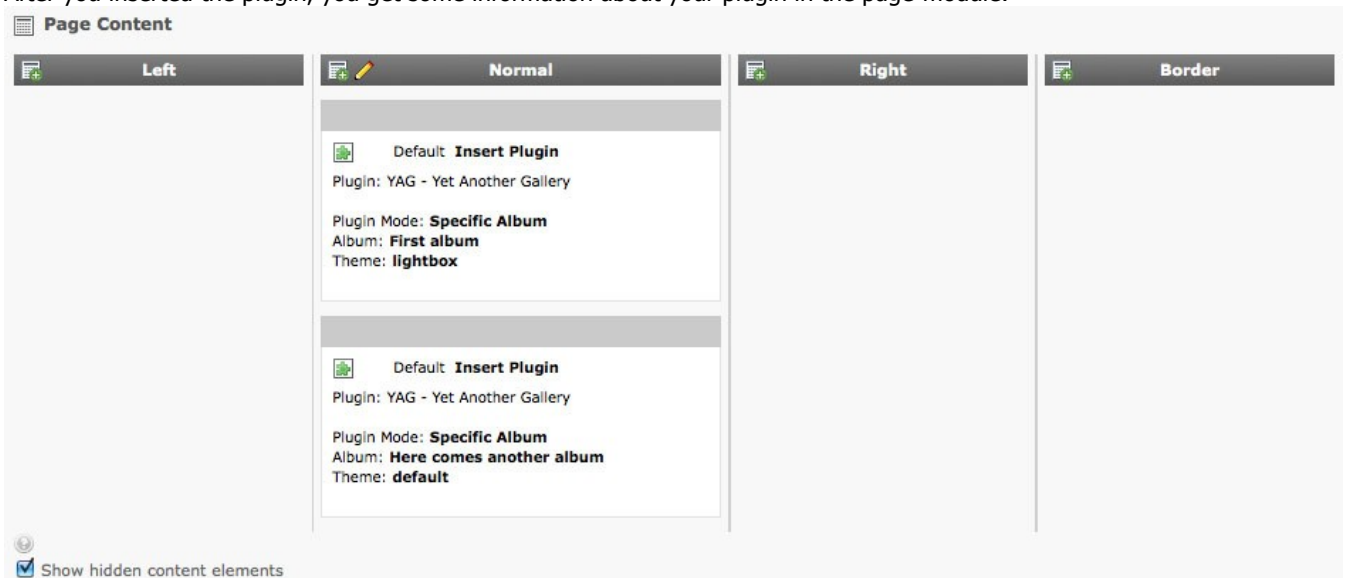
Users manual

We assume, that you have installed YAG according to the section "Installation". This chapter is about using the backend module. See chapter "Inserting YAG on your website" for setting up pages, templates and content element etc.

Setting up your first Gallery

After setting up a page on which you want to insert YAG – whether as a page content element or in standalone mode does not matter here – you can click on the YAG module icon in the backend and will see something like that:
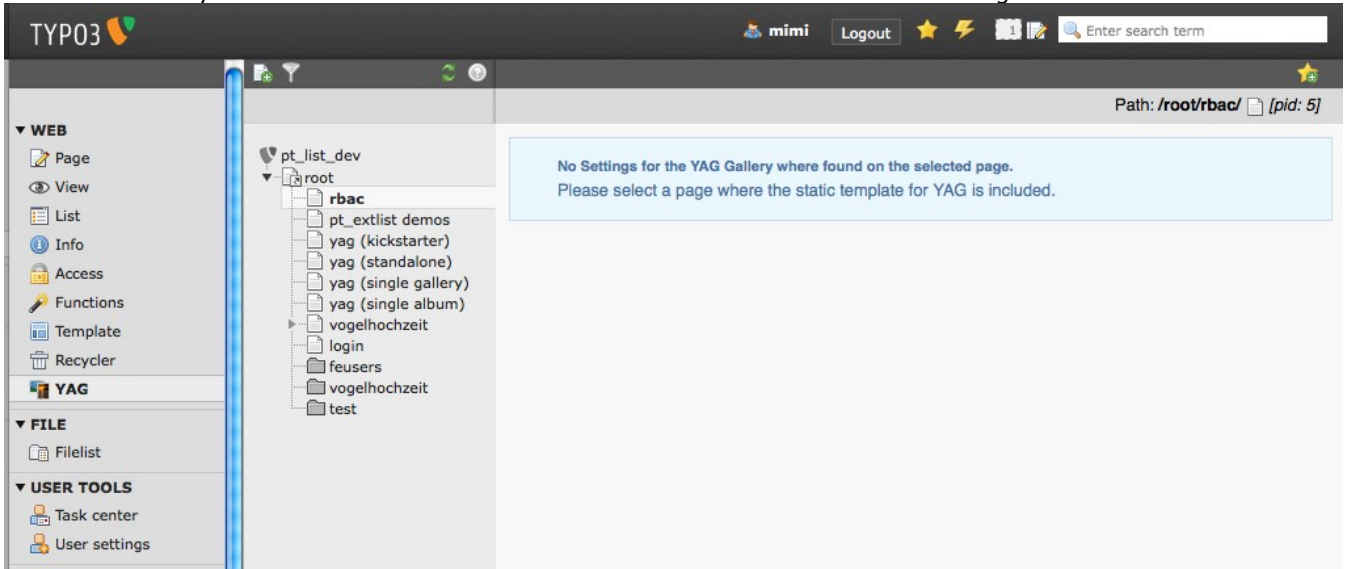
The message tells you, that you did not select a page on which YAG is installed (as content element or with standalone template), so go select a page! After that, you will get YAG module displayed in backend. Of course there is nothing to show yet, as you have not created a gallery:
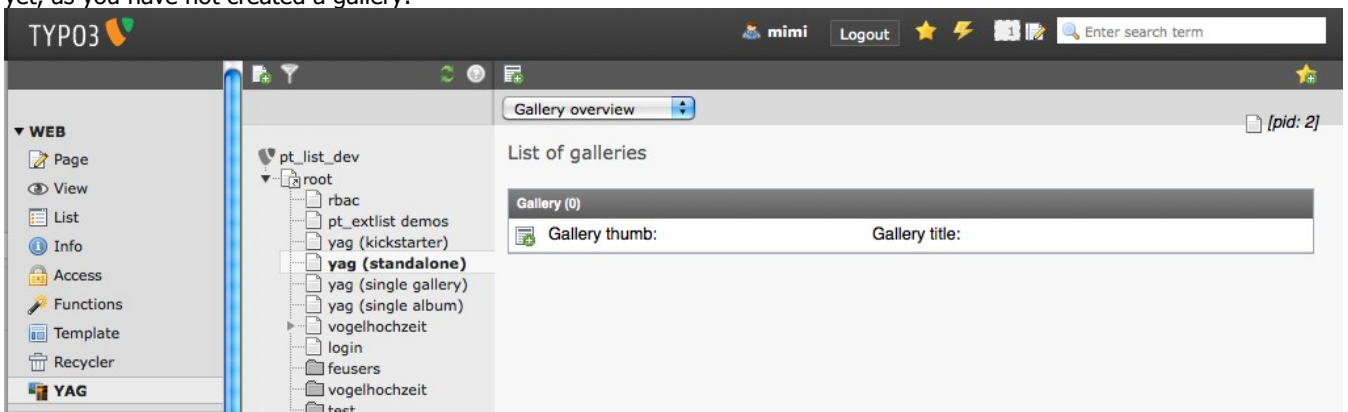
So let's get our first gallery created. Click on the green plus below 'Gallery' and you get a input form like this:

Fill in some text as name and description and press 'Save'. Now you should have a first gallery in your list:

So that's it – you just created your first gallery. Let's go on and create some albums inside this gallery.

Setting up your first Album

After you succesfully created a new gallery, you can now insert some albums. Therefore you can click on the green plus right beneath the garbage symbol in your gallery list:

You will get an empty form for creating a new Album. Fill in some text for Name and Description and click 'Save':
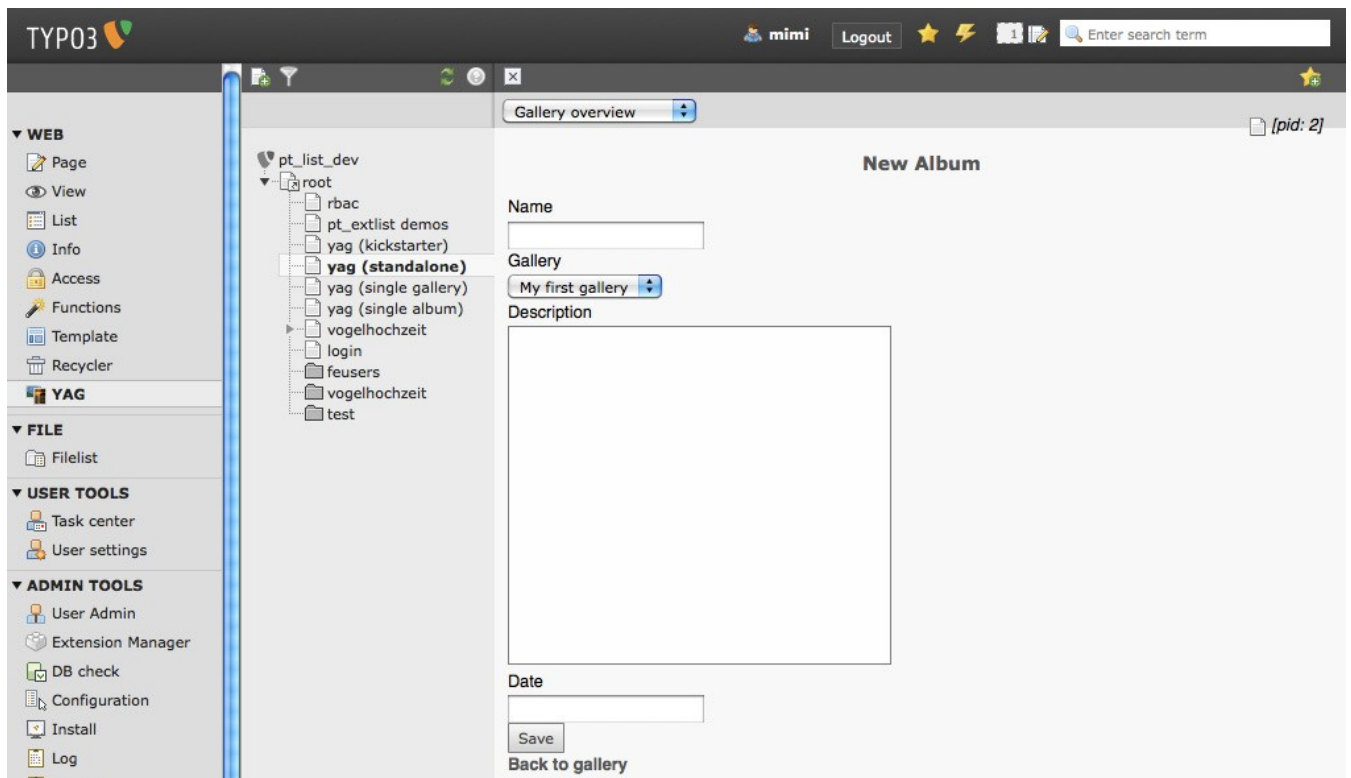
You will get an album list showing your newly created album:
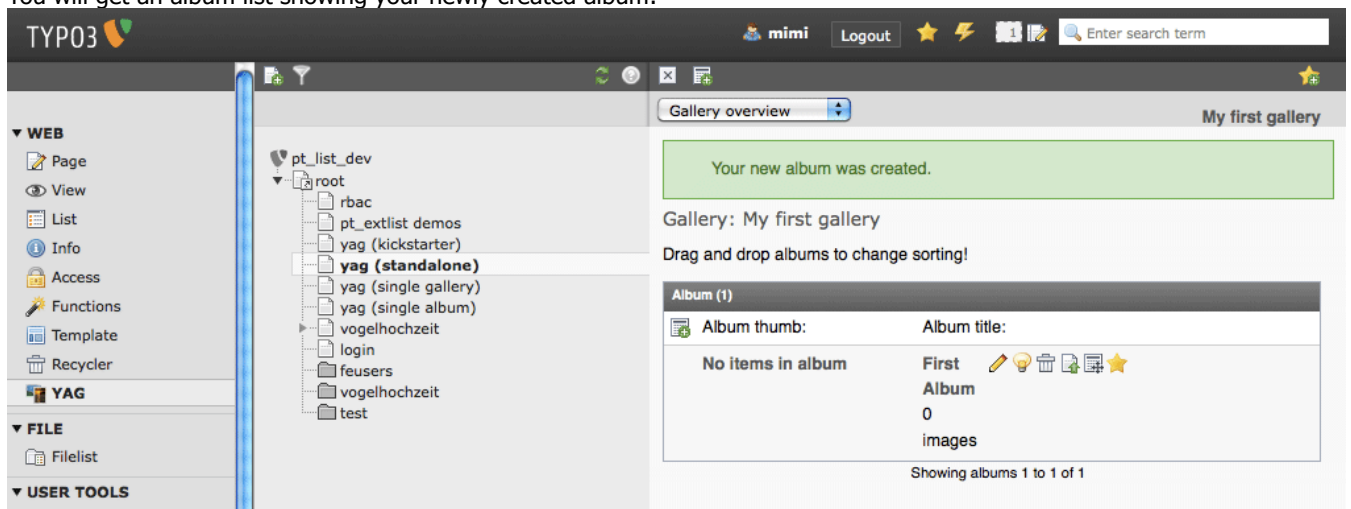
Now we can start uploading some images into our album.

Uploading Images into Albums

There are several ways for uploading images. We show you uploading from album list. So make sure, you have selected a gallery and get a list of albums inside this gallery. Now click on the green arrow-up symbol next to the garbage-bin symbol:
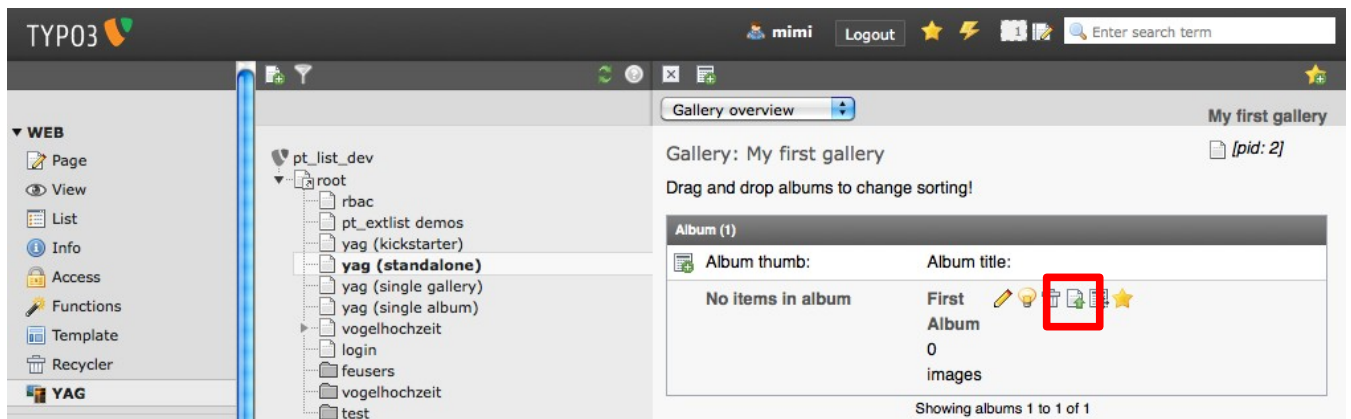
Multifile Upload

