Workshop: I'm a developer where do I start?

Presentation at the official Contao Conference 2012
Bad Soden, Germany

certo

Don't ask me why my nickname's Toflar. I remember having taken 6 letters on my keyboard and arranging them in random order because everything „cool" was already taken at that time. And I was very young which also excludes any influence of alcohol...probably. Today I'd probably choose „Yanick" as my nickname, why not :-)

certo

Contao

Table of contents

- Goals
- Your expectations
- The structure of Contao 2.11
- Drivers & The DataContainer
- The structure of a custom module
- Hooks & Callbacks
- Workshop (I shop, you work ;-))

certo

Contao

Goals

What you will know after this workshop

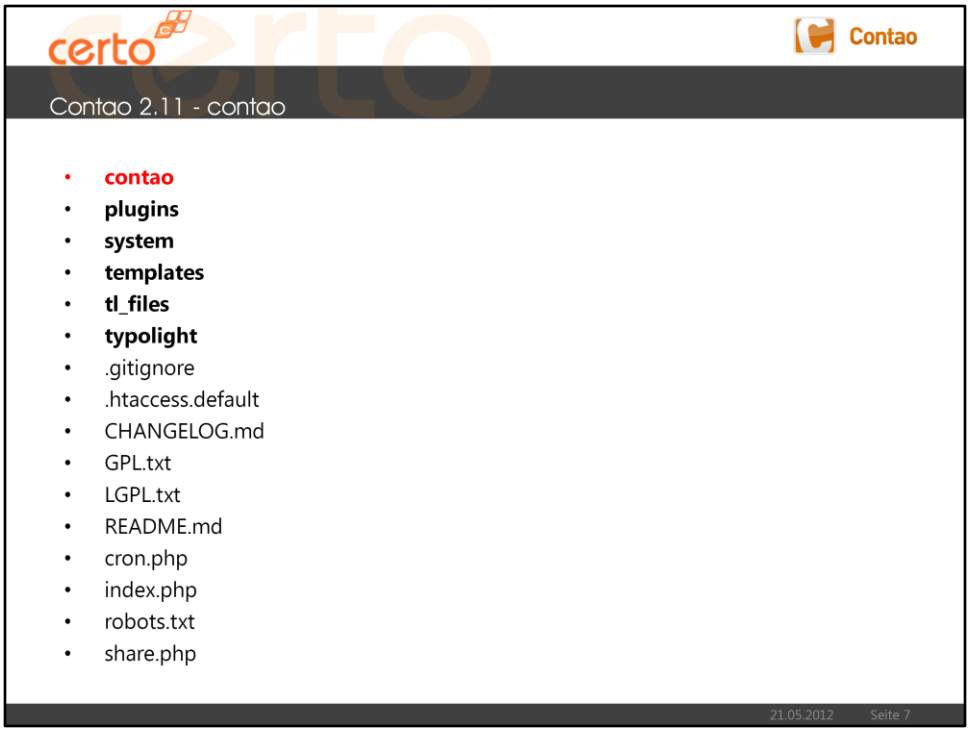
- Get to know the structure of Contao 2.11
- How to read the source and help yourself
- The structure of a custom module
- Hooks and callbacks

What you will not know after this workshop

- PHP
- Object Oriented Programming (OOP)
- How to use Contao
- Every single line of Code of Contao

Expectations

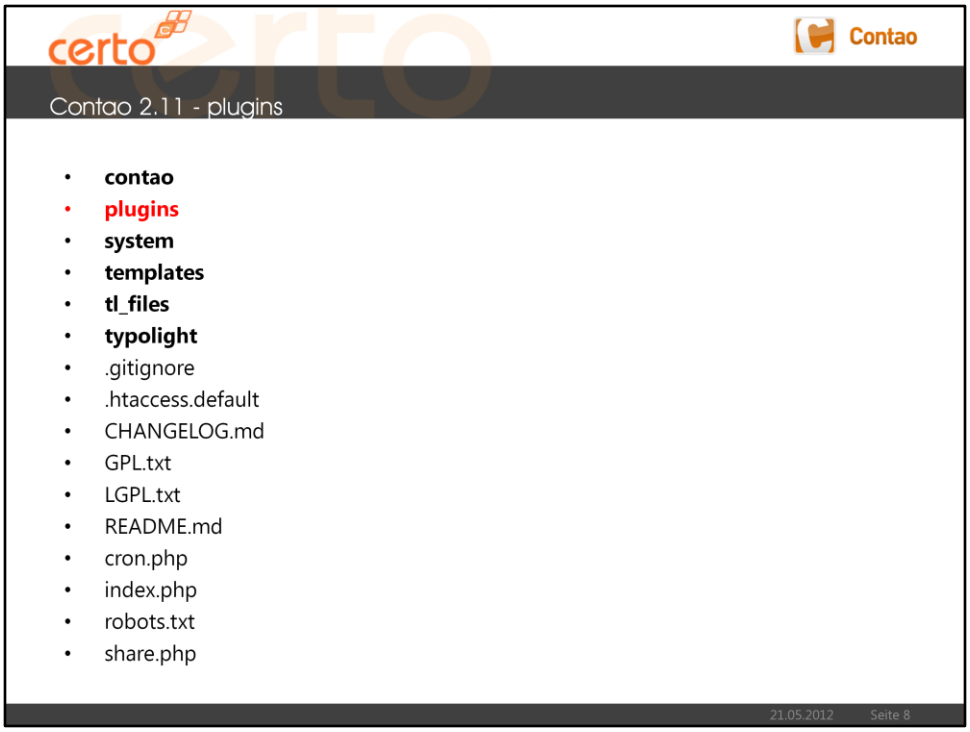

certo
Contao
The structure of Contao 2.11
21.05.2012
Seite 6



The directory "contao" contains all the files that are needed for the back end functionality such as

- The back end entry file "main.php"
- Javascript files
- The install.php file
- etc.

