

## SEO dynamic tag

You can generate values dynamically with this extension especially the <title>-tag, the <meta>-tag description and the <meta>-tag keywords. The <title>-tag can be the title and subtitle of a news in tt\_news or the title and category of a product in tt\_products for e.g. The <meta>-tag can be the short-field of a tt\_news or the description of a product in tt\_products for e.g. You can use the extension for every table and record, of course for tables and records of third party extensions too.



Version: 1.1.1, 2013-01-15

Extension Key: seo\_dynamic\_tag

Language: en

Keywords: forAdmins, forDevelopers, forBeginners, forIntermediates, forAdvanced, SEO, search engine optimisation, tags, meta tags, meta, dynamic

Copyright 2007 - 2013, Dirk Wildt, Die Netzmacher, <wildt.at.die-netzmacher.de>

This document is published under the Open Content License  
available from <http://www.opencontent.org/opl.shtml>

The content of this document is related to TYPO3  
- a GNU/GPL CMS/Framework available from [www.typo3.org](http://www.typo3.org)

## Table of Contents

SEO dynamic tag .....	1	Debug Mode .....	11
Introduction .....	3	How to set up the extension .....	11
What does it do? .....	3	Screenshots .....	12
Requirements .....	3	Reference .....	14
Screenshots .....	4	General Settings .....	14
tt_news .....	4	Query .....	15
tt_products .....	4	Files .....	16
Third Party Extension .....	4	FAQ .....	17
More Screenshots .....	4	SimulateStaticDocuments and RealUrl .....	17
Installation .....	5	Helpful suggestions .....	18
Quick start .....	5	Further Information .....	19
Extension Manager .....	5	About the plugin icon .....	19
Tutorial .....	6	Other extensions published by Die Netzmacher .....	19
How to set up the extension .....	6	To-Do list .....	20
tt_news .....	6	Changelog .....	21
tt_products .....	8	Illustration Index .....	22
Third Party Extension .....	9		
Keywords Improvement .....	10		

# Introduction

## What does it do?

You can generate values dynamically with this extension especially

- the <title>-tag
- the <meta>-tag description and
- the <meta>-tag keywords

The <title>-tag can be the title and subtitle of a news in tt\_news or the title and category of a product in tt\_products for e.g.

The <meta>-tag can be the short-field of a tt\_news or the description of a product in tt\_products for e.g.

You can use the extension for every table and record, of course for tables and records of third party extensions too.

## Requirements

- Nothing but a little experience in typoscript and the syntax of SQL queries.



# Screenshots

## tt\_news



Illustration 1: Headlines of a news and the browser-title



Illustration 2: News in the frontend and the HTML code

## tt\_products



Illustration 3: Category and title of a product and browser title



Illustration 4: product in the frontend and the HTML code

## Third Party Extension



Illustration 5: Headlines of a 3. party extension and browser title



Illustration 6: Third party extension and the HTML code

## More Screenshots

See "Screenshots" on page 12.

# Installation

## Quick start

- Install the extension with the extension manager.
- Configure the setup of the page, where you want to control title tags or other tags.

## Extension Manager

Open the extension manager and download the SEO-dynamic-tag extension (key seo\_dynamic\_tag) and install it:

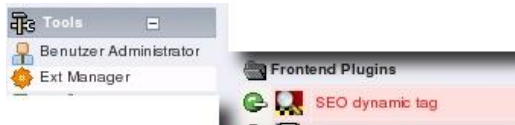


Illustration 7: Extensionmanager

# Tutorial

## How to set up the extension

Please read the section "How to set up the extension" in the chapter "Debug" on page 11.

### tt\_news

The result on the page with the single view of a news should be:

- The browser title is composed of the title (1) and the subtitle (2) of the news.
- That the <meta>-tag description (3) is the value of the short (3) field.
- That the <meta>-tag keyword (4) is the value of the title (1) and the subtitle (2) of the news.

