webEdition
manage your content

Customer Management
and
Shop Modules

The Customer Management and Shop Modules

User Guide

Standard 5.0.1
June 2007

Printed in Germany

Contents

List of figures

List of procedures

About this document

Purpose of this document

This document treats the webEdition Customer Management (PRO) and the Shop Modules and how to use them.

You can use this manual to learn:

- what the Customer Management Modules are
- how to install the modules
- how to use these modules to build and maintain a customer database
- how to create customer registration forms to manage external visitors to your Web site

Audience

This document is intended for personnel in the following groups:

- Web administrators
- Web editors

The webEdition customer documentation suite

The documentation team publishes new webEdition documents to support the release of all webEdition features, modules and enhancements.

You can consult our documentation suite for detailed information about the modules you have purchased or about webEdition products that you may wish to purchase in the future. All customer documentation is available in portable document format (PDF) on the webEdition documentation Web page.

On-line reference documentation

The webEdition customer documentation suite comprises the following books, all of which you can obtain at URL: http://www.living-e.de

Standard webEdition documentation

The following books support the webEdition Standard suite:

- *The webEdition User Guide*
- *The webEdition Installation Guide*
- *The webEdition Tag Reference*

Documentation for webEdition modules

The following books support the webEdition modules:

- *The Customer Management and Shop Module User Guide*
- *The Database/Object Module User Guide*
- *The Newsletter Module User Guide*
- *The User Management PRO Task/Messaging and Workflow Module User Guide*

What precautionary messages mean

webEdition documents include attention and caution messages, which are designed to draw your attention to important instructions.

Attention boxes

An attention box identifies information that is necessary for the proper performance of a specified task. For example:

> **ATTENTION**
> You must have the appropriate permissions in your user profile to complete this procedure. Permissions are assigned to you by your webEdition system administrator. Contact your webEdition system administrator for further details.

Caution boxes

Caution messages indicate that there are possible risks to your software or database if you perform a specified task without taking the suggested appropriate precautions. For example:

> **CAUTION**
> **Database warning**
> If you complete this procedure, your database will be overwritten.

How commands, system responses and we:tags are represented

The commands, system responses and webEdition tags (called we:tags) used in this document conform to the following conventions.

Web interface commands

Commands performed through a Web browser are shown in *italic* typeface. For example:

Click on the *Save* button.

Menu-based commands performed in a Web browser are shown in *italic* typeface. Drop-down or nested menu commands are separated by an input prompt (>). For example:

Select *Customers* > *New* from the main menu of the Customer Management Module.

webEdition tags and template code

The webEdition templates use a specialized set of programming tags based on the PHP programming language. These webEdition tags or we:tags are displayed in `courier` typeface and in angled brackets:

Opening tags appear thus: `<we:tag/>`

Closing tags appear thus: `</we:tag>`

The programming code used in webEdition templates is also represented in this document by `courier` typeface:

```
<we:sessionStart/>
<we:ifRegisteredUser>
Hello: <we:sessionField Last name="user name" type="print"/><br>
Logged in
</we:ifRegisteredUser>
```

Attribute variables

Attributes and variables appear in *`courier italic`* typeface. For example:

`<we:hidden name="`*`attribute1`*`">`

How to check the document version and issue

The information on the title page of this document indicates the version and issue for this publication. The version and issue also appear in the footer on every even-numbered page.

The first two digits in the document numbering scheme indicate the version. The version number increases each time the document is updated to support a new software release. For example, the first release of a document is 01.01. In the next software release cycle, the first release of the same document is 02.01.

The second two digits in the document numbering scheme indicate the issue. The issue number increases each time the document is revised and re-released in the same software release cycle. For example, the second release of a document in the same software release cycle is 01.02.

Customer service

For further information about webEdition, please consult our Web page, or contact our customer service department:

- Web Page:http://www.webedition.biz/
- E-mail:
  - Technical support: technik@living-e.de
  - Sales: sales@living-e.de
  - Information/Help: info@living-e.de

1 Shop Module: Introduction

This introduction is intended to help you familiarize yourself with the webEdition Shop Module. This chapter treats what the module does and how to install it. You can also find information here about the basic layout and command features for the Shop Module. These topics are treated in the following sections:

- Section 1.1, "What is the webEdition Shop Module?" on page 1
- Section 1.2, "Installation" on page 1
- Section 1.3, "General information and navigation" on page 2

1.1 What is the webEdition Shop Module?

The webEdition Shop Module is a toolkit that can assist you to set up a Web-based shopping site.

With the Shop Module you can

- create any number of items or item groups
- design item summaries (listviews) and detailed item views
- place shopping carts anywhere on your pages
- organize your order management
- perform transaction calculations (sales taxes, discounts, etc.)
- use an open interface to a payment provider
- integrate your shop data with the webEdition Customer Management Module

1.1.1 Intended users

The Shop Module is a toolkit intended for advanced users of webEdition, and those with a knowledge of e-commerce Web site architecture and design. Knowledge of the webEdition template development processes, and webEdition tags (we:tags) are required in order to develop an online shop.

1.1.2 Interaction with the Customer Management Module

The Shop Module needs the webEdition Customer Management (PRO) Module to be installed. You cannot use the Shop Module without the Customer Management Module.

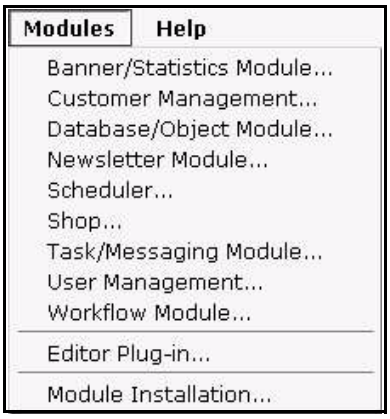
1.2 Installation

The installation procedure for all modules is described in The webEdition Installation, Update and Backup Procedures. A .pdf version of this guide is available at the following URL: http://www.living-e.de

1.3 General information and navigation

After installation, you will find a new menu item in the main menu called *Modules*, which contains all the modules in your installation of webEdition (see Figure 1).

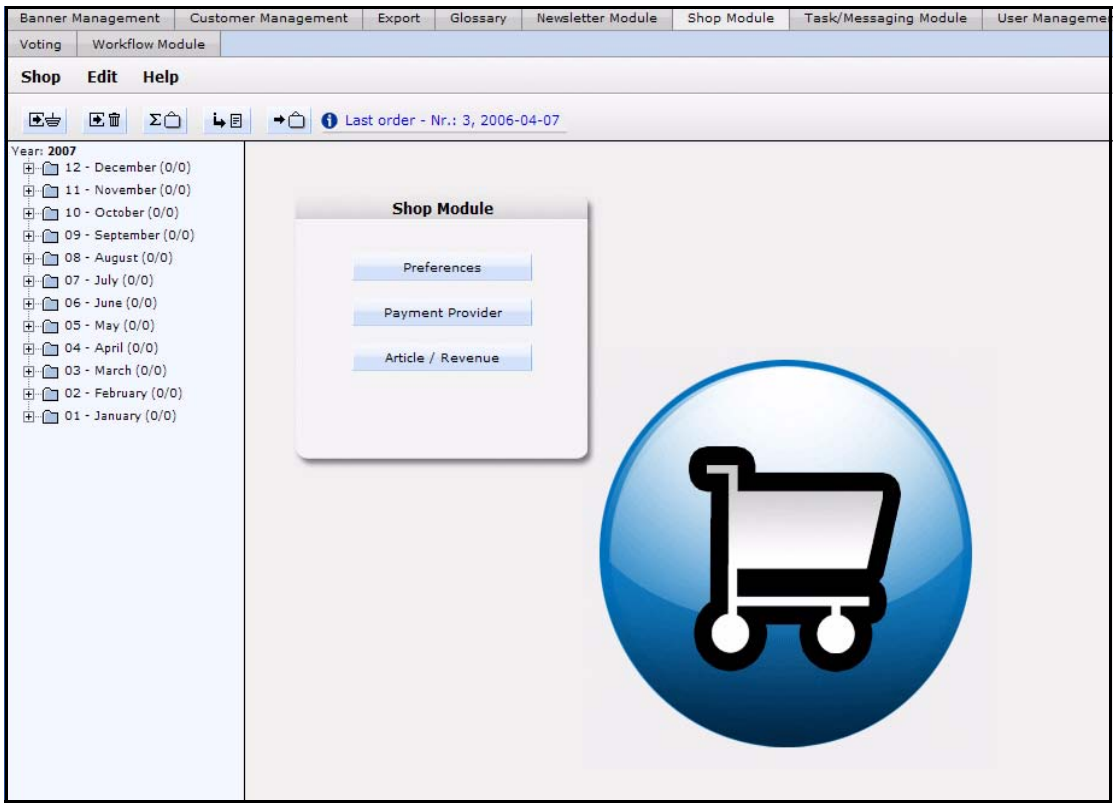
Figure 1 *Modules* drop-down menu



1.3.1 Shop Module *Quickstart*

Click on *Modules* > *Shop* to open the module summary window containing the Shop Module. The *Quickstart* screen for the Shop Module opens (see Figure 2).