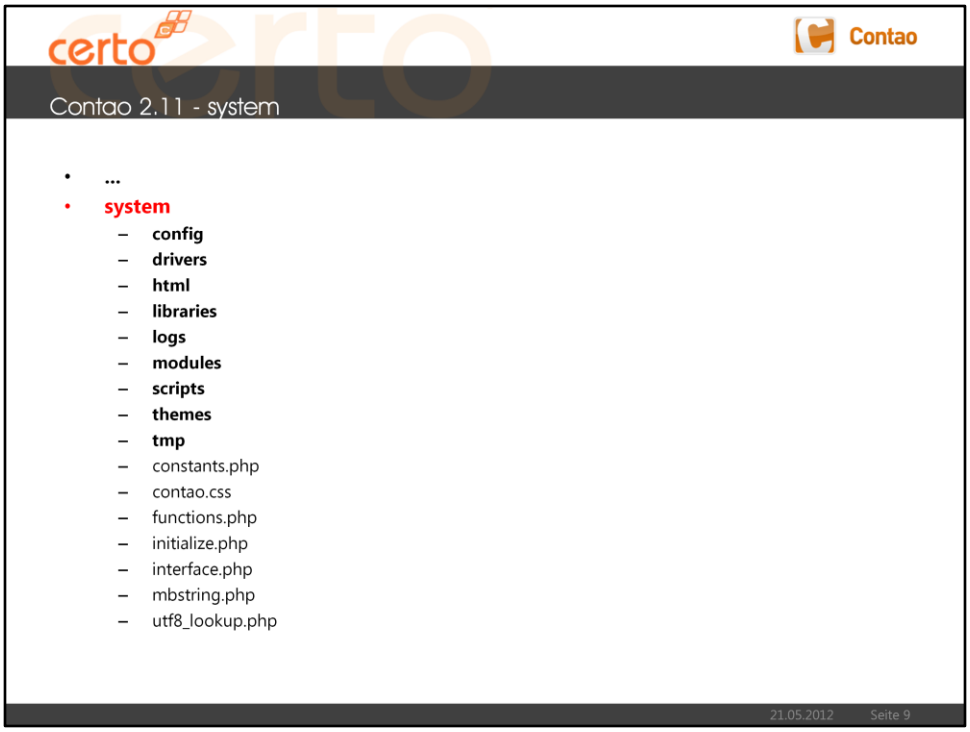
You will never have to add anything to that folder for a custom module.



The directory "plugins" contains third party scripts that are used within Contao such as

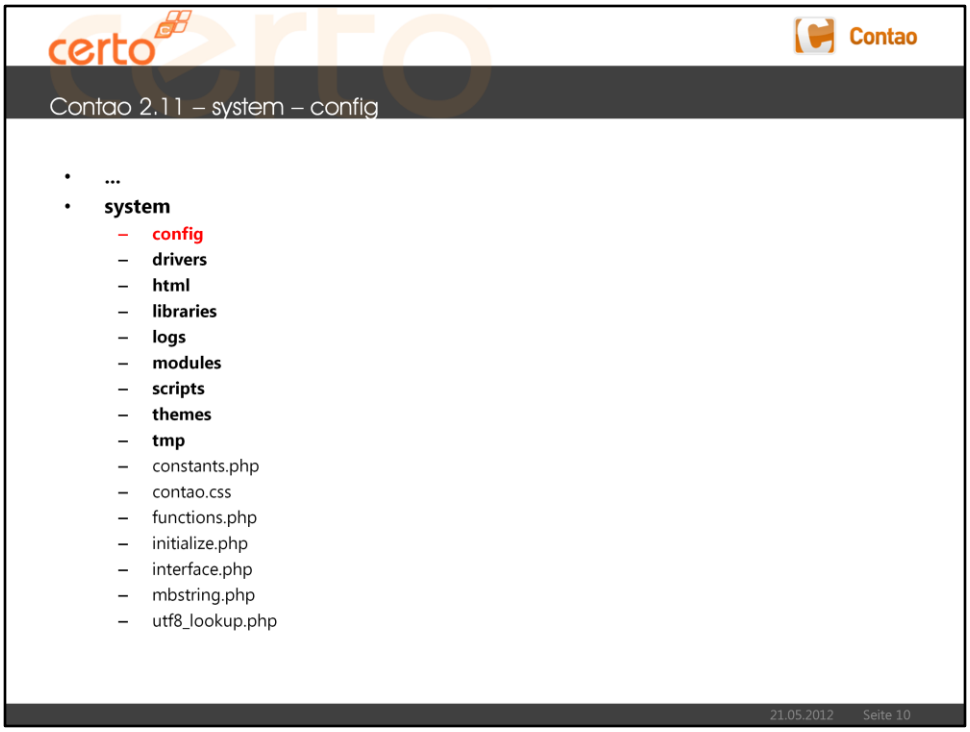
- chosen
- mediabox
- swiftmailer
- tinyMCE
- etc.

In general you don't need to add anything in here. It's a question of design whether you add plugins you use for your extension to this folder or not. There's no convention although it‘s always a good move if you think of other third party developers and outsource the plugins into their own extension. This way other developers don‘t have to ship the plugin again but can rather add a dependency of your library (e.g. I have done this for „rapidmail_library“ and „rapidmail_sync_recipients“).



As you might guess from the name of this directory, "system" is the most important directory in Contao and we will thus look into this in more detail.

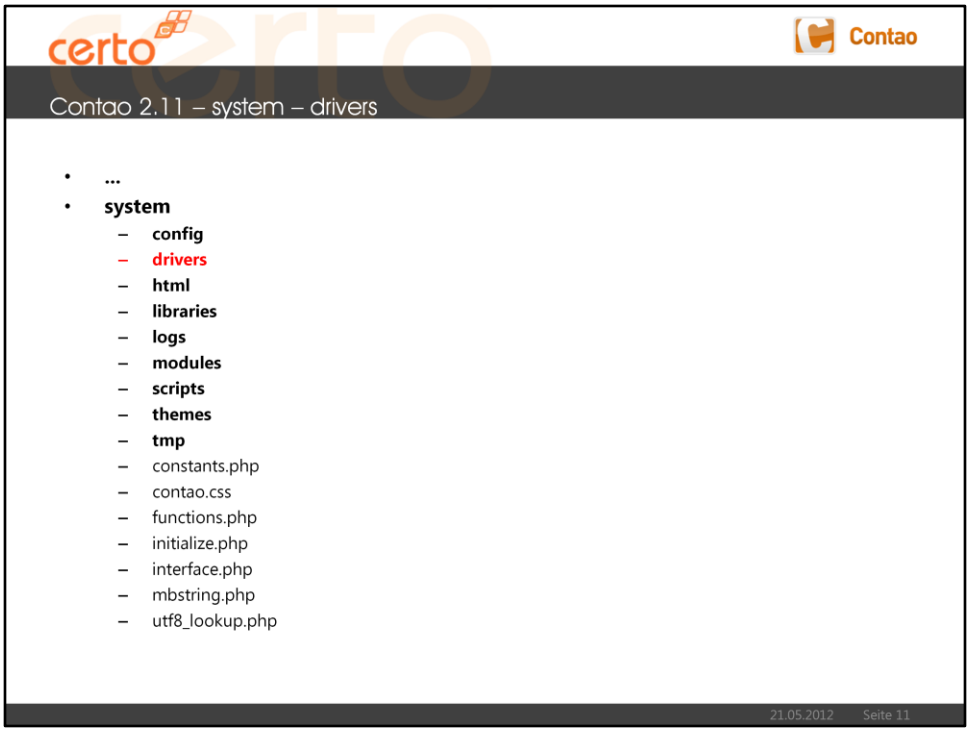
We won't cover all of the directories because some of them are just directories files are cached in etc. Be creative, read the source and try to find out what Contao does!