We need the following typoscript setup

```
temp.seo = COA
temp.seo {
    10 < plugin.tx_seodynamictag_pi1
    10 {
        # Example for the page title
        special = title
        query {
            select = CONCAT(`title`, ': ', `short`)
            from = `tt_news`
            where = `uid` = $1 && `deleted` = 0 && `hidden` = 0
            var.1 = tx_ttnews[tt_news]
        }
    }
    20 < plugin.tx_seodynamictag_pi1
    20 {
        # Example for a meta tag
        special = register
        register = description
        query {
            select = `bodytext`
            from = `tt_news`
            where = `uid` = $1 && `deleted` = 0 && `hidden` = 0
            var.1 = tx_ttnews[tt_news]
            maxLength = 200
        }
    }
    30 < plugin.tx_seodynamictag_pi1
    30 {
        # Example for a meta tag
        special = register
        register = keywords
        query {
            select = CONCAT(`title`, ' ', `short`)
            from = `tt_news`
            where = `uid` = $1 && `deleted` = 0 && `hidden` = 0
            var.1 = tx_ttnews[tt_news]
        }
        keywords = 1
        keywords {
            amount = 10
            minLength = 8
            positiveList = CMC, Nokia
        }
    }
}
```

The temp.seo object is the last content element of our page.

It has to be the last element, because we have to avoid, that other content elements override our browser page title or our <meta>-tags.

```
page {
    ...
    10 {
        subparts {
```



Illustration 8: The result

```

content = COA
# The single view of tt_news
content.20 < temp.single
# If the seo-plugin should overwrite values like the page title,
# the plugin have to be the last content element
content.1000 < temp.seo
...

```

## General Explanation

We choose a Content Object Array (COA), because we want more than one thing.

```

temp.seo = COA
temp.seo {
    10 < plugin.tx_seodynamictag_pi1
    ...

```

We set up the three arrays

- 10 for the browser title and
- 20 for the <meta>-tag description and
- 30 for the <meta>-tag keywords.

```

temp.seo = COA
temp.seo {
    10 < plugin.tx_seodynamictag_pi1
    ...
    20 < plugin.tx_seodynamictag_pi1
    ...

```

The special value defines the properties.

```

10 < plugin.tx_seodynamictag_pi1
10 {
    # Example for the page title
    special = title
    ...

```

There are three possibilities:

- title
- register
- no special value

"Title" effects, that the extension is overwriting the browsers <title>-tag. You can use this only for the browser title.

"Register" enables, that the needed value will be stored in a typo3 register and will be used later. We need this for every <meta>-tag in the HTML head.

The waiving of the special value effects, that the needed value will be echoed in the content area of our website. Though this is the default, but it seems to have less sense.

## The property "query"

The plugin.tx\_seodynamictag\_pi1 has the property "query". The syntax of this property is the syntax of SQL.

The typescript setup code:

```

query {
    select = CONCAT(`title`, ': ', `short`)
    from   = `tt_news`
    where  = `uid` = $1 && `deleted` = 0 && `hidden` = 0
    var.1 = tx_ttnews[tt_news]

```

The property query is very powerful, because you can define variables, which will get a value at runtime.

You can use a variable with any name in your SQL query everywhere.

- You have to label the variable with a dollar character (\$).
- You have to define the array var with an element named with your variable.
- Then you have to allocate the global get variable or the global post variable.

In the example above we defined the variable \$1 in the where clause (red coloured).



In the array var we defined the variable \$1 a second time as the element 1: var.1 (red coloured).

Then we allocate the value of global var tx\_ttnews[tt\_news] (red coloured).

If you don't know, which GET variables or POST variables are available use the debug mode. The mode will display the array.

```
special = title
debug = 1
query {
    ...
```

## tt\_products

Thank you for understanding, that we don't repeat explanations from above.

If you aren't close with the example tt\_news above, please read it first.

The result on the page with the single view of a product should be:

- The browser title is composed of the title (1) and the category (2) of the product.
- That the <meta>-tag description (3) is the value of the subtitle (3) field.
- That the <meta>-tag keyword (4) is a list of the words "Dentalgeräte", the words of the product title (1) and the product category (2).

