

# EXT: SEO dynamic tag

Extension Key: **seo\_dynamic\_tag**

Copyright 2007

Dirk Wildt, <dirk.wildt@think-visually.com>

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.com](http://www.typo3.com)

## Table of Contents

<b>EXT: SEO dynamic tag</b> .....	<b>1</b>	Third Party Extension .....	6
<b>Introduction</b> .....	<b>1</b>	<b>Debug Mode</b> .....	<b>8</b>
What does it do? .....	1	How to set up the extension .....	8
Requirements .....	1	Screenshots .....	9
<b>Screenshots</b> .....	<b>2</b>	<b>Reference</b> .....	<b>10</b>
tt_news .....	2	General Settings .....	10
tt_products .....	2	Query .....	10
Third Party Extension .....	2	<b>Files</b> .....	<b>11</b>
More Screenshots .....	2	<b>FAQ</b> .....	<b>11</b>
<b>Installation</b> .....	<b>3</b>	SimulateStaticDocuments and RealUrl .....	11
Quick start .....	3	<b>Helpful suggestions</b> .....	<b>11</b>
Extension Manager .....	3	<b>Further Information</b> .....	<b>12</b>
<b>Tutorial</b> .....	<b>3</b>	About the plugin icon .....	12
How to set up the extension .....	3	Other extensions published by think visually .....	12
tt_news .....	3	To-Do list .....	12
tt_products .....	5	<b>Changelog</b> .....	<b>12</b>

## 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

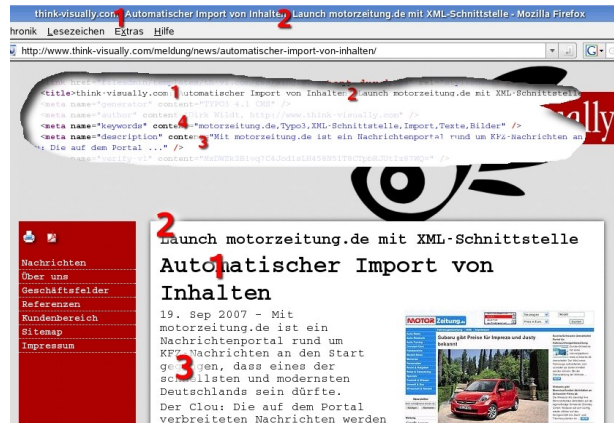


Illustration 2: News in the frontend and the HTML code

## tt\_products



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

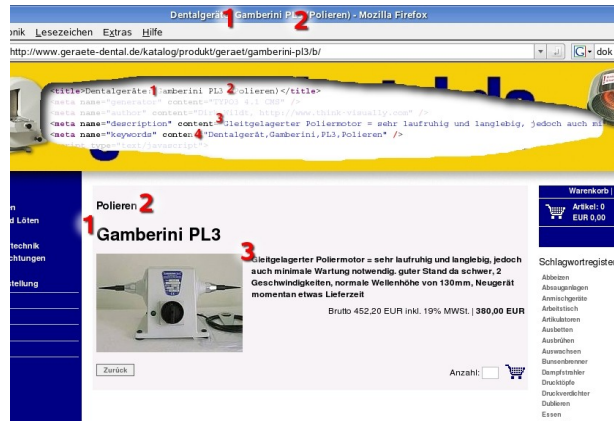


Illustration 4: product in the frontend and the HTML code

## 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 9.

# 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:

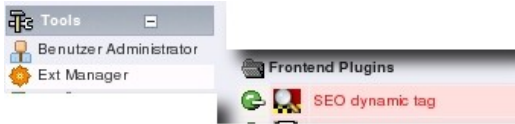


Illustration 7: Extensionmanager

# Tutorial

## How to set up the extension

Please read the section "How to set up the extension" in the chapter "Debug" on page 8.