The config directory contains the standard configuration files. There are also the local configuration files for your installation that go in here:

- localconfig.php
- dcaconfig.php
- langconfig.php
- initconfig.php
- Your own tinyMCE config files
- etc.

Normally you won't have to add anything to that folder but you might consider this place to put a custom configuration file if you think that your extension has too many parameters for being reasonably managed in the back end settings.



The "drivers" directory is very important for Contao. You are using drivers whenever you are using Contao. There are two different types of drivers:

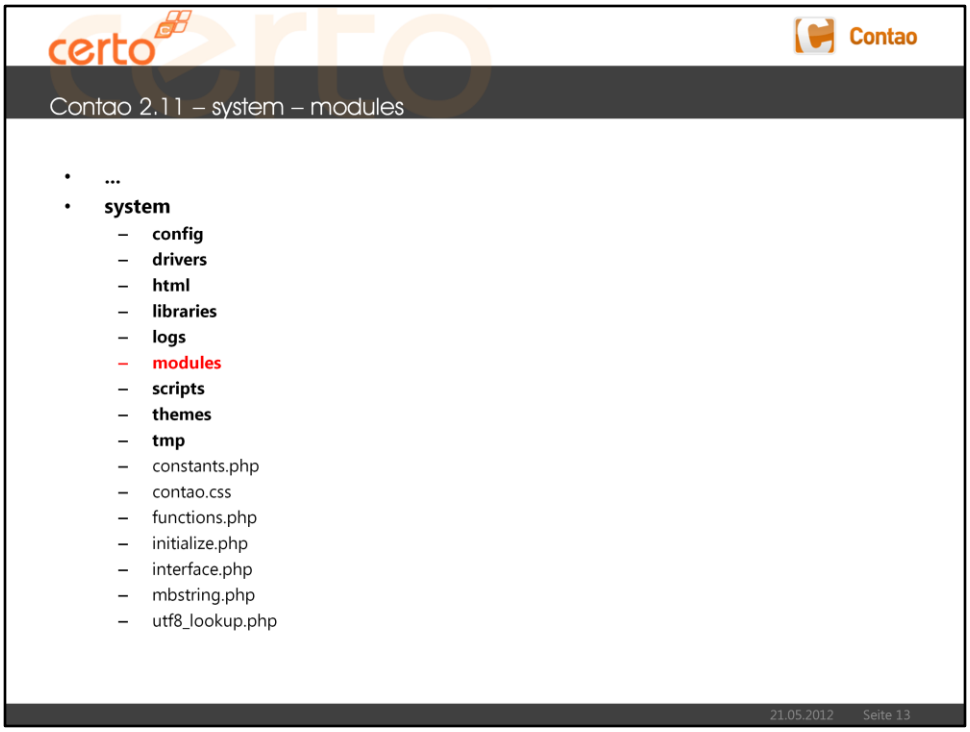
- DataContainer drivers (start with "DC_")
- Database drivers (start with "DB_")

The database drivers provide a very primitive abstraction layer for the database so that you can use Contao also with PostgreSQL or other databases.

We will see what a DataContainer is later on but if you develop your own DataContainers or database drivers - put them in here.

The „libraries“ directory contains some core functionality that can be used in your own modules.
Make sure you know the libraries in here because they contain a lot of methods you will need in your own modules too.

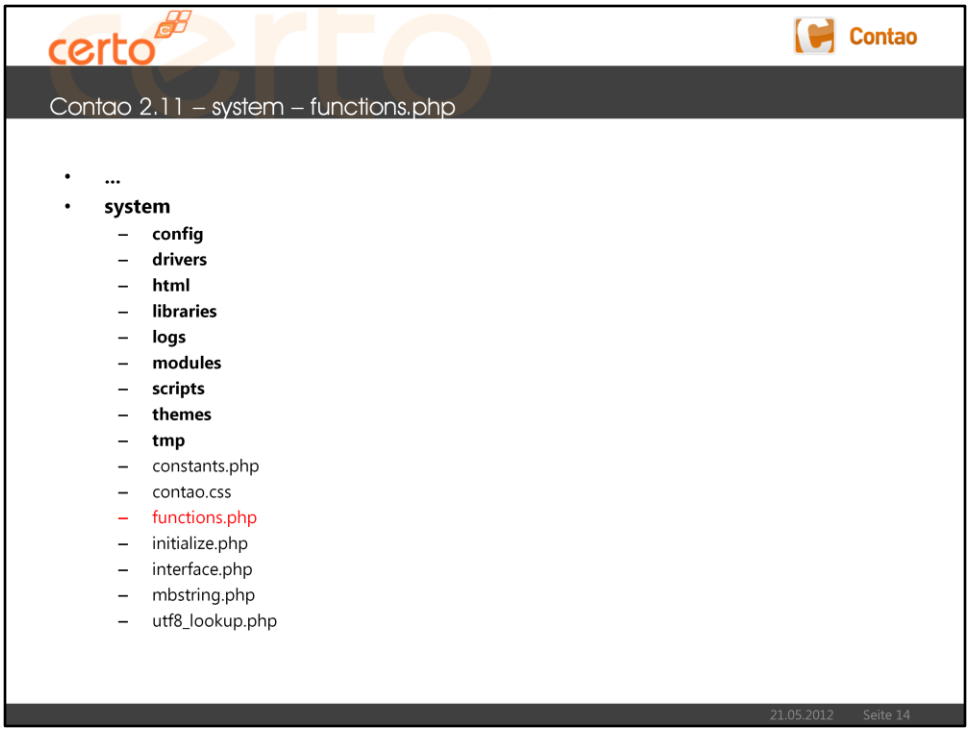
The core library‘s name is „Controller“. It‘s probably the class that provides the greatest number of methods all over the core and that‘s probably because Contao‘s grown that fast over the past years. Take a close look at it.