We need the following typoscript setup:

```
temp.seo = COA
temp.seo {
    10 < plugin.tx_seodynamictag_pil
    10 {
        # Example for the page title
        special = title
        query {
            select = CONCAT(product.title, ' ', category.title, ' ') as value
            from = tt_products as product, tt_products_cat as category
            where = product.uid = $1 AND product.category = category.uid \
                AND product.deleted = 0 AND product.hidden = 0
            var.1 = tx_ttproducts_pil[product]
        }
    }
    20 < plugin.tx_seodynamictag_pil
    20 {
        # Example for a meta tag
        special = register
        register = description
        query {
            select = `subtitle`
            from = `tt_products`
            where = `uid` = $1 && `deleted` = 0 && `hidden` = 0
            var.1 = tx_ttproducts_pil[product]
            maxLength = 200
        }
    }
    30 < plugin.tx_seodynamictag_pil
    30 {
        # Example for a meta tag
        special = register
        register = keywords
        query {
            select = CONCAT('Dentalgerät ', product.title, ' ', category.title) as value
            from = tt_products as product, tt_products_cat as category
            where = product.uid = $1 AND product.category = category.uid \
                AND product.deleted = 0 AND product.hidden = 0
            var.1 = tx_ttproducts_pil[product]
        }
        keywords = 1
        keywords {
            amount = 10
            minLength = 8
            positiveList = CMC, Nokia
        }
    }
}
```



Illustration 9: The result



```
}
}
```

The temp.seo object is the last content element of our page.

It has to be the last element, because we have to avoid, that other content elements override our browser page title or our <meta>-tags.

```
page {
  meta {
    description {
      field >
      data = register:description
    }
    keywords {
      field >
      data = register:keywords
    }
  }
}
10 < styles.content.get
1000 < temp.seo
}
```

The setup code is nearly the same like in the section tt\_news above (page 6), but the select clause for the browsers page title and the keywords.

```
query {
  select = CONCAT('Dentalgerät ', product.title, ' ', category.title) as value
  from   = tt_products as product, tt_products_cat as category
  where  = product.uid = $1 AND product.category = category.uid \
        AND product.deleted = 0 AND product.hidden = 0
}
```

Yes, it's possible to link tables!

The result is composed with values of records out of two tables.

## Third Party Extension

Thank you for understanding, that we don't repeat explanations from above.

If you aren't close with the example tt\_news above, please read it first.

The result on the page should be:

- The browser title is composed of the title (1) and the subtitle (2) of the displayed news.
- That the <meta>-tag description (3) is the value of the short (3) field.
- That the <meta>-tag keyword (4) is the value of the title (1) and the subtitle (2) of the news.

The setup code is nearly the same like in the section tt\_news above (page 6), but we have

- a different table name,
- different field names
- and a different global get variable.

```
temp.seo = COA
temp.seo {
  10 < plugin.tx_seodynamictag_pi1
  10 {
    # Example for the page title
    special = title
    query {
      select = CONCAT(`article_caption`, ': ', `article_roof`)
      from   = `tx_hptazarticle_list`
      where  = `uid` = $1 && `deleted` = 0 && `hidden` = 0
      var.1  = art
    }
  }
}
20 < plugin.tx_seodynamictag_pi1
20 {
  # Example for a meta tag
  special = register
}
```



Illustration 10: The result

```

register = description
query {
  select = `article_teaser_short`
  from   = `tx_hptazarticle_list`
  where  = `uid` = $1 && `deleted` = 0 && `hidden` = 0
  var.1  = art
}
}
30 < plugin.tx_seodynamictag_pi1
30 {
  # Example for the page title
  special = register
  register = keywords
  query {
    select = CONCAT(`article_caption`, ': ', `article_roof`)
    from   = `tx_hptazarticle_list`
    where  = `uid` = $1 && `deleted` = 0 && `hidden` = 0
    var.1  = art
  }
  keywords = 1
  keywords {
    amount      = 10
    minLength   = 8
    positiveList = taz, BILD, SZ, Zeit, Spiegel
  }
}
}

```

You see, that it is very easy to use the SEO dynamic extension for third party extensions.

