EXT: Shop System

Extension Key: **tt_products**

The content of this document is related to TYPO3

- a GNU/GPL CMS/Framework available from www.typo3.com

Table of Contents

Introduction

You should read the German tutorial or the book 'Der TYPO3 Webshop', which contains many necessary hints for beginners, before you start.

Version

This document is for version 2.6.0 of tt_products. The next version can be ordered from ttproducts.org and ttproducts.de .

Translations

A German translation of this document is available under the extension key doc_ttproducts_de. A French translation is under development at doc_ttproducts_fr.

Upgrade

If you upgrade from Version 1.2.7 you have to do all the administrative steps under the topic 'Important'.

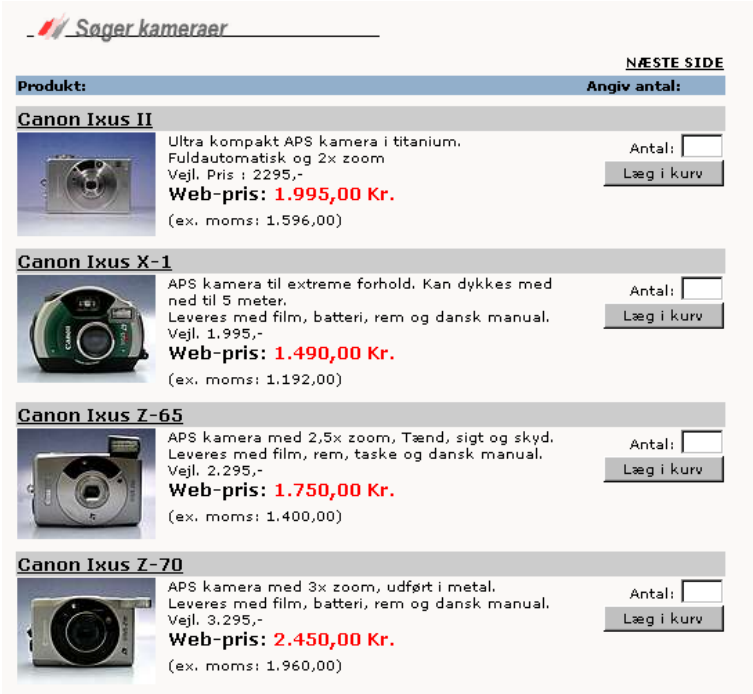
Starting with tt_products 2.7.0 PHP5 is a requirement.

What does it do?

The Typo3 shop extension gives you the facility for...

- Product listings with multiple images, details and languages
- Shopping basket
- Payment page - The orders will be indicated and can be checked over before the products are finalized.
- sponsors only: Payment gateways with Payment Library extension - Paypal and Transaction Central
- Tracking customers order status
- Automatic creation of bill and delivery sheet
- Different tax percentages per item, shipping and payment
- basic stock management
- Send a CSV for each order to the shop admin (2 choosable file formats)
- E-Mail-Attachments for the confirmation mails (for example AGB in German = General trading conditions)
- choosable item variants (colors, sizes, gradings, description, materials and qualities)
- Force customer to accept the General trading conditions (AGB) per checkbox
- Offers, highlights and newly added items
- Special preparation, weight and bulkily (can be used to calculate the shipping fee)
- Ability to limit payment methods to specific user groups
- Automatic creation of frontend users at first order
- Remember items in a memo, when a user is logged in
- Discount percentage per user
- Some methods for price calculation with rebate for resellers
- Display orders: order can be displayed on per fe-user basis (CODE=ORDERS)
- Creditpoint system: customers can save credit points per each order. Saved points will give them a discount for newer orders or certain products can be "bought" with these points.
- Voucher system: if a new customer indicates when registrating that she/he was tipped by another existing customer, this customers gets a credit point bonus. The new customer gets a discount on first order.
- Gift certificate: Users can buy certificates and send them as gifts to their friends. After signing on as front end users they can transform their gift certificates into creditpoints.

Screenshot



Developers

- Kasper Skårhøj: 1st developer 1.2.7 / TYPO3 3.8
- René Fritz: 1st developer 1.2.7 / TYPO3 3.8
- Franz Holzinger (trainer): price calculation, discount price, gift certificates, e-mail table for notification, error correction, billing, receipts, multi-column listings, product/articles tables, multiple languages, DAM list and category list
- Klaus Zierer (trainer of zk_products): more sizes and colors, entry lists, automatic registration of front-end users, category pages. In zk_products you can find a good example shop template.
- Els Verberne: Credit point system and voucher system, Order lists

Sponsors

- The development of gift coupons was carried out by Franz Holzinger. It was sponsored by BENDOO e-work solutions of the Netherlands (http://www.bendoo.nl). For more information contact BENDOO at hiddink@bendoo.com.
- The shopping of articles without attributes from a product, the category select box, the image naming using parts of table field entries and of the DAM table, and the shipping setup using the static_countries and a credit cards table was carried out by Franz Holzinger. It was sponsored by http://www.henrikjahn.de/ Germany. For more information contact Henrik Jahn at jahn@henrikjahn.de .
- pil.dk - Professionelle Internet Løsninger ApS is a main sponsor of the code for the Payment Library extension.
- lightimaging images database - superb images of nature, royalty-free
- Multi-categories per product with multilingual listings, import scripts for XML files from inventory control systems, custom shop tables, advanced search mask with producer select box, traffic light symbols to display the availability of articles. Sponsored by Geo.net IT GmbH.

Support

You can get support and installation service for this extension at http://jambage.com/index.php?id=162 .

You can download all needed extensions from the TYPO3 TER, http://typo3.org or from http://jambage.com/index.php?id=170 .

Users manual

Note:

The data path is renamed from 'pi' to 'pi1', thereby tt_products will now be put together like standard TYPO3 extensions. If paths have been employed from previous versions of tt_products then these must now be renamed in TypoScript setup. If you newly installl or update the extension, please observe the Depencies in TER (online repository of the extension manager).

You must fill in the product's 'In Stock (pcs)' field of your product properties, or no item will be shown in your frontend list if set to 0. This has been changed to the older version.

However, you should store the prices with included tax in the database; it makes it easier for you to reenter the price as well as for the customer to see the prices as you entered them. These prices should finally have the convention of pricing like 89.99 and not 89.72.

Before you start entering the prices of your products you have to decide whether you want to enter the prices with or without tax (see TAXincluded). All the calculations and configurations will use the prices as they have been stored into the price field of the tt_products table.
You should use a template with multiple columns (displayBasketColumns=1 or higher) even if you only have one column in the list table.

Use an example template from the directory tt_products/template, like the template, example_template_bill_de.tmpl, when you start your shop. You have to change the page id in the links (after the 'id=...') to the page id of the basket on your site. The marker ###DOMAIN### is the placeholder for your url. There are several placeholders for the different page ids of the shop. This is sensible because different hosting environments will be used. In the setup of the shop templates, the domain marker must become mydomain.com by using the following TypoScript : plugin.domain=mydomain.com).

tt_products is based on the Table Library (table) and FE/BE Library (fh_library). So you have to always update to the latest versions of these extensions when you update tt_products.

If you want to use the variant fields (color, quantity ...), you must also set the constants 'selectColors' and 'selectSize' at 0 or 1. Otherwise, the count in the product list of articles will be calculated incorrectly.

Many adaptations can be undertaken by way of the constant editor. For entering in the shop page, most should be directed in the template's setup field and constant editor.

Mini-Basket:

Beside the "normal" basket which shows a complete list of added products and some order options, there is also a "mini-basket" which only shows the number of products in the basket and the total price. This shows or indicates the actualization of the contents of the mini-basket displayed above the normal view (or as desired) while the process in shop steadily continues. The progress of accumulation that the mini-basket indicates can be controlled via typoscript. To use this mini-basket, insert a new content element "Plugin: Products" and in the display view insert 'Basket: overview' or if you use typo3script set the CODE-field to OVERVIEW.

Use the following markers in your tt_products template to enable and customize this mini-basket:

Code Listing:

```
<!-- ###BASKET_OVERVIEW_TEMPLATE### begin -->
  <div class="shop_minibasket">
     <div class="shop_minibasket_image"><!--###LINK_BASKET###-->###IMAGE_BASKET###<!--
###LINK_BASKET###--></div>
    <p>###NUMBER_GOODSTOTAL### Products (EUR ###PRICE_GOODSTOTAL_TAX###.-)</p>
<!-- ###BASKET_OVERVIEW_TEMPLATE### end -->



<!-- ###BASKET_OVERVIEW_EMPTY### begin -->
  <p>No products in basket.</p>
<!-- ###BASKET_OVERVIEW_EMPTY### end -->
   </div>
```

Calculation script:

If you write and use your own calculation scripts, then you should always use only the price variables. The priceTax and priceNoTax variables will be deleted in August 2008 because they can be calculated from the price variables and the TAXpercentage and TAXincluded constants.
You can use the global PHP variables $this->calculatedArray and $this->itemArray to make your own price calculations.

Discount:

There are several methods to get a discount.

1. Use the discount field for a frontend user and enter the percent of discount.
2. Use the price2 field for special feuser groups or other things. This setting is done in TypoScript via the IF-statement.
3. Use the discountprice calculation where the discount depends on the value of the total number of the products which have a special price. This will be available for the frontend user group 'discountGroupName'.

Search link for the products of the last X days

This is practical for “What is new?” pages. Create simply a link to the SEARCH page and add "newitemdays" as parameters.
http://www.my-typo3-shop.com/?id=8&newitemdays=10

The search result will contain all the products of the last 7 days.
You have to use the plugin code LISTNEWITEMS. You can also set the constant "newItemDays".

Offers and highlights

You have to use the plugin code LISTOFFERS for the items marked as 'offer' and LISTHIGHLIGHTS for the newest items.

AGB General Trading Conditions

You have to accept the general trading conditions before you can make an order and continue with the process. A page can be defined for this with "PIDagb". The AGBs have to be displayed there. You can set the target for a link by "AGBtarget".

A document (e.g. a PDF file) on the server containing the AGB can be set by "AGBattachment". This will be set to the customer as an email with an attachment to the order confirmation. The shop administration will not get this file. (he can get a CSV file of the order data instead, if wanted).

MEMO page

You can add items to a memo page when a user is logged in. This memo page will be saved in the fe_user table.

Simply create a new page with the Shop plugin as content and enter the code "MEMO". You must set the starting point there too, like in the SEARCH pages.

Voucher System

An interested customer is obtained through invitation by e-mail to visit a shop and purchase products. This e-mail is attached to a unique code which corresponds to a certain amount of credit points (voucher). When they make a purchase, they can redeem these credit points by inputing the unique code.

If a customer states in his registration that he has been recruited by another customer, then the referring customer gets credit points. The new customer gets a discount with his first order.

