EXT: oEmbed media render type

Extension Key: mediaoembed
Language: en
Keywords: media, oEmbed, rendering

Table of Contents

Introduction

What does it do?

This extension provides editors with a new “oEmbed” option in the “Render Type” dropdown for media content elements. If this option is selected the media will be embedded via an oEmbed request if a matching provider was found. For more information on oEmbed see http://www.oembed.com/.

Additionally the extension gives administrators the possibility to manage oEmbed providers. It comes with a huge amount of default providers out of the box.

Administrators have the possibility to use providers that have their own API like http://www.youtube.com/ or http://www.flickr.com/ but they can also use generic providers like http://api.embed.ly/ or http://oohembed.com/ that can deliver the embed code for many other providers.

Screenshots

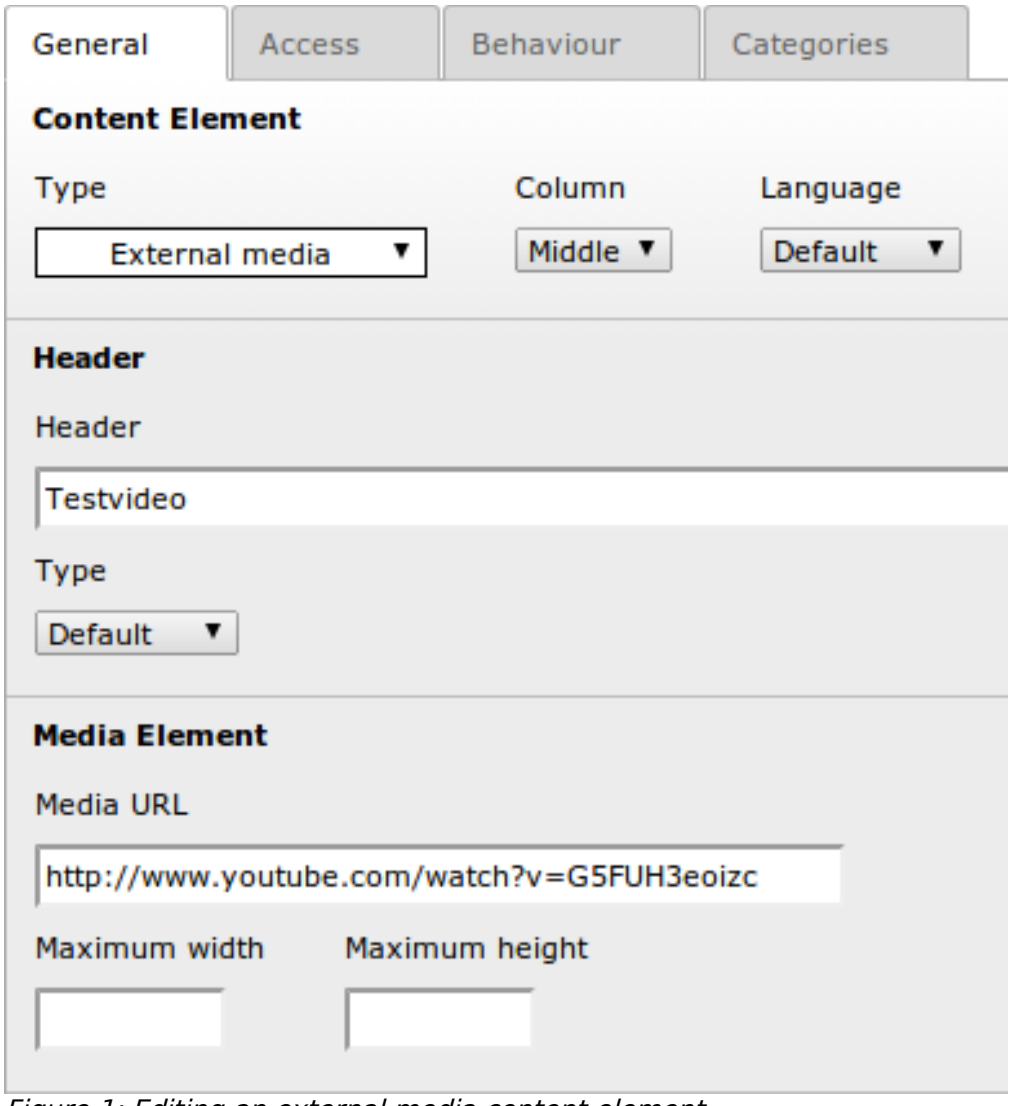