Figure 2 The Shop Module start screen with *Quickstart*



1.3.1.1 Preferences

The module opens with the *Shop Quickstart* dialog box. Click the *Preferences* button to set your initial Shop preferences. You can choose from the following options:

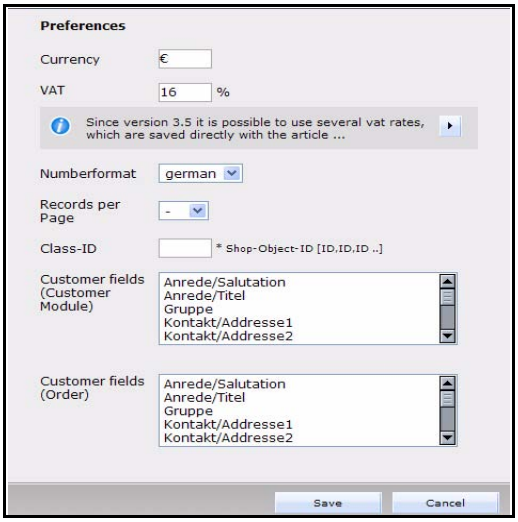


Figure 3 Shop settings screen

•*Currency*. Select the currency of choice.

•*VAT*. Set the tax rate (as a percentage) that you are obliged to collect.

•*Number format*. Choose the number format from the select box: *German*, *English*, *French.* or *Swiss*

•*Records per page* Set the number of records to be dispalyed per page.

•*Class-ID* This field is only displayed, if the *Database /Object Module* is installed. If objects are to be used as shop articles, the IDs of all classes containig shop articles must be entered here comma-separated. This includes the listing of all shop articles as well as the features to extend orders. In the field Class-ID many classes can be entered arbitrarily.

- *Customer fields (Customer Module)*: Select here the fields to be indicated (from the Customer Management Module). To select several fields, please hold the the Shift key while selecting. These preferences concern the announcement within the Shop Module.
- *Customer fields (Order)*: Select here the fields which should be indicated the customer. As a rule customer's fields of the Customer Management are used to hold on current dates of the customer, as for example comments to the customer of the web pages operating authority (payment behavior, etc. …). By contrast customer fields of the order are necessary, around address dates, etc. to be able to prove at the time of the order, so that also afterwards still valid order dates are available.

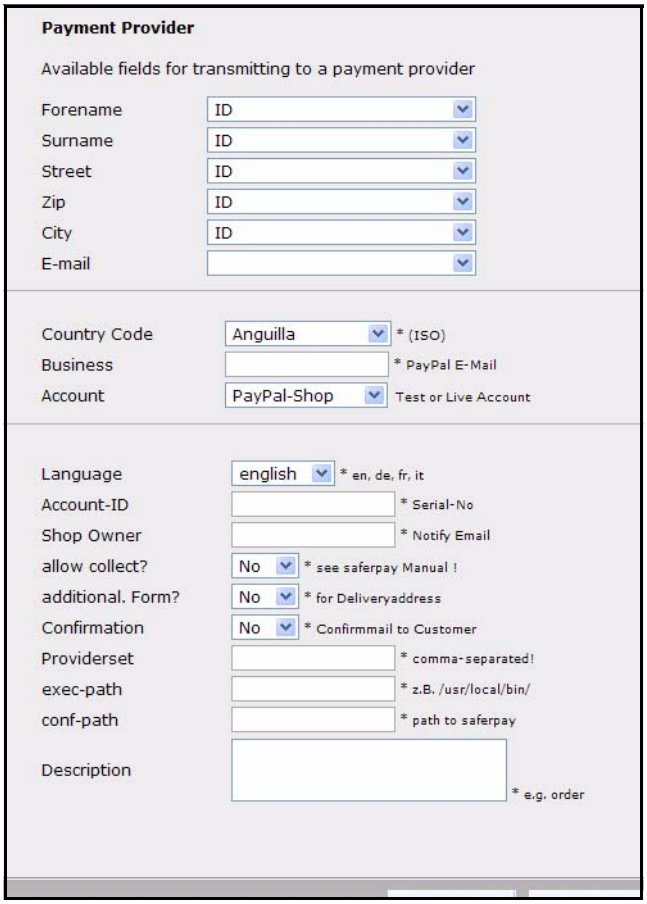
Save your preferences by clicking on the *Save* button.

Note: You can change these preferences at a later date by selecting *Shop* > *Settings*... from the main drop-down menu.

1.3.1.2 Payment Provider

Click in the Quickstart menu on the button Payment Provider, the corresponding window is opened (see

Figure 4 Pament Provider



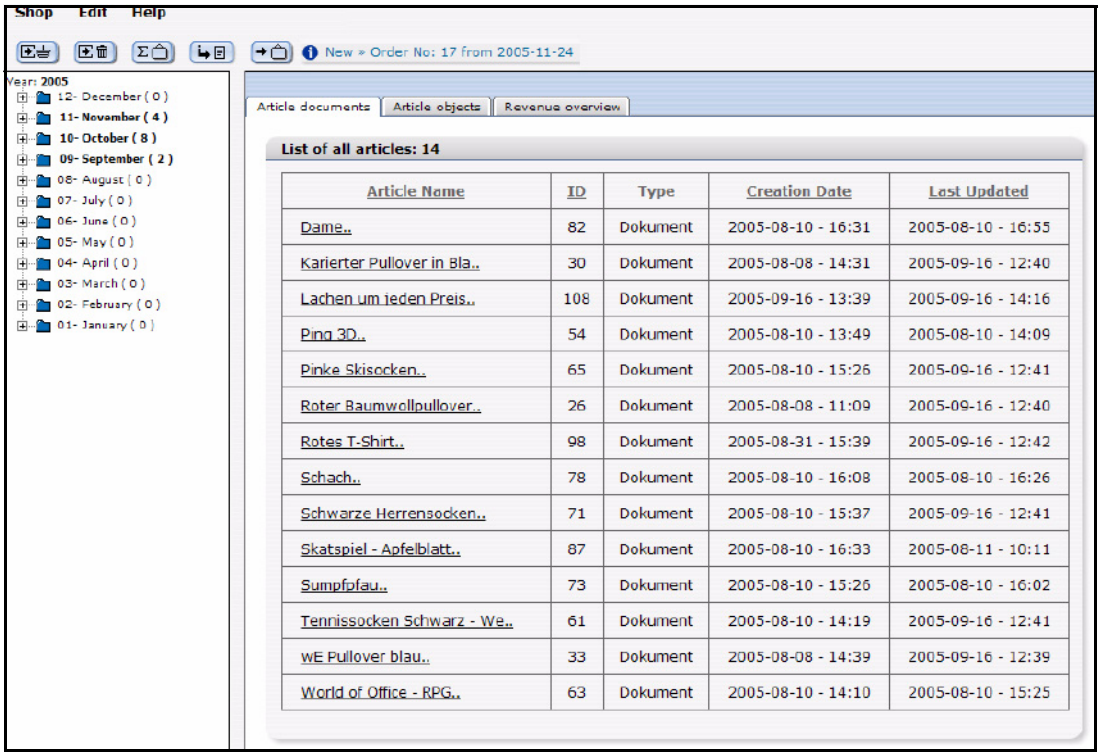
For more information about Payment Providers, please refer to 3, "Linking to a payment provider" on page 27.

1.3.1.3 Article/Revenue

Click the *Article/Revenue* button to open the view of that name.. In this view, the tabs *Article dokuments, Article objects* and *Revenue overview* are located(see Figure 5).