Simply put, **this** is where your extensions go! Every extension to Contao is a module and goes into "system/modules".

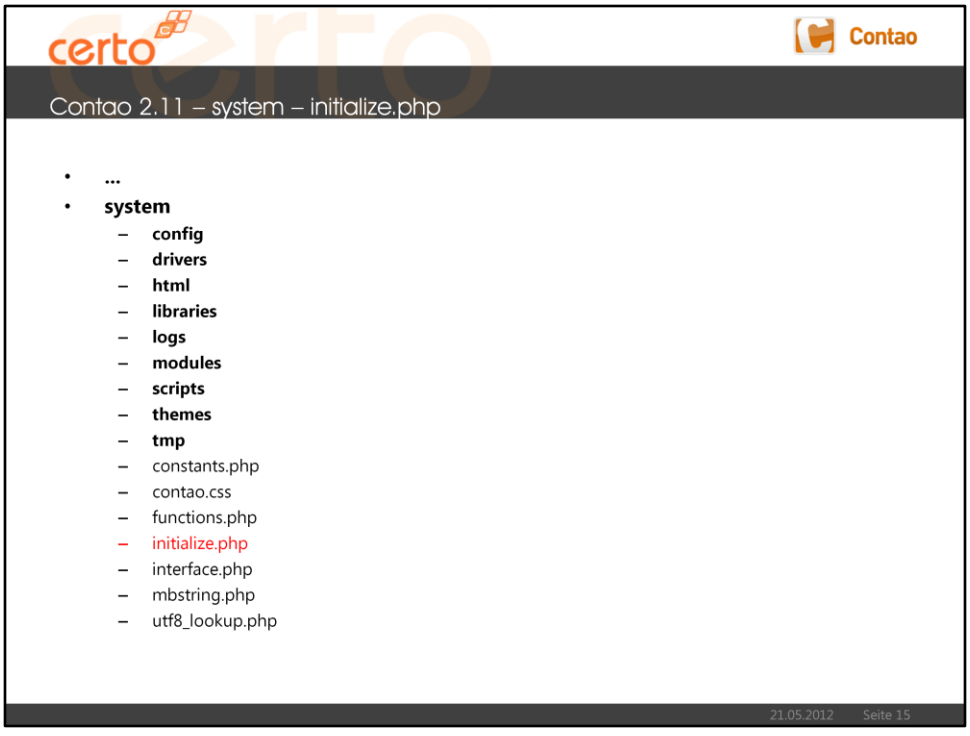
We will learn more about the structure of a module later on but for now, just remember that they have to be installed into this directory.

By the way: did you know that the back end and front end are just modules too?



When you happen to have a few minutes of spare time, also check out the functions.php. It defines some very useful, Contao-specific, functions that you can use in your modules.

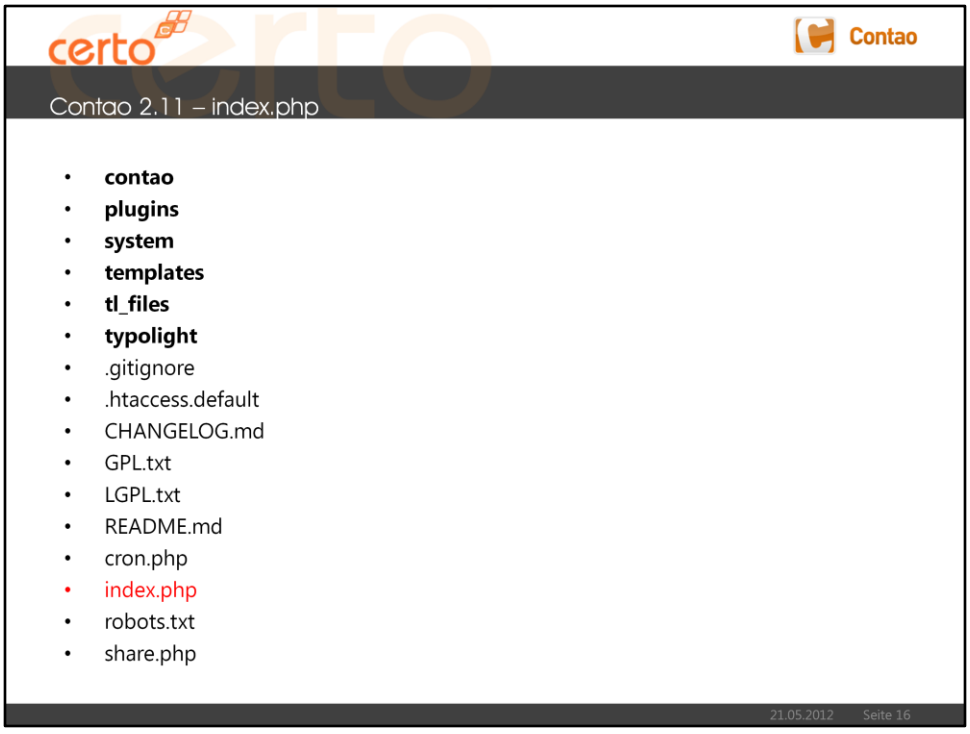
- scan() - to scan a directory for files
- specialchars() - to convert special characters to HTML entities
- standardize() - to convert a string to a "standardized string" (used e.g. for aliases)
- deserialize() - to unserialize a serialized string with some additional features
- array_insert() – to insert an array at a certain index of another array (quite useful!)
- ...



This is the file that is included by Contao to bootstrap the system. Normally you don‘t need it but knowing what it does and what components get loaded in what order is always useful!

Not only will it give you a better understanding of how Contao works but also will it help you debugging your code.

Some extensions provide other interfaces to Contao‘s index.php such as Andreas Schempp‘s „ajax“ extension or my „swekey“ module. Both include the initialize.php to bootstrap the system other than via the index.php.
Probably you‘ll need to do that one day too.



I hope I don't need to mention that the index.php file is the one that is initially called by the browser when a user visits your page.

Also take the time to check this file. You will see that it includes the "initialize.php" file we have seen before.

Just make sure you roughly know what it does so you aren't lost when you stumble across funny error messages.


certo
Contao
Drivers & The DataContainer
21.05.2012
Seite 17

Drivers - Overview

- The drivers that begin with „DB_“ represent a very simple database abstraction layer
  - Mssql
  - Mysql
  - Mysqli
  - Oracle
  - Postresql
  - Sybase
- The drivers that begin with „DC_“ represent a Contao specific „DataContainer“
  - File
  - Folder
  - Table

certo

Contao

What is a DataContainer?