Figure 1: Editing an external media content element

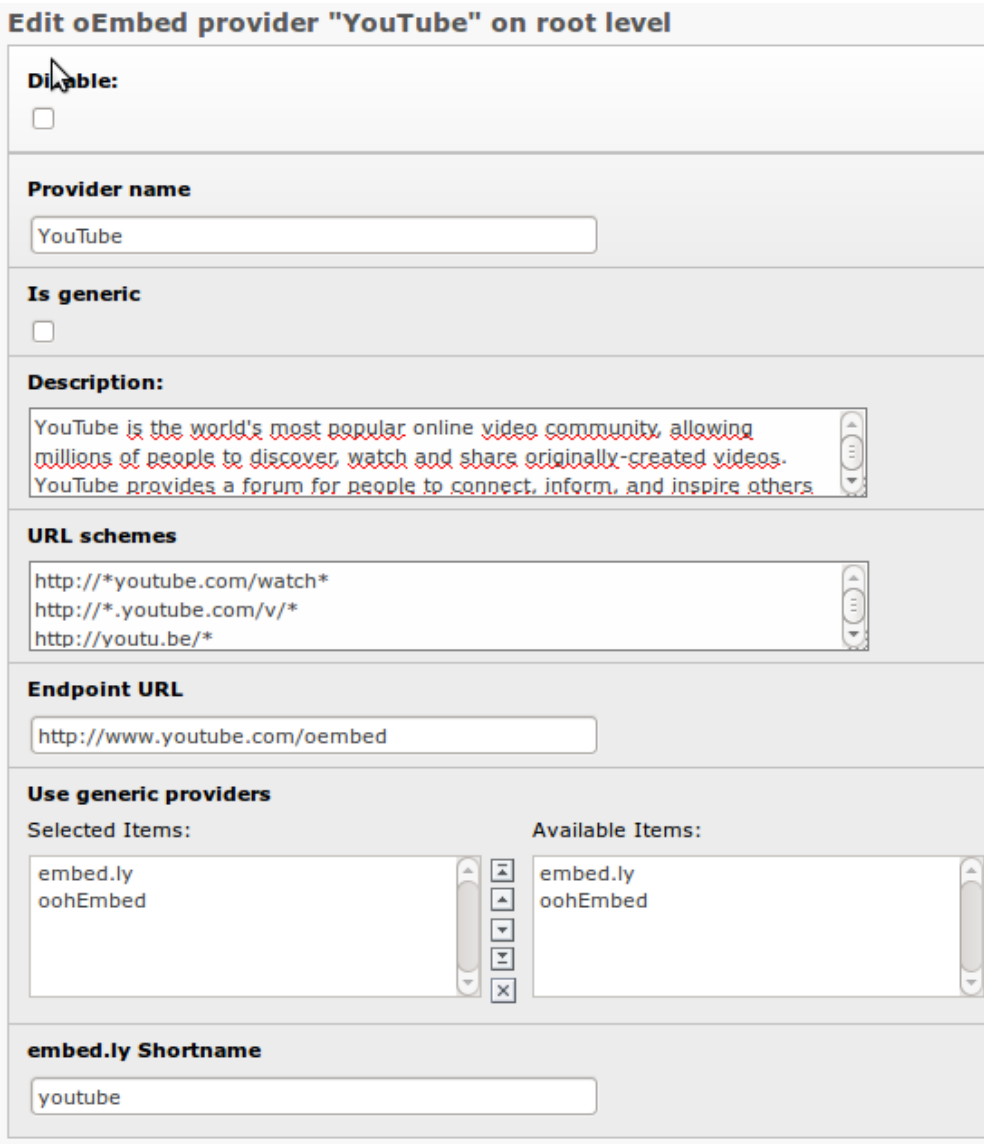

Figure 3: Editing a provider in the Backend

Figure 4: A YouTube video embedded with mediaoembed

Users manual

Using this extension is quite simple. You start editing or creating a content element and select “External Media” as content type. Then you only need to provide the URL to the media you want to embed (e.g. a YouTube video) and save the content element.

If a provider is known for this URL the embed code will be retrieved and the media will be embedded. Otherwise an error will be rendered.

You can use the width and height parameters so specify maximum width and height of the embedded element.

Administration

Installation

Import the extension in the extension manager and install it. If you don't want to set up all providers on your own import the static data and you will have a large number of providers per-configured.