## 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`)
```



Illustration 8: The result

```

        from = `tt_news`
        where = `uid` = $1 && `deleted` = 0 && `hidden` = 0
        var.1 = tx_ttnews[tt_news]
        keywords = 1
    }
}

```

The temp.seo object ist 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 {
            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 sepcial 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 typoscript 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

```

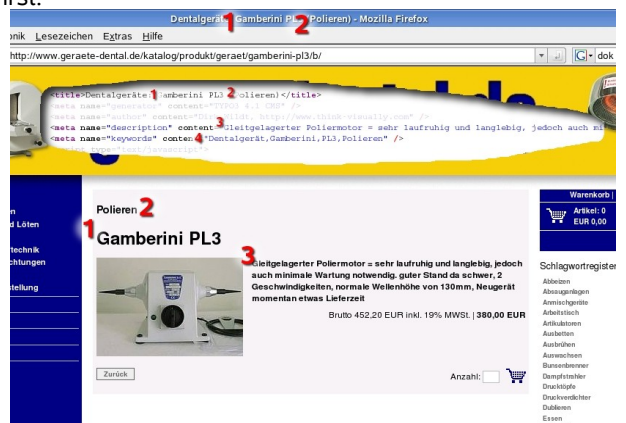


Illustration 9: The result

```

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
}
}
}

```

The temp.seo object ist 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 3), 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 3), 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_pil
  10 {
    # Example for the page title
    special = title
    query {
      select = CONCAT(`article_caption`, ': ', `article_roof`)
    }
  }
}

```

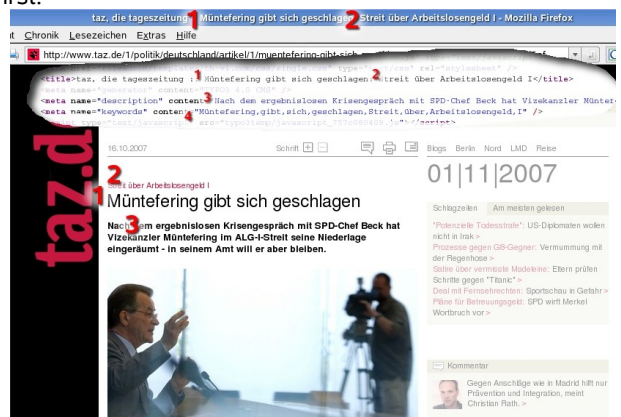


Illustration 10: The result