- A DataContainer is something Contao specific
- It's a mix of Model, View and Controller which makes it very easy to use for common patterns but very hard if you want to do something that's not planned
- It provides different ways to render a DataContainer Array (DCA) and list database records or files
- It's configuration is done via a DataContainer Array (DCA)

What is a DataContainer Array?

- Used to define the meta data of how the DataContainer displays the data of a database table
- Defines relations to other tables and their fields (database columns)
- Grown in the past and thus very powerful but also very complex and unfortunately not very well documented
- Consists of
  - Table configuration
  - Listing configuration
    - Global operations
    - Operations
  - Palettes and subpalettes
  - Fields
    - Evaluation

A method is worth a thousand words.
Let‘s say the DataContainer for the table „tl_news“ is being loaded. You can see that Contao checks every module for a file that is called „dca/tl_news.php“ and loads this. Then it calls a hook (we will see what this is later on) and eventually includes the dcaconfig.php which is one of the configuration files we saw before (remember, the ones within „system/config“).

Because every DCA is just a global array, every module can extend and modify the array as it likes. But let‘s see how such a DCA looks like.

Every DataContainer Array (DCA) is defined into the global array „TL_DCA“. This way it is possible to modify the array in whatever scope you‘re currently working.

You can see that the table „tl_news“ defines a DataContainer „Table“ that is responsible for displaying your data. Yes, you are right. This is the driver „DC_Table“ we saw before and now you can also see how flexible Contao is.
Just write your own „DC_Whatever“ and set „dataContainer“ to „Whatever“ and you‘re ready to go and define your own back end look and feel.

„ptable“ is an example for a table relationship. It tells Contao that a „tl_news“ entry belongs to a parent entry in „tl_news_archive“ and we all know that‘s true. Every news entry belongs to a news archive.
The callbacks are extremely useful but we will see that later on.

certo

Contao

Example for a DCA based on the table „tl_news" – listing configuration

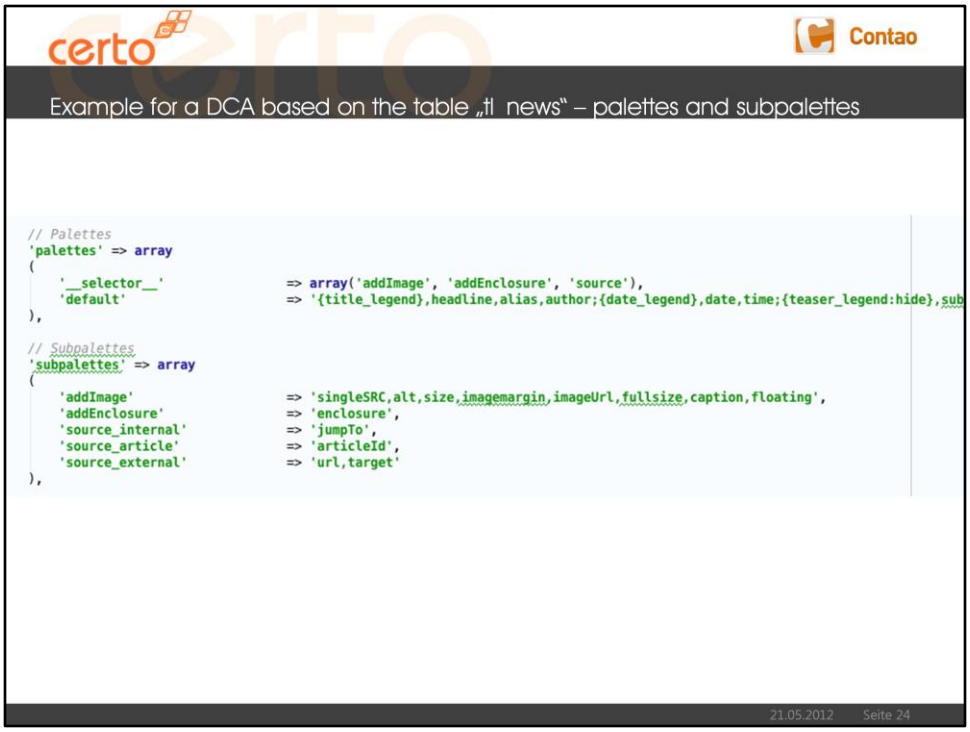
```
// List
'list' => array
(
    'sorting' => array
    (
        'mode'                    => 4,
        'fields'                  => array('date DESC'),
        'headerFields'            => array('title', 'jumpTo', 'tstamp', 'protected', 'allowComments', 'makeFeed'),
        'panelLayout'             => 'filter;sort,search,limit',
        'child_record_callback'   => array('tl_news', 'listNewsArticles')
    ),
    'global_operations' => array
    (
        'all' => array
        (
            'label'               => &$GLOBALS['TL_LANG']['MSC']['all'],
            'href'                => 'act=select',
            'class'               => 'header_edit_all',
            'attributes'          => 'onclick="Backend.getScrollOffset()" accesskey="e"'
        )
    ),
    'operations' => array
    (
        'edit' => array
        (
            'label'               => &$GLOBALS['TL_LANG']['tl_news']['edit'],
            'href'                => 'act=edit',
            'icon'                => 'edit.gif'
        ),
```

21.05.2012 Seite 23

Here we can define how the DC_Table should list our entries. Remember: Everything we define in this DCA has to be supported by the DC_Table. If you use your own DC_Whatever then you have to either implement all this configuration or use your own definitions.

You can see that there are several sorting modes and global_operations (such as „all" which is the „edit multiple" button) as well as operations which are listed for every record (edit, delete, copy etc.).

Go and see the reference (http://www.contao.org/en/reference.html) for more info.



This is an extract from contao.org:

A palette is a group of form fields which are required to edit a record. A palette typically does not include all columns of a table but only the ones that belong to a particular module or content element. Palettes can change dynamically depending on the user's permissions or type of element and certain subparts of the form (called subpalettes) can be loaded interactively via Ajax.