Somebody (lets say user 1) can enter a voucher code if he concludes an order. Then user 1 will obtain a discount. This credit entry code undergoes a change with each different user name (which is the e-mail address). Let's say user 2 gets 5 credit points altogether and applies his code to his purchase. Only one entry code out of a voucher system can be used by each user. In fe_users table there is a field given for this "tt_products_vouchercode" which contains the used credit entry code (e-mail of user 2).

Creditpoint System

If a customer purchases a certain number of items, then he gains credit points which are stored in his user's account. If he purchases more items at a later time, he can redeem credit points against the purchase price. The customer can gather credit points with each purchase.

```
creditpoints {
 pricefactor = 0.5
 10.type = price
 10.prod.1   = 0.02
 10.prod.101 = 0.04
 10.prod.501 = 0.06
}
```

Gift Certificates

A shop manager can sell, for example, electronic gift certificates for wine bottles. A client may order a certificate that corresponds to 50 credit points (=25 Euros) to send to someone as a gift. To redeem this gift certificate, then one must register as a shop customer and input the certificate code. Thereby, haveing 50 credit points credited to his account. This customer can purchase a particular product with these credit points.

Anyone can buy, for example:

- 3 certificates @ 25 credit points
- 5 certificates @ 50 credit points

- 8 certificates @ 75 credit points

...all within the same order.

Buy-a-Case

Wine will be sold in a shop, for instance. Normally, you sell a case of wine bottles (a case can contain 6 or 8 items). The additional service, “buy-a-case”, gives the shop customer the potential to put together his own case of 6 bottles: for example, 2 red, 2 white, 2 rose. The special handling of bottles and cases will be calculated under the buy-a-case concept. The handling per bottle will be calculated 1.50 Euro/bottle; likewise, the extra handling for 2 cases will amount to 2 x 6 x 1.5 = 18 Euros.

Using Product Articles (variants with new prices)

This example shows you how to have different sizes at different prices. The same procedure can be used with color and other variants. Make sure ‘Show secondary options (palettes)’ is tagged, otherwise some of the fields will not be available.

1. In the TSSetup for the template add

   ```
   plugin.tt_products.useArticles = 1
   plugin.tt_products.selectSize = 1
   ```

   If you are using columns to List your products like in the Bananaguard example, you also need

   ```
   plugin.tt_products.conf.tt_products.LIST.displayColumns = 1
   ```

   All this tells TYPO3 to read from the product articles and display a dropdown selection box if variants are used.

2. Add the parent product.
   This is the primary information that is displayed in the frontend List and Single views, ie description, image etc.
   Create a new entry in the Web > List mode. Select Product from the available options.
   You will need to fill out the following fields:

   - Title
   - InStock (must have at least 1 to display in the frontend)
   - Category (if applicable)
   - Price (this should be the cheapest price for the cheapest size - corresponding to the cheapest product article which we will set up next)
   - Size (same applies for Color/Description/Grading if applicable)
     This is crucial - in order for the frontend to be able to select a product variant you need to list all the size options here first, separated by a semi-colon eg S;M;X;XL - this will correspond to the product articles we'll create next.
   - Note
   - Image
   - Datasheet (if applicable).

3. Add the product variants.
   Here we create the different sizes for the parent product. The information is pretty basic, you use to clarify different prices, item numbers (for keeping track of stock), amount of stock etc. Each product variant must relate to one of the sizes (or color etc) we listed in the parent product.
   - Create a new entry in the Web > List mode.
   - Select Product Articles from the available options.
   - Fill out the Title (this will appear in the shopping basket) and
     - Instock
     - Price

       If this variant is the cheapest version this should be the same as the price listed in the Parent Product.
     - Product

       Very important, you need to select the parent product using the Element Browser, otherwise the price will not be changed in the shopping basket when a size is selected
     - Size (or Color, Grading is applicable)
       This must correspond to one of the sizes that was listed in the Product Parent eg XL, if the spelling is incorrect it will not work.

4. Changes to the html template.
   This is the last thing to do.
   - You'll need to add new fields to the ###ITEM_LIST_TEMPLATE### and ###ITEM_SINGLE_DISPLAY###.
   - The Code Listing below for the variant selection box needs to be added and must sit inside of the form

tags. Also note that any information between ###display_variant1### or ###display_variant2### tags will not be displayed if that variant is not used. This is a useful feature as we can use this to display or hide the word 'From' before the ###PRICE_TAX### value by putting the ###display_variant2### tags either side, giving us for example 'From $200' when $200 is the cheapest size (this is why the cheapest price is entered in the product parent - by default the parent product price is displayed in the List and Single view, the variant price is only displayed in the shopping basket once a size (or color etc) has been selected.) When no variant options are available the word From is hidden.

Code Listing:

```
<!-- ###display_variant1### -->
Color:><SELECT name="###FIELD_COLOR_NAME###" rows="1">###PRODUCT_COLOR###</SELECT>
<!-- ###display_variant1### --><!-- ###display_variant2### -->
Size:><SELECT name="###FIELD_SIZE_NAME###" rows="1">###PRODUCT_SIZE###</SELECT>
<!-- ###display_variant2### -->
```


Administration

In the normal case the categories from the category table and its enhancements are used. You have to set

$TYPO3_CONF_VARS['EXTCONF']['tt_products']['pageAsCategory'] = 0

to be able to use them. Otherwise the pages will form the category and the category will be used as subcategory (=1) or not used at all but replaced by the page (=2).

Activate now the flexforms in the backup. The code field will be replaced by a graphical user interface. However you have to reenter all your code fields.

$TYPO3_CONF_VARS['EXTCONF']['tt_products']['useFlexforms'] = 1

Installation

Install the extension with the extension manager. If you already use an older version of tt_products that's installed in the “global” location (typo3/ext/), it's recommended to install the new extension into the “local” folder (typo3conf/ext/) without overwriting the old one. By doing this you can easily switch back to the former version.

Deinstall the extension from the Extension Manager. Then download the version of your choice from the Online Repository with the Extension Manager.
Then use the Extension Manager's "Available extensions to install" and add the Shop system again.

This extension works best in union with static_info_tables, sr_feuser_register, rtehtmlarea and conf_userts.

Handling of Categories

There are multiple ways to create a shop. The usual way is to create sysfolders for the products and lists and single sites where the products are displayed. If you want to do a bigger shop with TYPO3, this will become a little bit unhandy.

If you want to make bigger shops with hierarchical categories you shall install the mbi_products_categories and maybe also the nsb_cat2menu extension. Sponsors will get an enhanced version of mbi_products_categories with which you can assign many categories to one products via a mm-table. Only with this it will be possible to list DAM images.

Example of tt_products/ext_localconf.php:

```
$TYPO3_CONF_VARS['EXTCONF']['tt_products']['pageAsCategory'] = 0;
```

Handling of Images

There are ways to use and display the infos about images of the DAM extension.

Important

This should be fully downwards compatible to the former tt_products 1.2.7. But it needs some adaption to the template file. In ###BASKET_TEMPLATE### you must have the lines

```
<input type="hidden" name="mode_update" value="1">
<input type="Submit" name="products_update" value="update basket">
```

You have to rename the marker ###FIELD_NAME### to ###FIELD_NAME_BASKET### in the BASKET_TEMPLATE.

This is compatible with zk_products 1.3.2.

Some of the TypoScript settings do not function via the constants field and must be entered in the setup field. Only those settings which can be found using the Constants Editor or in the file static/old_style/constants.txt are valid constants. You must include the 'Shop System Old Style' into your 'static template records' of your TYPO3 template.

A negative value of a product at 'in stock' does not have any more a special function. This has been replaced by a checkbox 'always on stock'.

Change the former input fields for a search into

<INPUT size="30" maxlength="100" type="text" name="sword" value="###SWORD###">

The marker ###SWORDS### must be changed into ###SWORD###, and the name into 'sword'.

Template files

In your template files for the shop you need to make the following settings so it will work:

<input type="hidden" name="mode_update" value="1">

<input type="Submit" name="products_update" value="update basket">

Put this into your FORM-attributes of the BASKET_TEMPLATE.

Template markers

The following template markers for the tmpl-file are used. You have to put ### before and after them. There are more

markers, so look into the example template files.

area markers:

Marker:	Description:	Area:
BASKET_TEMPLATE		top
BASKET_INFO_TEMPLATE	This is normally used to let people enter address information separately from the real basket. Exact same features as BASKET_TEMPLATE.	top
BASKET_ORDERCONFIRMATION_TEMPLATE	the final page after the order has been processed. It will not be used for the display, if PIDthanks is set. But this will always be used for the text in HTML emails. see: BASKET_ORDERTHANKS_TEMPLATE	
BASKET_ORDERCONFIRMATION_NOSAVE_TEMPLATE	This HTML data will be added to the display after the order has been stored to the database.	
BASKET_ORDERTHANKS_TEMPLATE	Used for displaying a thanks page, when PIDthanks is set. This will not be used in the HTML emails.b see BASKET_ORDERCONFIRMATION_TEMPLATE	
BASKET_OVERVIEW_EMPTY	message 'your basket is empty' for OVERVIEW	
BASKET_OVERVIEW_TEMPLATE		
BASKET_PAYMENT_TEMPLATE		
BASKET_REQUIRED_INFO_MISSING		
BASKET_TEMPLATE_EMPTY	message 'your basket is empty'	
BASKET_TEMPLATE_NOT_LOGGED_IN	error message that the user has not logged in	
BASKET_TEMPLATE_INVALID_GIFT_UNIQUE_ID	error message that a wrong unique id for a gift product has been entered	
BILL_TEMPLATE	how your bill file will look like	
DELIVERY_TEMPLATE	for the file of your delivery sheet	
EMAIL_PLAINTEXT_TEMPLATE	the email notification text The first line is the subject.	
EMAIL_NEWUSER_TEMPLATE	email after creation of new frontend use	
ITEM_LIST_TEMPLATE	the listing of products on the starting LIST page	
ITEM_LIST_GIFTS_TEMPLATE	listing of the gift products	
ITEM_SEARCH		
ITEM_SEARCH_EMPTY		
ITEM_SINGLE_DISPLAY		
ITEM_SINGLE_DISPLAY_GIFT	single display used when this is a gift product	
ITEM_SINGLE_DISPLAY_NOT_IN_STOCK	single display when item is not in stock	
ITEM_SINGLE_DISPLAY_RECORDINSERT	see displayCurrentRecord: render the $cObj->data	
MEMO_TEMPLATE		
MEMO_NOT_LOGGED_IN	error message that use has not been logged in for MEMO	
ORDERS_LIST_TEMPLATE	display of the order list	
TRACKING_EMAIL_GIFTNOTIFY_TEMPLATE	notification to the gift recipient in the order tracking	
TRACKING_ENTER_NUMBER		