## Keywords Improvement

You can improve your keywords since version 0.0.3.

This is the process of keyword handling:

- Keywords will be counted.
- The first words in the meta tag "keywords" will be the most frequented of the words
- You are controlling the weight with the frequency of words by repeating a field in the SQL query.

### Example for an SQL query

```

...
query {
  select = CONCAT(`title`, ' ', `title`, ' ', `title`, ' ', `title`, ' ', `short`)
  ...
}

```

The words in the field title will have the fourfold weight than the words in the field short, because the field title is repeated four times in the SQL query.

# Debug Mode

## How to set up the extension

If you don't know, how to set up the extension "SEO dynamic tag", or if there will be any failure, it is very helpful to activate the debug mode.

You will receive a lot of information like in our illustration above (page 12) about

- your typoscript code,
- the typoscript code changed by the extension,
- the success of substituting the used variables,
- the array with the available globals,
- the query string in the SQL database and
- the result.

Turn the debug mode on:

```
plugin.tx_seodynamictag_pil.debug = 1
```

Or in our tutorial examples above

```
temp.seo = COA
temp.seo {
    10 < plugin.tx_seodynamictag_pil
    10 {
        # Example for the page title
        special = title
        debug = 1
        query {
            ...
        }
    }
}
```

## Everything is ok but there is a failure

If you have a failure like no effect in the page title, but everything in the report is ok, than there will be only one cause:

The extension "SEO dynamic tag" isn't the last page element, another page element is overriding the page title or <meta>-tags.

Example code for page element:

```
page {
    ...
    10 < your_content
    # The seo-plugin have to be the last content element
    1000 < plugin.tx_seodynamictag_pil
    1000 {
        # Example for the page title
        special = title
        query {
            ...
        }
    }
}
```

Or in our tutorial examples above

```
page {
    ...
    10 {
        subparts {
            content = COA
            # Single view of a news in tt_news
            content.20 < temp.single
            # The seo-plugin have to be the last content element
            content.1000 < temp.seo
        }
    }
}
```

## Screenshots

### Example Page Title



Illustration 11: Values of the fields title and short of a tt\_news and the page title

The values of the fields title (1) and short (2) of a tt\_news and the values in the page title.

### The debug Report

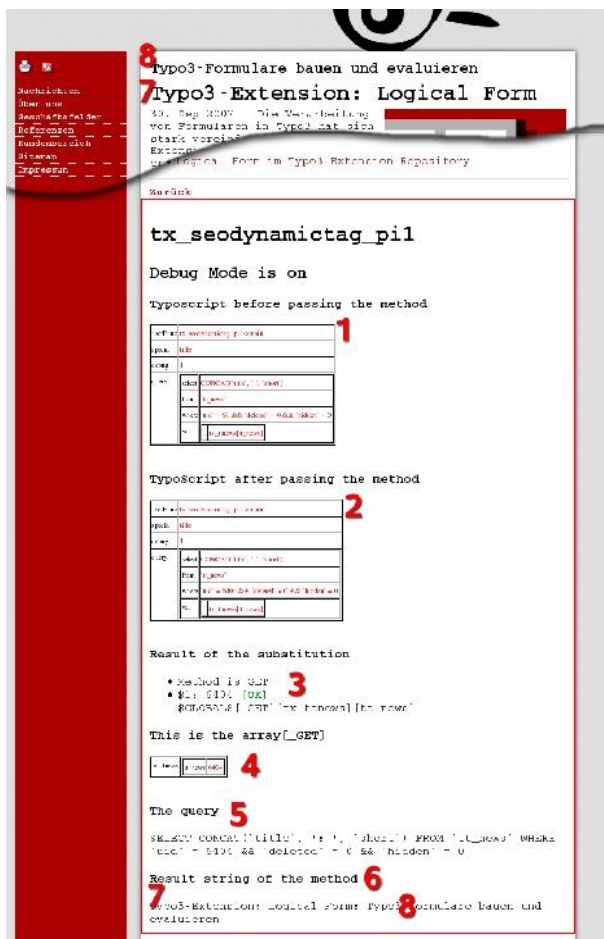


Illustration 12: Report in the debug mode

The same in the report.