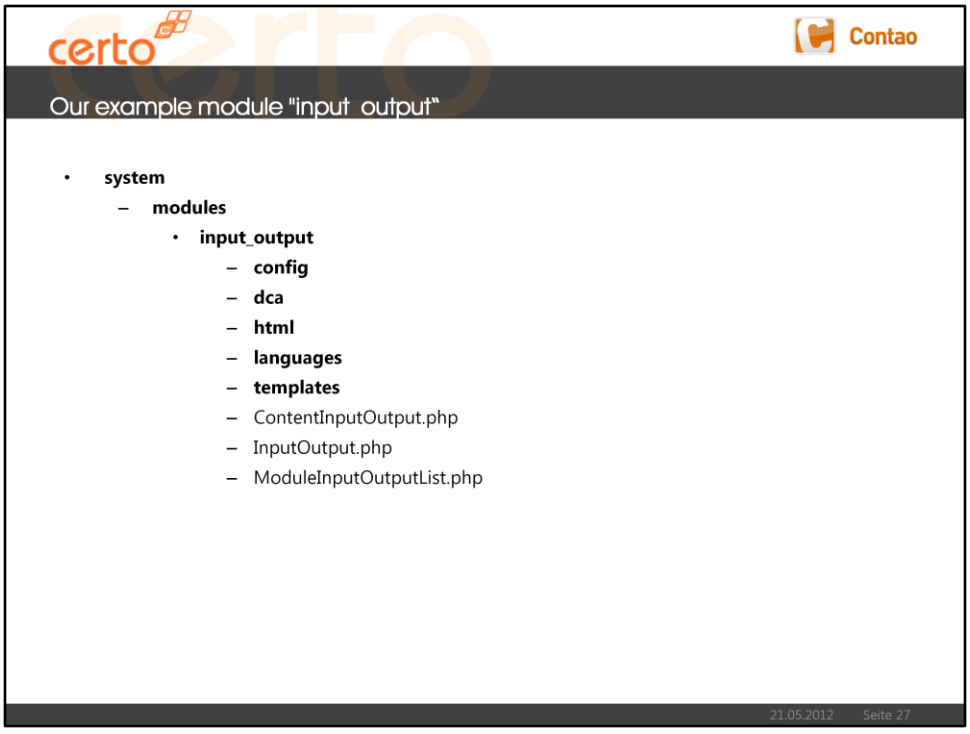
In this case you can see that we have a „default“ palette that adds a few fields and groups them into legends.
The „addImage“ field for example is a „__selector__“ which means that it is a field that is responsible for changing either a palette or loading a subpalette. In this case we can see that „addImage“ is defined within the „subpalettes“ array. So when I change the „addImage“ field, a subpalette containing the fields singleSRC, alt, size, imagemargin, imageUrl, fullsize, caption and floating is being loaded using Ajax.

Okay. So now let‘s define how our fields (columns) look like. As you can see from the sql definition on the slide, the fields are always named after the column name in the database. We will see how modules define what database tables and columns they need when we look into the structure of a custom module.

For now just check how the field is defined within the DCA.
There‘s a label with the reference to a language array so the descriptions can be translated and a lot of other stuff.
„inputType“ defines the form field widget which can be a normal text field like for the headline or a „pageTree“ for a page picker etc. Again you have all the flexibility and you can define your own widget/inputType.

Also note the „eval“ array which configures a field in detail. Check the documentation on contao.org: http://www.contao.org/en/reference.html#evaluation

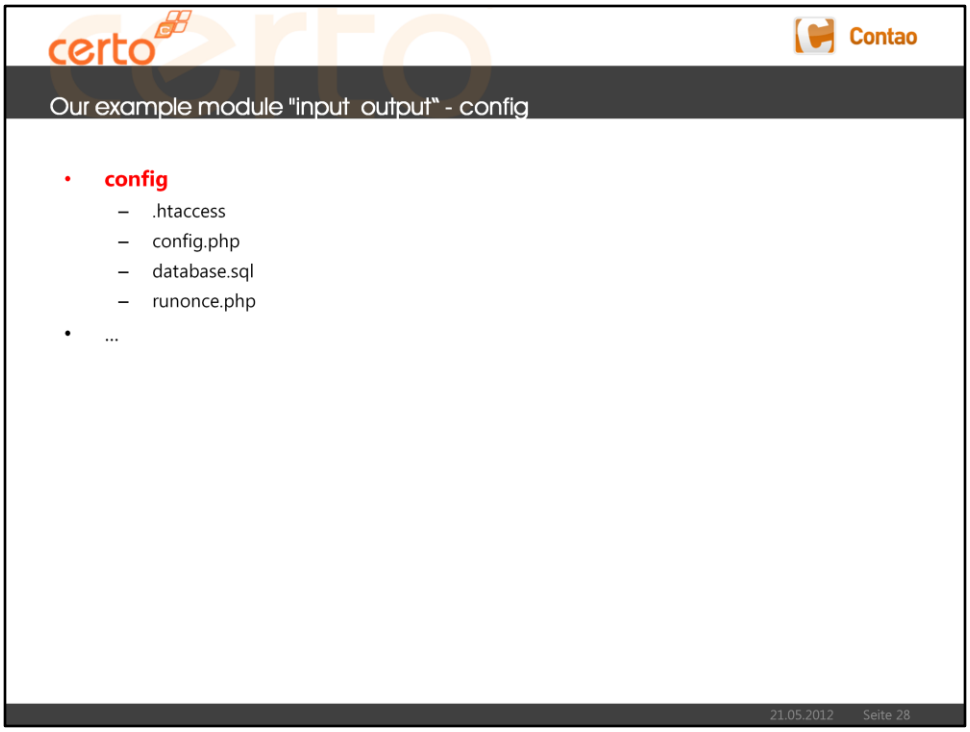
The structure of a custom module



A custom module almost always has the structure presented on this slide. Obviously you might have other directories for plugins you use or something similar and you don‘t necessarily need to have all of these folders for every module.
It depends on what you want to develop.

We will see what goes into the directories on the following slides.

Note that the Contao autoloader will load all the classes located in the module root automatically so that‘s why our classes „ContentInputOutput.php“, „InputOutput.php“ and „ModuleInputOutputList.php“ are in the module root.



The „.htaccess“ denies direct access to the module via the webserver. You‘ll find it in all folders within a module except for those where people need to have access to (css files, images etc.).