Marker:	Description:	Area:
TRACKING_WRONG_NUMBER		

single markers

Example for a wrap:

```
<!--###LINK_DATASHEET###> datasheet for the product <!--###LINK_DATASHEET###>
```

Marker:	Type:	Description:	Area:
BROWSE_LINKS		for browsing the display list over several pages	
DELIVERYCOSTS	value	sum of delivery costs and payment costs	
DELIVERY_... NOTE NOTE_DISPLAY DESIRED_DATE	value	see also PERSON_... input field note for order output field note with instead of linebreaks desired delivery date of the order	BASKET_INFO_TEMPLATE
EXTERNAL_COBJECT	value	extra preprocessing Cobject	
FIELD_NAME	value in form	the name of a field in a form	
FIELD_NAME_BASKET	value in form	the basket data in encrypted format	
GC1, GC2, GC3	value	global colors	all
GW1B, GW2B, GW1E, GW2E	wrap	global wraps	all
PRICE_TAX PRICE_NO_TAX PRICE_ONLY_TAX PRICE2_TAX PRICE2_NO_TAX PRICE2_ONLY_TAX	value	price/price2 of the item with, without and only VAT	ITEM_SINGLE
PRICE_TOTAL_TAX PRICE_TOTAL_NO_TAX PRICE_TOTAL_ONLY_TAX	value	total sum of the items with shipping and payment costs	all
PRICE_GOODSTOTAL_TAX PRICE_GOODSTOTAL_NO_TAX PRICE2_GOODSTOTAL_TAX PRICE2_GOODSTOTAL_NO_TAX	value	total sum ot the items for price or price2	all
LINK_BASKET	wrap	link to the basket page	basket
LINK_DATASHEET	wrap	link to the datasheet file in uploads/tx_ttproducts/datasheet	
ORDER_STATUS_TIME, ORDER_STATUS, ORDER_STATUS_INFO, ORDER_STATUS_COMMENT	value	order values	TRACKING_DISPLAY_INFO
PERSON_... NAME, ADDRESS, TELEPHONE, FAX, EMAIL, COMPANY, CITY, ZIP, STATE, COUNTRY	value	address information fields have to be connected with a prefix PERSON_... the customer of the order DELIVERY_... the recipient of the order	
PID_TRACKING	value	the tracking pid	
STATUS_CODE_60	wrap	used to allow the disappearance of the text with status code by the shop	TRACKING_DISPLAY_INFO
STATUS_OPTIONS	value	Select menu of state options	ADMIN_CONTROL inside TRACKING_DISPLAY_INFO
SHOPADMIN_EMAIL	value	Email-Address of the shop admin	all

Automatic creation of frontend users and address fields

It is possible to create frontend users automatically after each order. You have to set "createUsers" to "1", enter the PID of the sysfolder as PIDuserFolder in the Setup field. Then you have to set memberOfGroup to the ID of your frontend user group.
The customer will get an email with his account data after his first order. This email will contain his account name which is his email address, and his automatically created password.

If you do not want to use a single address field, but the address field for the name of the street and the housenumber and additional fields for the ZIP, city and country, then you have to set these in your template file. If you install static_info_tables and set “useStaticInfoCountry=1” then the small field static_info_country of the fe_users will be used instead of country. This is useful to make a select box for the country.

Product properties:

Color, Size, Additional and Gradings

To enter variants of products you have to separate the values by a semicolon. So for t-shirts with different color and size enter "red;green;blue" in variant1 and "S;X;XL;XXL" in variant2.
Color (Variant 1) and Size (Variant 2): enter here values with title and values separated by semicolon ';' like

```
M;L;XL;XXL
```

Change the template to support this:

```
<!-- ###display_variant1### -->
###GW2B### Color: ###GW2E### <SELECT style="font-size: 10px"
name="###FIELD_COLOR_NAME###" rows="1">###PRODUCT_COLOR###</SELECT><br>
<!-- ###display_variant1### -->
<!-- ###display_variant2### -->
###GW2B### Size: ###GW2E### <SELECT style="font-size: 10px"
name="###FIELD_SIZE_NAME###" rows="1">###PRODUCT_SIZE###</SELECT><br>
<!-- ###display_variant2### -->
```

If you do not make colors or size selectable, you have to set selectColor or selectSize to 0 and only use the ###PRODUCT_COLOR### and ###PRODUCT_SIZE### markers and within the template itself delete the corresponding ###display_variant### marker. You can, however, use markers as representations of the colors, also if no different colors can be chosen.

The field name of all the fields in the list view and the basket view will be addressed with the marker ###FIELD_NAME###. The markers will be internally replaced in shop through a field name with which the field can be chosen and attributed correctly.

When you have products with different mixes of colors, sizes, and gradations, then you must replace the notation ###FIELD_NAME#### in the field BASKET_TEMPLATE with the notation ###FIELD_NAME_BASKET###.

In the list view of the web module, you can now complete the readily available items with the variant, for example color. For that purpose, you select the table product article.

You must set useArticles=1 in the template setup in order to apply the article attributes/properties.

When you have prepared products, having arranged product qualities of different color, you use only one article number and one price, etc. for this product. However, when you need, for example, different colors, different article numbers, and perhaps prices, etc then this arrangement occurs in the article property (web module, list, product article table). The advantage is that you do not need to create a completely "new" product. The default quality will be transmitted. You only give the color a new article number and a new price. All other product statements will be accepted. Leave a field empty so the default product quality field content will be accepted.

It makes a differene, if an article with different colors or sizes in the product list or shopping cart can be selected. A color or size normally will be chosen by way of a selection box. The quantity for the first color/size will be indicated in the product list. The products in all variables will be, however, individually listed. The lists of products in the basket and in the payment page are, therefore, nearly the same, but will be handled otherwise/differently. You must deactivate selectSize and selectColor whenever you do not want to have selection boxes in the product list.

Weight, bulk and special preparation

Each product can have weight (kg) and bulkily (Yes/No). The total weight is calculated and can be used in the price calculation for the shipping costs.

If an item has been marked as 'bulkily' then a warning message will be displays ('bulkilyWarning' in setup). By setting 'bulkilyAddition' in setup you can add an additional shipping price for this bulkily item.

Special preparation does generally not have a function. Only a marker (like with BulkilyWarning) will be written. You can set here a link to the mail form page.
Example:


```
specialPreparation = special preparation is possible!
<A HREF="?id=999&tt_products=###PRODUCT_ID###">Order here</A>.
```

Basic stock management

The field inStock can be used for a stock administration. If "in stock" is at "0", the item will not be visible for the customers any more.
After each order the number of the ordered items will be erased from "in stock".

If a new item gets created, its number will be set to '1' by default. If you have set 'alwaysInStock=1' then this item will always be available and visible. Otherwise the number of products will be reducted with each order.

You can fill in the checkbox for each product to have it always in stock.

You can define the unit on the store by "inStockPieces" like "pieces".

Several tax rates

Each item can have its own tax. But you have to enter this separately for each item.

Configuration

FAQ

– If you do not use the Constant Editor to configure the extension, please note the form of the constants assignments in the constants section of your TS template:

```
plugin.tt_products {
    property = value
}
```

Files

File:	Description:
class.tx_ttproducts.php	Main class used to display the product list or the shopping basket. Call it from a USER cObject with 'userFunc = user_products->main_products'
products_comp_calcScript.inc	Example 'calculationScript'
products_template.tmpl products_template_htmlmail.tmpl	Example templates in English. '..._htmlmail.tmpl' is a HTML-wrap for the HTML-emails sent.
products_template_dk.tmpl	Example template in Danish
example_template_bill_de.tmpl	bananaguard.de template with examples for bill and delivery sheet in German
products_template_fi.tmpl	Example template in Finnish
products_template_fr.tmpl	Example template in French
products_template_se.tmpl	Example template in Swedish
product_detail.tmpl product_proefpakketten.tmpl shop-a-box.tmpl producten.tmpl	Example templates for gift certificates, creditpoints and voucher system in Dutch
products_css_en.html	CSS styled template. Use this if you need a barrier-free shop.
'handleScripts' for interfacing with external payment gateways	
payment_DIBS.php	Script for interfacing with DIBS (Danish Internet Payment System) in Denmark. You can reach them at http://www.architrade.com/uk/.
payment_DIBS_template.tmpl	Template file for DIBS payment.

[tsref:(cObject).TEST]

Description

Built-in shopping basket and products display within TYPO3. Has a clearing interface which lets you write your own implementation with existing payment-gateways.

Currently there's an implementation with DIBS in Denmark, found at www.architrade.dk .

Reference

Property:	Data type:	Description:	Default:

templateFile
Constants:
file.templateFile
	resource	The template-file.
		See example in 'tt_products/template/products_template.tmpl'
		You can also specify a CODE.
		(siehe display mode)
		Example:
		`plugin.tt_products.templateFile = EXT:tt_products/template/example_template_bill_de.tmpl`
		`plugin.tt_products.templateFile.LIST = EXT:tt_products/template/products_template_dk.tmpl`	

pid_list	string /stdWrap	The pids from where to fetch categories, products and so on. Default is the current page. Accepts multiple pid's separated by comma.	
defaultCode	string	The default code (see below) if the value is empty. By default it's not set and a help screen will appear. You should not set anything here. **Example:** `plugin.tt_products.defaultCode = HELP`	
code	string /stdWrap	see chapter 'display mode'	HELP
defaultArticleID	int+	The default article uid number for the single display is used when the link to the script did not contain a 'tx_ttproducts_pi1[article]' parameter.	