> ***Note:*** Notice please that the Article objects tab only exists if you have installed the DB/Object Module, article properties exist and the correct Class-ID is defined in the preferences! (cf also Section 1.3.1.1, "Preferences,")

Figure 5 Article/Revenue-view: Article documents

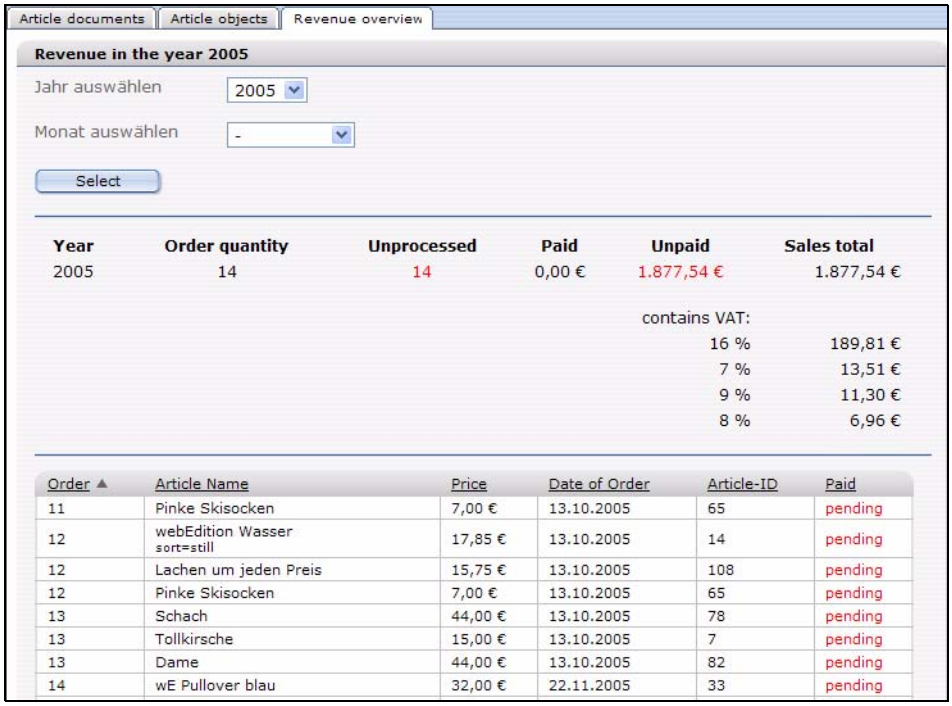


Article Name	ID	Type	Creation Date	Last Updated
Dame..	82	Dokument	2005-08-10 - 16:31	2005-08-10 - 16:55
Karierter Pullover in Bla..	30	Dokument	2005-08-08 - 14:31	2005-09-16 - 12:40
Lachen um jeden Preis..	108	Dokument	2005-09-16 - 13:39	2005-09-16 - 14:16
Ping 3D..	54	Dokument	2005-08-10 - 13:49	2005-08-10 - 14:09
Pinke Skisocken..	65	Dokument	2005-08-10 - 15:26	2005-09-16 - 12:41
Roter Baumwollpullover..	26	Dokument	2005-08-08 - 11:09	2005-09-16 - 12:40
Rotes T-Shirt..	98	Dokument	2005-08-31 - 15:39	2005-09-16 - 12:42
Schach..	78	Dokument	2005-08-10 - 16:08	2005-08-10 - 16:26
Schwarze Herrensocken..	71	Dokument	2005-08-10 - 15:37	2005-09-16 - 12:41
Skatspiel - Apfelblatt..	87	Dokument	2005-08-10 - 16:33	2005-08-11 - 10:11
Sumpfpfau..	73	Dokument	2005-08-10 - 15:26	2005-08-10 - 16:02
Tennissocken Schwarz - We..	61	Dokument	2005-08-10 - 14:19	2005-09-16 - 12:41
wE Pullover blau..	33	Dokument	2005-08-08 - 14:39	2005-09-16 - 12:39
World of Office - RPG..	63	Dokument	2005-08-10 - 14:10	2005-08-10 - 15:25

This view displays all articles (documents and objects). If you click on an article, the suitable article is opened in the webEdition main window and can be edited immediately. The listing itself is tabular with the most important info to the suitable articles including the info whether the respective article has variants. The list is sortable by click on the suitable heading. The representation corresponds in color usual webEdition rules, a not published article (document or property) is displayed, for instance, red.

Click on the card index rider Revenue overview to let indicate the orders of a certain calendar year. Also here the listings can be sorted by click on a heading. Also here there is a color convention- paid-up or worked on orders are displayed green.

Figure 6 *Artikel/Umsatz*-Ansicht: *Umsatzübersicht*



1.3.2 The Shop Module main screen

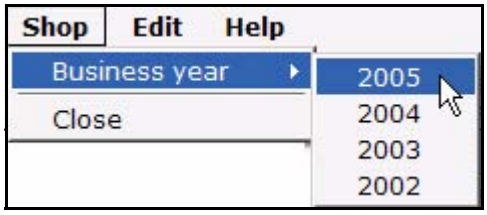
The Shop Module main screen has three drop-down menus: *Shop*, *Edit*, and *Help*.

1.3.2.1 The *Shop* menu

The *Shop* menu contains the following items (see Figure 7, "Shop drop-down menu" on page 6).

- *Business year*. Select the year you want to be diplayed.
- *Close*. Use this command to exit the module.

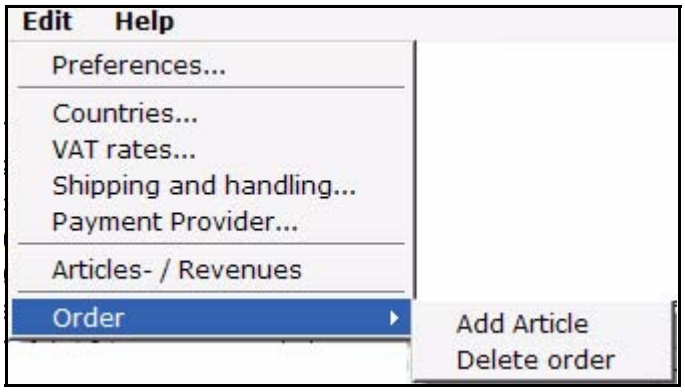
Figure 7 Shop drop-down menu



1.3.2.2 The *Edit* menu

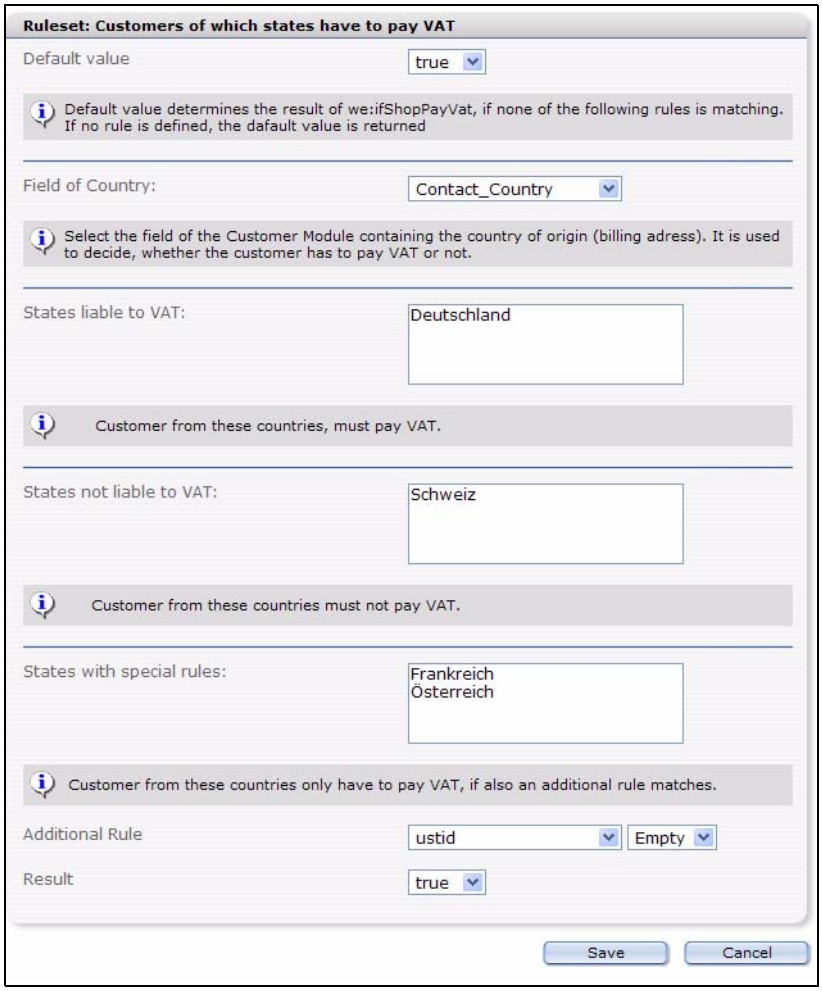
In the *Edit* menu, the following items can be slected (see Figure 8, "Edit drop-down menu" on page 7).

Figure 8 *Edit* drop-down menu



- *Preferences*: Opens the *preferences* window (see Section 1.3.1.1, "Preferences,")
- *Countries*: A window is opened in which you can determine the regulations specific for countries with regard to the value added tax clause.

Figure 9 Value added tax regulations specific for countries



— *Default value*: Standard return value for <we:ifShopPayVat> in case none of the opposed rules applies

— *Field of Country*: In this field the name of the field is stored in the customer management which contains the origin land (invoice address).