The „config.php“ contains the configuration of your module. The main concept behind the config.php is that all the config.php files of all the active modules get loaded when Contao is being initialized before anything else is done.
This means that we can configure all the configuration items that Contao is going to use such as

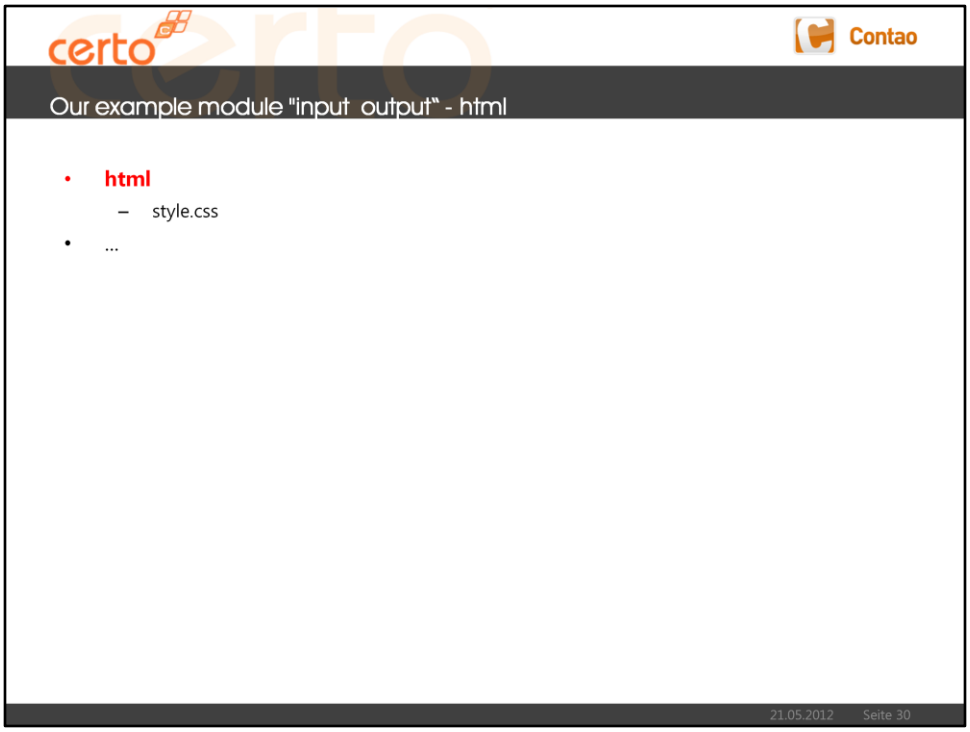
- register/unregister back end modules
- register/unregister front end modules
- register/unregister content element tymes
- register/unregister back end and front end form field types
- register/unregister hooks
- register/unregister page types
- etc.

The „database.sql“ contains the modifications on the database that are needed for our module to work.

The „runonce.php“ file is very useful. It‘s going to be included and thus executed only once and deleted afterwards. You can use this to provide update routines for your rmodules (e.g. database modifications or file adjustments).

The „dca“ directory is the directory where we put our DCA modifications. As we have seen before, Contao checks all the module folders for the corresponding DCA files within this folder when a DataContainer is being loaded.

I put „tl_content“ and „tl_module“ here because you will have to modify them quite often. The „tl_content“ table is the table where content elements are being stored in and the „tl_module“ does the same for the front end modules.

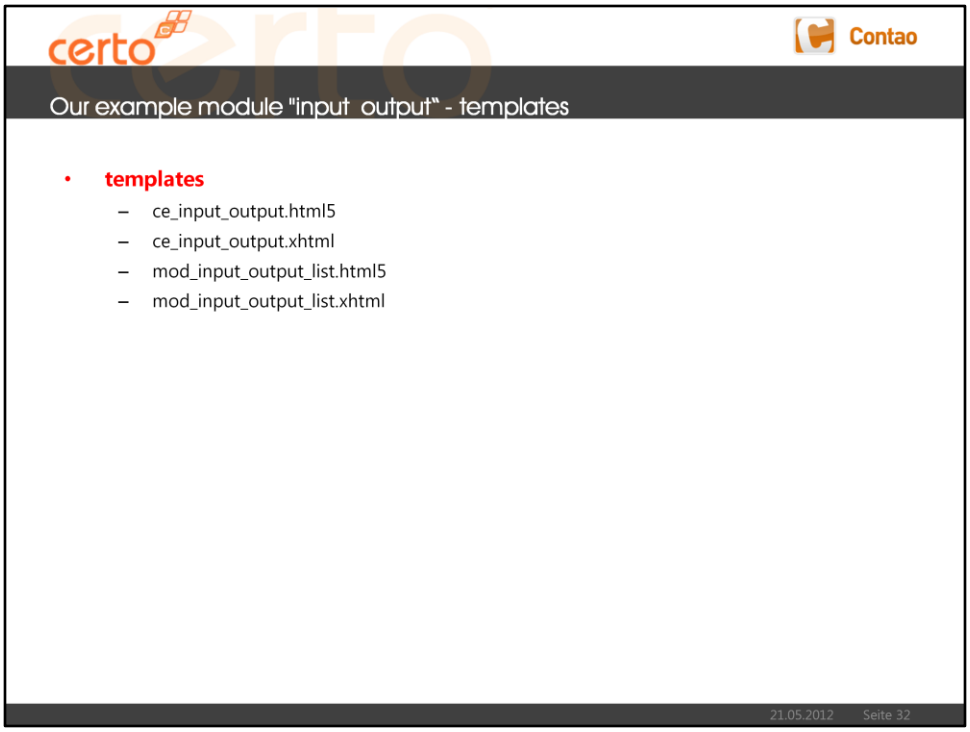


The „html“ folder is a resource directory. You can put your javascript, css and image files in here. The folder doesn‘t necessarily have to be named „html“ but it has become best practice amongst Contao developers.
So it‘s some kind of convention ;-)

The „languages“ directory contains the translations for everything within the module you want to have translatable.
The clever ones amongst you have already noticed that we have the table names for the DCA here again. That‘s right. They‘re automatically loaded when a DataContainer is being initialized.

