

## Thoughts about a link-directory for TYPO3

This paper is supposed to give the reader an idea of my thoughts regarding the making of a link-directory extension for TYPO3. First of all I would like to point out, that my experience in making extensions for TYPO3 is limited to working with the pibase. I have taught myself almost everything I know about web developing, I have had my own clients and have never worked with a consultancy. I study a master's degree in economics and business administration. Therefore you should be aware that there might be errors in the ER model and there might be better ways of doing things. To give you the best picture of my ideas around this extension I'll explain as much as possible, and do pseudo-code where I consider it to be helpful for the reader to gain a better understanding of my thinking.

I would like to elaborate a bit on the reasons I have for getting this link-directory made. By working with SEO on several websites of my clients, I have come to realize that many professional SEO companies provide their own directories. This enables them to provide a better product for their clients, because Google puts a great deal of weight on which words are used in the links that points to your site. TYPO3 already have extensions that serve the purpose of building lists of links. I find these extensions relatively rigid compared to the concept I'm about to explain. Moreover I have a new idea that hopefully could boost the value of this extension over the many free PHP platforms for managing a link directory. Enough with the introduction already, let's dig into it!

### Categorization with flexibility

I have described my views about categorization in TYPO3 in this blog-post:

<http://buzz.typo3.org/people/soeren-andersen/article/categorization-in-typo3/>. For your comfort I'll explain some of it here and relate it further to this extension. The problem with many extensions is that they have their own built in categorization. It typically works in a way that involves a GET variable, which determines what category the user is in. This means that no matter how deep you go into the categories, you will still be on the same page in the page tree (hint: same UID). This results in inflexibility because if you create a content element on this page, it will show up on every category page. In my opinion this disables a lot of great features and extensions in TYPO3. Instead I'm proposing that each page serves as a category, ideally by creating a doktype that allows the extension to identify which pages are categories and which are regular pages. Until the community creates a global way of determining categorization for TYPO3 this solution is the better one. It will allow you to make category specific things like: meta-tags, page titles and page content.