— *States liable to VAT*:Customers from these countries must pay VAT.

— *States not liable to VAT*:Customers from these countries must not pay VAT.

— *States with special rules*: Customer from these countries only have to pay VAT, if also an additional rule matches.

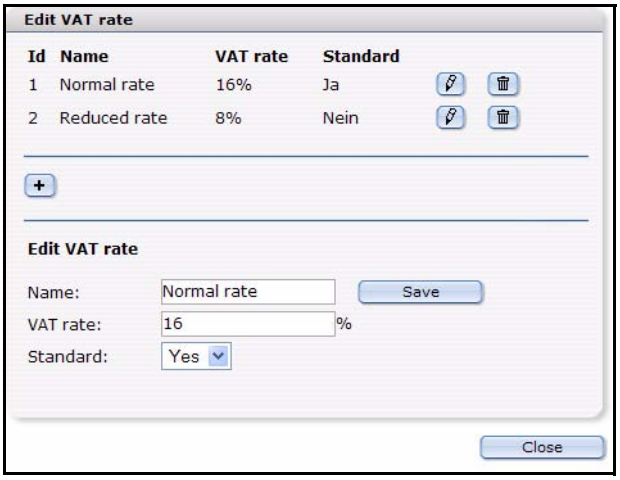
— *Additional rule*: The field in the customer management called (e.g.: ustid) corresponds to a condition (e.g., blank), then the value opposed here is returned (e.g.: A customer must pay VAT (true) if he has none ustid (ustid is empty) and must come from one of the defined countries).

Example

```
<we:ifShopPayVat>
  Customer has to pay VAT.
  (-> calculate VAT)
<we:else />
Customer has not to pay VAT.
</we:ifShopPayVat>
```

- *VAT rates*: The *Edit VAT rate* window opens (seeFigure 10)

Figure 10 Edit VAT rate

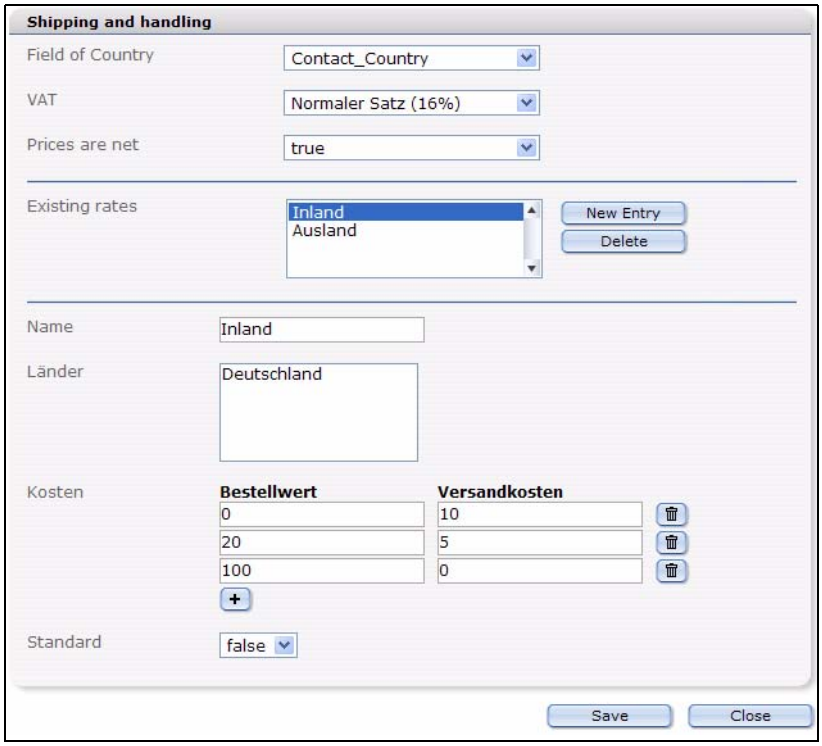


Here you can define different VATs. Select for a new tax rate a suitable name, enter the suitable percent number and determine whether it should concern the standard rate of taxation. Click then on *Save*

Should no value added tax clause be assigned to an article, the opposed standard clause is taken, if available. If still no values are entered here, the earlier entered VAT is used for compatibility reasons. Then the VATs entered here can be integrated by shop documents as well as by shop properties. Besides, during the complete purchasing process exclusively the id of the VAT is used. Only if the order is closed and is stored in the database, the VAT valid at the time of the order is saved. Therefore, an additional adaptation of the VAT does not alter finished orders

- *Shipping and handling*: The *Shipping and handling* window is opened

Figure 11 Shipping and Handling



— *Field of country*: In this field you select the field of the customer management in which the origin land of the customer is stored

— *VAT*: The value added tax clause which should apply to postal charges and packaging. This can be selected from the defined sets

— *Prices are net: If the prices are net or gross. This is important especially in the shop backend, all these specifications are stored with the order process and are taken into consideration within the order view. In the Template itself these values can be freely used*

— *Existing rates*: List of all defined rates

— *Name*: Internal used name. Is used for *Existing rates*

— *Countries*: List of countries to which this rule should apply.

— *Cost*: The dependence on order value and forwarding expenses can be entered in this list graded. In this example a purchasing to 20 euros of 10 euros of forwarding expenses, between 20 and 100 euros of order value 5 euros, and more than 100 euros costs nothing more.

— *Standard*: If no other rule of the forwarding expenses applies, the standard rule is used always.

- *Payment Provider*: Opens the *Pament Provider* window. See Section 1.3.1.2, "Payment Provider" on page 3

- *Articles-/Revenues*: Opens the corresponding window, see Section 1.3.1.3, "Article/Revenue" on page 4
- *Order*: Has the following sub-menu items:
  - *Add Article* Adds, from before defined articles, to the current order. You may perform this action also with the Quickbutton

Figure 12 *Add article* quickbutton

  - *Delete Order*: dletes the selected order.You may perform this action also with the Quickbutton

Figure 13 *Delete order* quickbutton

> ***Note:*** To be able to add an article to an order or to delete a complete order, you must at first have selected an existing order in the explorermenu.

1.3.2.3 The Help menu

The help menu has two options:

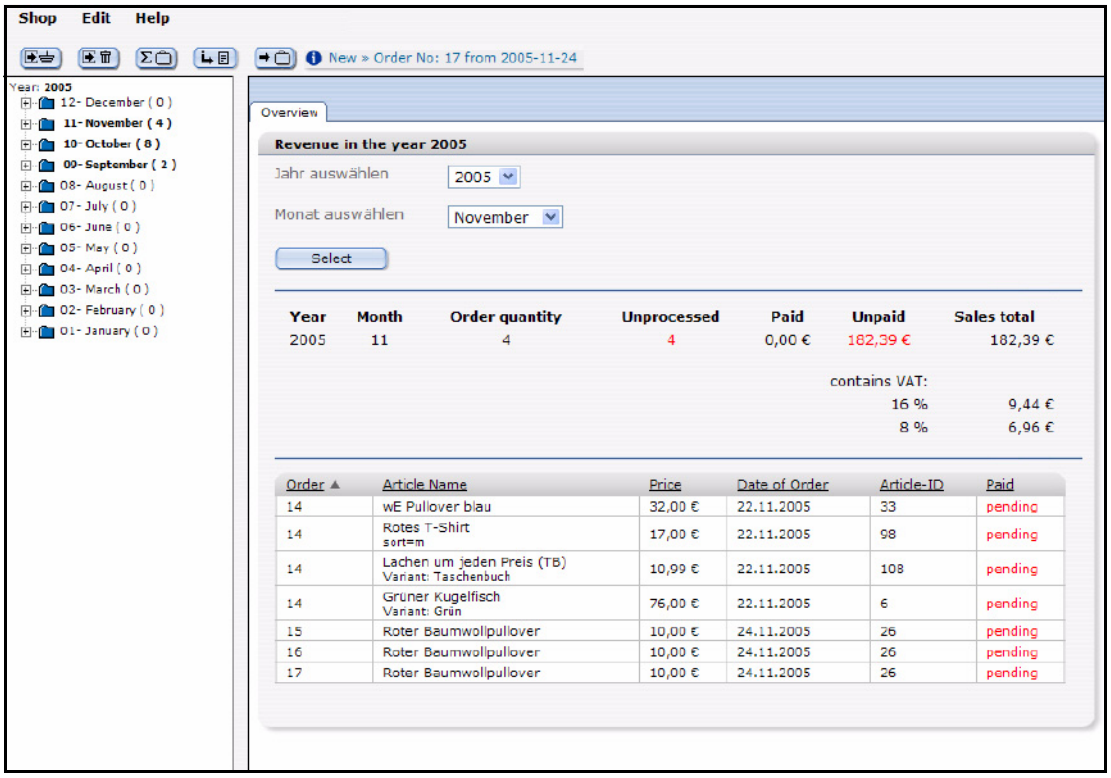
- *Info*... Select this option to access information about the license holder and the version of webEdition you are using.
- *Help*... Select this option to access the *webEdition Help System*.