This will show up a form containing different possibilities for uploading images into your album. First of all we use the Multifile-Uploader:
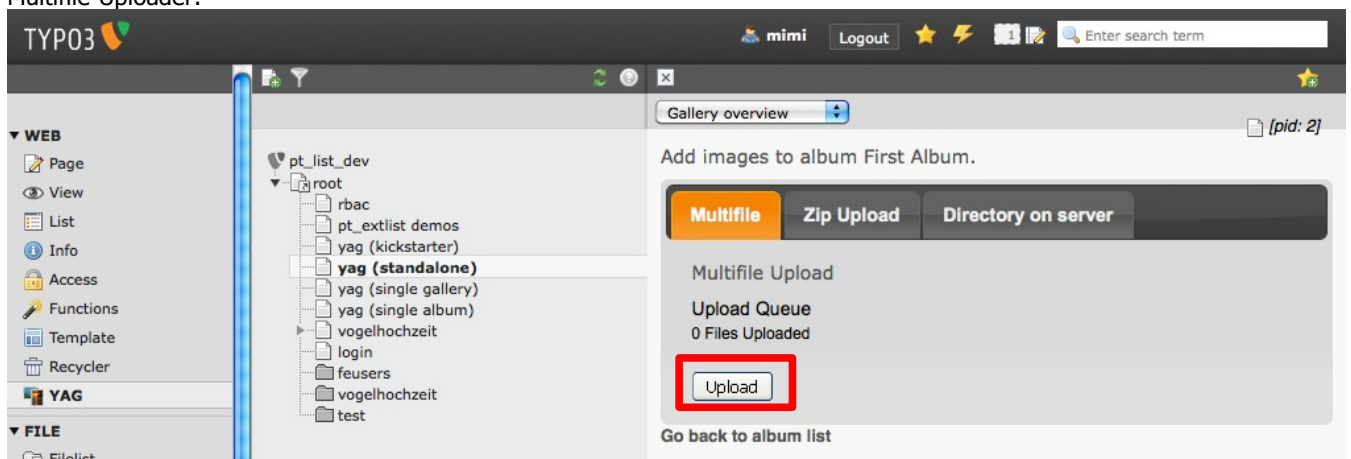
Click on the 'Upload-Button' and you will get a file dialog opened in which you can select multiple files that should be uploaded to your gallery:
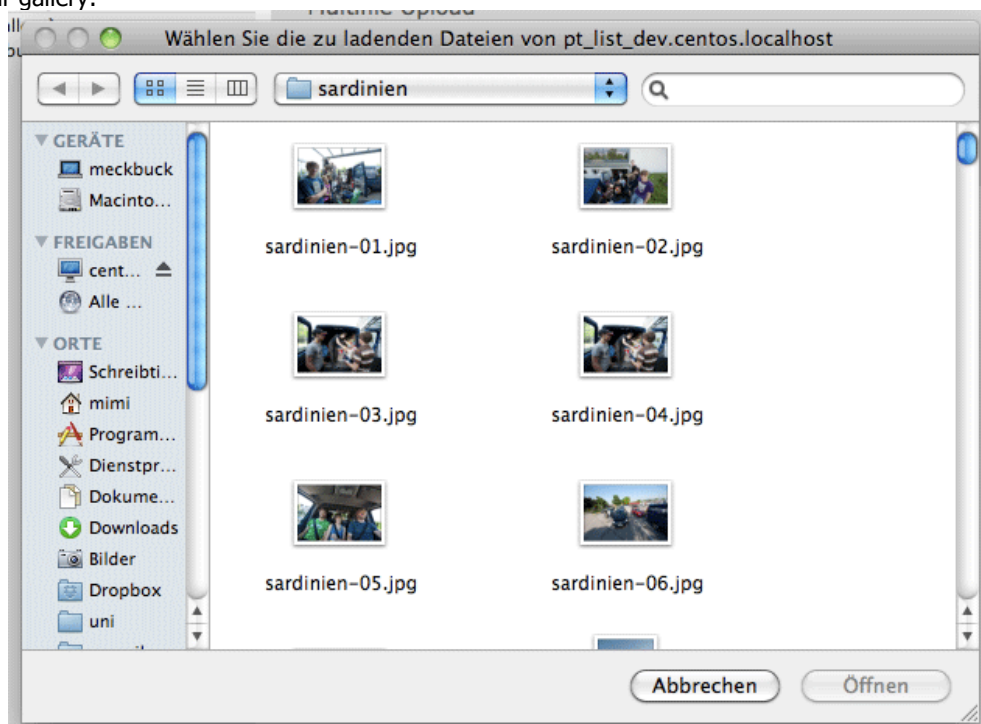
Select as many images as you like an click 'Open'. The uploading will start immediately. After you have finished uploading, click on 'Go back to album list':
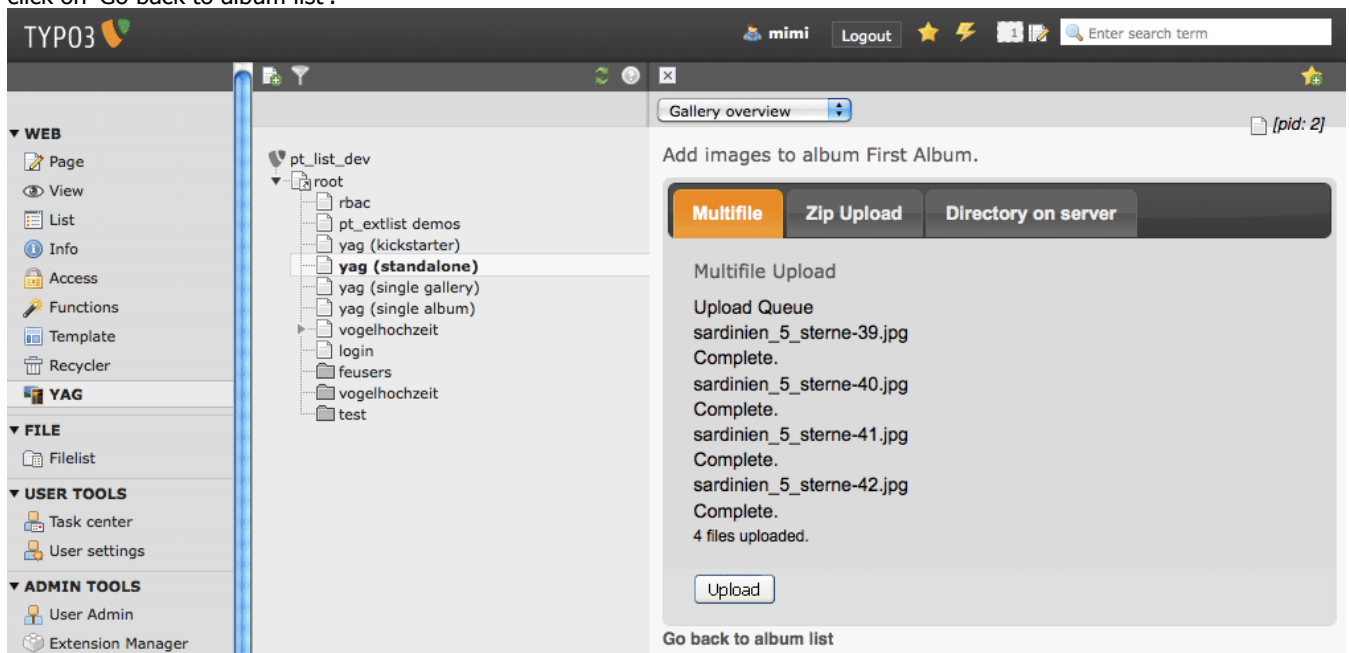
ZIP-File uploading

You can also upload images by putting them into a zip file and uploading this file. ZIP files may also contain folders. Any images in any folder of the ZIP file will be imported:

Importing images from directory on server

If you want to import images from a directory on your server, you can use the third import method:
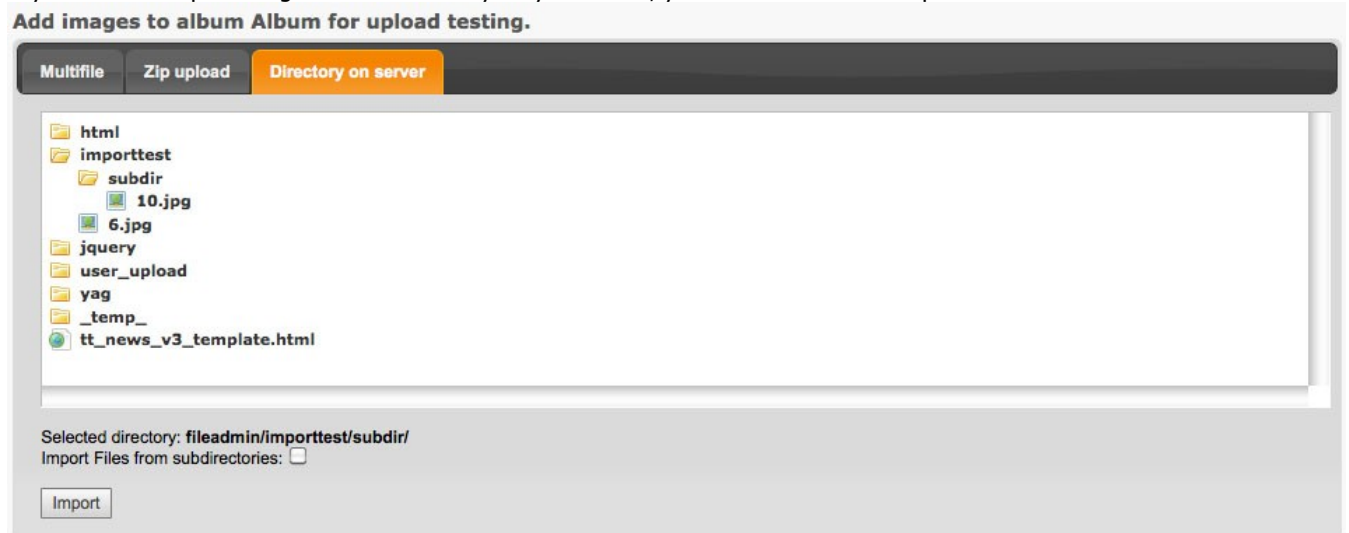
The checkbox at the bottom let's you import files from subdirectories of the folder you chose.

Overview of imported images

After importing items with any of the options a list will show up showing all the images just being uploaded to your album:
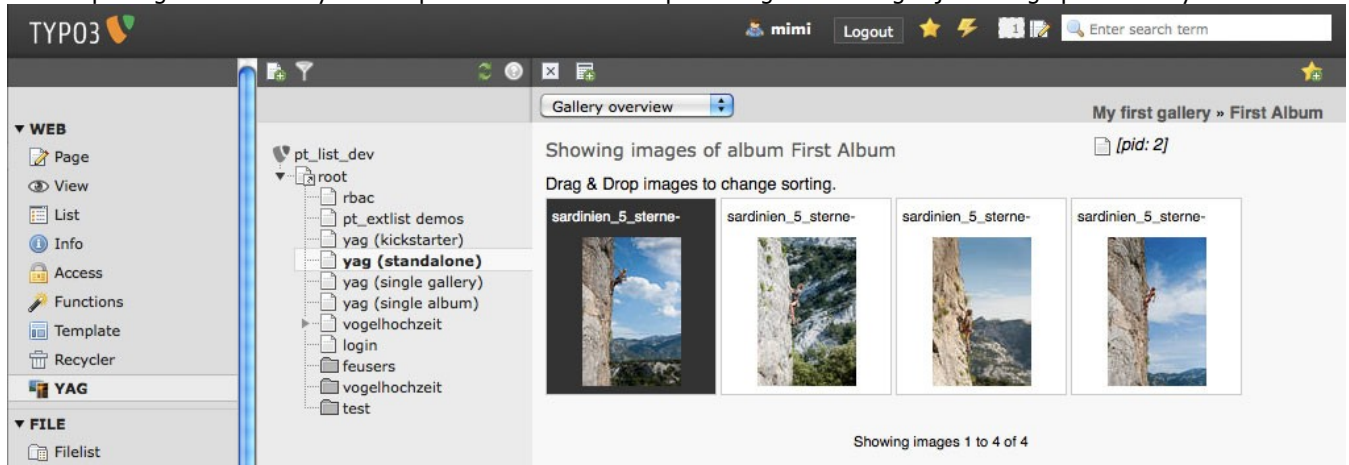
Congratulations – you just created your first album!

Editing Image data inside albums

Open up the YAG module, select a gallery and an album within this gallery. You should now see something like that:
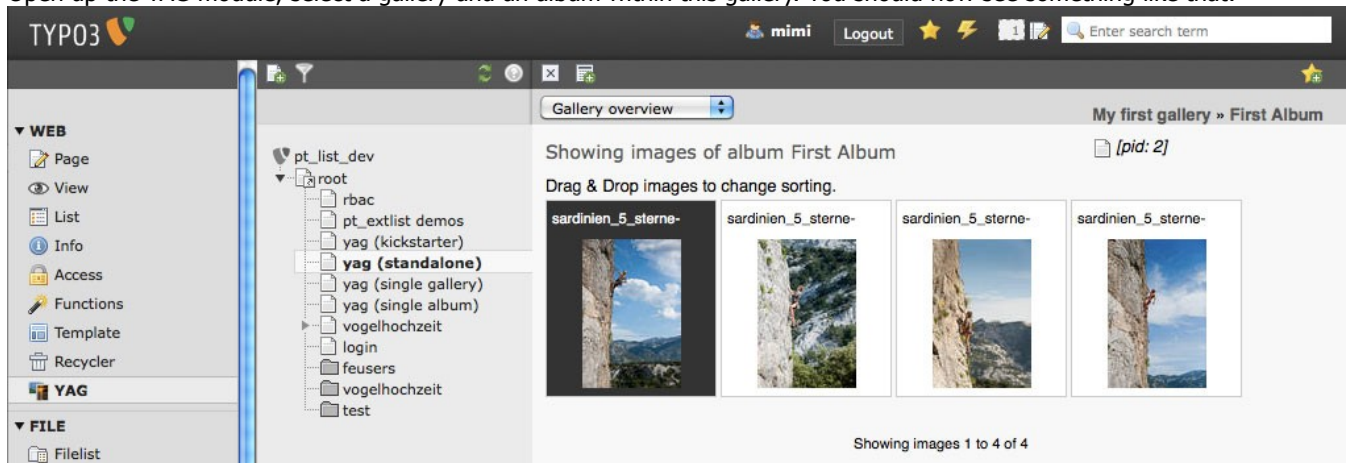
When you move your mouse over an image, a menu will fade in:
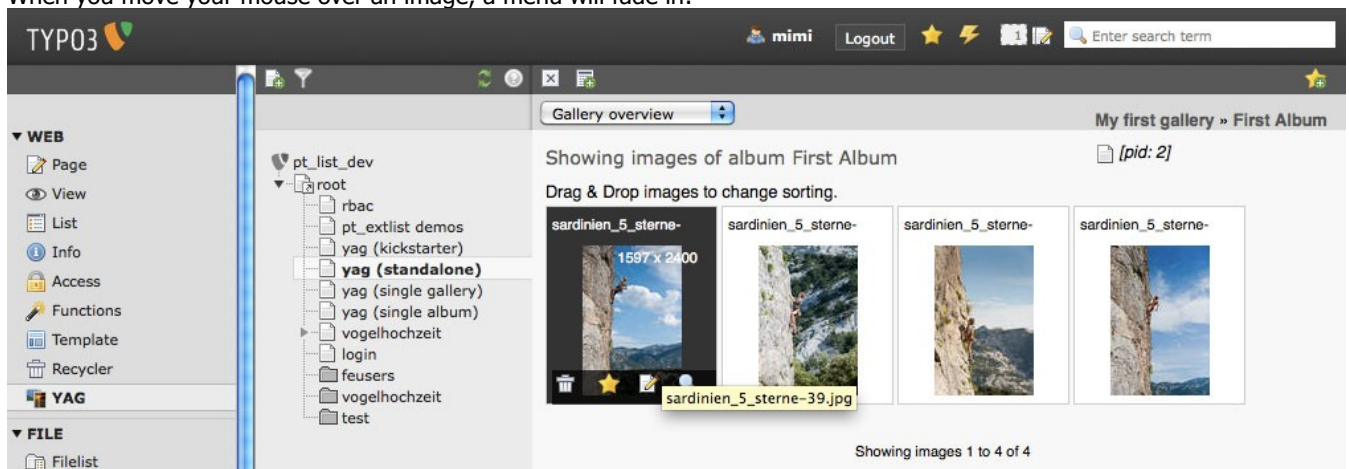
Besides the size of the image, you see some icons below your image, that let you manipulate it:

Using the garbage-bin, you can delete an image. Clicking on it, a dialog will appear and ask you again, whether you want to delete your image:
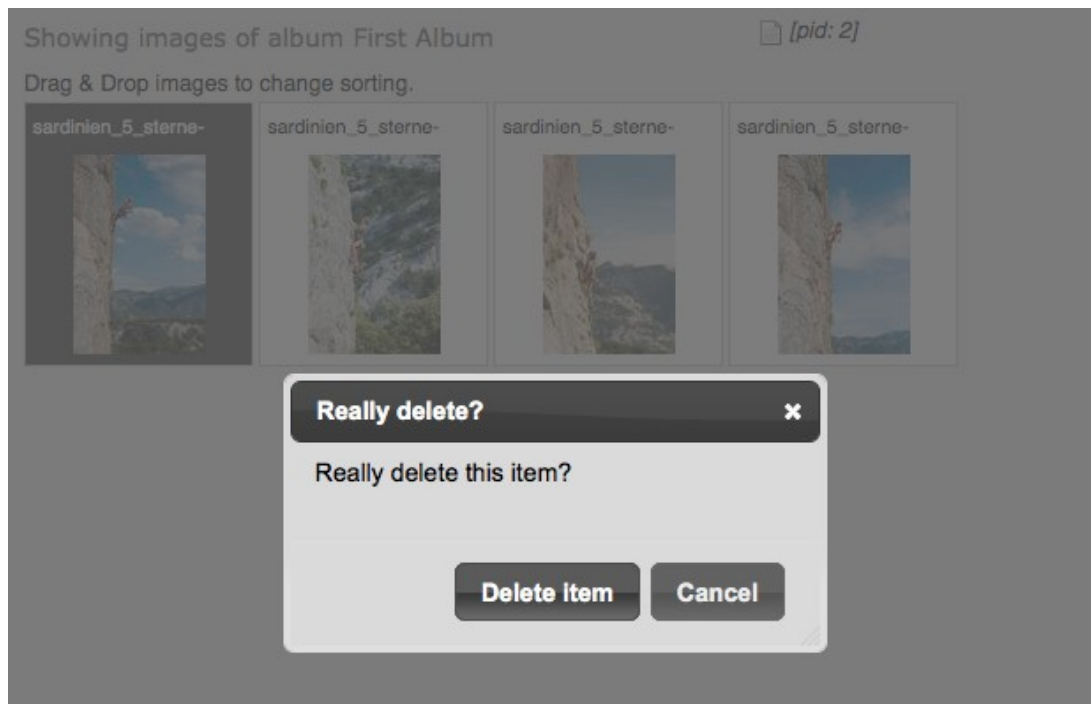
After clicking 'Delete item', the image will fade out and you will get a message, telling you what just has happened:
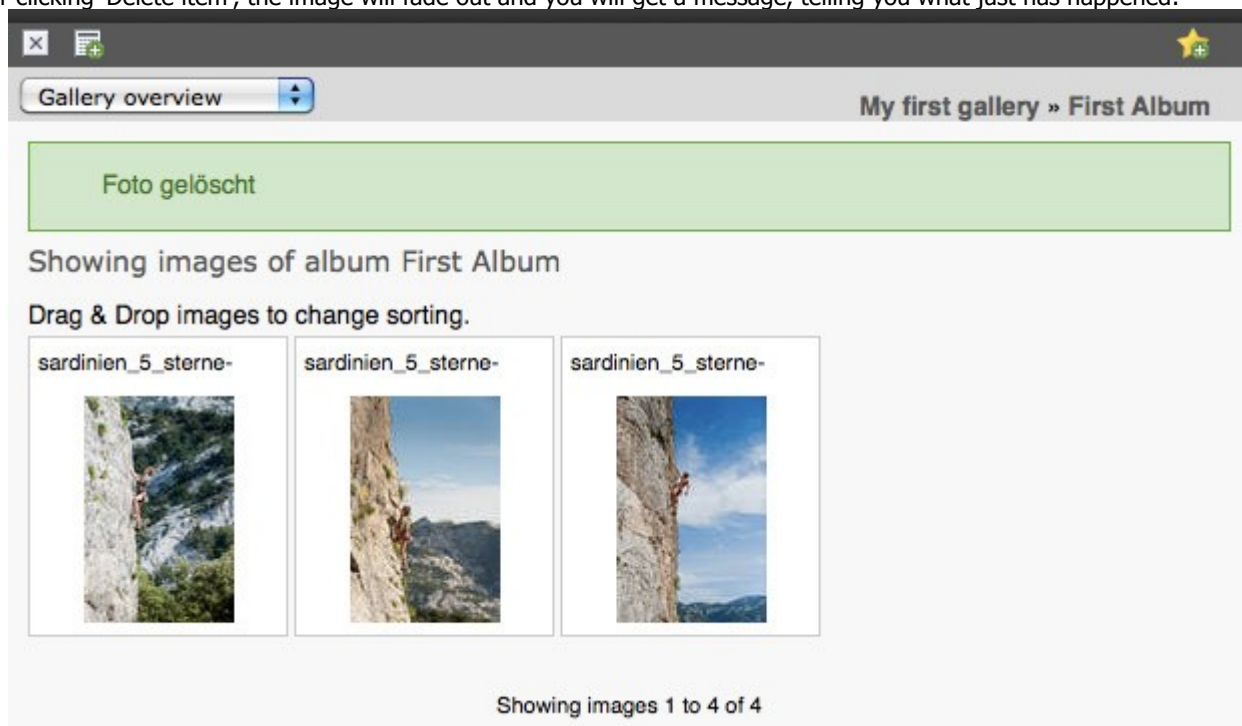
The yellow star sets your image as album thumb. Clicking on it will make the image the thumbnail image for you album:
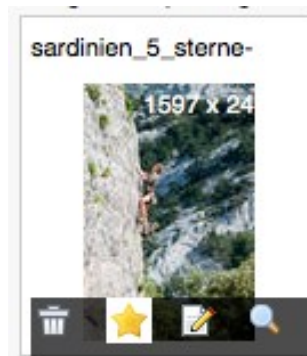
After clicking on it, your image will get a darker background which tells you, that this image is set as album thumbnail:

The edit icon lets you modify the description of your image:

Clicking on it will bring up a little form that lets you edit your image's description:

Drag & Drop images to change sorting.
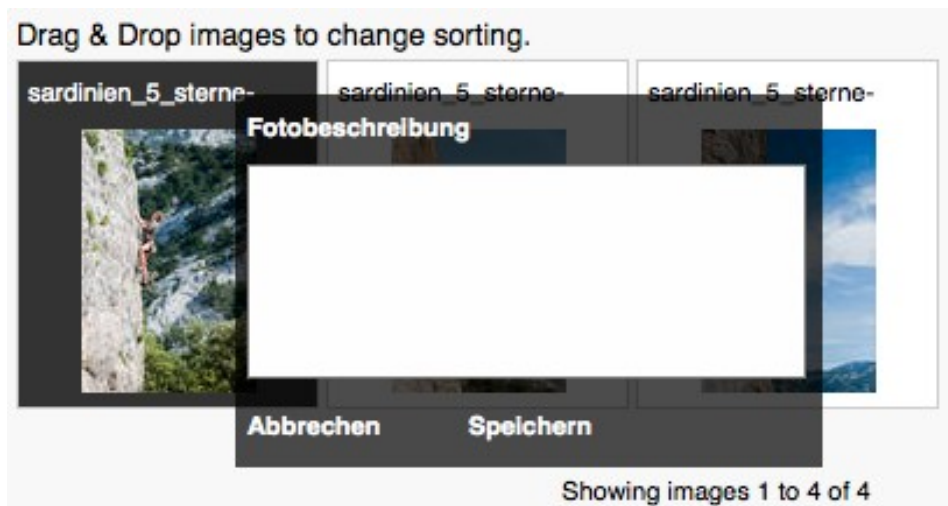
You can edit your image description and click 'Save' to save changes.

The last icon will open up a lightbox showing a bigger version of your image:

When moving over the heading of an image, the background will change to blue and by clicking the heading, you will be able to change the title of your image:

Last but not least, you can change the sorting of your images inside an album by dragging and dropping an image:
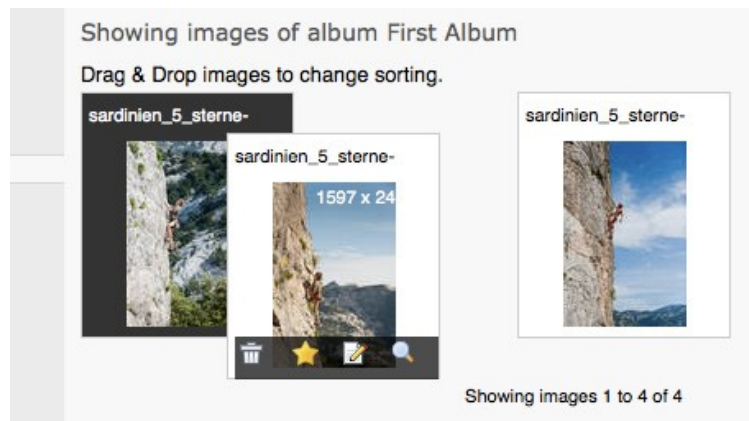
After dropping the image in your prefered position, the sorting will be saved:
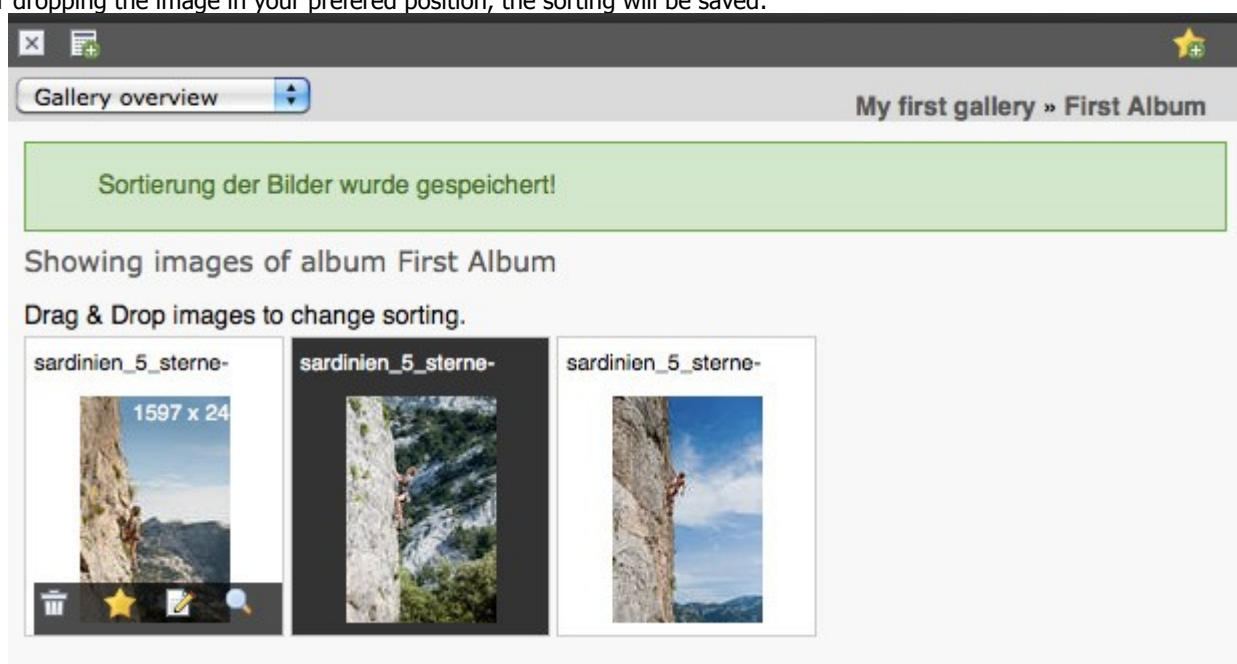
Clicking on the 'x' Symbol in the upper left corner will bring you back to your album list:
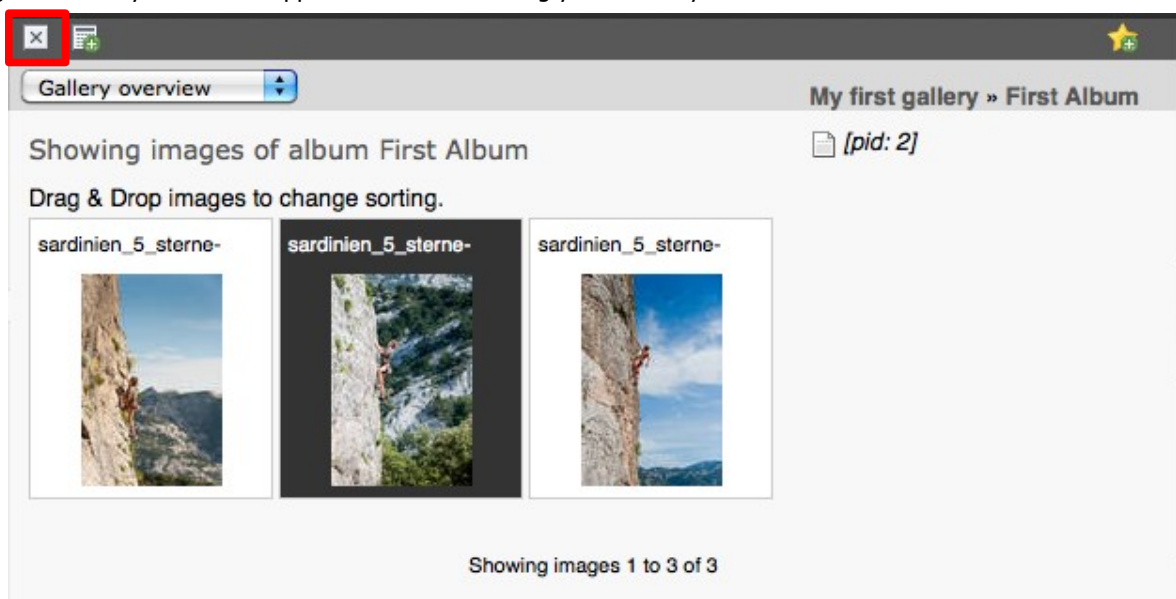
Editing Albums

Your album list comes with some icons for editing:

This lets you edit your albums data in a form:

The lamp will hide / unhide an album in the frontend. If you click on it, it will change to a switched-off lamp and the icon of the album will get lighter to mark it as hidden:

The Garbage-bin will delete your album.

The green arrow will open up an upload dialog that lets you upload some images to your album.

This will open up the image list that lets you change the sorting of your images.

The star sets your album as thumbnail for the gallery. **Please note that the background of your album entry will turn yellow, after you set it as gallery thumb.**

You can add a new album to your gallery by using this button.

Editing Galleries

The first screen you see when starting the YAG backend module is a list of galleries, available on your site:
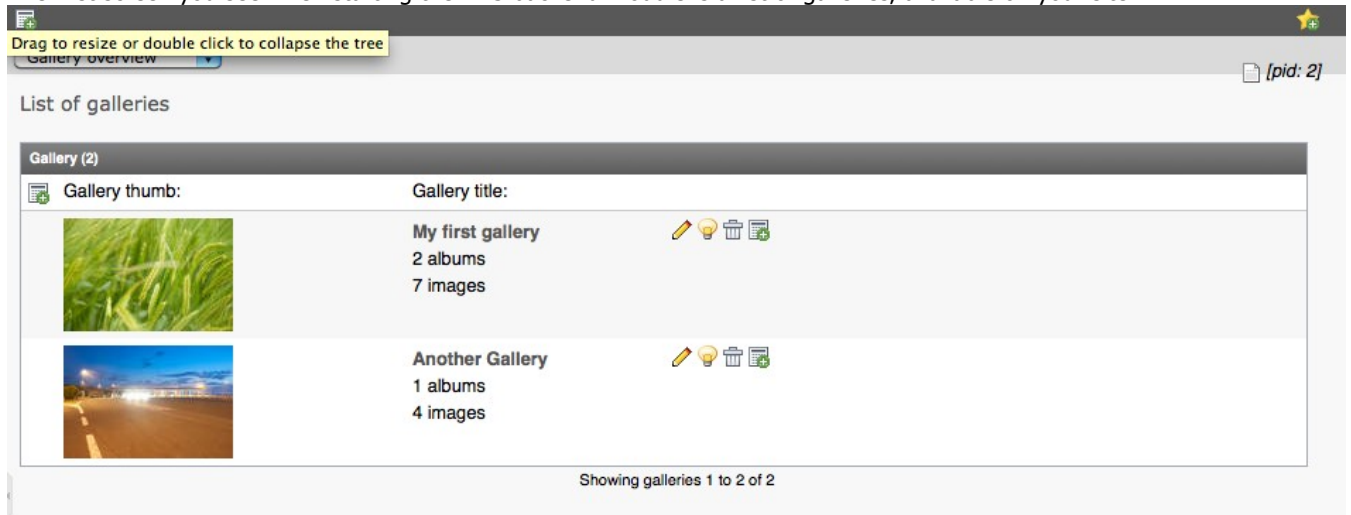
You can edit a gallery by clicking on the pencil icon. This will open a form that lets you edit your gallery's information:

Again, the bulb-icon hides and unhides your gallery. A hidden gallery will no longer be displayed in frontend. Hidden galleries are displayed with a switched-off light bulb and a half-transparent thumbnail:

TODO insert screenshot

Use the garbage-bin to delete a gallery and the green plus to add a new album.