Property:	Data type:	Description:	Default:
defaultProductID	int+	The default product uid number for the single display is used when the link to the scnotript did not contain a 'tx_ttproducts_pi1[product]' parameter. Set this default value when you get an error message like: "GET/POST var 'tx_ttproducts_pi1[product]' was missing."	
defaultCategoryID	int+	The default category uid number for the list display is used when the link to the script did not contain a 'tx_ttproducts_pi1[cat]' parameter. Use this if you want only products of this category displayed in the list view as a default.	
defaultDAMCategoryID	int+	See defaultCategoryID, but for DAM categories and the 'tx_ttproducts_pi1[damcat]' parameter.	
rootCategoryID	int+	The upper most category ID from where you want to start to list categories.	
rootDAMCategoryID	int+	The upper most DAM category ID from where you want to start to list DAM categories.	
rootPageID	int+	The upper most page ID from where you want to start to list them as categories.	
recursive	int+	Number of recursive sublevels of pids to select tt_products from in lists.	99
domain	string	The url of the shop. If not set, it will be detected automatically. Will replace ###DOMAIN### markers.	

altMainMarkers	*(array of strings)*	Lets you specify alternative subpart markers for the various main template designs in the shopping basket system.
		This is the list of main subparts you can override:
		Properties:
		`TRACKING_WRONG_NUMBER`
		`TRACKING_ENTER_NUMBER`
		`BASKET_REQUIRED_INFO_MISSING`
		`BASKET_TEMP`
		`ITEM_SINGLE_DISPLAY_RECORDINSERT`
		`ITEM_SINGLE_DISPLAY`
		`ITEM_SEARCH`
		`ITEM_LIST_TEMPLATE`
		`ITEM_LIST_GIFTS_TEMPLATE`
		`ITEM_SEARCH_EMPTY`
		`BASKET_TEMPLATE`
		`BASKET_INFO_TEMPLATE`
		`BASKET_PAYMENT_TEMPLATE`
		`BASKET_ORDERCONFIRMATION_TEMPLATE`
		`EMAIL_PLAINTEXT_TEMPLATE`
		`BILL_TEMPLATE`
		`DELIVERY_TEMPLATE`
		
		`/+ stdWrap`
		
		Example:
		`altMainMarkers.BASKET_TEMPLATE = BASKET_DESIGN2`
		`altMainMarkers.BASKET_TEMPLATE.wrap = ### | ###`
		
		This example changes the main subpart marker for the regular basket display from the default ###BASKET_TEMPLATE### to the custom supplied design ###BASKET_DESIGN2### (found in the same template HTML-file)	

stdSearchFieldExt	*list of fields*	Search fields Default internal list is title,subtitle,note. You can specify your default fields here.	
limit	int+	Max items displayed. The maximum number of items displayed on one page.	50
limitImage	int+	Max image items displayed. The maximum number of images for one item displayed on the list view.	1
limitImageSingle	int+	The maximum number of images for one item displayed on the single view.	1

~~usePageContentImage~~	~~boolean~~	**Deprecated.** See Article/Product configuration
		Use this instead:
		`plugin.tt_products.conf.tt_products.ALL.fetchImage {`
		`  type = foreigntable`
		`  table = tt_content`
		`}`	

separateImage	boolean	Normally all images are displayed together. With separateImage=on you can use a ###PRODUCT_IMAGE3### for each image number (starting with 1) separatly.	
image	IMAGE cObject	The image configuration in single display	

Property:	Data type:	Description:	Default:

listImage	IMAGE cObject	The image configuration in list display
		
		`listImage >`
		`listImage {`
		`  altImgResource.import = uploads/media/`
		`  altImgResource.import.field = media`
		`  altImgResource.import.listNum = 0`
		`  altText.data = field:title`
		`}`
		
		That way, attached images are not copied to and displayed from */typo3temp/* (which gives trouble with transparant backgrounds) but directly linked from /uploads/pics/.
		The line with altText leads to the drawing of an alternative text.	

listImageHasChilds	IMAGE cObject	DAM only:
		The image configuration in list display if there is a filter for a category on the page and this category has childs.
		
		`listImage >`
		`listImage {`
		`  altImgResource.import = uploads/media/`
		`  altImgResource.import.field = media`
		`  altImgResource.import.listNum = 0`
		`  altText.data = field:title`
		`}`
		
		see listImage	

basketImage	IMAGE cObject	The image configuration in basket display	
datasheetIcon	IMAGE cObject	The image icon for the datasheet. Replaces ###ICON_DATASHEET###	
basketPic	string	URL link to the basket image	
clickIntoBasket	boolean	If set you will be directed into the basket page after putting a product into the basket. This only works if PIDbasket has been set.	
clickIntoSubmenu	boolean	If set, the submenues in the LISTCAT category list will only be listed for the current category.	
basketMaxQuantity	int+ / string	The maximum integer value for the quantity of an item in the basket. 'inStock': Only the number of items which are currently in stock can be put into the basket.	100000
quantityIsFloat	boolean	If set the basket count can be a float value.	
noImageAvailable	resource	The image file displayed if no image was attached to a product. This image is processed by the IMAGE cObject which is active in the actual display of that image. That is, one of the above IMAGE cObjects.	
displayListCatHeader	boolean	Display Category Header in list If this option is set, the category headers (page titles) will automatically be displayed in the product lists. This is not always convenient because you might have chosen a header-title for the "list" content element.	1
displayBasketCatHeader	boolean	Display Category Header in basket. If this option is set, the category headers (page titles) will automatically be displayed in the basket lists.	0
displayCatListType	string	Define the HTML main tag for the display of the categories in the category list view. possible values: • ul • null	ul

Property:	Data type:	Description:	Default:
~~displayBasketColumns~~	int+	**Deprecated.** see table configuration: displayColumns Number of columns for the LIST, SEARCH listing of items in a table. You have to adapt your template using special template markers. The ITEM_SINGLE_PRE_HTML and ITEM_SINGLE_POST_HTML must surround your <TD>-</TD> tags, so the table will be created correctly. **Example:** `<!-- ###ITEM_SINGLE### begin-->` `###ITEM_SINGLE_PRE_HTML###` `<TD height="150" bgcolor="###GC1###" valign="bottom" align="center">` `<!--###LINK_ITEM###-->` `###GW1B###<b>###PRODUCT_TITLE###</b>###GW1E###` `<!--###LINK_ITEM###-->` ` ` `###PRODUCT_IMAGE###  ` `###GW1B### item count: ###GW1E### <INPUT size="3" maxlength="4" type="text" name="###FIELD_NAME###" value="###FIELD_QTY###">` `<p> </TD>` `###ITEM_SINGLE_POST_HTML###` `<!-- ###ITEM_SINGLE### end -->`	
CSS	*see below*	Cascading Stylesheets settings	
conf	*see below*	configurations of the tables	
NoSingleViewOnList	boolean	Usually you get the link to the single item view on the display page of the LIST code. If you however want to create your own pages for single view with SINGLE code you must set this to 1.	

itemMarkerArrayFunc	function-name	Every time a product is displayed be it in the basket, list or single view, the method getItemMarkerArray() in tx_ttproducts_marker is called. This function fills in and returns an array, so called markerArray(), with key/values for template substitution.
		If you enter a valid function name here (see datatype 'function-name' for details!) that array will be passed to that function as the second parameter. The first parameter will be the TypoScript properties to itemMarkerArrayFunc.
		
		Parent PHP-Object reference:
		.parentObj property is hardcoded to be a reference to the calling user_products object (PHP).
		
		Example:
		(provided that a function or class is included!)
		
		`itemMarkerArrayFunc = user_addFieldsMarkerArr`
		`itemMarkerArrayFunc.simpleOption = 1`	

PIDitemDisplay	int+/*Array of integers*	PID for single item display.
		If you want a certain page to be used for display of item details, please enter the PID (page-uid) here. If you set the type to sql, you can use conditions using several lines. The where string can contain the AND condition. The pid for the first fulfilled condition will be returned.
		
		`PIDitemDisplay {`
		`10.type = sql`
		`10.where = color=red`
		`10.pid = 142`
		`20.type = sql`
		`20.where = color=white`
		`20.pid = 143`
		`}`
		
		If you set the type to pid then the pid of the record will be used.
		`PIDitemDisplay {`
		`10.type = pid`
		`}`	

PIDlistDisplay	int+/*Array of integers*	PID for the item list display Similar to PIDitemDisplay, however the category table is used here.	
PIDsearch	int+	PID for search page. If you want all product searches to go to a specific page, enter the PID it here! NOTE: If you set this PID, all searchqueries will (must) be handled with a list content element with the display mode "Products: search" on that page.	
PIDbasket	int+	PID for the basket page. If you want ever change the number of items anywhere to go to a specific page (eg. the shopping basket page), enter the PID here.	

Property:	Data type:	Description:	Default:
PIDstoreRoot	int+	PID for store root. This is the PID of the rootPage of the store. If not set the store will operate over all pages from the root of the site. But if the site has many pages, performance may improve. You should better set pid_list instead of it.	
PID_sys_products_orders	int+	PID for the sys_products_orders records. By default they will get the pid of the payment (finalize) page.	
PIDGiftsTable	int+	PID for the tt_products_gifts table. The gift orders are stored here.	
PIDinfo	int+	PID for the info page where name and address is entered.	
PIDfinalize	int+	PID for the finalization page afther the user has confirmed the order data. The order will get stored here.	
PIDthanks	int+	PID for the thanks page. BASKET_ORDERTHANKS_TEMPLATE will be used. You must not set PIDfinalize if you use this.	
PIDtracking	int+	PID for the order tracking	
PIDbilling	int+	PID for the generation of the bill	
PIDdelivery	int+	PID for the generation of the delivery sheet	
PIDmemo	int+	The ID of the memo page	
PIDagb	int+	The PID of a page with the general trading conditions (“AGB” in germany) Only if this page id is set the AGB check will be active.	0
PIDuserFolder	int+	The sysfolder, where the new users should be stored	116
pidsRelatedProducts	int+	Allowed pages for related products.	
paymentActivity	string	When the payment with a payment script shall be executed. Possible values: payment, finalize	finalize

advanceOrderNumberWithInteger	int+	If this value is set, then each time a new order is created the order-number counter will be incremented with a random number between [first-integer] and [second integer] to cheat a little.
		
		Example:
		`1,10 (This will increment the counter randomly between 1 and 10)`
		`5,5 (This will increment the counter with 5 each time)`	

alwaysAdvanceOrderNumber	boolean	If set then the order number will always get increased and the empty order numbers are not reused. You have to set this if you use a payment script to pay via a payment system which does not accept duplicate order numbers.	
alwaysUpdateOrderAmount	boolean	If set then the entered order amount will always be updated and not increased by the entered number.	1

parseFunc	->parseFunc
	
	not used with 'CSS styled content'
		If the extension 'CSS styled content' has been installed, you have to make your settings in lib.parseFunc_RTE and not here.
		The product details are parsed by these properties. So if e.g. you want ot allow HTML-tags to create a table in the Note field you have to set the parseFunc.allowTags or use the denyTags.
		To make RTE working with HTML you have to set the following into the root page template.
		
		Example:
		`keepNonMatchedTags = 1`
		`RTE.default.proc.preserveTables = 1`
		
		Example:
		`parseFunc.allowTags = table,tr,td,b,i,u,a,img,br,div,center,pre,font,hr,sub,sup,p,strong,em,li,ul,ol,blockquote,strike,span,h1,h2,h3,h4,h5,h6`
		`parseFunc.denyTags = *`
			styles.content.parseFunc

categoryHeader	cObject	Generates the category header.
		