The other two files are pretty easy:

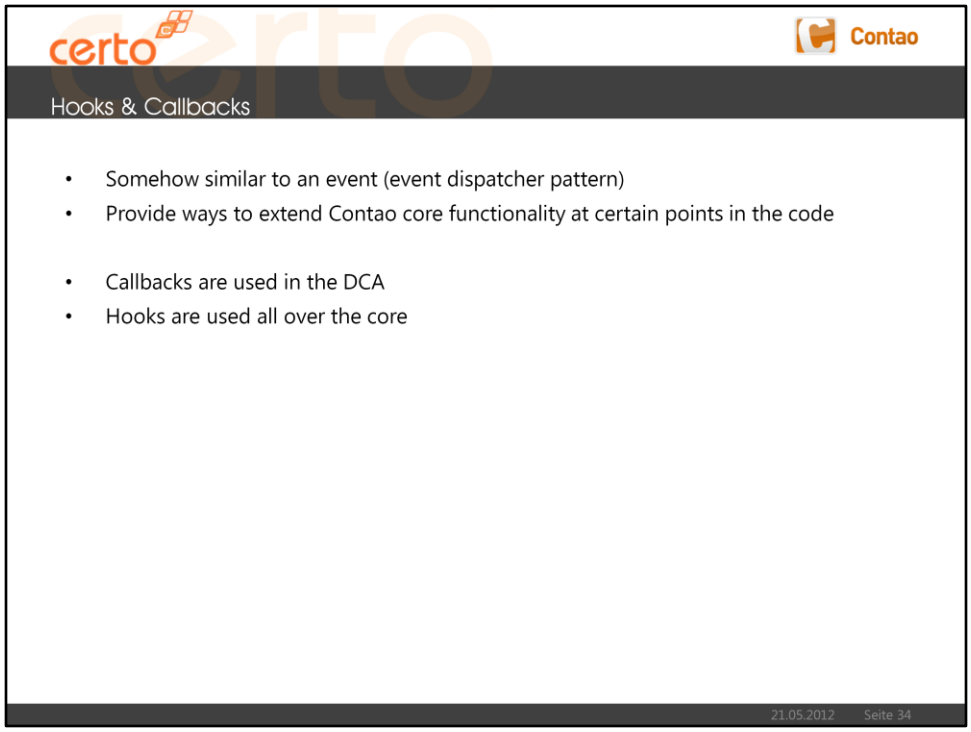
- „default.php“ is always included so you can specify things that have to be available everywhere in here such as error messages or more general things like „yes“, „no“, „probably“ etc.
- „modules.php“ contains the translations for the back and the front end modules.



This is something everybody who reads this presentation should know about: templates.

Here you can define your own templates for your modules or content elements. Obviously you can also override core templates here.
Again there‘s no convention to name a template after a particular pattern but we do begin with „mod_“ for module templates and with „ce_“ for content elements.
You just endear yourself to the other developers if you do ;-)

Hooks & Callbacks



If I had to define the difference between a hook and a callback I'd say they both provide an interface so that other code can extend the functionality at a certain point in the code but whereas a callback requires a return value, a hook doesn't.

But my definition doesn't matter. In Contao they're both merely the same. Some expect a return value, others don't.
But the real difference betwenn them is that callbacks are used within a DCA and hooks all over the core.

certo

Contao

Callbacks - Overview

- child_record_callback
- group_callback
- header_callback
- label_callback
- load_callback
- oncopy_callback
- oncreate_callback
- oncut_callback
- ondelete_callback
- onload_callback
- onrestore_callback
- onsubmit_callback
- onversion_callback
- options_callback
- paste_button_callback
- save_callback

system/modules/news/dca/tl_news.php

```
$GLOBALS['TL_DCA']['tl_news'] = array
(

    // Config
    'config' => array
    (
        'dataContainer'               => 'Table',
        'ptable'                      => 'tl_news_archive',
        'enableVersioning'            => true,
        'onload_callback' => array
        (
            array('tl_news', 'checkPermission'),
            array('tl_news', 'generateFeed')
        ),
```

certo Contao

Hooks - Overview

57 Hooks!

activateAccount	getAllEvents	loadDataContainer	removeOldFeeds
activateRecipient	getArticle	loadFormField	removeRecipient
addComment	getCacheKey	loadLanguageFile	replaceInsertTags
addCustomRegexp	getCombinedFile	outputBackendTemplate	reviseTable
addLogEntry	getContentElement	outputFrontendTemplate	setCookie
checkCredentials	getCountries	parseArticles	setNewPassword
closeAccount	getForm	parseBackendTemplate	sqlCompileCommands
compileDefinition	getFrontendModule	parseFrontendTemplate	sqlGetFromDB
createDefinition	getImage	parseTemplate	sqlGetFromFile
createNewUser	getPageIdFromUrl	postDownload	storeFormData
executePostActions	getRootPageFromUrl	postLogin	updatePersonalData
executePreActions	getSearchablePages	postLogout	validateFormField
generateBreadcrumb	getSystemMessages	postUpload	
generateFrontendUrl	getUserNavigation	printArticleAsPdf	
generatePage	importUser	processFormData	

system/modules/isotope/config/config.php

```php
/**
 * Hooks
 */
$GLOBALS['TL_HOOKS']['loadDataContainer'][]         = array('Isotope', 'loadProductsDataContainer');
$GLOBALS['TL_HOOKS']['addCustomRegexp'][]           = array('Isotope', 'validateRegexp');
$GLOBALS['TL_HOOKS']['getSearchablePages'][]        = array('IsotopeFrontend', 'addProductsToSearchIndex');
$GLOBALS['TL_HOOKS']['replaceInsertTags'][]         = array('IsotopeFrontend', 'replaceIsotopeTags');
$GLOBALS['TL_HOOKS']['generatePage'][]              = array('IsotopeFrontend', 'injectMessages');
$GLOBALS['TL_HOOKS']['executePreActions'][]         = array('ProductTree', 'executePreActions');
$GLOBALS['TL_HOOKS']['executePostActions'][]        = array('ProductTree', 'executePostActions');
```

21.05.2012 Seite 36

In contrast to the callbacks, hooks aren't registered in a DCA file. This (to me) seems quite obvious as DCA files are only loaded when needed and only in a DC view in the back end. Hooks, however, are spread all over Contao and are not limited to the DC. Therefore they have to be registered in the config.php file of the respective module.


certo
Contao
Workshop
21.05.2012
Seite 37

- This slide was in German as the presentation and the workshop were held in German.
- This is the module I created to give you something you can start off with: https://github.com/Toflar/iadwdis_input_output
- Leo Feyer's „Inside TYPOlight" presentation from the user meeting back in 2009 in Frankfurt, Germany: http://www.contao.org/tl_files/meetings/Inside-TYPOlight.pdf
  → A few things are not correct anymore (template loading order, referer check on forms etc.) but it might still be useful. (German only, unfortunately).

Contact

certo web & design GmbH
Yanick Witschi
Bielstrasse 1
CH - 3250 Lyss

www.certo-net.ch
yanick.witschi@certo-net.ch