Gallery Maintenance

There is a section for gallery maintenance in the backend. You can open it via the dropdownlist in the YAG module:
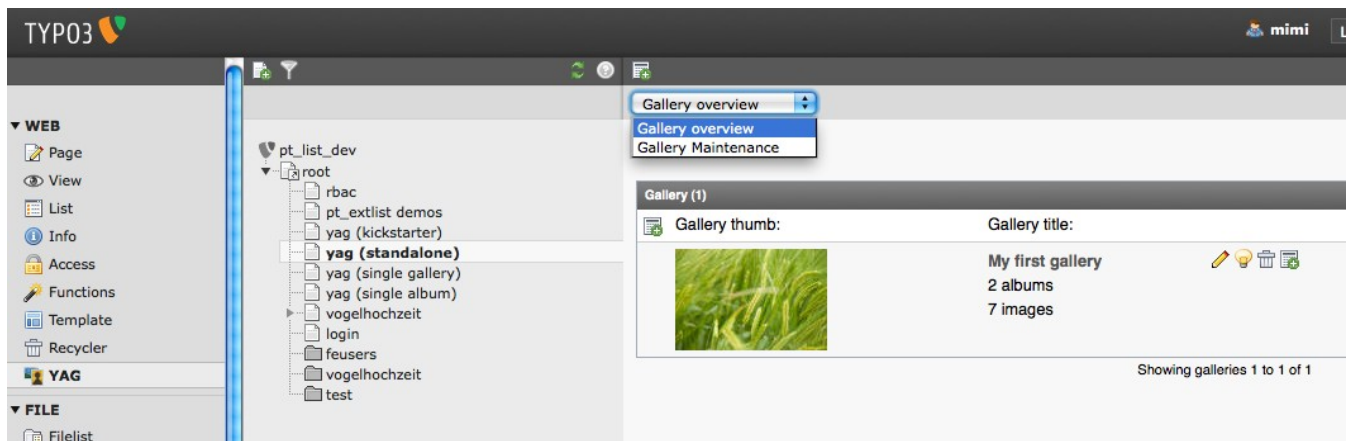
Select 'Gallery Maintainance' there and you will see the following screen:
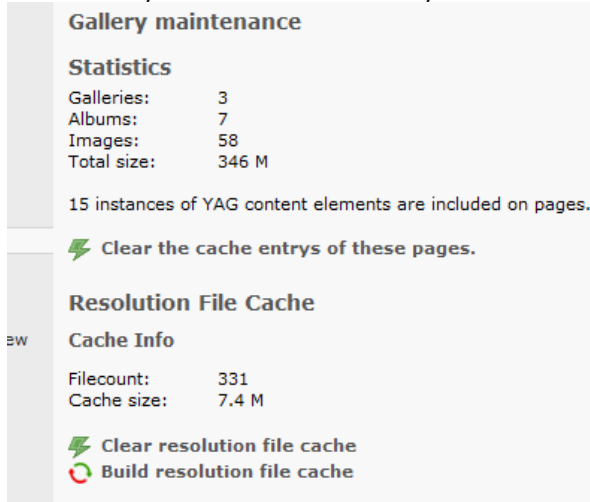
Besides a statistics of galleries, albums and images on your site, there is also some information about cached files and the amount of data used for image files on your harddisk.

The last two links of the site are used for maintaining the cached files. The cache is filled automatically if an image is requested in a specific size and configuration.

When clicking on 'Clear resolution file cache', a message informs you, that the cache has been deleted.

You can build all resolutions at once, this speeds up the page creation in frontend for the first user of your site. When you click on 'Build resolution file cache', there is a progress bar, that shows you the progress of rebuilding the cache:
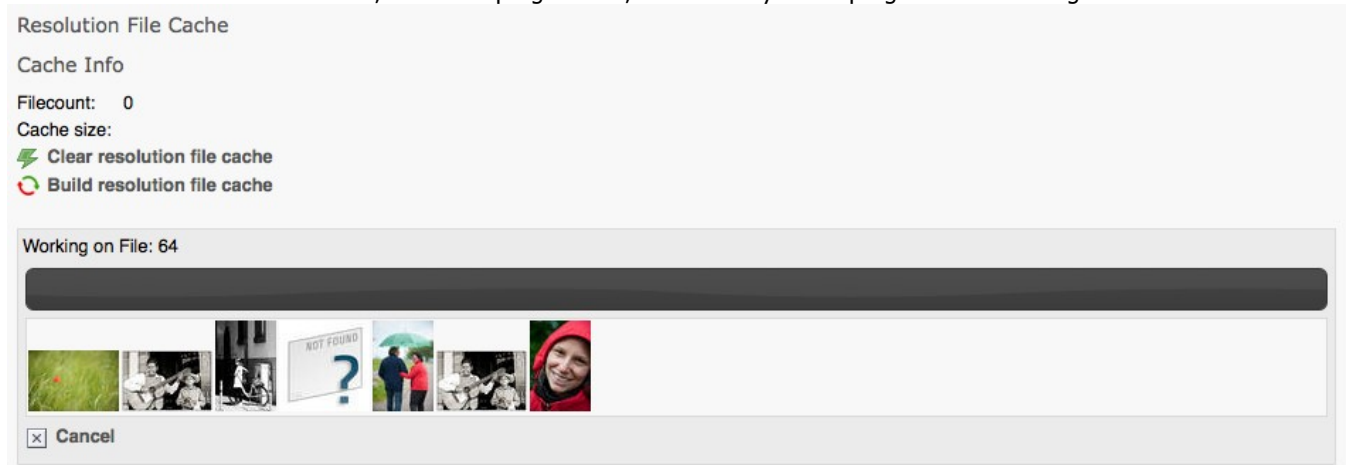
Installation

This chapter will give you a step-by-step introduction on how to install the extension.

Dependent extensions

At the moment, there are 3 dependencies for this extension:

- pt_tools – a collection of libraries used throughout this extension.

- pt_extlist – a list generator used to render all kinds of lists, pagers and filters.

In order to install YAG, you have to set up those three extensions in the order given above (pt_tools, pt_extlist, rbac). After that, the installation of YAG is quite straight-forward.

Importing YAG from TER

Open up the Extension Manager and select 'Import Extension' from the menu:

Search for 'yag':

Click on the install button:

Extensions in the TYPO3 Extension Repository (online) - Grouped by: Category

Look up extensions:

| yag | Look up |

	Title	Extension key:	Version	Upload date:	Author	Cur. Ver:	Cur. Type:	DL:	State
📁 Frontend									
	🚫 Simple Page Header Selector - Extended	bs_headerselector_yags	0.0.1	18-05-08	Markus Wanjura			56/56	Beta
📁 Frontend Plugins									
	jk_poll extended	yags_jk_poll_extended	0.1.1	24-06-09	Markus Wanjura			31/30	Beta
	YAG Theme Simpleviewer	yag_theme_simpleviewer	0.1.1	07-03-11	Daniel Lienert			4/4	Beta
	Yet Another Gallery	yag	1.0.2	08-03-11	Michael Knoll,Daniel Lienert	1.0.2		16/8	Beta

There will be a message informing you, that additional information has to be given:

Extension Manager

Extension: 🖼Yet Another Gallery (yag)

🔲 Installing 🖼Yet Another Gallery: DATABASE NEEDS TO BE UPDATED

Before the extension can be installed the database needs to be updated with new tables or fields.
Please select which operations to perform:

ℹ This extension provides additional configuration options which become available once it is installed.

Make updates

Click on 'Make updates' and you will get a form you have to submit:

✅ The extension "yag" has been installed.

Extension Manager

Extension: 🖼Yet Another Gallery (yag)

Current status:

The extension is installed (loaded and running)!
Click here to remove the extension: 📦

Configuration:

(Notice: You may need to clear the cache after the configuration of the extension. This is required if the extension adds TypoScript depending on these settings.)

Path of directory where YAG should ... [hashFilesystemRoot]
Path of directory where YAG should store all cached images generated for albums - relative to Typo3 base path. This path should not be changed unless you know what you do! (Image base path)

| typo3temp/yag |

Path of directory where YAG stores ... [origFilesRoot]
Path of directory where YAG stores all original files.

| fileadmin/yag |

Importing TypoScript RBAC settings ... [updateMessage]
Importing TypoScript RBAC settings into database

RBAC data has been imported into database.

Update

There are two settings you have to make:

- **Path of directory where YAG should ...[hashFilesystemRoot]**: YAG will create a hashed image for each image shown in different sizes. Those cached files need to be stored on your server. You have to determine here, which folder will be used to do that. Default is typo3temp/yag

- **Path of directory where YAG stores ...[origFilesRoot]**: YAG will store each original file on your server. You have to determine here, where YAG should store original files on your server. Default is fileadmin/yag

Besides those settings, some data is imported into your database for Permissions handled by RBAC.

Click 'Update' to confirm those settings. A message will be shown informing you about the state of the installation:

Extension Manager

Extension: **Yet Another Gallery** (yag)

Current status:

The extension is installed (loaded and running)!
Click here to remove the extension:

That's it. You have now successfully installed YAG on your server.

Installing additional themes

To shape the look-and-feel of your gallery, you can use different themes available in TER. Installation is straight forward. Go to the Extension Manager and search for 'yag'. There should be at least one theme:

Extensions in the TYPO3 Extension Repository (online) - Grouped by: Category

Look up extensions:
[yag] [Look up]

		Title	Extension key:	Version	Upload date:	Author	Cur. Ver:	Cur. Type:	DL:	State
📁 **Frontend**										
	⊘	Simple Page Header Selector - Extended	bs_headerselector_yags	0.0.1	18-05-08	Markus Wanjura			56/56	Beta
📁 **Frontend Plugins**										
		jk_poll extended	yags_jk_poll_extended	0.1.1	24-06-09	Markus Wanjura			31/30	Beta
		YAG Theme Simpleviewer	yag_theme_simpleviewer	0.1.1	07-03-11	Daniel Lienert			4/4	Beta
		Yet Another Gallery	yag	1.0.2	08-03-11	Michael Knoll,Daniel Lienert	1.0.2		16/8	Beta

Use the install button to import the extension on your server:

Extensions in the TYPO3 Extension Repository (online) - Grouped by: Category

Look up extensions:
[yag] [Look up]

		Title	Extension key:	Version	Upload date:	Author	Cur. Ver:	Cur. Type:	DL:	State
📁 **Frontend**										
	⊘	Simple Page Header Selector - Extended	bs_headerselector_yags	0.0.1	18-05-08	Markus Wanjura			56/56	Beta
📁 **Frontend Plugins**										
		jk_poll extended	yags_jk_poll_extended	0.1.1	24-06-09	Markus Wanjura			31/30	Beta
		YAG Theme Simpleviewer	yag_theme_simpleviewer	0.1.1	07-03-11	Daniel Lienert			4/4	Beta
		Yet Another Gallery	yag	1.0.2	08-03-11	Michael Knoll,Daniel Lienert	1.0.2		16/8	Beta

After import, a message will be shown and you can install the theme click on 'Install extension':

Extension Manager
Extension import results

✓ **Extension imported**
Folder created: /var/www/kunden/pt_list_dev.centos.localhost/typo3conf
/ext/yag_theme_simpleviewer/
ext_emconf.php: /var/www/kunden/pt_list_dev.centos.localhost/typo3conf
/ext/yag_theme_simpleviewer/ext_emconf.php
Installation Type: Local

Install/Uninstall Extension:
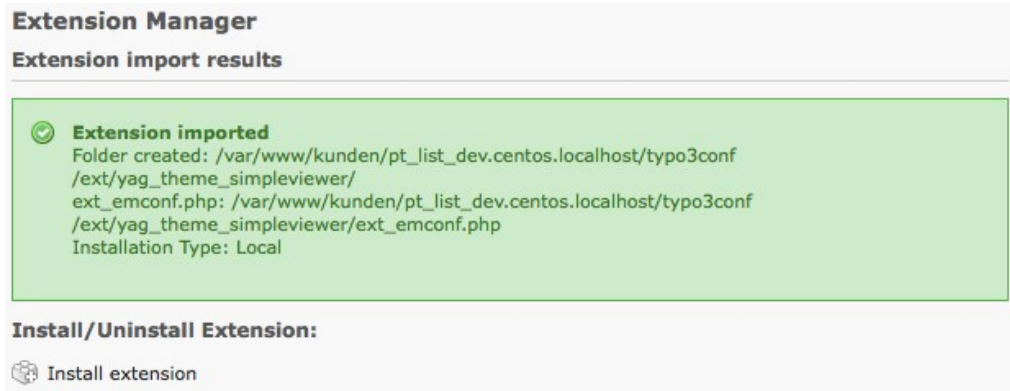
🗔 Install extension

There should be a message, telling you that the extension has been installed:

✓ The extension "yag_theme_simpleviewer" has been installed.

Extension Manager
Group by: [Category ▼]   Show: [Details ▼]
Only my extensions: ☐   Show obsolete: ☐

Extensions in the TYPO3 Extension Repository (online) - Grouped by: Category

Look up extensions:
[yag]   [Look up]

	Title	Extension key:	Version	Upload	Author
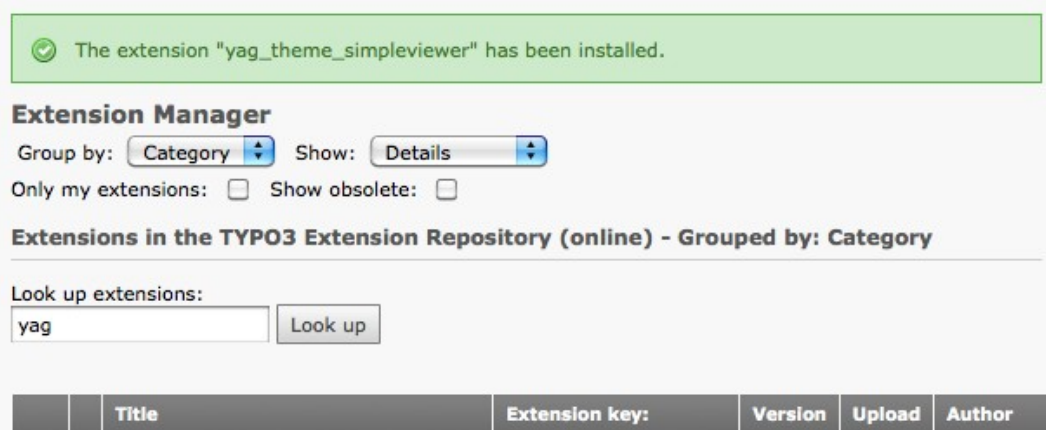
Now you have to include the theme's static template on the page you would like to use the theme. This is done via Template-Module on the page you want to have the new theme:
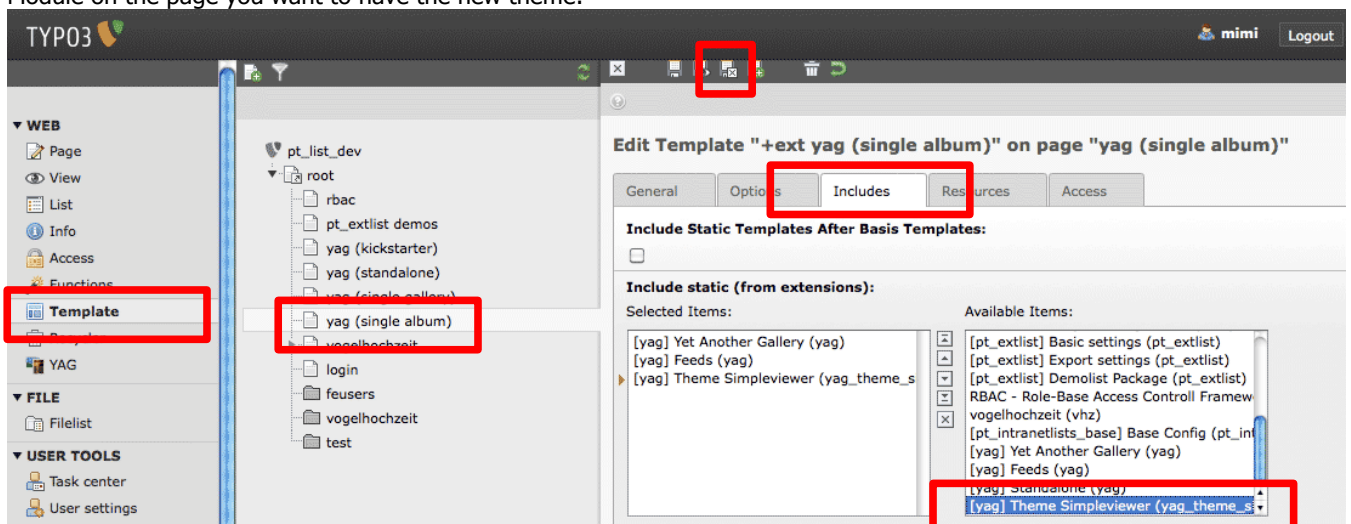
Right after that, you should be able to select the theme from within your FlexForm of the content element that includes your gallery widget:

Administration

This chapter gives you a step-by-step introduction on how to set up a page hierarchy and templates to bring YAG on your site.