```

        from = `tx_hptazarticle_list`
        where = `uid` = $1 && `deleted` = 0 && `hidden` = 0
        var.1 = art
    }
}
20 < plugin.tx_seodynamictag_pil
20 {
    # Example for a meta tag
    special = register
    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_pil
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
    }
}
}
}

```

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

# 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 9) 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:

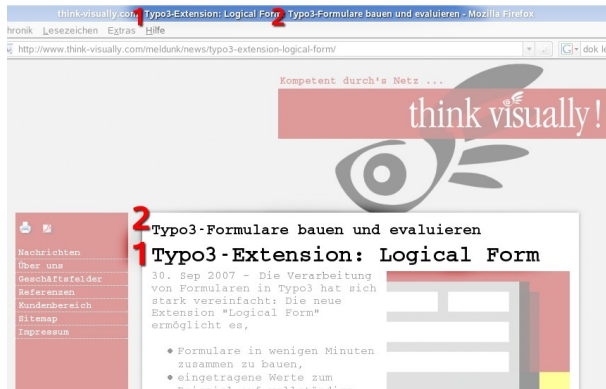
```
page {
    ...
    10 < your_content
    # The seo-plugin have to be the last content element
    1000 < plugin.tx_seodynamictag_pil
    1000 {
        # Example for the page title
        special = title
        query {
            ...
        }
    }
}
```

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

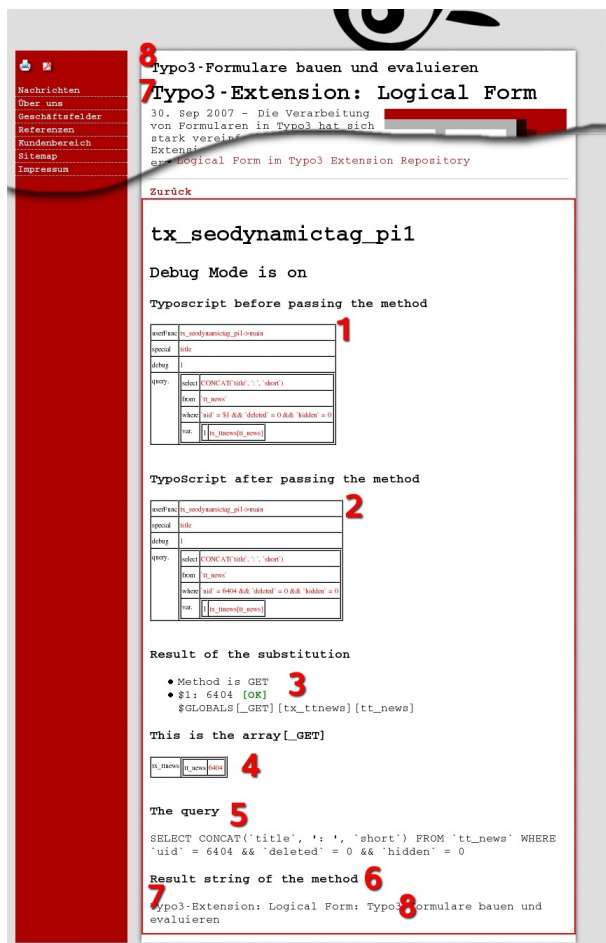
## Example Page Title



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

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

## The debug Report



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:
7. tt\_news.title and
8. tt\_news.short

Illustration 12: Report in the debug mode

## SQL error message

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



Illustration 13: SQL error message

# Reference

plugin.tx\_logicalform\_pi1 properties: TS configuration.

## General Settings

Property:	Data type:	Description:	Default:
debug	boolean	Debug mode (see "Debug Mode" on page 8 for any explanation) <b>Example:</b> <pre>plugin.tx_logicalform_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 4). 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_logicalform_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_logicalform_pi1 {     special = register     register = my_name     query {         ...     } }</pre> <pre>page.meta.description {     # Delete the default value     field &gt;     data = register:my_name }</pre>	empty
query	->QUERY	SQL query with typoscript properties. <b>Example:</b> <pre>plugin.tx_logicalform_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	SQL select command. <b>Example 1:</b> <pre>plugin.tx_logicalform_pi1.query {   select = CONCAT(`title`, ': ', `short`)   from   = `tt_news`   where  = `uid` = \$1   var.1  = tx_ttnews[tt_news] }</pre> <b>Example 2:</b> <pre>plugin.tx_logicalform_pi1.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_pi1[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	Variable in a SQL query. You can use a variable with any name in your SQL query everywhere. <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> If you need a post variable see property "method" below. See the example in the tutorial 'The property "query"' on page 4.  <b>Example:</b> <pre>plugin.tx_logicalform_pi1.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> <i>It's only an example. There is no global var tx_ttnews[pid]</i>	empty
method	string: <i>get, post</i>	You can choose the global array GET or POST for substituting variables in your SQL query. <b>Example:</b> <pre>plugin.tx_logicalform_pi1.query {   method = post }</pre>	get
keywords	boolean	If set, the result will be changed in a comma separated list of words <b>Example:</b> <pre>plugin.tx_logicalform_pi1.query {   keywords = 1 }</pre>	0
maxLength	integer	If set, the result will be cut after maxLength chars. <b>Example:</b> <pre>plugin.tx_logicalform_pi1.query {   maxLength = 100 }</pre>	0
dontStripTags	boolean	If set, the result value won't be stripped of it's HTML-tags. <b>Example:</b> <pre>plugin.tx_logicalform_pi1.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 8).

## 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 stripe "follow me" is symbolizing content management as an powerful help.
- The colour red is symbolizing the colour of think visually, the company which created the seo\_dynamic\_tag extension.

### Other extensions published by think visually

-  logical\_form: A frontend plugin for evaluating forms
-  majordomo: A web frontend for subscribing in majordomo lists and for unsubscribing from majordomo lists.

### To-Do list

- Nothing to do.

## Changelog

- 0.0.1: Initial release