		Example:
		`categoryHeader = TEXT`
		`categoryHeader.current = 1`	

TAXpercentage	double	Sales TAX/VAT percentage. Double value (!) (means, "use . as decimal point")	

Property:	Data type:	Description:	Default:
TAXincluded	boolean	Set this, if TAX *is* included in the database prices! (... and of course: Clear this, if TAX *is not* included in the database prices and should be added in the display of items) All processing will take this flag into account and calculate prices accordingly.	1
TAXmode	int+	tax mode 1: The net sums are added first. The tax is added on the final total net sum. 2: The gross price is calculated for every product. The total sum is calculated on the single gross prices.	1
priceDec	int+	Price decimals	
priceDecPoint	string	Price decimal point	
priceThousandPoint	string	Price Thousand point Enter the thousand separator, if any.	

priceNoReseller	int+	Price number for reseller, which can only be 2 at the moment. The price2 will however only be taken when its value is greater than 0. Here is the way to get the price2 for a special user group:
		
		Example:
		`[usergroup = 1]`
		`priceNoReseller = 2`
		`[global]`
		`...`	

currencySymbol	string	Currency symbol. Used in shop administration.
		
		Example:
		`EUR`
		`DKR`
		`USD`
		`$`	

lockLoginUserInfo	boolean	If set and a user is logged in, the address info of that fe_user gets filled in as billing address of the user. It is not possible to change this data.	
editLockedLoginInfo	boolean	If set and lockLoginUserInfo is set, then the filled in data is still editable for the order. You have to set the input HTML tags for this.	
loginUserInfoAddress	boolean	If lockLoginUserInfo is set, this switch makes that the address field is filled in from address, country, zip and city of the fe_user	
requiredInfoFields	*list of string*	List of the fields which are required in the address information This example gives you all possibilities. **Example:** `requiredInfoFields = name,address,telephone,fax,email,company,city,zip,state,country`	
~~orderBy~~	~~string~~	**Deprecated.** See Article/Product configuration Use this instead: `plugin.tt_products.conf.tt_products.ALL.orderBy = title`	
~~orderByCategoryTitle~~	~~boolean~~	**Deprecated.** Use this instead: `plugin.tt_products.conf.tt_products_cat.ALL.orderBy = title`	
orderByItemNumberSg	boolean	If the single item display should be sorted by ItemNumber instead of uid; used for ###LINK_NEXT_SINGLE### and ###LINK_PREV_SINGLE###	
orderNumberPrefix	string	Prefix to the order numbers. Max 10 chars. If this string starts with '%' then the rest will be interpreted as a PHP date format.	
orderEmail_from	string	From email address for the confirmation email to customer	
orderEmail_fromName	string	From name for the confirmation email to customer.	
orderEmail_to	*list of email-addresses*	Comma separated list of recipients of the order email. Shop and administrator/supervisor email addresses go here!	
orderEmail_toDelivery	boolean	If set, the email notification will be sent to the delivery email address and not to the billing email address.	
orderEmail_subject	string	Contents of the subject line if the first line in ###EMAIL_PLAINTEXT_TEMPLATE### is empty.	
orderEmail_htmlmail	boolean / string	If set, the order confirmation email is sent as HTML If orderEmail_htmlmail.removeImagesWithPrefix is set, then the images and their HTML tags will not be sent in an email.	
email_notify_default	boolean	If email-notification to the delivery email address of the customer is enabled by default for tracking (he can change it himself in the tracking module later)	

Property:	Data type:	Description:	Default:

statusCodes	*Array of integers*	Status codes used in the tracking module.
		Numbers above 100 removes the order from the tracklist.
		Number zero is the status of a non-finalized order (and non-finalized orders in the database may by time be regarded as garbage...)
		Numbers of 50-59 is available for the customer to choose from.
		Number 1 is reserved to be selected when an order goes from zero to 1 because it's confirmed. Number 1 cannot be selected by shop admin.
		These will be written into ###STATUS_OPTIONS### markers.
		
		Example:
		`statusCodes {`
		`1 = Order submitted by user`
		`2 = Order is received and accepted by store`
		`10 = Shop is awaiting goods from third-party`
		`11 = Shop is awaiting customer payment`
		`12 = Shop is awaiting material from customer`
		`13 = Order has been payed`
		`20 = Goods shipped to customer`
		`21 = Gift certificates shipped to customer`
		`30 = Other message from store`
		`50 = Customer request for cancelling`
		`51 = Message from customer to shop`
		`60 = Send gift certificate message to receiver`
		`100 = Order shipped and closed`
		`101 = Order closed`
		`200 = Order cancelled`
		`}`	

update_code	string	The 'password' used by the administrator of the shop to go into the tracking system in the front end. The password form field will appear if a BE_USER is logged in, but this password is still needed.	password
statusDate_stdWrap	->stdWrap	stdWrap for status date **Example:** `statusDate_stdWrap.strftime = %d-%m-%Y %H:%M`	
orderDate_stdWrap	->stdWrap	stdWrap for the order date **Example:** `orderDate_stdWrap.strftime = %d-%m-%Y`	
displayCurrentRecord	boolean	If set, certain settings are manipulated in order to let the script render a single item - the $cObj->data. If this setting is set, the subpart marked ###ITEM_SINGLE_DISPLAY_RECORDINSERT### will be used instead of the regular subpart ###ITEM_SINGLE_DISPLAY### if it is found.	
externalProcessing	cObject	This cObject may be used to call a function which manipulates the shopping basket. This manipulation could be based on settings in an external order system. The output is included in the top of the order (HTML) on the basket-page. This cObject is executed each time the main_products method of the user_products class in productsLib is called and it's executed before any of the main processing. See the class for details.	
externalProcessing_final	cObject	cObject for the final order confirmation template	
externalFinalizing	cObject	This cObject may be used to call a function which clears settings in an external order system. This is a sister to the above function and they should probably be used in conjunction somehow. This function is called immediately after the finalize-function has been called. For instance this function would be suitable for clearing any external basket facilitated by the .externalProcessing cObject **Note:** The output is NOT included anywhere.	
wrap1	-> stdWrap	Global Wrap 1. This will be splitted into the markers ###GW1B### and ###GW1E###. Don't change the input value by the settings, only wrap it in something. **Example:** `wrap1.wrap = <B> |</B>`	
wrap2	-> stdWrap	Global Wrap 2 (see above) markers ###GW2B### and ###GW2E###	
selectColor	boolean	If true the color of a product is selectable in a select box.	1
selectColor2	boolean	If true the 2nd color of a product is selectable in a select box.	1
selectSize	boolean	If true the size of a product is selectable in a select box.	1
selectSize2	boolean	If true the 2nd size of a product is selectable in a select box.	1
selectDescription	boolean	If true the decription of a product is selectable in a select box.	1

Property:	Data type:	Description:	Default:
selectGradings	boolean	If true the gradings of a product are selectable in a select box.	1
selectMaterial	boolean	If true the material of a product are selectable in a select box.	1
selectQuality	boolean	If true the quality of a product are selectable in a select box.	1
color1	string /stdWrap	Value for ###GC1### marker (Global color 1)	
color2	string /stdWrap	Value for ###GC2### marker (Global color 2)	
color3	string /stdWrap	Value for ###GC3### marker (Global color 3)	
nl2brNote	boolean	If true the linefeeds in the note field are replaced by .	1
useArticles	integer	• 0: If you do not want to use the articles table then this must remain 0. • 1: If you want to use products in different variants and have special prices for them in the articles table, then you have to set this to 1. The instock of the articles table will be used instead of the products table. • 2: Like 1 but you do not need to have corresponding fields for the variants between a product and its articles.	
priceTagObj	IMAGE cObject	Image for the display of the price	
usePriceTag	boolean	If true, the priceTagObj will be used. If false, the text will be displayed.	0
useStaticInfoCountry	boolean	If true the short fe_user static_info_country field will be used for country instead of the country field.	0
whereGift	string	This SQL where clause defines which products will be treated as gifts.	
max_note_length	int+	In the list view: The item note will be cut after max_note_length characters, and three dots are added... This can be dangerous if you use html in your product notes, because a tag could stay open and the following items look ugly	100
specialPreparation	string	This text will substitute your ###PRODUCT_SPECIAL_PREP### markers.	
createUsers	Boolean	If you set this and PIDuserFolder, for each customer that is not logged in and his email address is not listed as user already, a new frontend user will be created. An email with the username (his e-mail address) and the password will be sent to him (see ###EMAIL_NEWUSER_TEMPLATE###)	0
outputFolder	string	Folder where the bill and delivery sheets are created.	fileadmin/data
memberOfGroup	int+	The ID of the frontend user group, the new users should be member of	1
discountGroupName	string	Name of the group for discount pricecalculation	
getDiscountPrice	boolean	If set the discount price will be used even for normal customers without having a frontend user in the discount group.	
AGBtarget	Small	The target for the PIDagb link	_blank
AGBattachment	Resource	If this file exists, it will be attached to the order confirmation email to the customer. The shop administrator won't receive the agb attachment, but the CSV if you enable it	fileadmin/agb.pdf
GiftAttachment	Resource	If this file exists, it will be attached to the order gift email to the customer.	
generateCSV	boolean	Generates a CSV from each order. This file will be saved on the server and sent to the shop administrator per email (with the order confirmation)	0
CSVdestination	../csvorders/	The directory on the server, where the CSV files should be saved. This directory should not be accessible over http, because everyone can read your customers and their orders. So don't put this directory unter fileadmin, or rename it to something cryptical at least.	../csvorders/
CSVfields	string	Select which product data should be added to the CSV. By default these are all important fields except the note	(Long, long list)
CSVinOneLine	boolean	If you set this to 1, all additional information (shipping, payment, address data, delivecatry note) will be appended to the first ordered item. This is good if you want to import this files somewhere. If you leave this at 0, the file will be formated for best view in programs like Excel.	0
CSVnotInEmail	boolean	If set no csv file will be appended to the email.	
alwaysInStock	boolean	If set the stock will not get reduced if something has been bought.	1
showNotinStock	boolean	If set the items will be shown even if no product is in the stock.	
notInStockMessage	string	See above. This will be shown if no items are in stock and showNotinStock is set.	Not in stock

Property:	Data type:	Description:	Default:
warningInStockLimit	int+	Amount of items in stock at which when reached a warning message is sent.	
inStockPieces	string	This is the unit for items inStock.	pieces
newItemDays	int+	In LISTNEWITEMS, the newly added items of the last n days will be showed	7
bulkilyWarning	string	Text for ###BULKILY_WARNING### for bulky products	
bulkilyAddition	int+	Factor to multiply with a product which is bulkily	
bulkilyFeeTax	int+	Tax fee in percent for shipping of bulkily	

javaScript	*array of integers*	Some JavaScript which will be included for ###JAVASCRIPT_10### markers.
		
		Beispiel:
		`javaScript {`
		`  10 = function addValues (a, b) { return a+b; }`
		`  20 = function multiplyValues (a, b) { return a*b; }`
		`}`	

payment / shipping	(see below)	Configuration of payment and shipping methods, their values and costs and additional calculation scripts and payment gateways. See description below!	