Setting up YAG for standalone usage

YAG ships with a TypoScript Template that lets you install YAG standalone, without requiring a page hierarchy or anything. You only have to set up a single page in backend and include the standalone TS-Template. Here are the steps:

1. Create a new, empty page where you want to have YAG standalone installed.

2. Create a new TypoScript Template using the template module on this page. Click "Create template for a new site"
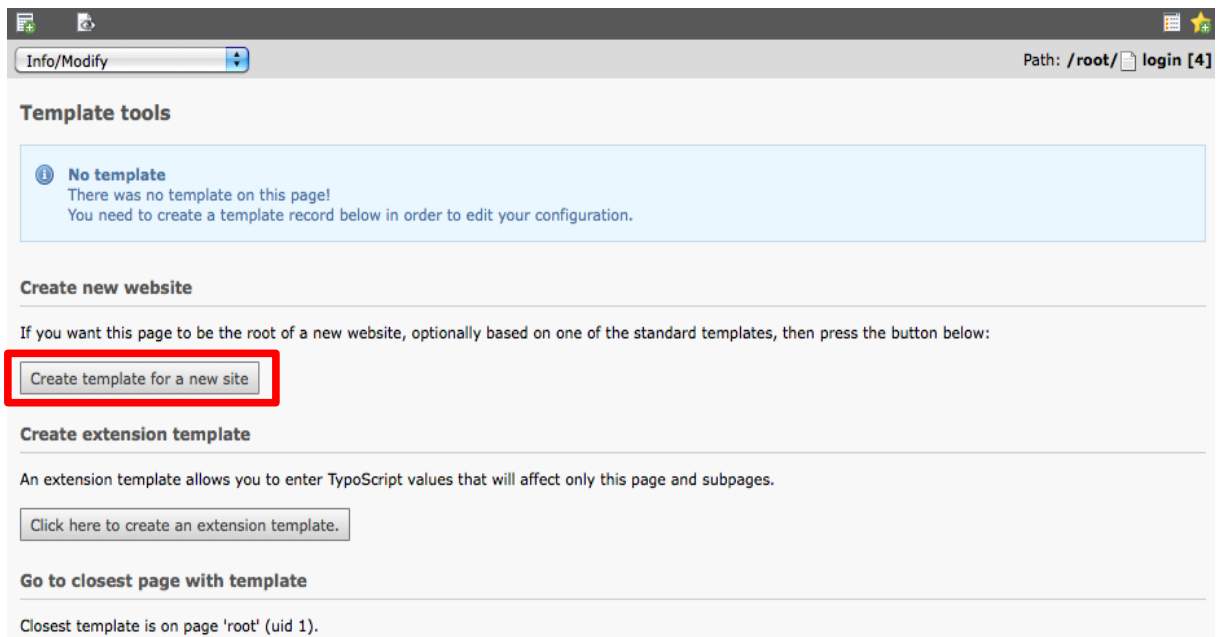
3. Within your new template, remove anything from "Setup" and open the tab "Includes". Select "Standalon (yag)" from Include statics:
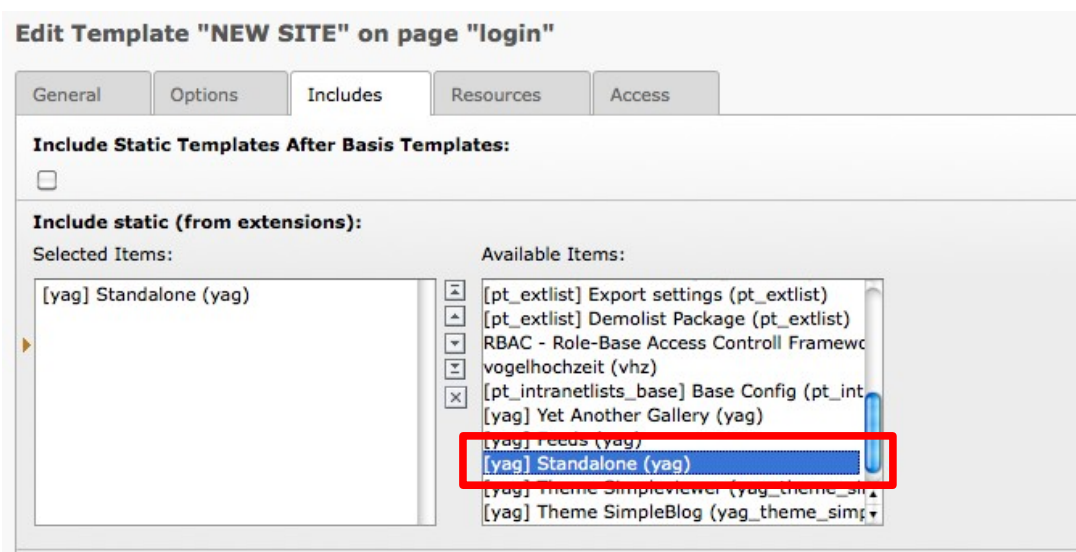
4. Save your TS template and you are done!

Setting up YAG for usage as content-element

If you want to use YAG as content elements, here are the steps, you have to follow:

1. Create a new page on which you want to include YAG.
2. Create an extension template using Template module:
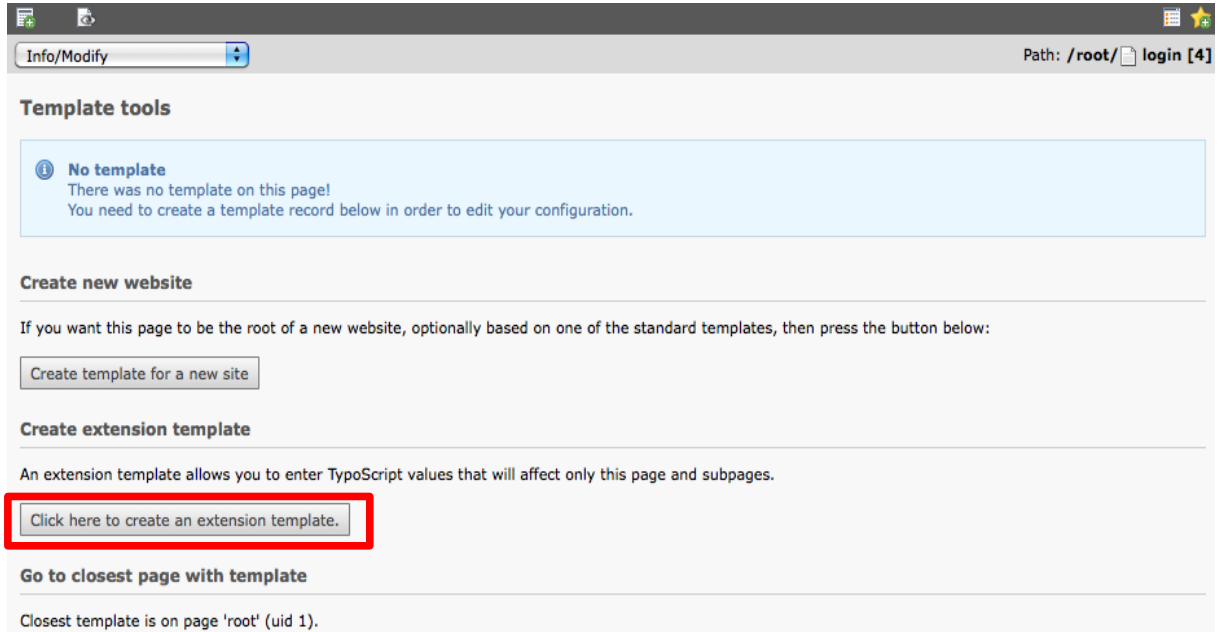
3. Edit the extension template and switch to the "Includes" tab. Include "Yet Another Gallery (yag)" and "Feeds (yag)":
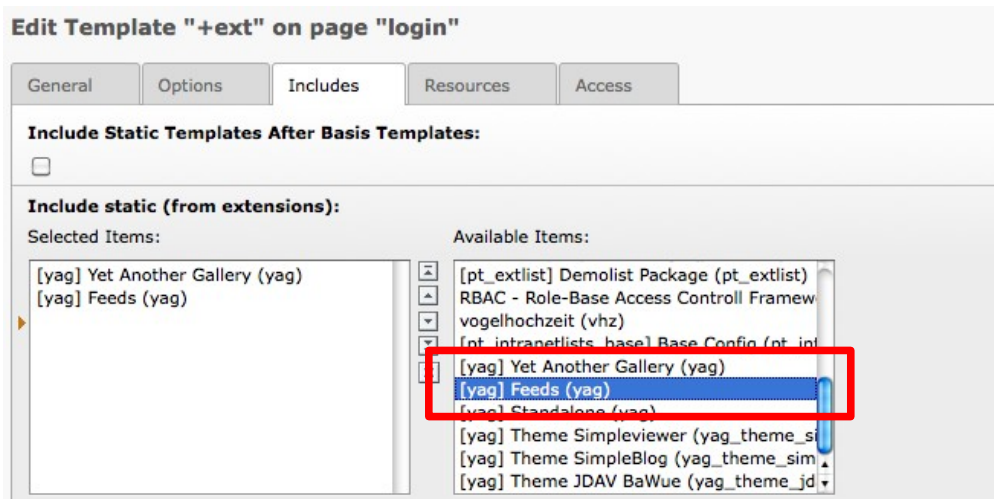
4. Save your TS template.
5. Open the "Page" module and go to the page you just created. Insert a page content element. From the list of contents, chose "Plugins → YAG – Yet Another Gallery":
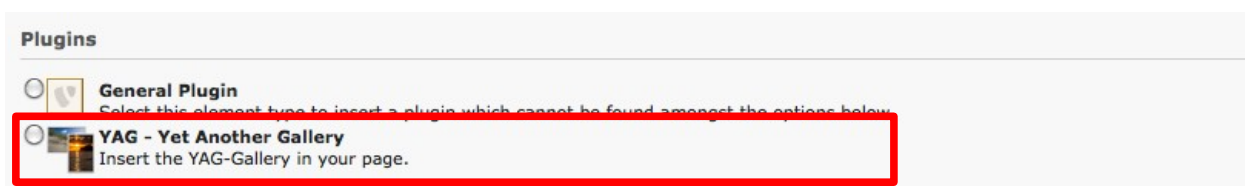
6. Switch to the "Plugin" tab and you can configure your content element:

Configuration possibilities in FlexForm

Tab "General"

- **Plugin type** lets you chose what you want the plugin to do:
    - **Gallery** list shows a list of galleries
    - **Specific gallery** shows a single gallery you can define in the "Gallery" Tab
    - **Specific album** shows a single album you can define in the "Album" Tab
    - **Specific image** shows a single image you can define in the "Image" Tab
- **Theme** lets you chose the theme you want to use to style your gallery
- **Context Identifier** whenever you want to put more than one gallery plugin on a single page, you have to set up a context to make them work independently.

Tab "Gallery"

You can select a gallery here. Only has an effect, if plugin is in "Gallery"-mode (selected plugin type on first tab).
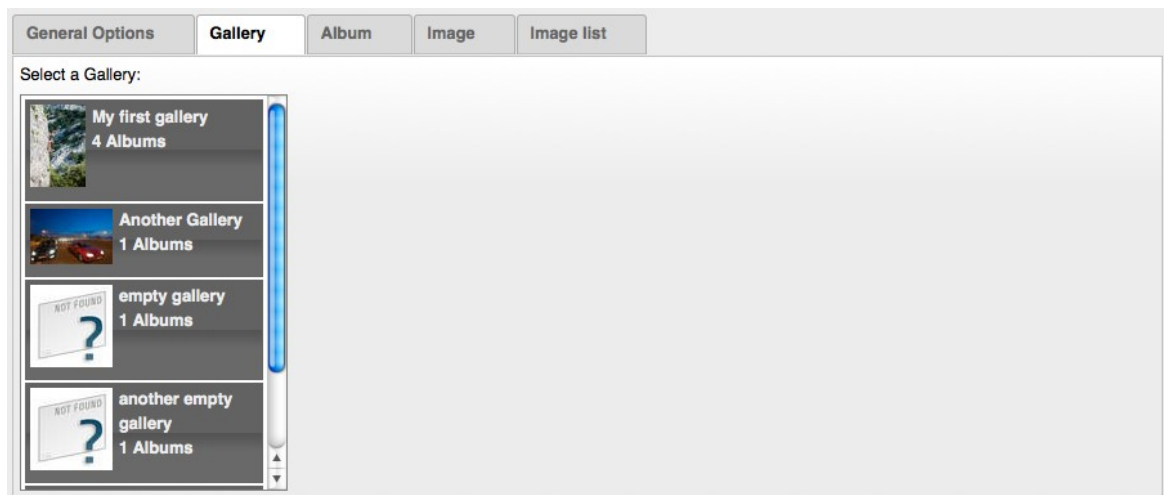
Tab "Album"

You can select an album here. Only has an effect, if plugin is in "Album"-mode (selected plugin type on first tab).
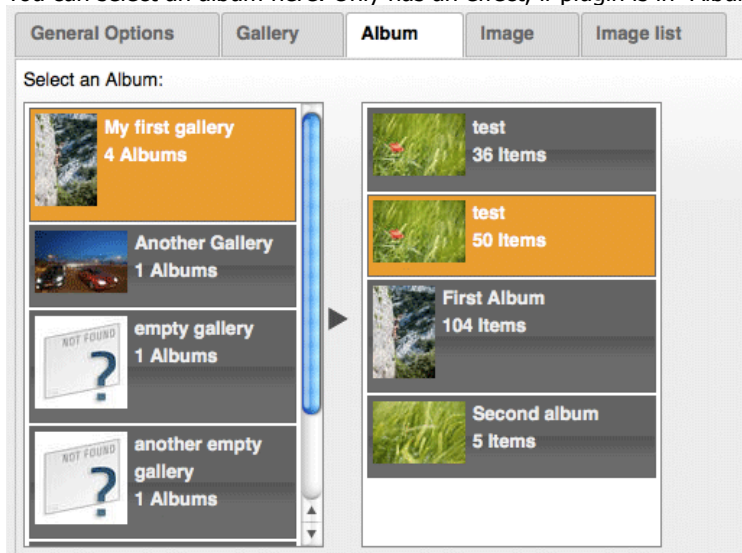
Tab "Image"

You can select an image here. Only has an effect, if plugin is in "Image"-mode (selected plugin type on first tab).
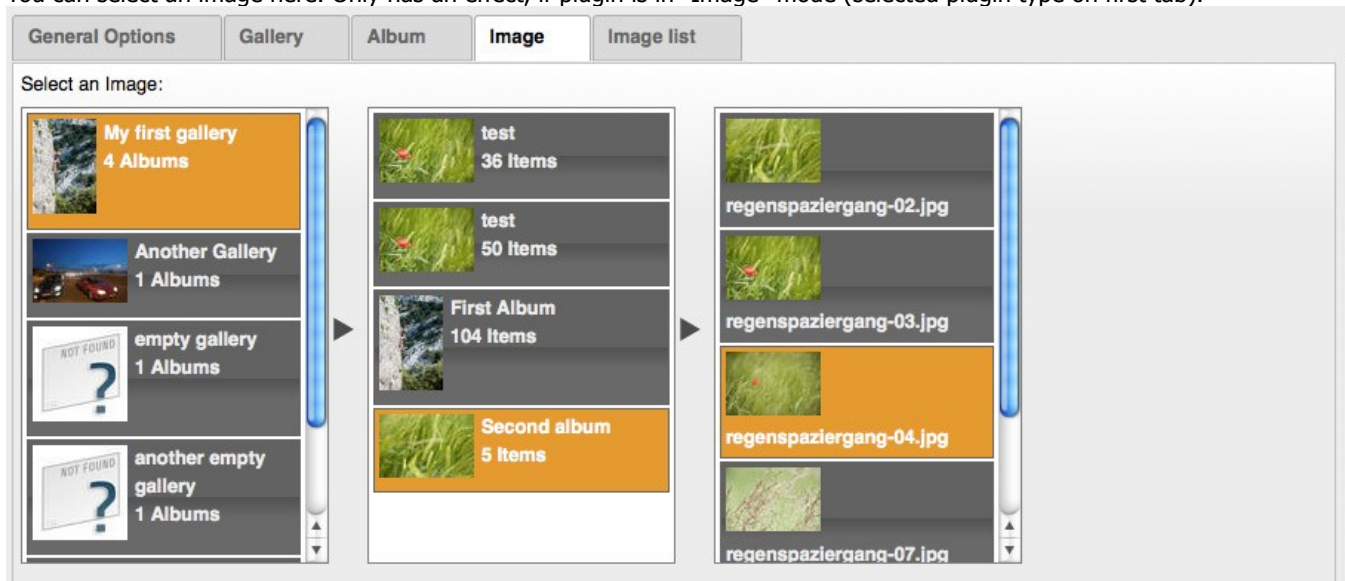
Configuration

As most TYPO3 extensions, YAG ships with a lot of TypoScript settings to be made. To get a first impression of what is configured via TypoScript, open up a page on which YAG static template is included using the Template module and open 'TypoScript Object Browser':

The following reference will give you a in depth explanation of all TypoScript settings used in YAG.