#### Report structure

1. Your typoscript code.
2. The typoscript code changed by the extension.
3. The result of Substitution:
  - There is an "OK", if the global variable is available.
  - There is an "DANGER", if the global variable isn't available.
4. The array with the available globals.
5. The SQL query string. If you don't know, if the query has the valid syntax, copy the query to PhpMyAdmin e.g. and test it.
6. The result value. In this example:
  1. tt\_news.title and
  2. tt\_news.short

### SQL error message

If there is any syntax error in your query, you get an SQL error message, but only in debug mode.

```

File: /opt/typo3_src-4.1.2/typo3/sysext/core/Classes/Controller/Backend/Controller.php
Line: 2907
Warning: Cannot modify header information - Headers already sent by (see past errors) at
/opt/typo3_src-4.1.2/typo3/sysext/core/Classes/Controller/Backend/Controller.php on line
2907

```

Illustration 13: SQL error message

# Reference

**plugin.tx\_seodynamictag\_pi1 properties: TS configuration.**

## General Settings

Property:	Data type:	Description:	Default:
debug	boolean	Debug mode (see "Debug Mode" on page 11 for any explanation) <b>Example:</b> <pre>plugin.tx_seodynamictag_pi1 {     debug = 1 }</pre>	0
special	string: <i>title,</i> <i>register</i>	There are two special values title and register (see "General Explanation" on page 7). In the default mode the special value is empty.  Special value "title": The result of the query will be written in the browsers page title. <b>Example 1:</b> <pre>plugin.tx_seodynamictag_pi1 {     special = title     query {         ...     } }</pre> Special value "register": The result of the query will be written in a first step in the register with the name "my_name" and than in a second step in <meta>-tag description. <b>Example 2:</b> <pre>plugin.tx_seodynamictag_pi1 {     special = register     register = my_name     query {         ...     } }</pre> <pre>page.meta.description {     # Delete the default value     field &gt;     data = register:my_name }</pre>	empty
keywords	boolean	If set, the result will be changed in a comma seperated list of words <b>Example:</b> <pre>plugin.tx_seodynamictag_pi1 {     keywords = 1 }</pre>	0
keywords.minLength	int	The length of a keyword at least. <b>Example:</b> <pre>plugin.tx_seodynamictag_pi1 {     keywords.minLength = 6 }</pre>	4
keywords.amount	int	The maximum amount of keywords <b>Example:</b> <pre>plugin.tx_seodynamictag_pi1 {     keywords.amount = 12 }</pre>	10
keywords.positiveList	comma seperated values/string	The positive of possible keywords which hasn't the minimum length. <b>Example:</b> <pre>plugin.tx_seodynamictag_pi1 {     keywords.positiveList = VCD, ABS }</pre>	empty
query	->QUERY	SQL query with typoscript properties. <b>Example:</b> <pre>plugin.tx_seodynamictag_pi1.query {     select = CONCAT(`title`, ': ', `short`)     from = `tt_news`     where = `uid` = \$1     var.1 = tx_ttnews[tt_news] }</pre>	empty