[tsref:(script).class.tx_ttproducts_pi1.php]

Display Modes (CODE)

Here comes a list of the possible display types of the plugin.

Code to define, what the script does. In the backend these entries are made using flexforms instead of the Codes (capital letters). Use the codefields ony in TypoScript setup.

Display Mode:	CODE:	Description:
Products: list	LIST	listing of the products
Products: list gifts	LISTGIFTS	listing of gifts
Products: list highlights	LISTHIGHLIGHTS	listing of the products marked as highlights
Products: list offers	LISTOFFERS	listing of the products marked as offers
Products: list new items	LISTNEWITEMS	listing of the new items entered to the sysfolder
Products: list DAM	LISTDAM	list DAM images or media files
Products: single view	SINGLE	single view of an article (LIST code can be used also) or GET/POST var 'tt_products' can be set.
Products: search	SEARCH	displays a search dialog for searching product
Currency: selector	CURRENCY	currency selector
Basket: content	BASKET	Displays the shopping basket. The code 'BASKET' works in general but using the specific codes INFO, PAYMENT and FINALIZE, you can split out the function over multiple pages
Basket: overview	OVERVIEW	a minimum basket containing only the number of items
Basket: input customer data	INFO	enter address information
Basket: control and payment	PAYMENT	last check and payment gateway
Basket: finalize order	FINALIZE	finalize the order and send emails – thanks page for the order
Orders: tracking	TRACKING	to track the order state, bill and delivery
Orders: bill	BILL	creates a file containing the bill
Orders: delivery	DELIVERY	creates a file containing the delivery sheet
Orders: list	ORDER	display orders on per fe-user basis
General: memo	MEMO	memo of products for frontend users
General: help	HELP	information how to use tt_products
Categories: list	LISTCAT	listing of categories
Categories: select	SELECTCAT	categories inside of select boxes
Categories: DAM list	LISTDAMCAT	listing of DAM categories
Articles: list	LISTARTICLES	listing of articles

CSS configuration

The CSS id names can be set here. You have to provide a CSS file that will use these ids however.

You have to provide also the name of the table in the setup.

The last but one/two value will be the name of the view. It can be 'ALL', if it is valid for all views..

Views correspond to the code field:

SINGLE, LIST, BASKET

```
plugin.tt_products.CSS.tt_products.LIST.row.even = 35
```

Property:	Data type:	Description:	Default:
row		even: Cascading Stylesheets (CSS) even rows in the products display. uneven	
list		default: CSS for default entries in a list view. current: CSS for the currently selected item a list view.	
menu	string	CSS for the menu	
itemSingleWrap	wrap	HTML part to replace the markers ###ITEM_SINGLE_PRE_HTML### and ###ITEM_SINGLE_POST_HTML### at single item level.	| or <td>|</td>
itemRowWrap	wrap	HTML part to replace the markers ###ITEM_SINGLE_PRE_HTML### and ###ITEM_SINGLE_POST_HTML### at item row level.	empty or <tr>|</tr>

Configuration of Articles, Products, Categories, Pages and Images

The last but one/two value will be the name of the view. It can be 'ALL', if it is valid for all views..

Views correspond to the code field:

SINGLE, LIST, BASKET

additional possible values are

EMAIL, PAYMENT

Example:

```
plugin.tt_products.conf.tt_products_articles.LIST.generatePath.base = fileadmin/images
plugin.tt_products.conf.tt_products.LIST.orderBy = sorting
```

Property:	Data type:	Description:	Default:

generatePath	*array of string*	path to the image folders where the images for generateImage are located.
		
		Pairs of field names and the count of the first characters to be used to form the name of the image file.
		type ... tablefields
		fieldname ... name of the table field
		
		Example:
		`ALL.generatePath {`
		`  type = tablefields`
		`  base = fileadmin/images`
		`  field.itemnumber = 2`
		`}`
			fileadmin/img

generateImage	*array of string*	Pairs of field names and the count of the first characters to be used to form the name of the image file.
		type ... tablefields, foreigntable (for field of another table)
		field.*fieldname* ... name of the table field
		table ... use another table and its configuration to get the image
		uid_local ... use the value of this local field of the current table
		uid_foreign ... use this field of the foreign table to find a match
		
		Example:
		`ALL.generateImage {`
		`  type = tablefields`
		`  field.itemnumber = 6`
		`}`
		
		`ALL.generateImage {`
		`  type = foreigntable`
		`  table = tt_products_articles`
		`  uid_local = uid`
		`  uid_foreign = pid`
		`  field.itemnumber = 6`
		`}`	

Property:	Data type:	Description:	Default:

imageMarker	*array of string*	Defines how the marker for the image is composed. In this example image names like 30_P1_001.jpg can be used, where the second part P1 and the third part 001 form the marker. So the marker will be ###CATEGORY_IMAGE_P1_001###.
		
		Example:
		`ALL.imageMarker {`
		`  type = imagename`
		`  parts = 2,3`
		`}`	

orderBy	string	List of the fields by which the items will be ordered.	sorting

fetchImage	string	If set, the images for the table are taken from the images of another table
		
		Example:
		`plugin.tt_products.conf.tt_products.ALL.fetchImage {`
		`  type = foreigntable`
		`  table = tt_content`
		`}`	

language	*array of string*	The name of a language file with translations from the default language into another language.
		type:
		• csv ... The values are separated by ';' and newline characters
		• noTranslation ... do not use the language overlay table
		• field ... the translation is in fields
		• table ... the translation overlay table
		
		file: Path and name of the file
		field: name of the field on left and new value on right side
		notinst
		Example:col
		`[globalVar = GP:L = 1]`
		`language {`
		`  type = csv`
		`  file = fileadmin/data/EnglishCategories.csv`
		`}`
		`[GLOBAL]`
		
		Example:
		`language {`
		`  type = field`
		`  field.title = subtitle`
		`}`	

image	IMAGE cObject	Image is copied into the others via TypoScript and can be used for several code fields. **Example:** `###PRODUCT_IMAGE1:M###` `plugin.tt_products.conf.tt_products.ALL.image.m {` `  wrap = | ` `  file.maxW = 320` `  file.maxH = 280` `}`	

filter	*array of string*	Use only table records which apply to a filter on a field base.
		type:
		
		• regexp ... use a regular expression
		field: name of the field on left and value on right side
		
		Example:
		`filter {`
		`  type = regexp`
		`  field.title = [:alpha:]+[:blank:]+1[:blank:]+`
		`}`
		
		Will filter all records of the table to use only those where the title has characters and a 1 on the last position.	

urlparams	string	Comma separated list of tt_products URL parameters which must have a value. Otherwise no items will be displayed. Normally no products shall be shown below a category list when no category has been selected yet. If you leave this empty, then all products will be listed in the list view when no category parameter is given and you have a category list view on the page.	

Property:	Data type:	Description:	Default:
displayColumns	string	Number of columns on the display You have to adapt your template using special template markers. The ITEM_SINGLE_PRE_HTML and ITEM_SINGLE_POST_HTML must surround your <TD>-</TD> tags, so the table will be created correctly. The first number is the order in the category hierarchy. **Example:** `displayColumns {` `  1 = 3` `}` **Example:** `<!-- ###ITEM_SINGLE### begin-->` ` ###ITEM_SINGLE_PRE_HTML###` ` <TD height="150" bgcolor="###GC1###" valign="bottom" align="center">` ` <!--###LINK_ITEM###-->` `###GW1B###<b>###PRODUCT_TITLE###</b>###GW1E###` ` <!--###LINK_ITEM###-->` `  ` ` ###PRODUCT_IMAGE###  ` ` ###GW1B### item count: ###GW1E### <INPUT size="3" maxlength="4" type="text" name="###FIELD_NAME###" value="###FIELD_QTY###">` ` <p> </TD>` ` ###ITEM_SINGLE_POST_HTML###` ` <!-- ###ITEM_SINGLE### end -->`	

Form configuration

Setup only. There are several forms which can be configured. Put the code after the form settings.

Example:

```
plugin.tt_products.form.SEARCH.name = ShopSearchForm
```

Property:	Data type:	Description:	Default:
name		name of the form. If empty the whole subpart will not be drawn.	*depends on codefield*

Basket configuration

You can configure the behaviour of the basket here.

Example:

```
plugin.tt_products.basket.minPrice {
    type = price
    collect = goodstotal
    value = 250
}
```

Property:	Data type:	Description:	Default:

minPrice	*array of string*	minimum price which the products must reach to get a permission to buy them. E.g. only products at a total price of at least 250 shall be accepted.
		
		Example:
		`minPrice {`
		`  type = price`
		`  collect = goodstotal`
		`  value = 250`
		`}`	

Payment and shipping configuration

Payment and shipping are very similar in configuration and therefore shared the same property list with special notes if something is for the one type only. The configuration of payment and shipping is in short a question of defining the items to choose from on the basket page. That is, a choice of one out of many transportation methods and one out of many payment methods. Therefore you can for instance select either radio-button representation or selector box.

The number of the selected payment method or shipping method is reflected in the html-template certain places and you may also want special PHP scripts executed based on the settings. That's all allowed.

Property:	Data type:	Description:	Default:
radio	boolean	If set, you get radio button layout. If not, selector-box.	0

Property:	Data type:	Description:	Default:
template	string	**(Radio layout only)** If .radio is true, this string is the 'template' for the radio items. **Default is (in one line):** `<nobr>###IMAGE### <input type="radio" name="recs[tt_products][.$key.]" onClick="submit()" value="###VALUE###"###CHECKED###> ###TITLE###</nobr> `	
wrap	string	**(Select layout only)** If .radio is false, this string wraps the <option> tags in a <select>-tag! **Default is (in one line):** `'<select name="recs[tt_products][.$key.]" onChange="submit()">|</select>'`	

TAXpercentage	double	TAX/VAT percentage.
		Double value (!) (means, "use . as decimal point")
		This substitutes priceNoTax. This can be different to the global tax with the same name.
		
		Example:
		`# Danish payment TAX is 25%:`
		`payment.TAXpercentage = 25.00`
			from global

TAXincluded	boolean	Set this, if TAX *is* included in the payment/shipping prices.	from global

Array of integers		Configuration, see below
		
		Examples:
		`TAXpercentage = 12`
		`10.title = Credit card`
		`10.image.file =`
		`typo3/sysext/cms/tslib/media/logos/dankort.gif`
		`10.price =`
		`10.percentOfGoodstotal = 0`
		`10.calculationScript =`
		`EXT:tt_products/pi1/products_comp_calcScript.inc`
		
		`30.title = By mail`
		`30.image.file =`
		`typo3/sysext/cms/tslib/media/logos/postdanmark.gif`
		`30.price = 40`	

Configuration of payment / shipping items			
title	string	Title of item, eg. “Master card” or “Ground mail” The title will be cloned, if markers like ###STATIC_COUNTRIES_CN_ISO_3### are used together with where.static_countries.	
image	IMAGE cObject	Logo image for the item	

price	double or array of integers *see below for additional parameters*	Price of item, including or excluding VAT, depends on TAXincluded.
		