1.3.3 The Shop Module main screen

The start screen (see Figure 14, "Shop Module main screen" on page 11), is divided into two parts:

- explorer menu, on the left side
- order processing screens, on the right side

Figure 14 Shop Module main screen



1.3.3.1 The explorer menu

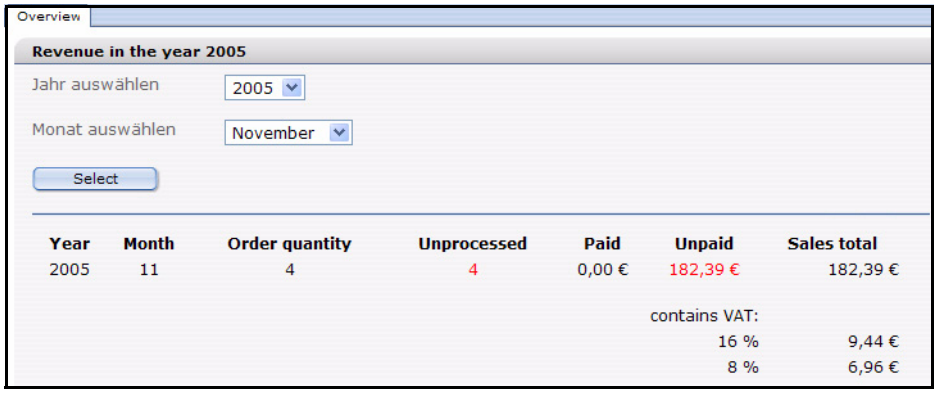
The explorer menu acts as a directory of all Shop Module orders in a business year. Each month is represented by a folder. By clicking on the "+" character to the left of the folder, you can open the directory for that month to view a list of all orders. The number in brackets adjacent to the month indicates the number of orders for that month.

1.3.3.2 The order processing screens

The order processing screen has two views. Monthly overview, and the order processing screen, each of which are accessed by clicking on items in the explorer menu.

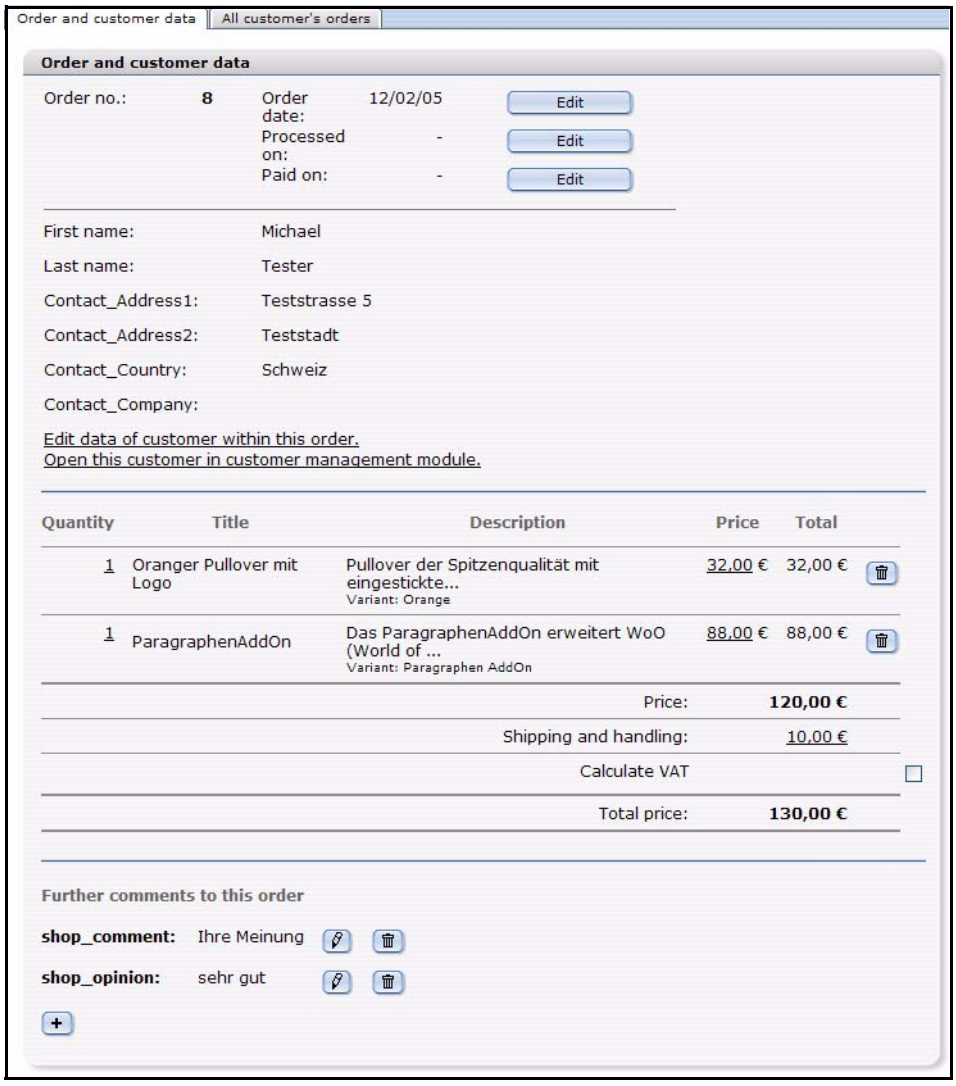
1.3.3.2.1 Monthly overview. If you click on a month folder itself, a summary appears in the right side of your screen , showing the number of processed and unprocessed Shop orders. Below this area is a statistical evaluation that reports the total sales, and the number paid/unpaid invoices for that month (see Figure 15).

Figure 15 Monthly overview



1.3.3.2.2 The *Order Data* view If you click on a specific Shop item listed in one of the monthly directories, webEdition will display the processing data for that order on the right side of the screen (see Figure 16).

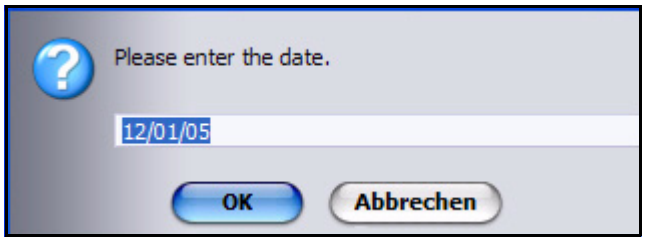
Figure 16 *Order Data* view



The *Order Data* view has two areas: *Order and user details* and *Items ordered*.

You can modify the order data by clicking on the *Edit* buttons. webEdition will open a dialogue box in which you can edit the corresponding values as seen in Figure 17.

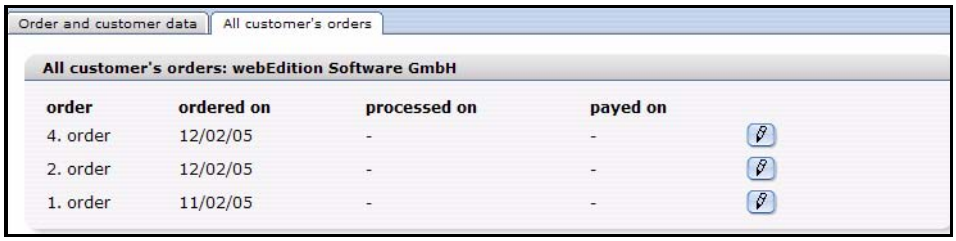
Figure 17 Editing an order



1.3.3.3 Viewing all of a customer's orders

You can view all orders placed by an individual customer by selecting the *All Customer's Orders* tab, which is adjacent to the *Order Data* tab (see Figure 18). Thereafter, if you click on the (underlined) item number or date for an order, webEdition will open the corresponding *Order Data* screen for that order.

Figure 18 Viewing all of a customer's orders



2 Configuration of the Shop Module

In this chapter you find out how you configure the shop module. The following items are discussed:

2.1 Variants

From webEdition version 3.5 it is possible to use variants of articles.

2.1.1 Variants of shop articles

An improvement desired for a long time for the shop module is the possibility of variants for shop articles. With the sales of similar articles it is very laborious to put on a completely new article every time, although merely the color differs. Now from webEdition 3.5 it is possible to create several variants for a shop article. These own all properties of the original article and "overwrite" merely the varying elements. Hence, an article variation is dependent always from the original article and can be called only in its context.