The extension was developed for TYPO3 version 6.2 and was not yet tested in older 6.x versions. If you are running older versions of TYPO3 please share your experience.

Hint! This extension will not work with TYPO3 4.x any more. You can find older versions in the Github repository.

Update

The database structure and the TYPO3 integration have changed very much between version 0.0.1 and 0.1.0! In the current version a new content type is used for embedding external media called “External Media”. The media data is stored in some new database fields in the tt_content table and not in the media FlexForm any more.

After updating the Extension please use the Update wizard in the Install tool to upgrade you existing media elements to the new version. There are two steps you need to execute in the following order:

1. mediaoembed - Create required columns
2. mediaoembed - Migrate content elements

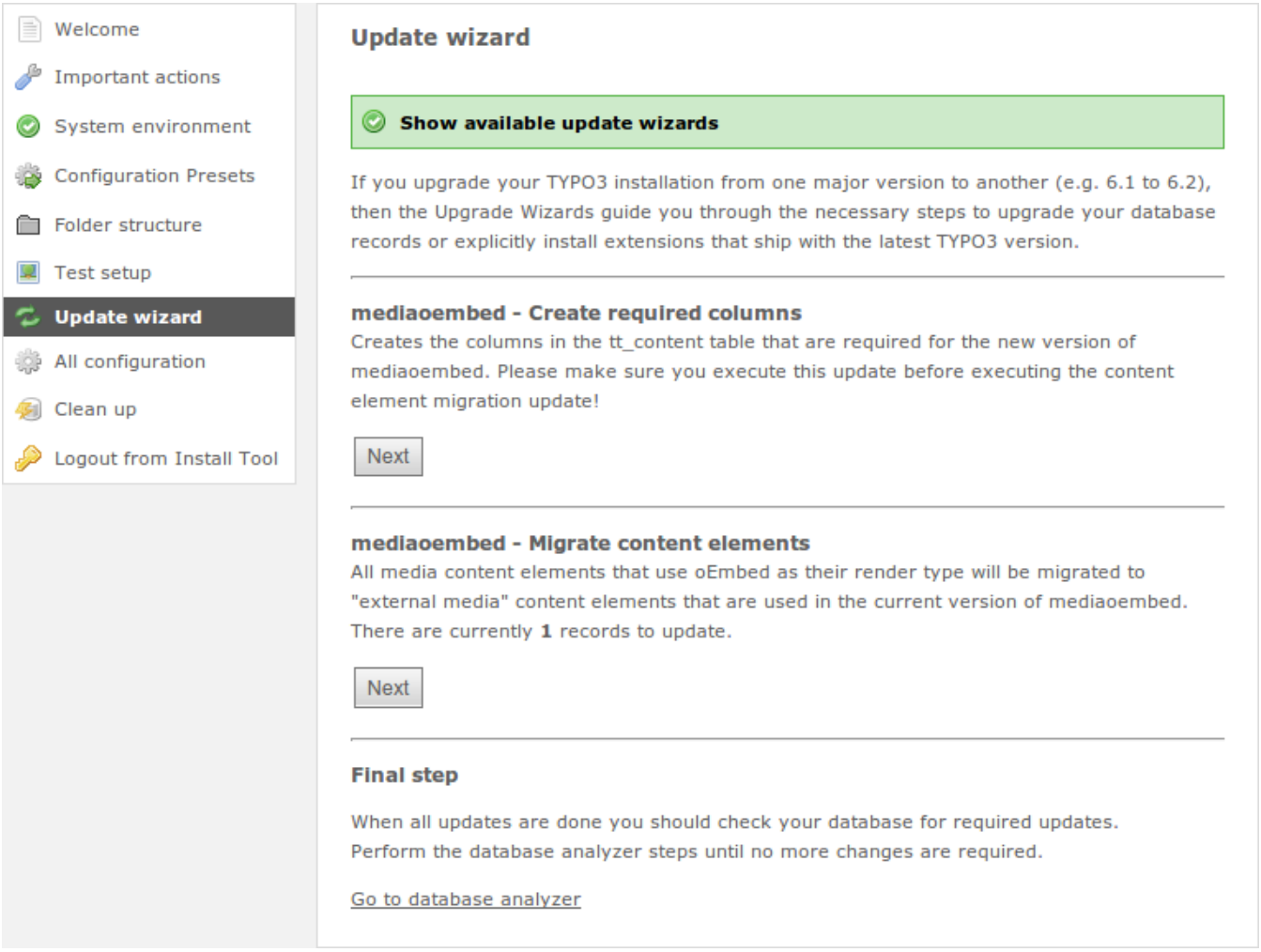

Figure 5: mediaoembed update steps in the install tool

That's it. You can now use the new version of the mediaoembed Extension and all your existing contents are migrated.