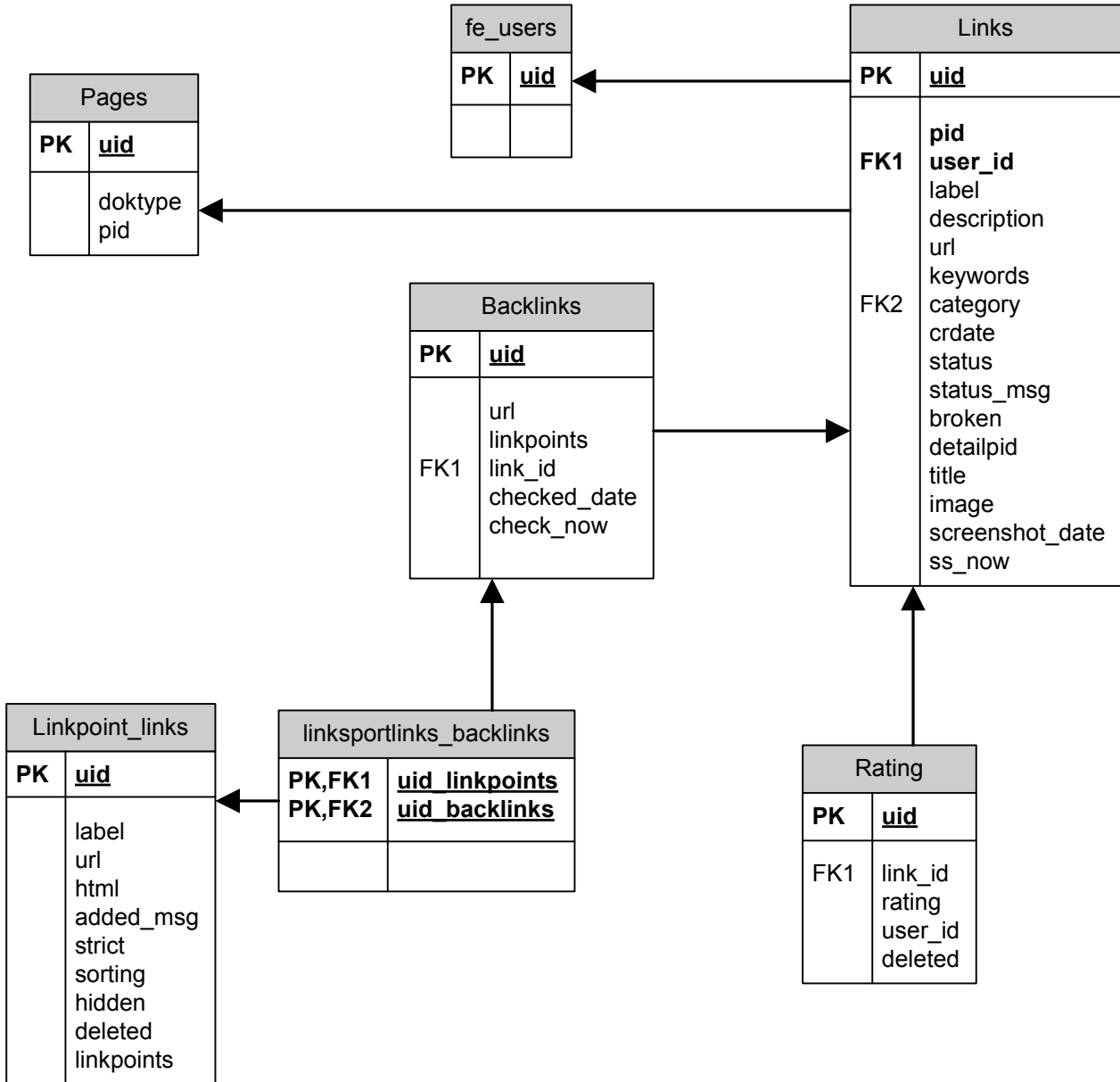
### Building on an extendable platform

Normally a TYPO3 extension will extend the class that is called "the pibase". But this class is old and outdated, and limits development in a number of ways, that I won't describe further. Therefore I have decided that this extension should be built in the lib/div framework, which also should ease the porting to V5 when that is needed. This framework is a great foundation for flexibility and enhances the users' ability to create uniquely structured templates.

The two sections above lays out the foundation for programming this extension, and should ease the understanding of how the extension is kept most flexible. Now let's move on to some of the functionality.

## Data structure of the extension

I would like to start this part with an introduction to an entity-relation model that I have made, to show you how I think of the data structure.



## Links entity

The central part of this model is the *Links* entity (note that I will be using italic font for entity names and ***bold-italics*** for columns in the entities). The table has – as most TYPO3 tables – has the ***uid*** for indentifying records. Other TYPO3 specifics are ***pid*** and ***crdate***. For relating the link to a front-end user we have the ***user\_id*** field. To determine the link-text I have dedicated the ***label*** field; this will enable the user to specify a link text that’s more SEO friendly. ***Description*** is just what it says, a description of the link like many directories have. ***url*** also speaks for itself, we need the URL to make the link. The ***keywords*** field should be

used for a later search implementation. Now the **category** field is open for discussion. Normally I would just let the links be located on category pages, thereby making the **pid** field sufficient for distinguishing in which category the link is located. Note that this will only work if links can't be placed in multiple categories.

**Status** should determine if the link is pending, approved, denied, removed or other things. The **status\_msg** field is thought to be a way to let the user know, why the link is removed, denied or something like that.

**Broken** is a status code that could be set by a spider, when the link returns 404 and another code when the users report the link broken. If you want to have a whole page for each link, the **detailpid** should tell which page is related to the link. Again this is because of flexibility, so you don't use the same page for displaying all the links. **Title** should be used for the title attribute in the <a> tag, or perhaps the alt attribute. **Image** could determine a picture you would want to show with the link. Because the most probable image would be a screenshot, the process should be automated. Therefore the field **screenshot\_date** should determine when a screenshot was last taken. **Ss\_now** is a flag, that when set, should prompt for a screenshot to be taken.