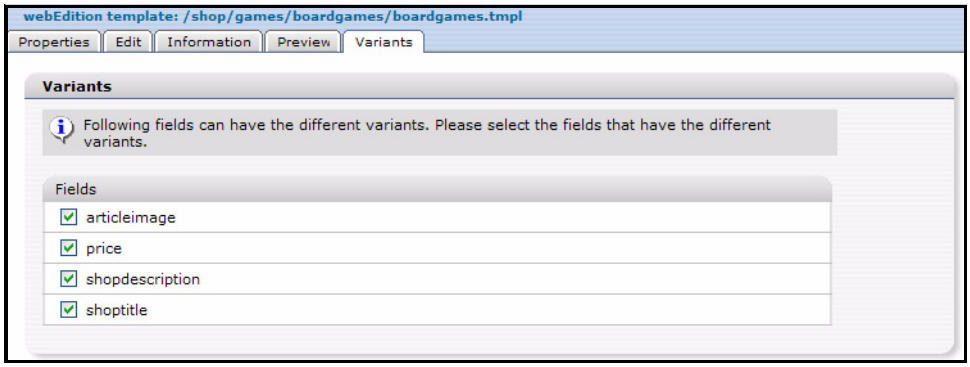
At the moment only variants of shop articles can be created, i.e. the template class must contain the fields specific for shops: *shopdescription*, *shoptitle* and *price*. Only if these fields can be found, it is possible to create variants.

Which articles and fields own variants is set directly in the class or template. One creates variants on a concrete article directly within aforesaid object or document.

2.1.2 Variants in article documents

To create variants for articles based on documents, fields for the variants have to be specified in the template. Variants can be created only if the template contains the shopspecific fields *shoptitle, shopdescription* and *price*. If this is the case, the template has the tab "Variants". In the Variants view of a template a list of all suitable fields of the template is displayed which can be marced with a checkbox as active.

Figure 19 Variant fields in the template



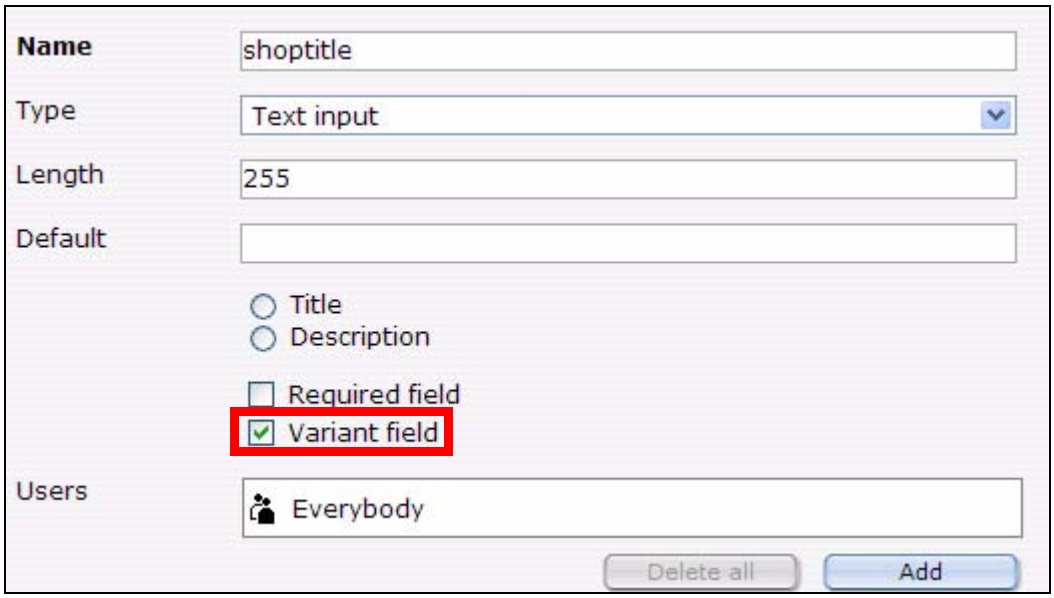
Note: Variants can only be created for dynamic documents!

If variant fields are defined in a template, all documents which are based on this template have the new tab *Variants*. Here the different variants can be created and maintained. The editing of the variants is analogue to the editiong of blocks. In addition to the fields defined in the template a field required for the internal use "Name" is created by which the variant can be called (by request `we_variant = <NAME>`)

2.1.3 Variants in article-objects

If objects with variants should be used, the corresponding filed must be declared at first in the class. Analogously to variants with documents, a class permits only variants if the fields *shoptitle*, *shopdescription* and *price* exist. Whether and which fields should be used for variants, then can be determined directly at the field administration. Also here only picture fields or text fields can be used for variants.

Figure 20 Variant fields for objects in classe



Object variations are analogously edited to document variations. Only the preview button is absent here, because it depends on the operating range which template should be used for the view. Object variations function only if in the template the we:tag `<we:useShopVariant/>` exists. Also here the real object is initialized and then, when

required (? we_variant = <NAME>), overwritten by the data of the variant. Likewise <we:useShopVariant/> must be integrated , before the first field is displayed.

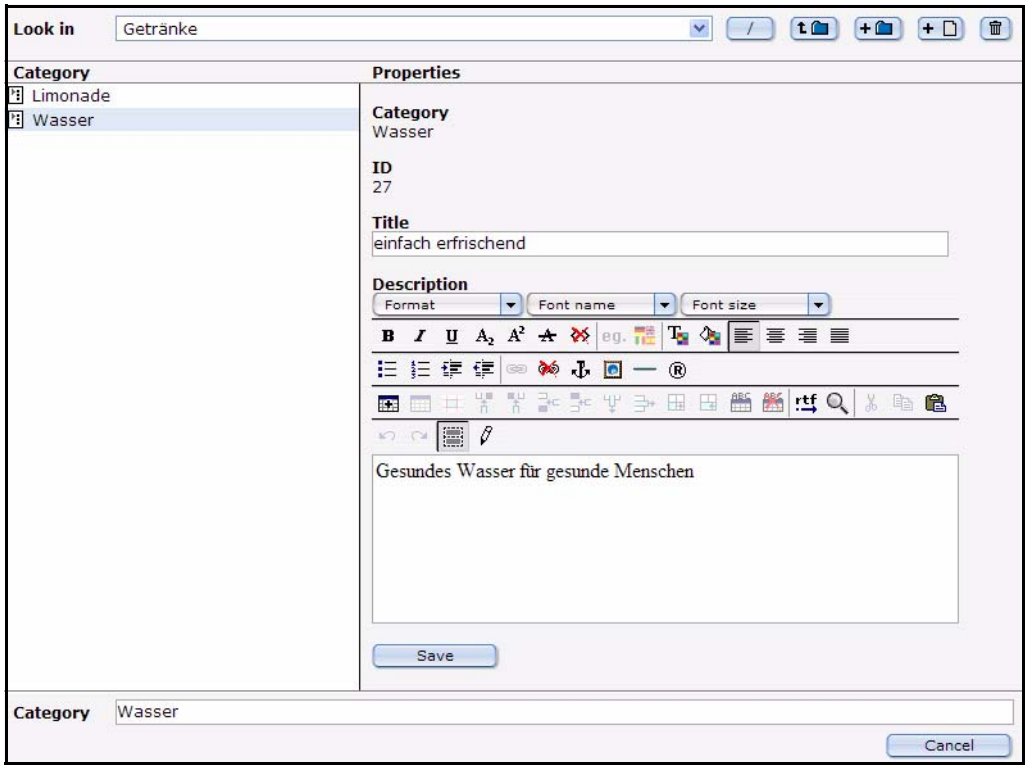
2.2 Product groups by categories

Online shops require gradation of the offered articles in product groups. In webEdition this can be realized with categories. For this pupose, the categories were extended with some functions.

2.2.1 Advanced category functions

Categories can be provided from webEdition 3.5 with a title and description text. Select for this the menu item *Options> categories* in the webEdition main window (see). If a category is selected, title and description can be edited.

Figure 21 Kategorie Ansicht



These fields can now be put out with a *<we:listview type="category">*. Thus farther operational areas offer. Now beside category names - titles, heading, Id and ParentId - can be displayed, too. With this, these informations can also be used with a listview of documents or objects.

2.2.2 *<we:listview type="category">*

The tag *<we:listview type="category">* has been added from version 3.5. With it, it is possible to display all entries of a category folder.

Example

`<we:listview type="category" parentid="5">` Listview for all categories which are in the category folder with id 5. Beyond this, it is possible to set the parentId with `$ _REQUEST ['we_parentid']`. You can determine the name of these request variables for the attribute parentidname by yourself. If none is

specified (neither as an attribute nor as request), a Listview about the Root directory of the categories is created.