TypoScript Reference

plugin.tx_yag.settings

This is the main section of our extension. All non-framework-specific configuration goes here.

Property:	Data type:	Description:	Default:
rbacSettings	See RBAC reference	Holds actions, objects, domains and role definitions for Role-Based Access Controll (RBAC). **These settings ship with YAG's static template and should not be changed, as they have no effect on configuration once the extension is installed.** Use RBAC administrator to set up further RBAC settings.	
crawler	array	Settings for the YAG file crawler used for directory import.	

accessDenied	array	Holds a controller / action pair, that defines which controller and action is called whenever access is denied for an action. Example:
		
		`accessDenied {`
		`    controller = Gallery`
		`    action = list`
		`}`
		
		This will show list action of Gallery controller, whenever access is denied.	

sysImages	array	Holds an array of paths for different images used throughout the extension.	
themes	array	Holds an array of themes.	
extlist	array	Holds settings for pt_extlist extension. Take a look at the pt_extlist documentation for further information.	

plugin.tx_yag.settings.crawler

Configuration for file crawler

Property:	Data type:	Description:	Default:
fileTypes	string	Comma-separated list of file-endings that should be indexed by crawler. Example: `jpg,jpeg`	jpg,jpeg

plugin.tx_yag.settings.sysImages

Configuration for all kinds os images used for skinning.

Property:	Data type:	Description:	Default:

imageNotFound	Item file description	Configures a path, title and description for an item.
		
		Example:
		
		<pre>sysImages {
		     imageNotFound {
		          sourceUri =
		typo3conf/ext/yag/Resources/Public/Icons/imageNotFound
		.jpg
		          title = No image found.
		          description = No image found.
		     }
		}</pre>
		
		Mind that the sourceUri of the image must be relative to TYPO3 root.	

plugin.tx_yag.settings.themes

Most of the configuration for YAG can be found in themes. We have a default theme, where you can find all the settings available in YAG. See section 'Themes and Templates' in the Developers' chapter for further information on how to extend themes and write your own themes.

Property:	Data type:	Description:	Default:
[your_theme_name]	array	You can define your own themes here. YAG ships with a default theme and a backend theme.	

plugin.tx_yag.settings.themes.default

In this section, you can find the settings for the default theme which acts as basis for all other themes. Best practice for developin your own themes is to extend this theme with your own theme like that:

```
plugin.tx_yag.themes.[your_theme_name] < plugin.tx_yag.themes.default
plugin.tx_yag.themes.[your_theme_name] {
   # … your theme specific settings
}
```

Property:	Data type:	Description:	Default:
showBreadcrumbs	bool	If set to 1, breadcrumbs are shown as navigation.	1

Property:	Data type:	Description:	Default:

resolutionConfigs	array	Configuration for image resolutions. You can define the resolutions of thumbnails, single images etc. here.
		
		``
		resolutionConfigs {
		    thumb {
		        width = 150c
		        height = 150c
		    }
		
		    medium {
		        maxW = 800
		        maxH = 600
		    }
		}
		``
		
		In the default theme, thumb for thumbnails and medium for medium sized images in single view are defined and used. For your own template, you can define any kind of resolutions with the name of your choice.
		
		A resolution configuration can consist of any parameter that the TYPO3 IMAGE type provides, including image manipulation via GIFBUILDER.	

gallery	array	Gallery specific settings of your theme. See section below	
album	array	Album specific settings of your gallery. See section below	
extlist	array	This section configures pt_extlist specific settings for YAG. See pt_extlist documentaiton for further information.	
itemList	array	This section configures the list of images shown, when you click on an album. See section below for further information.	
item	array	This section configures single view of an item. See section below for further information.	
includeLibJS	CSV	Comma-separated list of defined librarys from wich you want to include javascript files. Defined libraries are jQuery, jQueryUi, jQueryShadowBox	
includeLibCSS	CSV	Comma-separated list of defined librarys from wich you want to include CSS files. Defined libraries are jQuery, jQueryUi, jQueryShadowBox	
includeJS	arary	Define JS files which should be included in the page header. Same schema as in page.	
includeJS	array	Define CSS files which should be included in the page header. Same schema as in page.	

plugin.tx_yag.settings.themes.galleryList

Gallery specific settings of your theme.

Property:	Data type:	Description:	Default:
columnCount	int	Number of columns used for rendering gallery overview.	2
GalleryThumbPartial	String	Pathand filename of the gallery thumb partial.	Gallery/GalleryThumb.html

plugin.tx_yag.settings.themes.albumList

Album specific settings of your theme.

Property:	Data type:	Description:	Default:
itemsPerPage	int	Number of albums shown on album list	12
showBreadcrumbs	bool	If set to 1, breadcrumbs are shown on album page.	1

Property:	Data type:	Description:	Default:
columnCount	int	Number of columns used for rendering album list.	2
AlbumThumbPartial	String	Pathand filename of the album thumb partial.	Album/AlbumThumb.html

plugin.tx_yag.settings.themes.extlist

pt_extlist specific settings of your theme. See pt_extlist documentation for further information.

plugin.tx_yag.settings.themes.itemList

Configuration of image list of your theme.

Property:	Data type:	Description:	Default:
itemsPerPage	int	Number of images shown on a single page.	12
columnCount	int	Number of columns used to render images on image list.	4
showTitle	bool	If set to 1, album title is shown on overview page.	1

imageThumbPartial	path	Path to partial used to render an image in image list. This can be Extbase path (relative to EXT:yag/Resources/Private/Partials):
		
		`Image/ImageThumb`
		
		or common TS resource path to set offer paths:
		
		`EXT:yag/Resources/Private/Partials/Image/ImageThumb.html`
			`Image/ImageThumb`

imageAdminThumbPartial	path	Not used at the moment.	

plugin.tx_yag.settings.themes.item

Configuration of image single view of your theme.

Property:	Data type:	Description:	Default:
showItemMeta	bool	If set to 1, metadata of image will be shown in single view.	1

itemMetaPartial	path	Path to partial used to render image meta data (EXIF etc.). This can be Extbase path (relative to EXT:yag/Resources/Private/Partials):
		
		`Image/ImageMeta`
		
		or common TS resource path to set offer paths:
		
		`EXT:yag/Resources/Private/Partials/Image/ImageMeta.html`
			`Image/ImageMeta`

module.tx_yag.settings

Holds settings for the backend of YAG. The content of this setting is the same as plugin.tx_yag.settings.

Tutorial

How to create your own Themes as a third-party extension

Although you can create your themes with TypoScript and some Fluid templates, we will show here how to create a theme as a third party extension. We use Extbase's Kickstarter to create a basic extension and then show how to bring up all the rest.

1. Creating a basic extension using FLUID kickstarter

Start Extbase kickstarter:
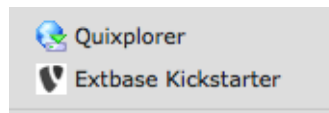
Open the properties pane and give your theme a descriptive name and some more information:

Save your extension. If everything worked fine, you should get the following message:
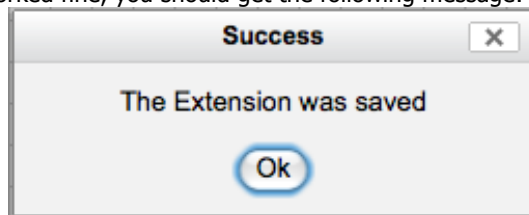
2. Open you extension in some Editor / Create a new project

Open your extension in any editor you like – e.g. ZendStudio and create a new project. First of all, you can delete some files and folders, as we won't need them:

- /ext_localconf.php (we won't have any plugins configured and won't have any tables or TCA)

- /ext_tables.sql (as we won't have any tables, we don't need this file)

- /kickstarter.json (there is no need in keeping this file)

- /Resources/Private/Language/locallang_db.xml (we won't have any DB changes or fields)

- /Configuration/TCA (we don't need no TCA changes)

So after that, your folder structure looks something like that:

```
▼ 🔲 (centos) yag theme simple blog
    ▶ 📁 .settings
    ▼ 📁 Configuration
        ▼ 📁 TypoScript
            📄 constants.txt
            📄 setup.txt
    ▼ 📁 Resources
        ▼ 📁 Private
            ▼ 📁 Language
                🗴 locallang.xml
            📄 .htaccess
        ▼ 📁 Public
            ▶ 📁 Icons
    🗴 .project
    📄 ext_emconf.php
    📄 ext_icon.gif
    📄 ext_tables.php
```
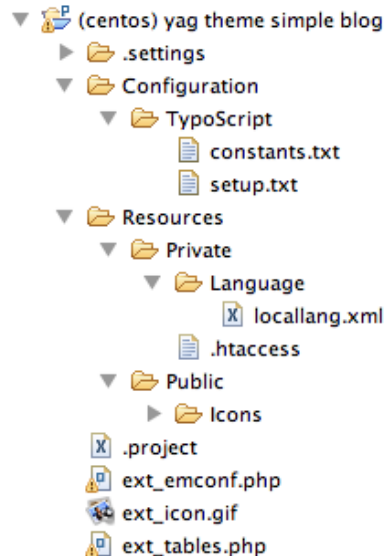
3. Modify ext_tables.php

We will later include a static template holding the configuration for our theme. This has to be included via ext_tables.php:

```php
<?php

if (!defined ('TYPO3_MODE')) die ('Access denied.');


// Include static template for SimpleBlog Theme

t3lib_extMgm::addStaticFile($_EXTKEY, 'Configuration/TypoScript', '[yag] Theme SimpleBlog');

?>
```

Save your ext_tables.php and you're done with it.

4. Set up basic TypoScript Template

We need at least one TS-Template for registering our theme and copying all the basic settings from the default theme. Copying the settings from default prevents you from writing all configuration again. So first you copy everything and then you overwrite the settings you would like to change. Here is our first draft of the /Configuration/TypoScript/setup.txt:

```
##################################################
# YAG theme SimpleBlog
#
# @author Daniel Lienert <daniel@lienert.cc>
# @author Michael Knoll <knoll@punkt.de>
##################################################

# Copy default settings from default theme
plugin.tx_yag.settings.themes.simpleBlog < plugin.tx_yag.settings.themes.default
```

As long as we do not want to change any settings, that's all we have to do. We will come back to this file later.

5. Hands on FLUID Templates

Most of the look and feel of our themes is set via FLUID html templates. So let's create some. The only template we need for

our theme is a simple image list, that shows the images from a single album.

So here's, what you have to do step-by-step:

1. Create a new folder in /Resources/Private and call it 'Templates'

2. Create a new folder in /Resources/Private/Templates and call it 'ItemList'

3. Create a new html file 'List.html' inside this folder (/Resources/Private/Templates/ItemList/List.html)

4. Remove all content from this html file your editor probably auto-generated

5. Open up your setup.txt again and add the following lines to overwrite the template used for the ItemListController and list action:

```
plugin.tx_yag.settings.themes.simpleBlog {
   controller {
      ItemList {
         list.template = EXT:yag_theme_simpleblog/Resources/Private/Templates/ItemList/List.html
      }
   }
}
```

6. Write some template code inside '/Resources/Private/Templates/ItemList/List.html':

```
{namespace yag=Tx_Yag_ViewHelpers}

<div class="tx-yag-theme-simpleblog-imagecontainer">
   <f:for each="{listData}" key="rowIndex" as="listRow">
      <f:render partial="{config.itemListConfig.imageThumbPartial}"
         arguments="{config: config, image: listRow.image.value,
            rowIndex: listRow.specialValues.absoluteRowIndex,
            pager: pager, pagerCollection:pagerCollection}" />
   </f:for>
</div>
```

The first line gives us access to some YAG viewhelpers we probably need later on. The rest is quite straight-forward: We open up a div container for all images. The we iterate through the rows we get from pt_extlist as listData object. **Each row stands for a single image in our album**. Do not confuse row here with the visualized rows containing some images in a row. A row here is a record in a database table containing one record, which is an image here!

We then use a partial to render the thumbnail for an image. This partial will be created in the next step.

One line that show get your attention is '{config.itemListConfig.imageThumbPartial}'. 'config' holds a so called ConfigurationBuilder which gives you access to almost all the configuration of YAG within your templates and partials. We will come back to this later.

7. You are done with your template. Save it!

6. Hands on FLUID Partials

Although it is not really needed here to use partial, we will use them for practice. Again step-by-step what you have to do:

1. Create a new folder in '/Resources/Private' and call it 'Partials'

2. Create a new file in your new folder and call it 'ImageThumb.html'

3. Open this file and remove any content eventually created

4. Open your setup.txt again and add the following lines to make the partial known to yag:

```
plugin.tx_yag.settings.themes.simpleBlog {
   controller {
      ItemList {
         list.template = EXT:yag_theme_simpleblog/Resources/Private/Templates/ItemList/List.html
      }
   }

   itemList {
      imageThumbPartial = EXT:yag_theme_simpleblog/Resources/Private/Partials/ImageThumb.html
   }
}
```

5. Write some code to make the partial work (for your own themes, you can copy and paste the content of the partial in YAG's default theme for a start):

```
{namespace yag=Tx_Yag_ViewHelpers}

<f:if condition="{image.width} > {image.height}">
    <div class="tx-yag-theme-simpleblog-thumb tx-yag-theme-simpleblog-thumb-landscape">
</f:if>
<f:if condition="{image.width} < {image.height}">
    <div class="tx-yag-theme-simpleblog-thumb tx-yag-theme-simpleblog-thumb-portrait">
</f:if>
<f:if condition="{image.width} == {image.height}">
    <div class="tx-yag-theme-simpleblog-thumb tx-yag-theme-simpleblog-thumb-square">
</f:if>
    <a href="{yag:resource.image(item: image, resolutionName: 'lightbox')}"
       rel="shadowbox[images_{image.album.uid}]" title="{image.title}">
        <yag:image item="{image}" resolutionName="thumb" alt="{image.title}"/>
    </a>
    <ul>
        <li class="tx-yag-theme-simpleblog-thumb-title">{image.title}</li>
    </ul>
</div>
```

So this one's a little trickier. The outer div's class changes, depending on whether the image is a landscape image or a portrait image. Therefore we use an inline if-viewhelper.

For a single click, we use a link to a down-sized image whose URL we get using imageLink viewhelper. We later have to set up a resolution configuration with the name 'lightbox'.

6.   Add the required resolutions to setup.txt:

```
plugin.tx_yag.settings.themes.simpleBlog {
    resolutionConfigs {
        thumb {
            width = 150
            height =
            quality =
        }
        lightbox {
            width = 1200
            height = 800
        }
    }

    controller {
        ItemList {
            list.template = EXT:yag_theme_simpleblog/Resources/Private/Templates/ItemList/List.html
        }
    }

    itemList {
        imageThumbPartial = EXT:yag_theme_simpleblog/Resources/Private/Partials/ImageThumb.html
    }
}
```

7.   Save your partial.

7. Add some CSS

The last step in creating your theme is CSS styling. We therefore create a new folder '/Resources/Public/CSS' and in there a new file 'styles.css'.