		You use integers to specify the minimal number of items for which the price is valid. 6 items and more will cost 5.8 in this example.
		
		Example:
		`30.price.type = count`
		`30.price.1 = 4`
		`30.price.6 = 5.8`	

~~priceTax~~	double	Depricated (1st July 2008) Price of item, including VAT	use price, TAXpercentage and TAXincluded instead
~~priceNoTax~~	double	Depricated (1st July 2008) Price of item, excluding VAT Notice: you have to calculate the VAT amount here by yourself!	use price, TAXpercentage and TAXincluded instead
replaceTAXpercentage	double	see TAXpercentage If set, the general TAXpercentage will be overwritten by this value.	
priceFactWeight	double	**shipping ONLY:** Price is calculated from weight of all products. Will be added to price. The weight is mulitplied with this factor to increase the shipping price.	
percentOfGoodstotal	double	Price of item, calculated from a percentage of the total amount before payment/shipping	

Property:	Data type:	Description:	Default:
percentOfTotalShipping	double	**payment ONLY:** If set the payment costs are calculated in the percentage of the total product tax price inclusive the shipping tax price.	
creditcards	string	**payment ONLY:** Comma separated list of allowed uids for the creditcards. See file localland_db.xml about the values. (sys_products_cards.cc_type.I) 0 ... American Express 1 ... Diners Club 2 ... Mastercard 3 ... Visa	
accounts	boolean	**payment ONLY:** If set the payment with booking from the entered bank account will be allowed.	
calculationScript	resouce	PHP script which is included in a “blank” function and it should be written to manipulate amounts in the internal arrays. This script could be used to calculate a special fee regarding a payment/shipping item. For an example of application, see media/script/products_comp_calcScript.inc which shows you how to raise the final amount with 5.75% of it's own value as to compensate for fees to international credit card organisations. Properties of the calculation script is passed to the function as $conf array.	
handleScript	resource	PHP script which is included in a “blank” method called from products_basket() in user_products class when the order is finalized. This function must take care of displaying templates during the payment process with a payment gateway as well as finalizing the order afterwards. See pi/payment_DIBS.php for an example. A HTML-template file follows. Properties of the handle script is passed to the function as $conf array. The content of the variable $content is returned as content.	
handleLib	string	**payment ONLY:** Name of the TYPO3 library to handle the Payment. Currently you can set only 'paymentlib' here to use Rober Lemke's Payment Library Extension. **Example:** `30.handleLib = paymentlib`	
handleURL	string	If set, this handleURL is called instead of the THANKS-url (by PIDthanks) in order to let eg. a handleScript process the information if payment by credit card or so.	
handleTarget	string	Alternative target for the form.	

excludePayment	*list of integers*	**shipping ONLY:**
		This is a list of payment method keys (their numbers) which are not available given a certain delivery form. For instance if people pick up goods in the store, you don't want them to transfer money or pay online but just order the goods. So you can exclude those payment methods.
		
		Example:
		`...`
		`  40.title = Pick up in store`
		`  40.excludePayment = 10,40`
		`}`	

replacePayment	*list of integers*	**shipping ONLY:**
		This is a list of payment settings which will be overridden if this shipping method has been selected.
		
		Example:
		`...`
		`  40.title = China`
		`  40.replacePayment.10.title = Payment with China`
		`  40.replacePayment.10.price = 100`
		`}`	

show	boolean	If set, the item is shown in the list.	1
showLimit	double	If set, then this item will only get shown when there is at least this number of products in the basket. 0 ... always show this item	0
type	string	**Payment ONLY:** fe_users ... the payment can be configured via the fe_users table	
visibleForGroupID	int+	**Payment ONLY:** This payment method is only available, if a user is logged in and member of this frontend user group	

Property:	Data type:	Description:	Default:
addRequiredInfoFields	string	**Payment ONLY:** Additional required fields in the INFO page, if this payment method is selected. Useful for credit card payment.	
Configuration price Parameters for payment / shipping			
type	string	Meaning of the number: count ... the products count weight ... the calculated weight in Gramm price ... the total products price	

WherePIDMinPrice	int+	**shipping ONLY:**
		Set a minimum price for shipping if there is an item in the basket which is from the sysfolder of this PID.
		Where 155 is the PID and 7.5 is the minimum price taken for shipping costs when at least one product in the basket comes from the page with that PID.
		
		Example:
		`plugin.tt_products.shipping {`
		`10.title = Parcel`
		`10.price.type = weight`
		`10.price.WherePIDMinPrice.155 = 7.5`
		`10.price.1 = 1.5`
		`10.price.500 = 2.5`
		`10.price.1000 =3.5`
		`}`	

where.static_countries	string	**shipping ONLY:**
		Set a SQL WHERE condition to follow for the selected country in the static_countries table of the static_info_tables extension.
		
		Example:
		`plugin.tt_products.shipping {`
		`10.title = Parcel Germany`
		`10.where.static_countries = cn_short_local == 'Deutschland'`
		`10.price = 5.9`
		`20.title = Parcel EU`
		`20.where.static_countries = cn_eu_member = 1 AND cn_short_local != 'Deutschland'`
		`20.price = 8.9`
		`30.title = Outside EU`
		`30.where.static_countries = cn_eu_member <> 1`
		`30.price = 15`
		`}`	

~~productsNoTax~~	~~int+~~	~~**shipping ONLY:**~~ ~~Set if the taxes are included in the price for all products.~~	

noCostsAmount	double	When the total amount for the products reaches this value then no costs will be calculated.
		
		Example:
		`plugin.tt_products.shipping {`
		`10.title = Parcel`
		`10.price.type = weight`
		`10.price.noCostsAmount = 200`
		`10.price.1 = 1.5`
		`}`	

Pricecalc, discountprice and creditpoints configuration

The pricecalc gives you the possibility to build the price sum of products using a calculation table. The discount price will be used for all users who belong to the group set in discountGroupName.

Property:	Data type:	Description:	Default:
prod	*two-edged list of integers*	The left edge of integers correspond to lines belonging together, the meaning of the right edge depends on the settings for each line.	0

		Pricecalc:
		Special Prices for the products. Where 1 product costs 4.99, 2 products will cost 8.99. With discount price this will form the price for one product, with pricecalc it is the price for all products together where 1 has cost 4.99 in the products folder. The discountprice overrides the pricecalc if possible, because this should be cheaper then. A price calculation from here will get replaced if price2 is used.
		
		Example:
		`pricecalc {`
		`10.type = count`
		`10.field = price`
		`10.where =`
		`10.prod.1 = 4.99`
		`10.prod.2 = 8.99`
		`10.prod.5 = 19.99`
		`20.type = count`
		`20.field = price`
		`20.where =`
		`20.prod.1 = 6.99`
		`20.prod.2 = 13.98`
		`20.prod.5 = 29.99`
		`}`	

		Discountprice:
		Here the single prices for products are calculated depending on the count of articles, if type=count.
		The additive settings tells if all the products are counted together even from different lines.
		
		Example:
		`discountprice {`
		`10.type = count`
		`10.field = price`
		`10.additive = 1`
		`10.where =`
		`10.prod.1 = 4.99`
		`10.prod.100 = 2.49`
		`10.prod.1050 = 2.39`
		`20.type = count`
		`20.field = price`
		`20.where =`
		`20.prod.1 = 6.99`
		`20.prod.100 = 2.59`
		`20.prod.1050 = 2.49`
		`}`	

		Creditpoins:
		This tells you how many creditpoints someone will get if he buys articles in the shop. The right values are the percentage of the price of the ordered articles, if type=price.
		
		Example:
		`creditpoints {`
		`pricefactor = 0.5`
		`10.type = price`
		`10.prod.1 = 0.02`
		`10.prod.101 = 0.04`
		`10.prod.501 = 0.06`
		`}`	

additive	double	Only valid for discount price. If set all the products with any of these discount prices are counted together to calculate which discount price will apply. If unset only the products of the same price are counted.	
type	string	Meaning of the right edge integer which usually gets calculated: count ... the products count (pricecalc and discountprice only) price ... the price field is used (creditpoints only)	

Property:	Data type:	Description:	Default:

pricefactor	double	Used to calculate how much money someone will get for his creditpoints. 2 creditpoins will give 1 Euro or the currency of your choice.
		
		Example:
		`creditpoints {`
		`  pricefactor = 0.5`
		`}`	

[tsref:(script).productsLib.payment/(script).productsLib.shipping]

payment_DIBS.php

Properties of the handleScript for DIBS interfacing (http://www.architrade.com/uk/):

Property:	Data type:	Description:	Default:
templateFile	resource	Template file for use with DIBS You have to put the following line in the form of the tt_products template before the DIBS script will be called: `<input type="hidden" name="products_cmd" value="cardno">`	
soloe	boolean	If set, the script uses sub-template with marker ###DIBS_SOLOE_TEMPLATE### instead of the default which is ###DIBS_CARDNO_TEMPLATE###	
direct	boolean	If set, the script uses sub-template with marker ###DIBS_DIRECT_TEMPLATE### instead of the default which is ###DIBS_CARDNO_TEMPLATE###	
merchant	boolean	Merchant id	
currency	int+	Currency number, ISO4217 format	
relayURL	string	The url of the shop where their secure server is going to fetch the basket.	
test	boolean	If set, the test-field is set in the forms.	

cardType	string	Card type,
		
		Example values:
		`DK = Dankort`
		`V-DK = Visa-Dankort`
		`MC(DK) = Mastercard/Eurocard issued in Danmark`
		`VISA = Visakort issued abroad`
		`MC = Mastercard/Eurocard issued abroad`
		`DIN(DK) = Diners Club, Denmark`
		`DIN = Diners Club, international`	

account	string	DIBS account feature	
addOrderInfo	boolean	If set, order info is added to the form. DIBS can pickup this info and simply display it with the payment information.	
k1 k2	string	DIBS key values	

[tsref:(script).productsLib.paymentDIBS]

Tutorial

If you are a beginner with the shop system, you should start with the step by step tutorial which is available under the extension key *tut_ttproducts_de*.