With the attribute categoryids one or several categories can be displayed: `<we:listview type="category" categoryids ="7,9">`. The attribute *categoryids* may also be used in a standard *listview*. Then it serves as a substitute for the attribute *categories*. Should *categories* and *categoryids* be set, precedence has always *categoryids* !

Within the category Listview the single fields of the category can be displayed, as usual, with we:field. The following Keys (name of the fields) exist:

- *ID* or *WE_ID* = ID of the category
- *Category* = Name of the category
- *ParentID* = ID of the parent category
- *Title* or *WE_TITLE* = Title of the category
- *Description* or *WE_DESCRIPTION* = Descriptiontext of the category
- *Path* oder *WE_PATH* = pathe of the category

Here, too, the attribute *hyperlink="true"* may be used. Then if necessary a hyperlink is placed on the same document with the request variables *we_parentid* (or parentidname). Now within the *<we:repeat>* tag, with the tag*<we:ifHasChildren>*, can be requested, if the actual category-folder contains categories. From version 3.5 it is possible to create interlockíng listviews! Here an example for hierarchical listing of categories:

```
<we:listview rows="2" type="category" name="outer">
  <we:repeat>
    <b><we:field name="Category" hyperlink="true"></b><br>
    <we:field name="Title"><br>
    <we:field name="Description">
    <we:setVar to="global" nameto="cat" from="listview"
namefrom="Category" />
    <we:ifHasChildren>
      <i>The category has children</i><br>
    </we:ifHasChildren>
   <we:listview type="document" categories="\$cat" name="inner"
rows="2">
      <we:ifFound>
        <we:repeat>
          - <we:field name="WE_PATH" type="text"
hyperlink="true"><br />
        </we:repeat>
        - <we:back>back</we:back>
        - <we:next>next</we:next>
      </we:ifFound>
    </we:listview>
  </we:repeat>
  <we:back>back</we:back>
  <we:next>next</we:next>
</we:listview>
```

This makes it very easy to display multi-hierarchical product groups.

2.2.3 *<we:category>*

<we:category> could display up to now only information about the category(ies) of the current document. This was extended with the attributes *field* and *id*. The attribute *field*

permits the access to the new fields of the category, with the attribute *id* a certain category (independent of document) can be read out

Example

- `<we:category field="Title" />` displays the title of the category of the current document (or Listviewentry within Listviews)
- `<we:category field="Description" />` displays the description of the category of the current document (or Listview entry within Listviews)
- `<we:category id="7" />` displays the name of the category with id 7
- `<we:category id="7" field="Description" />` displays the description of the category with id 7

2.3 Freely determinable fields

Now in the shop from webEdition version 3.5 free input fields are also realizable for shop articles. Thereby it is possible without variants to let adjust special values of an article only by the customer (e.g., sizes or colors of articles). The shop operator can determine for these free fields then either values (-> make available Select menu) or a free input field in which the customer can enter a desired text himself.

Beside free fields in shop articles the free fields can be also added to an order which are then displayed in the webEdition shop module. Thus, for instance, comments can be saved directly with the order.

2.3.1 Freely determinable fields in shop articles

With the tag <*we:shopField*> input fields can be generated on a shop article page. If an article is put in the shopping cart, these fields are saved there. This is possible for document articles as well as for object articles. In addition, these free fields are saved in separate fields of the document (in the shopping cart) to be able to recognize them later again. Besides, <*we:shopField*> generates arbitrary input fields with a certain name. If the article (together with own fields) transmits then with a form to the shopping cart, these fields are added to the article. It is possible to define arbitrarily many free fields and to save them with the article. The destination page of this form must contain the tag <*we:createShop*>, so that the fields can be created.

<we:shopField> has the following parameters:

- *name*: Name of the field; with this name the field is saved in the article (shopping cart).
- *reference*: Determines, wether the free field belongs to an article or shopping cart.
- *shopname*: Name of the shop being used
- *type*: Set the entry field to be created
- *value*: preset value
- *values*: comma-separated list for type="choice" and "select"
- *checked*: Set the checkbox activated or not

- *mode*: Only for type=choice. Several values can be selected, these are attached comma-separated
- *xml*: Create input field xhtml-valid

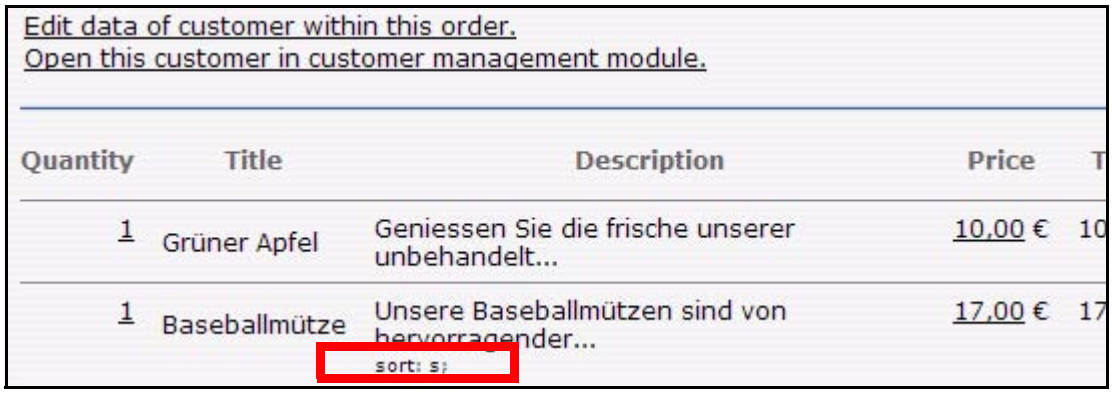
Depending on reference from we:shopField the names of the generated input fields vary. With reference = "article" an input field is generated which is added to an article, this has the name we_sacf [<name>].

These free fields are saved in every article of the shopping cart and can be displayed with we:field or we:shopfield ... type = "print" within we:listview for the shopping cart.

```
<table>
<we:repeatShopItem shopname="demoshop">
<tr>
  <td><we:field name="shoptitle" hyperlink="true"></td>
  <td><we:field name="sort"></td>
  ...
</tr>
</we:repeatShopItem>
</table>
```

In the order view in the shop module these free fields are visible under the description of the article.

Figure 22 Freely determinable field in article



2.3.2 Freely determinable fields in orders

The free fields in orders work analogously to free fields of articles . However, besides, is used *we:shopField reference = "cart"*. Also here many free fields can be added arbitrarily to an order. The free order fields are indicated in the order view of the shop, can be displayed during the order, however, also with *<we:shopField type = "print" ./>*. These fields are used for the input as follows

```
Comment:
<we:shopField reference="cart" name="shop_comment"
shopname="demoshop" type="textarea" />
```

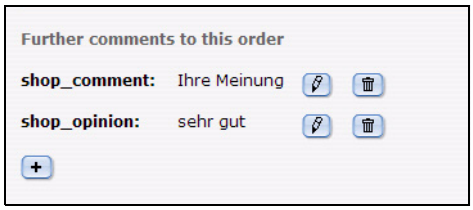
If you want to display this again to send it, for example, by mail or as a control for the user, following syntax can be used:

```
Komment:
<we:shopField reference="cart" name="shop_comment"
shopname="demoshop" type="print" />
```

only the field is returned.

Also here is to be noted that the page, from which the form is sent off absolutely must include *we:createShop*. Then the fields are listed with the accompanying order.

Figure 23 Free fields in order



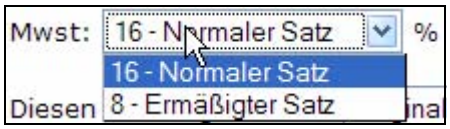
These fields can be deleted or edited after click on the suitable icon .

2.4 Value added tax

2.4.1 Value added tax set associate to a document

With <we:shopVat> a certain value added tax set can be assigned to an article. Besides, a list with the sets entered in the shop module from which one can select the desired one is generated. <we:shopVat type = "select"/>, for instance, generates a dropdown menu for the edit mode of the document with the pre-defined VATs

Figure 24 VATs in the dropdown menu (german example)



With <we:shopVat id = "1"/> one can display the value added tax set of the selected Id. This is sensible to be able to expel different VATs in the shopping cart decollated.