## Query

Property:	Data type:	Description:	Default:
select	string	<p>SQL select command.</p> <p><b>Example 1:</b></p> <pre>plugin.tx_seodynamictag_pil.query {     select = CONCAT(`title`, ': ', `short`)     from   = `tt_news`     where  = `uid` = \$1     var.1  = tx_ttnews[tt_news] }</pre> <p><b>Example 2:</b></p> <pre>plugin.tx_seodynamictag_pil.query {     select = CONCAT(product.title, ' \     (' , category.title, ')') as value     from   = tt_products as product, \     tt_products_cat as category     where  = product.uid = \$1 \     AND product.category = category.uid \     AND product.deleted = 0 \     AND product.hidden = 0     var.1  = tx_ttproducts_pil[product] }</pre>	empty
from	string	SQL table name. See example "select" above.	empty
where	string	SQL where expression. See example "select" above.	empty
var	array	<p>Variable in a SQL query.</p> <p>You can use a variable with any name in your SQL query everywhere.</p> <ul style="list-style-type: none"> <li>You have to label the variable with a dollar character (\$).</li> <li>You have to define the array var with an element named with your variable.</li> <li>Then you have to allocate the global get variable or the global post variable.</li> </ul> <p>If you need a post variable see property "method" below.</p> <p>See the example in the tutorial 'The property "query"' on page 7.</p> <p><b>Example:</b></p> <pre>plugin.tx_seodynamictag_pil.query {     ...     where          = `uid` = \$my_var AND \     `pid` = \$my_other_var     var.my_var      = tx_ttnews[tt_news]     var.my_other_var = tx_ttnews[pid] }</pre> <p><i>It's only an example. There is no global var tx_ttnews[pid]</i></p>	empty
method	string: <i>get, post</i>	<p>You can choose the global array GET or POST for substituting variables in your SQL query.</p> <p><b>Example:</b></p> <pre>plugin.tx_seodynamictag_pil.query {     method = post }</pre>	get
maxLength	integer	<p>If set, the result will be cut after maxLength chars.</p> <p><b>Example:</b></p> <pre>plugin.tx_seodynamictag_pil.query {     maxLength = 100 }</pre>	0
dontStripTags	boolean	<p>If set, the result value won't be stripped of it's HTML-tags.</p> <p><b>Example:</b></p> <pre>plugin.tx_seodynamictag_pil.query {     dontStripTags = 1 }</pre>	0

## Files

Files for the configuration of language items and for documentation.

File:	Description:
pi1/class.tx_seodynamictag_pi1.php	Main PHP-class used to generate the browsers page title or the <meta>-tags description and keywords dynamically.
ext_php_api.dat	API documentation of the seo_dynamic_tag extension.



## FAQ

### SimulateStaticDocuments and RealUrl

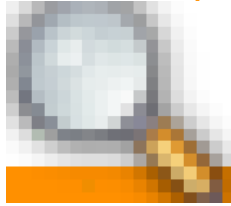
The extension is compatible. In RealUrl you can't see the global get variables. The extension displays this variables in debug mode (see page 11).

## Helpful suggestions

If you have helpful suggestions, feel free to contact us: Dirk Wildt, [dirk.wildt@think-visually.com](mailto:dirk.wildt@think-visually.com).

## Further Information














### About the plugin icon



The concept of the icon:

- The reading-glass is symbol for a search machine.
- The color of the stripe, orange, is the color of the corporate design of *Die Netzmacher*, the company which created the extension SEO dynamic tag.