Example of a configuration from Inter-Photo A/S (www.inter-photo.dk):

```
xxxxx.xxxx {
  10.title = Dankort, VISA-Dankort
  10.image.file = media/logos/dankort.gif
  10.image.params = align=absmiddle vspace=2
  10.handleScript = media/scripts/payment_DIBS.php
  10.handleScript {
    merchant = xxxxx
    test = 0
    k1 = xxxxx
    k2 = xxxxx
    currency = 208
    addOrderInfo = 1
    account =
    cardType = DK,V-DK
    relayURL = http://www.inter-photo.dk/index.php?id=204
  }
  10.handleURL = index.php?id=204
  10.handleTarget = _top

  20 < .10
  20.title = Unibank e-betaling
  20.image.file = media/logos/soloe.gif
  20.handleScript.soloe=1
}
```

Example in product view make product title the page's title

If you want to fetch the product title into some marker, i.e. for use in automake_template or similar, try this code:

Code Listing:

```
# first set pagetitle to the page's title
temp.pagetitle = TEXT
temp.pagetitle.field = title

# now overwrite with the product's title in case it's non-empty
[globalVar = GP:tt_products > 0]
temp.pagetitle = COA
temp.pagetitle {
    10 = RECORDS
    10 {
        source.data = GPvar:tt_products
        tables = tt_products
        conf.tt_products = TEXT
        conf.tt_products {
           field = title
        }
    }
}
// For single record display, cache has to be disabled.
config.no_cache = 1
[global]

temp.mainTemplate = TEMPLATE
temp.mainTemplate {
    # Feeding the content from the Auto-parser to the TEMPLATE cObject:
    template =< plugin.tx_automaketemplate_pi1
    # Select only the content between the <body>-tags
    workOnSubpart = DOCUMENT_BODY
    subparts.title < temp.pagetitle
....
```

Example templates

Here comes the template part, when a new user has been registered automatically as front end user.

```
<BR><BR><BR><BR><HR><BR><BR><BR>
<h3>EMAIL_NEWUSER_TEMPLATE</h3>
<em>
Subpart used as template for the account-creation-emails. First line is
used as subject for the mail.<br>
</em><br>
<pre>
```

```
<!-- ###EMAIL_NEWUSER_TEMPLATE### begin
Subpart used as template for the account-creation-emails First line is
used as subject for the mail.
-->
New user account created
Dear ###PERSON_NAME###,
you have made an order at http://...../ for the first time.
To make orders in the future more easy a user account has been created.
Your account data:
user name: ###USERNAME###
password: ###PASSWORD###
Regards,

the Shopmaster
<!-- ###EMAIL_NEWUSER_TEMPLATE### end -->
</pre>
```

Example bananaGuard

(see file example_template_bill_de.tmpl delivered from http://bananaguard.de)

This example includes special price calculations and the automatic creation of a bill and delivery sheet. You have to install the extensions feuser_admin and conf_userts if you want to build a similar shop. Move the example_template_bill_de.tmpl to fileadmin/tmpl_files/products_eur_.tmp. To use this template you have to search for '?id=' in your text and substitute to following PIDs with those of your system.

The pages and plugins for the page tree are:

- 'BananaGuard' with main template of your choice and

Constants:

```
plugin.tx_srfeuserregister_pi1.email = info@shopms.de
plugin.tx_srfeuserregister_pi1.confirmPID = 83 ... You have to use your confirm PID.
styles.content.loginform.pid = 108 ... Your Users sysfolder PID.
```

-- 'Home' which is a shortcut to BananaGuard

-- 'Preise und Versand' with special information about the prices and shipping

-- 'BananaShop' which has its own template especially for usage with the shopping system

Constants:

```
plugin.tt_products.file.templateFile = typo3/ext/tt_products/pi/products_eur_.tmpl
plugin.tt_products.TAXpercentage = 16
plugin.tt_products.priceDecPoint = .
plugin.tt_products.maxW_list = 80
plugin.tt_products.outputfolder = fileadmin/data

plugin.tt_products.color2 = #003399
plugin.tt_products.color1 = #FFFFFF
plugin.tt_products.wrap2 = <font face="verdana" size=2 color=white> | </font>
content.tableCellColor = #003399
```

Setup:

```
plugin.tt_products.code.field = select_key
plugin.tt_products.alwaysInStock = 1

plugin.tt_products.statusCodes.1 = Bestellungseingang
plugin.tt_products.statusCodes.11 = Der bananaSHOP wartet auf Ihren Zahlungseingang
plugin.tt_products.statusCodes.20 = Ihre Ware wird versendet
plugin.tt_products.statusCodes.101 = Bestellung abgeschlossen
plugin.tt_products.statusCodes.200 = Bestellung storniert

plugin.tt_products.orderEmail_from = info@bananaguard.de
plugin.tt_products.orderEmail_fromName = bananaGUARD.de
plugin.tt_products.orderEmail_to = info@bananaguard.de
orderNumberPrefix = order2005_

plugin.tt_products.discountGroupName = Team
plugin.tt_products.lockLoginUserInfo=true

plugin.tt_products.conf.tt_products.LIST.displayColumns = 3
plugin.tt_products.outputFolder = {$plugin.tt_products.outputfolder}

plugin.tt_products {
  payment >

payment {
  radio = 1
  TAXpercentage = 16
  10.title = Vorkasse
  20.title = PayPal
  20.percentOfTotalShipping = 0.04
  30.title = Nachnahme
```



```
  30.discountDeactive = 1
  30.price.1 = 4
  30.price.6 = 5.8
  30.showLimit = 99
 }

shipping {
  radio = 1
  TAXpercentage = 16
  10.title = Deutschland
  10.image.file =
  10.price.type = count
  10.price.1 = 2.5
  10.price.6 = 4.8
  10.price.50 = 10
  10.price.100 = 25
  10.price.120 = 30
  10.price.300 = 150
  10.percentOfGoodstotal = 0

  20.title = Europa (nicht Deutschland)
  20.image.file =
  20.price.type = count
  20.price.1 = 8.8
  20.price.6 = 11.8
  20.price.50 = 11.8
  20.price.100 = 25
  20.price.120 = 30
  20.price.300 = 150
  20.percentOfGoodstotal = 0

  30.title = Selbstabholung
  30.image.file =
  30.price.1 = 0
  30.percentOfGoodstotal = 0
  }

pricecalc {
  type = count
  field = price
  10.prod.1 = 4.99
  10.prod.2 = 8.99
  10.prod.5 = 19.99
  20.prod.1 = 6.99
  20.prod.2 = 13.98
  20.prod.5 = 29.99
}

discountprice {
  10.type = count
  10.field = price
  10.additive = 1
  10.where =
  10.prod.1 = 4.99
  10.prod.100 = 2.89
  10.prod.1050 = 2.77
  20.type = count
  20.field = price
  20.where =
  20.prod.1 = 6.99
  20.prod.100 = 3.00
  20.prod.1050 = 2.89
}
}

plugin.tt_products.basketImage.imageLinkWrap.height = 800
plugin.tt_products.listImage.file.maxW = 150
plugin.tt_products.shipping.40 >
plugin.tt_products.PIDagb =

plugin.tt_products.createUsers = 1
plugin.tt_products.orderEmail_htmlmail = 1
```

Plugin:

Products, CODE: LIST

The next pages are:

--- 'Shopping Basket'

Plugin:

Products, CODE: BASKET, At the BORDER a login content type with send to page 'Warenkorb'

--- 'Cash Box'

Plugin:

Products, CODE: PAYMENT, FINALIZE, INFO

--- 'Order Status'

Plugin:

Products, CODE: TRACKING

--- 'Invoicing'

Plugin:

Products, CODE: BILL

--- 'Delivery'

Plugin:

Products, CODE: DELIVERY

--- 'Articles' Sysfolder

Plugin:

Products, CODE: LIST

-- 'Contact/Order' contains a form, text and login

-- 'BananaINSIDER' only visible after frontend login, contains a login plugin at the right margin

Access of the visibility settings is the group 'Team'

--- 'My profile'

Plugin:

Frontend User Registration

--- 'INSIDER Infos'

--- 'bananaBOARD'

Plugin:

Board, Tree, CODE: FORUM, POSTFORM

-- 'Users' Sysfolder of the group 'Team'

Known problems

Checklist:

- include a 'Shop System' static template file (from extensions)
- **never** use the 'Shop System Test' or the 'plugin.tt_producs [DEPRECATED]'
- set 'in stock' greater 0 for the products and articles
- do not use languages or have products in the products alternative languages tables
- the shop template file is found
- an error in the shop produces an entry in the PHP error_log file (activate this in Install Tool and php.ini)
- the cache has been cleared before
- set the pid_list and recursive in TypoScript or set the startingpoint/recursive inside of the shop plugins

General:

– If you make an update with the Extension Manager from an older version of tt_products and did not install the Table Library 'table' and FEBE Library 'fh_library' before that, you will end up in the error message

TYPO3 Fatal Error: Extension key "table" was NOT loaded! (t3lib_extMgm::extPath)

Steps to undo the TYPO3 fatal error:

```
1. edit typo3conf/localconf.php
2. remove the tt_products entry
3. delete the temp_CACHED_ files in typo3conf
4. press reload from the browser
```

– Please look at the website http://wiki.typo3.org/index.php/Ext_tt_products .

– Get the latest development version at http://ttproducts.de

– .htaccess must be properly configured otherwise tt_products doesn't work as expected because the plugin can't find all necessary scripts with poor configuration of .htaccess.

– **Wrong parameters, GET/POST var 'tx_ttproducts_pi1[product]' was missing or no product with uid = 0 found.** You should set the PIDitemDisplay to inform the shop how the link parameter 'tx_ttproducts_pi1[product]' to the single view shall be generated.

– **Pay attention to all legal claims of all the countries to where you send goods!**

– Plugin doesn't produce any output
See the checklist above.

To-Do list

– Wishlist: http://wiki.typo3.org/index.php/Ext_tt_products#Wishlist

– Rewriting of the code for PHP5.

Changelog

See the file tt_products/ChangeLog for more details

- 26.03.2005 bring in of zk_products from Klaus Zierer (zk_products), multiple column listing, bill , delivery and special price calculations by Franz Holzinger
- 06.05.2005 second price and price for additional accessory from Jens Schmietendorf, example template from http://bananaguard.de , zk_products 1.3.2 from Klaus Zierer, VAT by Franz Holzinger
- 23.06.2005 products_mail.inc deleted, example template by Franz Holzinger
- 26.07.2005 Display orders, creditpoint and voucher system by Bert Hiddink
- 03.08.2005 Notes from the zk_products forum, PIDtracking
- 11.09.2005 Gift certificates
- 14.09.2005 Flexforms instead of CODEs
- 28.10.2005 Accessory has been replaced by sizes with article table
- 23.02.2006 Better English translation sponsored by Bill Alexy
- 14.07.2006 You must now insert the static shop template manually from the template setup.
CSS styled template by Robert Markula
- 06.10.2007 The CODE field will be shown in the page module for the chosen flexforms.