Here is our CSS:

```
@CHARSET "UTF-8";

div.tx-yag-theme-simpleblog-imagecontainer:after {
    content: ".";
    display: block;
    height: 0;
    clear: both;
    visibility: hidden;
}

div.tx-yag-theme-simpleblog-thumb {
    float: left;
    height: 200px;
    width: 200px;
```

```
    border: 1px solid;
    border-color: #AAA #444 #444 #AAA;
    margin: 10px 10px;
    padding: 24px;
    -moz-border-radius: 3px 3px 3px 3px;
    -moz-box-shadow: 3px 3px 4px #aaa;
    -webkit-box-shadow: 3px 3px 4px #aaa;
    box-shadow: 3px 3px 4px #aaa;
    /* For IE 8 */
    -ms-filter: "progid:DXImageTransform.Microsoft.Shadow(Strength=4, Direction=135, Color='#aaaaaa')";
    /* For IE 5.5 - 7 */
    filter: progid:DXImageTransform.Microsoft.Shadow(Strength=4, Direction=135, Color='#aaaaaa');
}

div.tx-yag-theme-simpleblog-thumb ul {
    display: none;
}

div.tx-yag-theme-simpleblog-thumb img {
    border: 1px solid;
    border-color: #444 #AAA #AAA #444;
}

div.tx-yag-theme-simpleblog-thumb-landscape img {
    height: 133px;
    width: 200px;
    margin: 33px 0;
}

div.tx-yag-theme-simpleblog-thumb-portrait img {
    height: 200px;
    width: 133px;
    margin: 0 33px;
}
```

There is one thing left to do: Include CSS file via TypoScript so for a last time, open up setup.txt and bring it to its final version:

```
# Include CSS for this theme
page.includeCSS.yag_theme_simpleBlog = EXT:yag_theme_simpleblog/Resources/Public/CSS/styles.css

# Copy default settings from default theme
plugin.tx_yag.settings.themes.simpleBlog < plugin.tx_yag.settings.themes.default

# Some theme-specific settings
plugin.tx_yag.settings.themes.simpleBlog {

    resolutionConfigs {

        thumb {
            maxH = 200
            maxW = 200
        }

        lightbox {
            maxH = 1200
            maxW = 800
        }

    }

    controller {

        ItemList {
            list.template = EXT:yag_theme_simpleblog/Resources/Private/Templates/ItemList/List.html
        }

    }

    itemList {
        imageThumbPartial = EXT:yag_theme_simpleblog/Resources/Private/Partials/ImageThumb.html
    }

}
```

8. Installing your theme

You can now install your theme using the Extension Manager. Chose 'Install Extensions' from the dropdownlist and click on

the gray brick to install your new theme.

After installing the extension, you can include the static template on the page you want to use the new theme. If you haven't yet done so, create a new extension template and add the static template you just created.
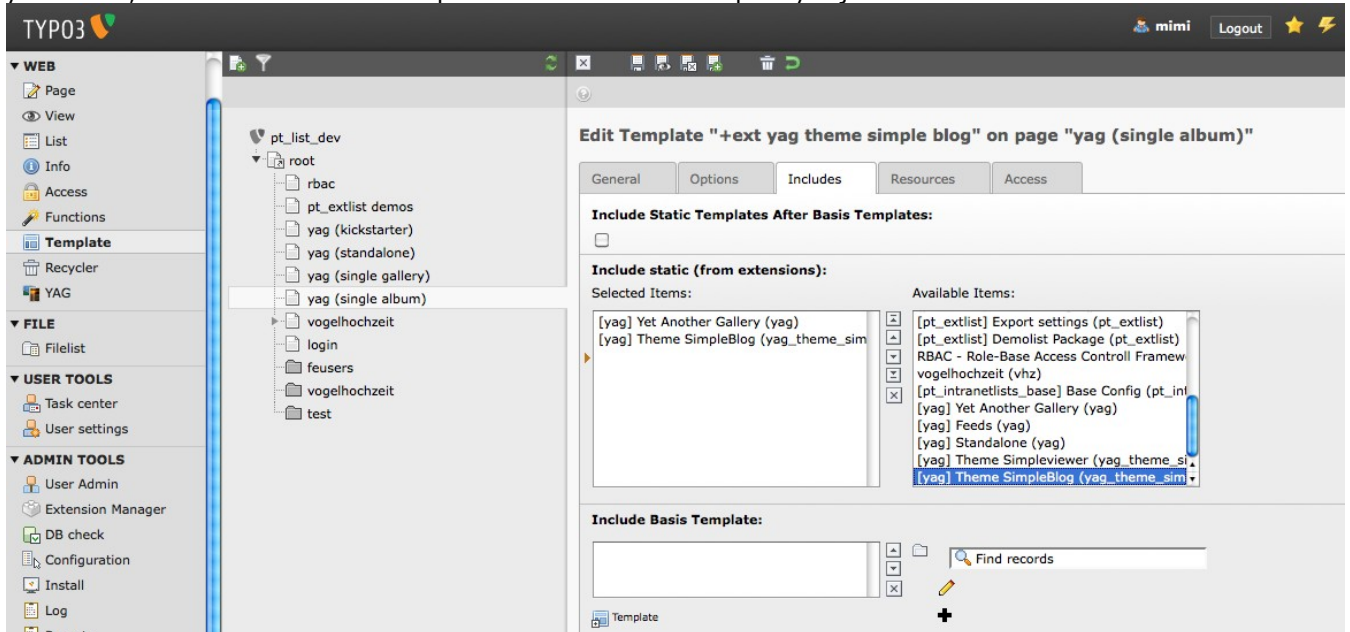
Now you can chose the theme from within your plugin's FlexForm on the page you want to include the album:
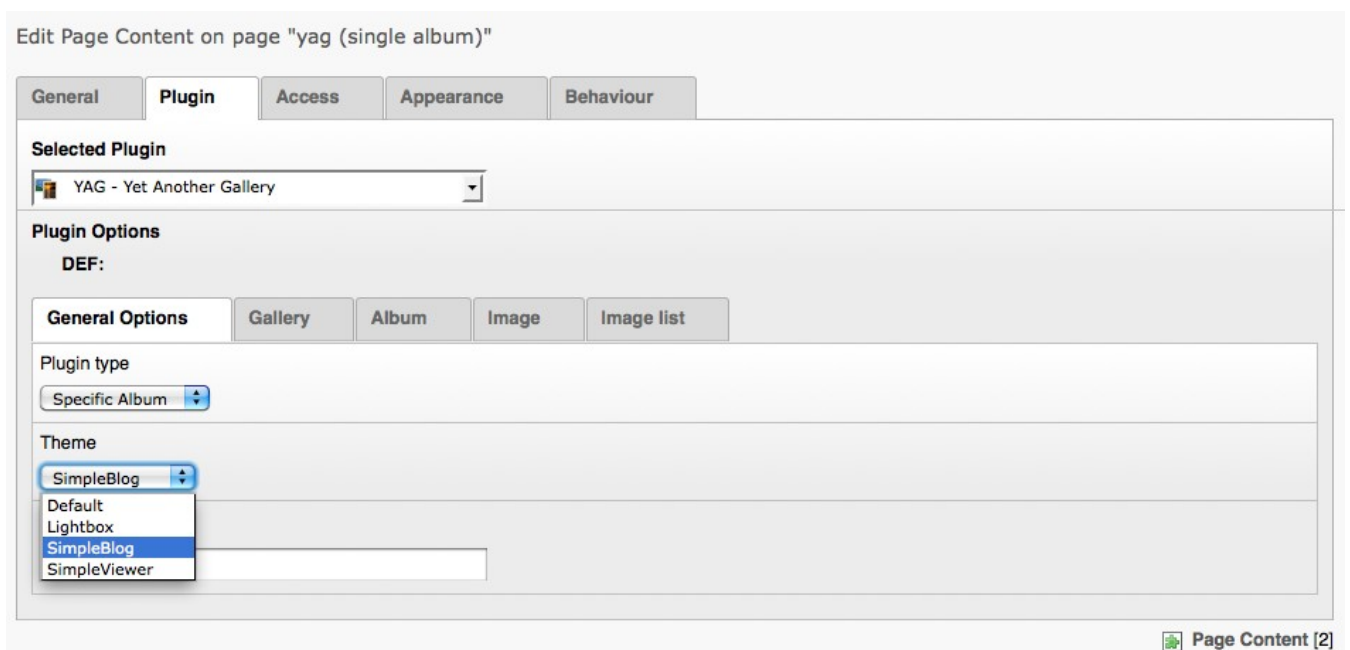
And basically that's it :-) You have just created your new theme. Depending on your settings, it could look something like this:

Developers Guide

We implemented YAG so that it is easily extendable by developers. Here are some basic things you need to know for writing your own extensions for YAG.

YAG Architecture

Usage of other extensions

We mainly use 3 dependent extensions:

- **Extbase** as basic TYPO3 Extension Framework

- **pt_extlist** for rendering all sorts of lists like lists of galleries, albums or images. pt_extlist does all the stuff like paging, filtering and sorting for us.

- **rbac** for role- and access management. rbac implements a role-based access controll system that we use for controlling which user has access to which actions.

Domain Model

The domain model of YAG is quite simple:
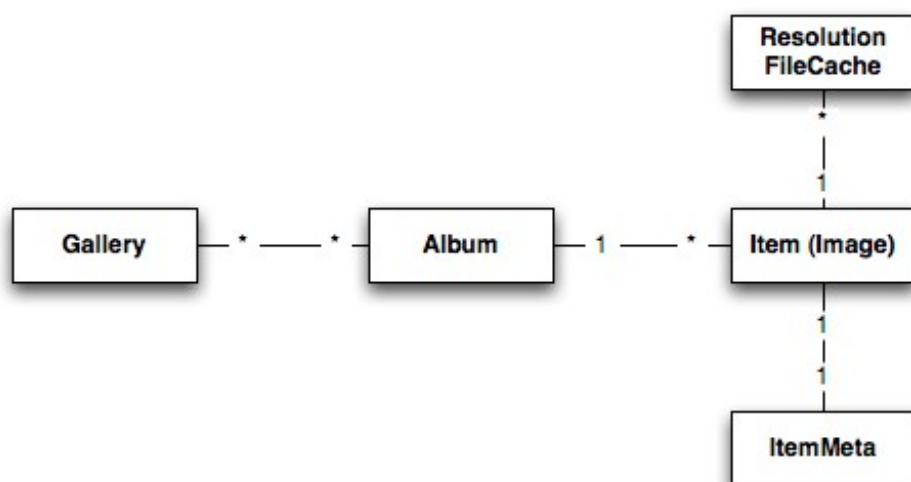
We have the following domain objects:

- **Gallery** acts as a 'container' for everything. Inside a gallery, you can have multiple albums. Mind that an album can also belong to different galleries, what makes things a little more complicated...

- **Album** holds a set of images, which we call items in YAG (we call it items, because that could be other things except images, like videos etc.).

- **Item (Image)** holds all information of an image like its source path, filename and other data.

- **ItemMeta** holds meta-data information for an item. E.g. EXIF or IPTC.

- **ResolutionFileCache** holds caching information

We refer to the domain model of YAG as all objects that directly have to do with the domain of our extension (remember – it's a gallery) and have to be persisted in the database.

Further Domain Objects

Besides the model there are other domain objects used to handle gallery related stuff. Those objects are not persisted and are therefore not part of what we called the domain model above:

- **Configuration** keeps a set of configuration objects that implement an object oriented way to handle TypoScript configuration and validation.

- **Context** context is a container of other objects used to make it easier to set and access certain information. E.g. getting all images stored in an album is implemented via a list of images that is filtered by an album uid. Setting the album for the filter and accessing it afterwards is made quite simple using the context.

- **ImageProcessing** Where there are images, there has to be done different kinds of processing. E.g. resizing of images is done by the objects kept in this part of the domain.

- **Import** holds all objects and classes required for importing images to YAG.

Controllers

As Extbase is implemented using the MVC paradigma, we have some controllers handling all the requests coming from Browsers or other applications from outside TYPO3.

Our controllers each extend an abstract controller that holds some functionality we require for YAG. This abstract controller extends Extbase's ActionController:
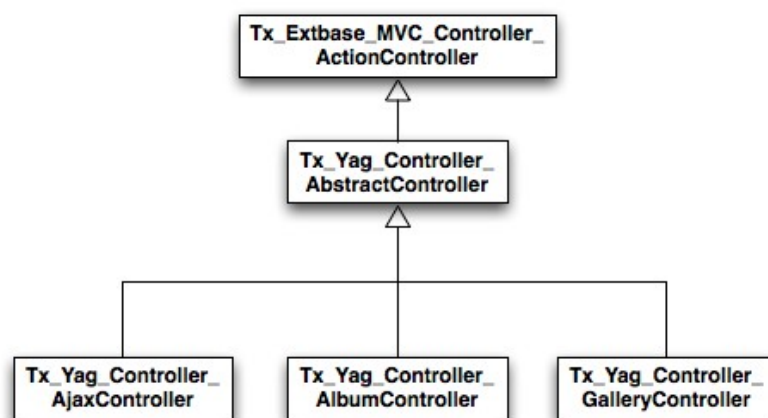
For a detailed explanation of how controllers and actions are handled within TYPO3 and Extbase, refer to the Extbase documentation.

Filesystem Structure

The filesystem structure follows Extbase conventions. Each directory starts with an uppercase letter.

📁 Classes   ▶
📁 Configuration   ▶
📁 doc   ▶
📁 Documentation   ▶
⊙ ext_autoload.php
📄 ext_conf_template.txt
⊙ ext_emconf.php
🔳 ext_icon.gif
⊙ ext_localconf.php
⊙ ext_tables.php
📄 ext_tables.sql
⬛ kickstarter.json
📄 README
📁 Resources   ▶
📁 Tests   ▶
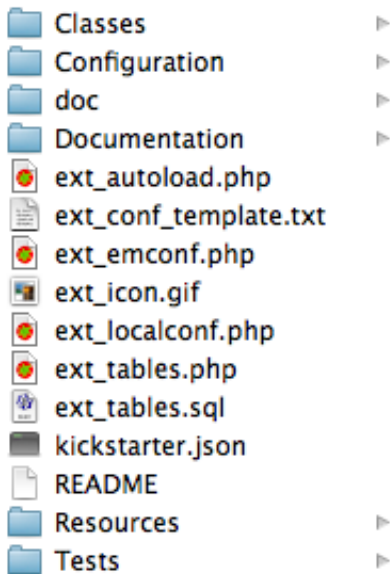
Image Storage and Resolution File Cache

A problem for every gallery application is the storage of original and resized images.

Storage of original images

Each image in YAG is stored as original image. This image is never changed by YAG. You find those images in the folder you set in the Extension Manager as 'origFilesRoot':

Path of directory where YAG stores ... [origFilesRoot]
Path of directory where YAG stores all original files.

    fileadmin/yag

Whenever you use an original image in an album, an item object is created. This item object has a UID and an albumUid as well as a sourceUri pointing to the file inside the origFilesRoot where the original file is stored. For each album there is an individual folder in the origFilesRoot named like the UID of the album. Each item (image) is stored, using its UID as filename:
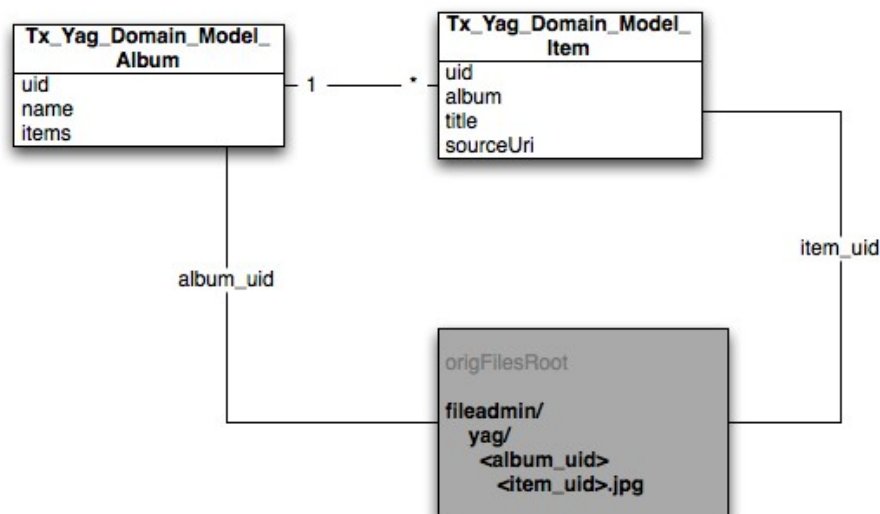
Storage of resized (cached) images

Whenever you use an item in your albums, YAG will generate resized versions of this item according to the resolution settings in your theme. So for each item and for each resolution there will be a cached version on your server. YAG stores those images in a chaos-filesystem. Chaotic means, that YAG will automatically create a folder structure so that no more than a hundred files are located in one folder. Many files in the same folder will make the filesystem slow.

So here is how it works:

- Each item holds a **Tx_Yag_Domain_Model_ResolutionFileCache** object for each resolution set in TypoScript.

- A cached image is written to the server for each resolution. A hashed string is used for the filename to prevent traversing access to the files.

- The resolution file cache object holds the resolution and the path to the cached file on the server.