*It's not that easy to automate the screenshot process, but I have found a firefox extension that can take the screenshots automatically when combined with a PHP script. I have already written this script, which can supply the server which hosts the link directory with images where the filename corresponds to the **uid** field. This would make it very easy to have a link directory with lots of pictures.*

## The concept of LinkPoints

Different directories use different methods for sorting the links. Some sort by date of creation some in alphabetical order. I have thought out a new way of sorting links, I call this LinkPoints. I know nothing about whether this has been thought of before, because it's very simple. The link that has the most LinkPoints has the best place in each category. You could let users earn the LinkPoints by linking back to your page, or you could simply sell the LinkPoints on a monthly basis. This will eliminate the need for toplink deals. By selling each LinkPoint at 1€ you could let the users decide how much they pay for the best place. Since links to your website enhance your website's performance in the search engines, I require the extension to have this aspect from the very start. You specify a number of links that the users can link to, when they link to these pages, they earn LinkPoints.

## Backlinks entity

This entity holds the links that will be used to earn LinkPoints. Let's assume someone has registered the domain i-love-typo3.org and he wants a good placement on our site, so he links back to us. The **url** is the URL where we can find the link fx <http://i-love-typo3.org/links.html>. **Linkpoints** is the amount of LinkPoints that the link has gathered. So if he links back to us he earns an amount, but if he also links to other sites we have decided to reward the amount will be higher. **Link\_id** is the relation to the **Links** entity. **Checked\_date** is a date that shows when the LinkPoints was last estimated, this is needed because we want to check the URL regularly so people don't just earn their LinkPoints and then removes our link. **Check\_now** would allow a backend admin to request a cronscript to reevaluate the amount of LinkPoints that a given link should be given.

## Linkpoint\_links entity

This entity holds the links that you would like users to link to. When a user adds a link from this table to the webpage they have just added to your directory, they are granted an amount of LinkPoints that equals the

field *linkpoints*. You can then decide which links earns the most LinkPoints and supply the users with more possibilities to earn a better placement on your directory. The fields *label*, *url*, *sorting*, *hidden* and *deleted* should not be hard to understand. The field *html* could supply the users with a predefined HTML code, ready to insert on his/hers website. *Added\_msg* would be a field with a message to the user, if the script finds that the user has added this link. I added the *strict* as a flag that should determine whether users are allowed to link to [www.i-love-typo3.org](http://www.i-love-typo3.org) or if it has to be <http://i-love-typo3.org> without the www.

This information about the data structure should give you an idea of the concepts I have thought about in relation to the extension.

## Specific functionality of the extension

There are several aspects to the coding that should be discussed. One of the things is registration of the links; another is the display of links. I will describe the different aspects I can think of as a minimum requirement for the extension.

### Registration

For most flexibility I have decided that I would like to split the registration process into 3 pages. First page should consist of the field: email, password, url. The email address is the username of the frontend user. If the username isn't found in the database, the user should be created "on the fly" and logged in at the same time (this is done by making a service that interrupts the login process). The URL is of course the URL of the page that the user would like to add to our directory. If the user is found but password incorrect then it should return to first page with an error message. Perhaps there should be some captcha to protect from robots.

On the next page the URL will be used to gather the HTML of the URL, then the meta-tags and title-tag will be gathered automatically (I have made a simple script to test the possibility of this, and it's actually very easy). The user can then correct the title, description and keywords or write them from scratch if they weren't found. This page should also inform the user of the possibility to earn LinkPoints and provide a text-field where the user can paste all the pages where he links to pages that will provide him with LinkPoints. He will of course not earn the LinkPoints twice, just because he has the links on two pages. The links should be inside the registered domain, to prevent abuse.

Last page will sum up the registration and give some final information.

I imagine this aspect as one plugin, with its own models, views and controllers. There could be some who would like to have the whole registration on one single page, therefore a controller switch should be build in the flexforms of the plugin. Again it's about flexibility.

### Display

As with the registration process the desired way of displaying categories and links will vary from person to person. I imagine that standard link directory setup should be supplied. Flexibility in this plugin should ensure that others ways of display can be implemented easily.

## **Backend administration**

There should be a backend module for handling all the things like: reviewing, disabling, overview of pages needed to get screencaptured and so on.

## **Some last thoughts**

There are very many functions that could be implemented into a link directory extension. The idea of this extension should be to provide a framework that can easily be extended to fit different needs. This way we would avoid being just another link directory extension that serves the needs of individuals. It's the same reason I keep talking about flexibility 😊