Manage Providers

Switch to the “List” module and select the root of your TYPO3 installation in the page root. Scroll down to the oEmbed providers table. There you can:

– create new providers

– edit existing providers

– change the order of providers

You might ask yourself “Why should I change the order of the providers?”. To answer this you need to know how mediaoembed resolves the provider for a mediaurl:

– each provider has a number of URL schemes he is responsible for

– to resolve the correct provider all providers will be traversed and all their URL schemes will be checked if they match the current media URL

– the order in which the providers are checked depends on the order in the backend

So if you have a provider that you will use very often you can improve the performance of the lookup process by moving this provider to the top.

Note! No benchmarks have been done yet to monitor the performance impact of the provider order. Maybe it is so small that you do not need to care...

Editing a provider

A provider has a number of properties that will be explained in this chapter.

Disable

Good old disable property. When you set this the provider will not be available for your content editors. If they use URLs from this provider and there is no other provider that can also handle this type of URL they will get an error.

Provider name

This is the readable name of the provider. It will be used as the title of the provider record in the backend and is also accessible during the rendering process of the media content element.

Is generic

This property marks a provider as “generic” which means the provider can serve embed codes for multiple content providers. Providers with this property will be listed as options for the “Use generic providers” property. Normally generic providers will not have any own URL schemes attached to them.

Description

Here you can put in a description for the provider which will also be available during the rendering process of the media content element. This might be useful if you want to provide more information about the used provider to your website users.

URL schemes

Here you can put in the URL schemes this provider is responsible for (one scheme per line). At the moment you can only use the asterisk (*) as a wildcard.

Endpoint URL

This is the URL where the query will be sent to and that should provide the embed code to us.

Use generic providers

Here you can select a generic provider. You can use this either in combination with the endpoint URL or on it's own. If you combine it the endpoint URL will have priority. Only if the request to the native endpoint will fail the generic providers will be used in the order you spcified.

embed.ly Shortname

Http://embed.ly provides an API call that returns a list of all supported providers. One property of the results is the shortname, which is unique for each provider. To make updates more easy in future, the embed.ly shortname is stored here. Normally you will not need this.

Configuration

Rendering

The rendering mechanism of this extension is very flexible since all the rendering is done with TypoScript. Have a look at *Configuration/TypoScript/setup.txt* and it will probably remind you of the setup of css_styled_content.

Accessing data

But there is something special in the TypoScript configuration because we somehow need to access the data of the the provider, the request and the response. This is why there is a new getText type available called *registerobj*. With this type you are able to access register data like you can do with the *register* type. The new thing about *registerobj* is, that you can access object getters and array data. Do do that simply use the pipe character (|) like you know it from the *GP* type for example.

All relevant data for the oEmbed rendering is stored in a single register called *tx_mediaoembed* which is an object of the type Tx_Mediaoembed_Content_RegisterData array that contains 4 child objects:

- Tx_Mediaoembed_Content_Configuration
- Tx_Mediaoembed_Request_Provider
- Tx_Mediaoembed_Request_HttpRequest
- Tx_Mediaoembed_Response_GenericResponse

With the new getText type you can access all public available getter Methods of these object. For a complete documentation of all available methods please have a look in the Documentation/Api folder that contains the full phpdoc for this extension. In the future, the most important methods will also be documented here.

If a method returns an array you can traverse the array by using the pipe character.

Example

```
video = HTML
video.value.data = registerobj : tx_mediaoembed|response|html

# This code will call the following methods:
Tx_Mediaoembed_Content_RegisterData->getResponse()->getHtml()
```

Function reference

TODO: Document the most important getter methods.

Extending it

TODO: Document how this extension can be extended.

Known problems

This is extension is quite new and not much tested so no problems are known so far but there might be some...

To-Do list

- Get rid of the XCLASS that is currently needed because of a wrong TYPO3 behaviour
- Document and improve the import scheduler tasks
- Better performance when resolving provider e.g. by ordering the providers by usage count
- Implement security features, e.g. use an iframe for embedded content
- Write more documentation
- If you have feature requests or want to view the most up to date to-do list, please visit the forge page of this extension.

ChangeLog

To see what has changed please have a look at the roadmap on forge.