### Other extensions published by Die Netzmacher

-  autositemap: A smart site-map optimised for the footer. It groups menus in columns. Great menus will get two columns. Configuration is based on TypoScript HMENU.  
<http://typo3.org/extensions/repository/view/autositemap/>
-  Browser – TYPO3 without PHP. Develop your TYPO3 extension 8 times faster! You need 1 line TypoScript for a result list with a search form, a record browser and an index browser. Images are wrapped self-acting. SEO, Search Engine Optimization, is integrated. <http://typo3.org/extensions/repository/view/browser/>
-  Flip it! offers lovely and smooth page flip transitions. It enables you to run over pages in PDF documents like in a real magazine. It is based on flash. Flipt it! can convert PDF documents to swf files automatically.<sup>1</sup>  
<http://typo3.org/extensions/repository/view/flipit/>
-  Green Cars (Grüne Autos) - Database optimized for ecological cars. It is a case study and demonstrates, how to get a complex database with the browser (see above) in three hours only.  
[http://typo3.org/extensions/repository/view/green\\_cars/](http://typo3.org/extensions/repository/view/green_cars/)
-  Jobmarket is a catalogue with job offers. Views, the a-z-browser, the page-browser, the search, social bookmarks and a lot of other stuff can be configured by the Browser plugin with the mouse. Job Market hasn't any PHP code, it should be easy to adapt it to your needs.  
[http://typo3.org/extensions/repository/view/job\\_market/](http://typo3.org/extensions/repository/view/job_market/)
-  Majordomo: For subscribing to and unsubscribing from a majordomo mailing list.  
<http://typo3.org/extensions/repository/view/majordomo/>
-  Organiser provides a lot of features for handle news, events, staff, headquarters, locations, repertoire, workshops and a calendar. Sell your online tickets! Available from April of 2011  
<http://typo3-organiser.de/>
-  Quick Shop (extkey: quick\_shop): The fastest shop in the history of TYPO3. Install it with one mouse click only! Quick Shop is based on the browser (see above) and powermail.  
<http://typo3-quick-shop.de/>
-  PDF Controller: Easy to install. Add to your HTML page the PDF-controller-button. Link from the button to the controller. Adjust the controller by mouseclicks. The PDF Controller supports CSS 3.  
<http://typo3-pdfcontroller.de/>
-  seo\_dynamic\_tag: Search Engine Optimisation. You can generate values dynamically with this extension especially for the <title>-tag, for the <meta>-tag description and the <meta>-tag keywords.  
[http://typo3.org/extensions/repository/view/seo\\_dynamic\\_tag/](http://typo3.org/extensions/repository/view/seo_dynamic_tag/)
-  TSconfig Pages and Users by extManager (extkey: tsconf): Configure the the eight most commonly used TSconfig porperities with the mouse - like page tree uids, activated extended view, activated clipboard, ...  
<http://typo3.org/extensions/repository/view/tsconf/>
-  tt\_news select configuration (extkey: ttnews\_selectconf) enables to select tt\_news by any SQL clause. The extension adds an andWhere clause to the SQL query of the tt\_news plugin.  
[http://typo3.org/extensions/repository/view/ttnews\\_selectconf/](http://typo3.org/extensions/repository/view/ttnews_selectconf/)
-  Wine Catalogue provides a data base for wine with regions, wineries, styles, variety and ageing among others. It is localized. English, German and Spanish ist of the box. Wine based on the extension browser (see above).  
<http://typo3.org/extensions/repository/view/wine/>

<sup>1</sup> We purpose to publish Flip it! in the TYPO3 repository in April of 2013

## To-Do list

Nothing to do.

# Changelog

- 1.1.1 **Bugfix**
  - \* #44545: Class 't3lib\_utility\_Debug' not found  
Downgrade for TYPO3 4.4  
Effected file:  
\* pi1/class.tx\_seodynamictag\_pi1.php
- 1.1.0 **Improvement**
  - \* #42405: t3lib\_div::view\_array is deprecated  
Update for TYPO3 4.7  
Effected file:  
\* pi1/class.tx\_seodynamictag\_pi1.php
- 1.0.0 **New Main Version**
  - Corporate Design Die Netzmacher GbR
- 0.0.5 **Improvement**
  - Documentation
- 0.0.3 **Improvement**
  - Keywords
- 0.0.2 **New Feature**
  - Keyword configuration
- 0.0.1 **Initial release**

# Illustration Index

Illustration 1: Headlines of a news and the browser-title.....	4
Illustration 2: News in the frontend and the HTML code.....	4
Illustration 3: Category and title of a product and browser title.....	4
Illustration 4: product in the frontend and the HTML code.....	4
Illustration 5: Headlines of a 3. party extension and browser title.....	4
Illustration 6: Third party extension and the HTML code.....	4
Illustration 7: Extensionmanager.....	5
Illustration 8: The result.....	6
Illustration 9: The result.....	8
Illustration 10: The result.....	9
Illustration 11: Values of the fields title and short of a tt_news and the page title.....	12
Illustration 12: Report in the debug mode.....	12
Illustration 13: SQL error message.....	13