The root folder of the hash filesystem root is set in your Extension Manager:

Path of directory where YAG should ... [hashFilesystemRoot]
Path of directory where YAG should store all cached images generated for albums - relative to Typo3 base path. This path should not be changed unless you know what you do! (Image base path)

typo3temp/yag

The following diagram shows how the resolution file cache works:
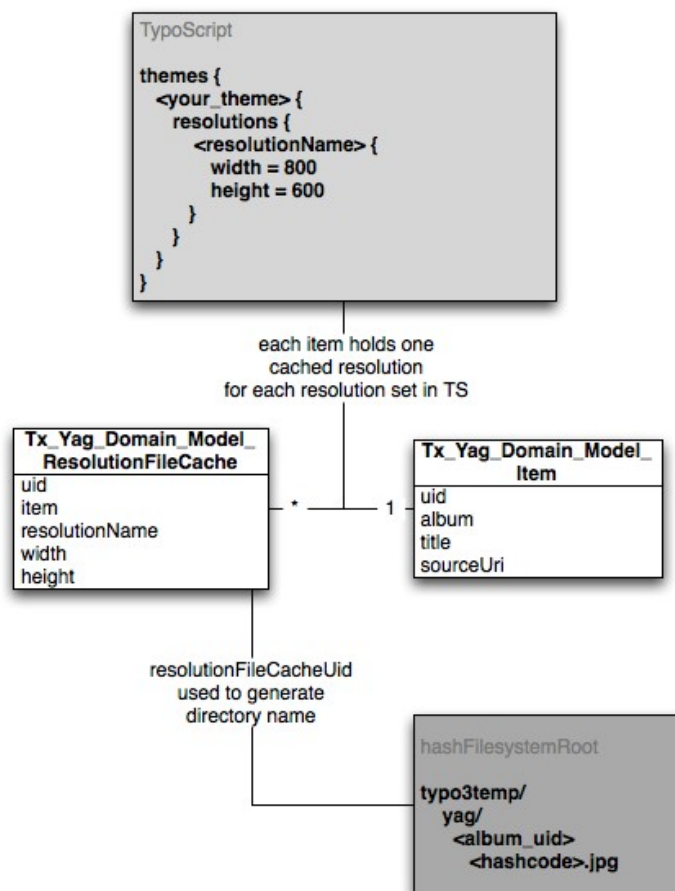
Themes and Templates

Themes are used to configure almost everything in YAG. This enables you to change almost everything using your own themes. The theme selector in your FlexForm lets you chose the theme you want to use for showing your gallery widgets.

What is included by a theme

A theme mainly consists of some TypoScript configuration and a set of FLUID templates. There also might be some FLUID partials and third party JavaScript libraries and CSS files. Take a look at the filestructure to get an impression of what are the contents of a theme:

```
▼ 🗂 (centos) yag theme simpleviewer
    ▶ 📂 .settings
    ▼ 📂 Configuration
        ▼ 📂 TypoScript
            ▼ 📂 Themes
                ▼ 📂 SimpleViewer
                    🔵 General.ts
                    🔵 ItemList.ts
                    🔵 Theme.ts
            📄 setup.txt
    ▼ 📂 Resources
        ▼ 📂 Private
            ▼ 📂 Language
                Ⓧ locallang_db.xml
                Ⓧ locallang.xml
            ▼ 📂 Templates
                ▼ 📂 ItemList
                    📄 List.html
                    📄 XmlList.html
        ▼ 📂 Public
            ▼ 📂 SimpleViewer
                ▶ 📂 css
                ▶ 📂 img
                ▶ 📂 js
                ▶ 📂 swf
                📄 full.html
    Ⓧ .project
    📄 ext_emconf.php
    🖼 ext_icon.gif
    📄 ext_localconf.php
    📄 ext_tables.php
    📄 ext_tables.sql
    📄 kickstarter.json
```

How to overwrite templates used for Controller/Action pairs

We extended Extbase's default ActionController in such a way, that you can overwrite each template for each controller / action pair using TypoScript. Let's say, you have a controller called 'ItemList' and an action 'list'. Then you can overwrite the template used for this controller/action pair using the following TypoScript configuration:

```
##Overwriting template path for ItemList->listAction()
plugin.tx_yag.settings.controller.ItemList.list.template = EXT:<ext_key>/<path_to_template>

## For example:
plugin.tx_yag.settings.controller.ItemList.list.template = EXT:yag/Resources/Private/Template/ItemList/
    list.html
```

This feature enables you to set templtes for YAG controllers to html files outside the extension itself, which is currently not possible using Extbase.

So here is another example on how to set template paths for a theme to template files that ship with the theme's extension:

```
plugin.tx_yag.settings.themes.simpleViewer {
    controller {
        ItemList {
            list.template = EXT:yag_theme_simpleviewer/Resources/Private/Templates/ItemList/List.html
            xmlList.template = EXT:yag_theme_simpleviewer/Resources/Private/Templates/ItemList/XmlList.html
        }
    }
}
```

You can see, that you can set the template paths to files included by your theme-extension and not by YAG.

Which objects are available in your templates

After you know how to set templates for a controller / action pair, the next question should be how you can access data from within your templates. Therefore you have to know, how data is assigned to your template. The easiest way to find this out, is to take a look in your controller's code:

```
public function newAction(
        Tx_Yag_Domain_Model_Gallery $gallery=NULL,
        Tx_Yag_Domain_Model_Album $newAlbum=NULL) {

    $selectableGalleries = $this->objectManager->get(
        'Tx_Yag_Domain_Repository_GalleryRepository')->findAll();

    $this->view->assign('selectableGalleries', $selectableGalleries);
    $this->view->assign('selectedGallery', $gallery);
    $this->view->assign('newAlbum', $newAlbum);
}
```

So here you can see, that there are three objects passed to your template: 'selectableGalleries', 'selectedGallery' and 'newAlbum'. You can access those objects using FLUID's mechanism of accessing properties:

```
<f:layout name="Default" />
<f:section name="main">
<h1><f:translate key="tx_yag_controller_album_new.header" /></h1>

            <f:render partial="FormErrors" arguments="{for: 'newGallery'}" />
            <f:form method="post" controller="Album" action="create" name="newAlbum"
             object="{newAlbum}">
                <f:render partial="Album/FormFields" arguments="{album:album,
                 selectableGalleries:selectableGalleries, selectedGallery:selectedGallery}" />
                <f:form.submit class="submit" value="{f:translate(key: 'general.save'
                 default:'Save')}"/>
            </f:form>

            <f:link.action controller="Gallery" action="index">
            <f:translate key="tx_yag_controller_gallery.backToGallery" />
    </f:link.action>
</f:section>
```

The YAG context object

Whenever YAG is used on a page (whether as content-element or in standalone mode), there are many dependencies to be resolved and a lot of environment to be set up. To make it easier for a developer to handle all this stuff, we implemented what we call a **context**. A context is a container giving you a nice way to set and get information gathered within a lifecycle of YAG. This might be configuration as well as currently set parameters. To get an impression of what is stored within the context container, here is a little diagram again:

TODO: finish me!

YAG ViewHelpers

Here is a list of viewhelpers available from within YAG:

```
▼ 📂 ViewHelpers
    ▼ 📂 CSS
        📄 IncludeViewHelper.php
    ▼ 📂 Form
        📄 ErrorsViewHelper.php
    ▼ 📂 Javascript
        📄 IncludeViewHelper.php
        📄 TemplateViewHelper.php
    ▼ 📂 Link
        📄 AlbumAdminViewHelper.php
        📄 AlbumViewHelper.php
        📄 GalleryViewHelper.php
    ▼ 📂 Namespace
        📄 GPArrayViewHelper.php
    ▼ 📂 Resource
        📄 ImageViewHelper.php
    ▼ 📂 Widget
        ▼ 📂 Controller
            📄 BreadcrumbsController.php
        📄 BreadcrumbsViewHelper.php
        📄 ImageViewHelper.php
```

If you want to use YAG's viewhelpers in your templates, you have to include them with the following line of code at the beginning of your template:

```
{namespace yag=Tx_Yag_ViewHelpers}
```

ViewHelper:	Parameters:	Description:
BreadcrumbsViewHelper	none	Renders a root path menue from gallery to image. Travel & Passion » Sardinia » sardinien-015.jpg Example: `<yag:breadcrumbs />`
ImageViewHelper	Item: item object resolutionName: The name of a defined resolution.	Renders an image in the given resolution. Example: `<yag:image item="{image}" resolutionName="thumb" />`

CSS

The followin viewhelpers are available for handling CSS related stuff:

ViewHelper:	Parameters:	Description:
IncludeViewHelper	Library: library name File: path to file	Includes CSS Files to the header section. If library is given, the css files defined in the library are included (see: Typoscript/BaseConfig/HeaderInclusion/) Example: `<yag:CSS.Include library="jQueryShadowBox" />`

Javascript

The followin viewhelpers are available for handling Javascript related stuff:

ViewHelper:	Parameters:	Description:

IncludeViewHelper	Library: library name
	File: path to file
		Includes JS Files to the header section. If library is given, the js files defined in the library are included (see: Typoscript/BaseConfig/HeaderInclusion/)
		
		Example:
		
		`<yag:Javascript.Include library="jQuery" />`
		
		In order to make this work, you have to configure your libraries in TypoScript. You find a list of predefined libraries in Configuration/TypoScript/BaseConfig/HeaderInclusion/JQuery.ts:
		
		`plugin.tx_yag.settings.frontendLib {`
		`    jQuery {`
		`        include = {$config.yag.addjQuery}`
		`        includeJS.jQuery =`
		`EXT:yag/Resources/Public/Js/JQuery/jquery-1.5.1.min.js`
		`        # includeCSS.jQuery =`
		`EXT:yag/Resources/Public/CSS/JQuery/base.css`
		`    }`
		`}`

TemplateViewHelper	TemplatePath: path to a jsTemplate
	Arguments: the arguments to replace in the template.
		This viewhelper is in some way a pragmatic approach to avoid the fluid restrictions with javscript inline markup in templates. All arguments given to the viewhelper are replaced in the Javascript template in the form ###argument### with the given value.
		
		There are some implicit defined markers:
		extPath: relative path to the extension
		extKey: Extension Key
		pluginNamespace: Plugin Namespace for GET/POST parameters
		
		Example (usage of viewhelper):
		`<yag:Javascript.Template`
		`templatePath="EXT:yag/Resources/Private/JSTemplates/ItemAdminList.js" arguments="{ajaxBaseURL :`
		`'{f:uri.action(controller:\'Ajax\')}'}" />`
		
		Example (usage of template markers in JS templates – so it's JavaScript what you see here):
		`var del_url = '###ajaxBaseURL###' +`
		`'&###pluginNamespace###[action]=deleteItem';`

Link

The following viewhelpers are available for rendering links:

ViewHelper:	Parameters:	Description:
AlbumViewHelper	Album: album object	Renders a link for an album
AlbumAdminViewHelper	Album: album object	Renders a link for administrating an album
GalleryViewHelper	Gallery: gallery object	Renders a link for a gallery

Namespace

The followin viewhelpers are available for using namespaces:

ViewHelper:	Parameters:	Description:
GPArrayViewHelper	###TODO daniel###	###TODO daniel###

Resource

The followin viewhelpers are available for getting URIs for resources:

ViewHelper:	Parameters:	Description:
ImageViewHelper	Item: item object resolutionName: The name of a defined resolution.	Renders URI for an image. Used in XML view for example. Example: `<yag:resource.image item="{listRow.image.value}" resolutionName="thumb" />`

Writing your own importers

TODO: finish me!

Using remote services

TODO: finish me!

Usage of pt_extlist

TODO: finish me!

Our extensions of Extbase

TODO: finish me!

Known problems

Although we tried to make the first version of YAG as stable as possible, there are still some issues left. For getting a complete overview of what's missing, take a look at:

http://forge.typo3.org/projects/extension-yag/issues

Open Issues for version 1.x

- Usage of storage PID: At the moment, YAG does not respect a storage pid, when adding records to database. That means that all records are stored with PID 0 and all records can be accessed from everywhere. This doesn't make YAG multi-user usable in backend, if you want to hide certain records from certain persons.

- Ajax-updating problems: There are some widgets like pagers in the backend, that are not updated when an Ajax request is handled. For example the number of items in an album is not updated, if an album gets deleted.

- Frontend-Editing: Almost everything required for editing the whole gallery in the frontend is implemented, but we didn't have the time to write all the JavaScript and templates for version 1.0. So this will come in a future-version.

- Role-Management: Although we have a role-management implemented in YAG, there is no administration view to set up users etc. that are equipped with roles. This will come with Frontend-Editing in a future version.

- RealUrl configuration: We are working on this. It will be released with version 1.2.0

- Categories: We are working on this. It will be released with version 1.2.0

To-Do list

Find our To-Do list in forge.typo3.org:

http://forge.typo3.org/projects/extension-yag/issues

Feel free to add your bugs and wishes!

ChangeLog

For detailed change logs, visit https://github.com/michaelknoll/yag/commits/master

Version	Changes:
1.1.6	FIX: Bug #27172 – Umlaute are now correctly displayed in Front- and Backend.
1.1.5	FIX: Bug #26740 – Insert plugin in backend crashes under some circumstances. FIX: Bug #26111 - Fileadmin importer is not able to import folders with blanks
1.1.4	DEL: Removed RBAC installation routine FIX: Added some escaping for title and description RFT: Added some frontend styling CHG: Added .jpeg, .JPG and .JPEG as possible file endings for importers RFT: Removed unused gallery:album mm table from SQL definition FIX: Some minor bugfixes
1.1.3	CHG: Improvements in performance. Tested handling of up to 50k images. Seems to be quite fast now :-) CHG: Directory importer comes with directory picker now. CHG: ZIP import now can handle zipped folders. FIX: BUG #25454, fixed 1st level resolution file cache. ADD: Added some documentation.
1.1.2	CHG: Changed TypoScript structure. Previously inserted plugins still remain functional, but if you edit the Plugin configuration, you have to select your gallery / album / item again. FIX: Paging in SpecificAlbum mode throws an exception. You have to edit your album and select the mode again. CHG: Plugins now displays mode / album / theme in the page content element overview CHG: Album / gallery description is displayed in the module
1.1.1	CHG: Galleries and Albums are now again sortable. (a change in the database was necessary!) CHG: Complete Extension is now translatable. ADD: Added german translation (Thanks to Matthias Kuchem). CHG: Add all parameters to the URL instead of using the stateHash CHG: Removed all tables from the list module. All data should be administrated by the YAG module. CHG: ReolutionFileCache-Files are now identified by parameter hash. FIX: Many more minor bugs.
1.1.0	RFT: RBAC is no longer a dependency. Features will be outsourced to yag_feedit extension FIX: German translations are removed from JS files FIX: Added lots of translations RFT: Removed lots of CSS and JavaScript to make Backend work better (thx to Matthias!) ADD: Page cache is cleared, if objects change FIX: Thumbs are now generated on Windows platforms FIX: Directory import now respects filetypes correctly RFT: Image processing now uses T3 standard libs and has many configurations now
1.0.10	Bugfix release
1.0.9	Bugfix release
1.0.8	FIX: Fixed some bugs concerning contextIdentifier to enable tt_news integration
1.0.7	FIX: Multiple instances of the plugin can now be positioned on the same page with different themes FIX: Bug #13820 – SWUploader not working without FE Session. Thanks to PETIT Yann FIX: Bug #13822 - No thumbnails are created on Windows servers. Thanks to PETIT Yann ADD: Caching has been refactored RFT: Image ViewHelper has been moved to another directory ADD: Implemented automatic cache cleaning, when objects change CHG: Added lazy loading for domain models ADD: Single image view now has Download-Link for full-res image ADD: Documentation
1.0.6	ADD: Implemented caching ADD: Documentation RFT: Reduced number of SQL queries in Domain Model
1.0.5	Problems with TER upload – no changes

Version	Changes:
1.0.4	ADD: Documentation FIX: Bug #13763 / display error message when static template is not included ADD: Breadcrumbs show "all galleries" when gallery list is shown ADD: Implemented pageCacheManager, clearAllPageCacheAction to Backend Controller FIX: #13775 Adding a new album to a gallery shows right gallery now FIX: #13776 After importing from directory on server, the album list is shown FIX: Fixed bug in directory crawler
1.0.3	ADD: Documentation ADD: Some translation FIX: Dependencies are set correctly in ext_emconf.php
1.0.0	First release of this extension.

We are currently using GitHub.com for collaborative development. You can find all commit messages and an up-to-date trunk of this extension on:

https://github.com/michaelknoll/yag

If you would like to join the team, send us an e-mail (info@yag-gallery.de)