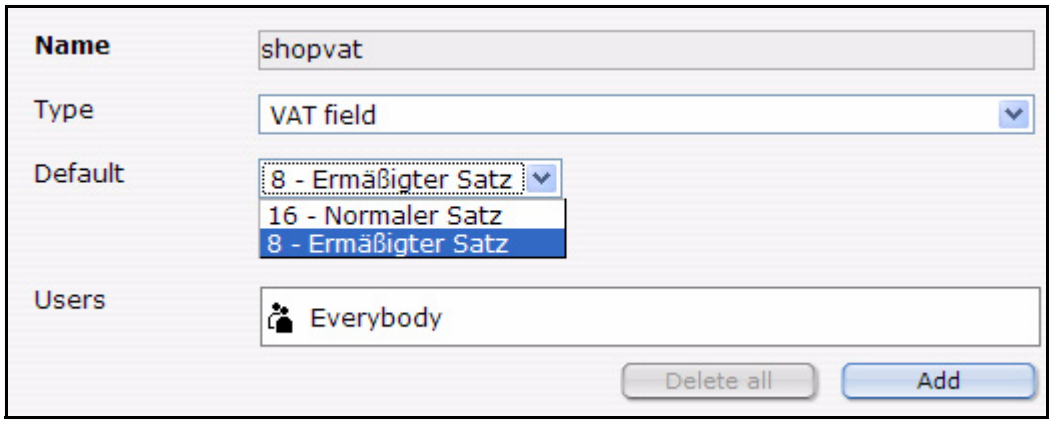
2.4.2 Value added tax sets for an object

If objects are used as shop articles, these can also use the value added tax sentences described on top. From version 3.5 the field type shopVat (value added tax field) has been added to classes. The name shopvat is a constant and cannot be changed. Objects of a class with this field can select the value added tax set then analogously to documents from within the shop module to built in sets. Then the output of the VAT set on the web site occurs like for other elements

```
<we:field type="shopVat" /> respectively <we:var type="shopVat" />
```

.

Figure 25 Taxes with *shopvat* in a class



2.4.3 Using VATs

With *<we:shopVat />* for documentc and *<we:field type="shopVat" />* or *<we:var type="shopVat" />* for objectes or Listviews (also shopping cart) the VATs are displayed on the website.

To allow a flawless work with value added tax sentences, these are called on the complete web site by their Id. This offers numerous possibilities to work with the different sets and enables you to change the sets at a later time without influencing the functionality of the underlying templates. The tag <we:ifShopVat id = "..."> controls whether the current document, or the current entry has the value added tax set with the specified Id in a Listview. Thus, different value added tax sentences can be decollated and expeled easily. A shopping cart could look as follows:

```
<we:repeatShopItem shopname="demoshop">
  <tr>
    <td><we:field name="shoptitle" hyperlink="true" /></td>
    <td class="shoppingCartNumber">
      <we:showShopItemNumber shopname="demoshop" />
    </td>
    <td class="shoppingCartNumber textBold">
      <we:calculate sum="totalPrice" num_format="german">
       <we:showShopItemNumber type="print" shopname="demoshop" /> *
        <we:field name="price" />
      </we:calculate> &euro;
    </td>
  </tr>
  <we:ifShopPayVat>
    <we:ifShopVat id="1">
      <we:calculate print="false" sum="totalVat1">
      <we:showShopItemNumber type="print" shopname="demoshop" /> *
        <we:field name="price" /> /
         100 *
        <we:field type="shopVat" />
      </we:calculate>
    </we:ifShopVat>
    <we:ifShopVat id="2">
      <we:calculate print="false" sum="totalVat2">
       <we:showShopItemNumber type="print" shopname="demoshop" /> *
        <we:field name="price" /> /
        100 *
        <we:field type="shopVat" />
      </we:calculate>
    </we:ifShopVat>
  </we:ifShopPayVat>
</we:repeatShopItem>
```

In the above example different sums of the complete value added tax are computed according to Id of the used value added tax set. Depending on which value added taxes exist the sums "totalVat1" and/or "totalVat2" are thereby computed.

```
<we:ifVarSet name="totalVat1" type="sum">
  <tr>
    <td class="shoppingCartNumber">zzgl. MwSt. <we:shopVat id="1"
/>%</td>
    <td class="shoppingCartNumber">
      <we:sum name="totalVat1" num_format="german" /> &euro;
    </td>
  </tr>
</we:ifVarSet>
<we:ifVarSet name="totalVat2" type="sum">
  <tr>
    <td class="shoppingCartNumber">zzgl. MwSt. <we:shopVat id="2"
/>%</td>
    <td class="shoppingCartNumber">
      <we:sum name="totalVat2" num_format="german" /> &euro;
    </td>
  </tr>
</we:ifVarSet>
```

Then these both sums must be added for the computed price and the shopping cart is finished.

```
<tr>
  <th class="shoppingCartNumber">Gesamtpreis</th>
  <td class="shoppingCartNumber textBold">
    <we:calculate num_format="german">
      <we:sum name="totalVat1" /> +
      <we:sum name="totalVat2" /> +
      <we:sum name="totalPrice" /></we:calculate> &euro;
  </td>
</tr>
```

Then the page generated looks as follows

Figure 26 Articles with VAT (german example)

Artikel	- Anzahl	+ x	Preis
Pinker Pullover (16% MwSt)	- 1	+ x	10,00 €
World of Office - RPG (16% MwSt)	- 1	+ x	129,00 €
	Preis		139,00 €
	Versandkosten		10,00 €
	zzgl. MwSt. 16%		23,84 €
	Gesamtpreis		172,84 €

2.4.4 VATs in the shopping cart

When an order is concluded, and all article data are saved in the database, also the used VATs are saved. Thus it is prevented that additional changes of the VAT alter already existing and concluded orders. In the backend of the shop module, the order can still be edited concerning the value added tax afterwards. An order with variants and user-defined fields can look as follows:

Figure 27 VATs in the order overview

Quantity	Title	Description	Price	Total	Sales tax
1	Grüner Pullover mit Logo	Pullover der Spitzenqualität mit eingestickte...	32,00 €	32,00 €	(16.00%)
1	Skatspiel - Klassik	Das altbekannte Skat-Spiel in verschiedenen A...	25,00 €	25,00 €	(16.00%)
1	Skatspiel Blau	Das altbekannte Skat-Spiel in verschiedenen A... Variant: Blau	27,00 €	27,00 €	(16.00%)
1	Mineralwasser	Tolles Wasser herrlich erfrischend. Je nach G... sort: still;	17,85 €	17,85 €	(8.00%)
1	Luxus Wasser	Exquisites Wasser. Dieses Wasser stellt bei w... sort: normal;	125,58 €	125,58 €	(16.00%)
		Price:		227,43 €	
		Shipping and handling:		0,00 €	
		plus VAT:			
		16 %:		33,53 €	
		8 %:		1,43 €	
		Total price:		262,39 €	

If you want to change afterwards the VAT of an article, merely click the order and enter the desired new value in the opening prompt. In addition, can be decided in the order view afterwards whether the value added tax should be computed. Depending on whether the forwarding expenses were entries net or gross, or whether the data saved in the goods basket were built in as net/gross prices the prices are anew computed in the order then

2.4.5 VAT: special we:tags

2.4.5.1 we:ifShopPayVat

With <*we:ifShopPayVat*> the surrounded contents are only displayed if the logged in customer must also pay value added taxes. Whether a customer must pay taxes, depends primarily on the country to which the product must be delivered. In addition country rules can be defined in the Shop Module under *edit* > *countries*.

2.4.5.2 we:ifShopVat

With <*we:ifShopVat*> can be controlled whether an article has a certain VAT. This tag is used in shops with several VAT sets.

2.4.5.3 we:shopVat

With <*we:shopVat*> a value added tax set can be assigned to a shop article from the defined VATs. If the attribute id is assigned, merely the value added tax set with the given Id is returned.

2.4.5.4 we:field type="shopVat"

Delivers within a *we:listview* shopping cart, *we:object* the value added tax set of the current article of the entry.

2.4.5.5 we:var type="shopVat"

Analogously to *we:field type = "shopVat"* the accompanying value added tax set is returned here on a detailed page for an object

2.4.5.6 we:writeShopData netprices="true|false" usevat="true”

With *<we:writeShopData>* can be indicated whether the passed prices are Net or Gross. And whether the different value added taxes should be taken into consideration, or not. Default value of both entries is true, i.e. by Default prices are accepted Net and the value added tax with is taken into consideration..

2.4.5.7 we:ifVarSet type="sum"

Controls whether the variable of the type sum (originates by *<we:calculate sum = "...">*) is set

2.4.5.8 we:ifNotVarSet type="sum"

Analogously to we:ifVarSet type="sum"

2.5 Shipping

2.5.1 Shipping on the webseite

The defined postage and forwarding expenses can also be accessed by we:tags, according to order value and origin land of the logged in customer the suitable set is determined and returned. However, besides, only the entered costs are returned, then these can be used as desired on the page. These expenses are defined as shown in Section 1.3.2.2.

The value added tax consists of the VAT of the price and the forwarding expenses. This was computed in the template with *<we:calculate>*. The different forwarding expenses can be acessed by means of the *<we:shipping>* tag. Following example shows the application of *<we:shipping>*.

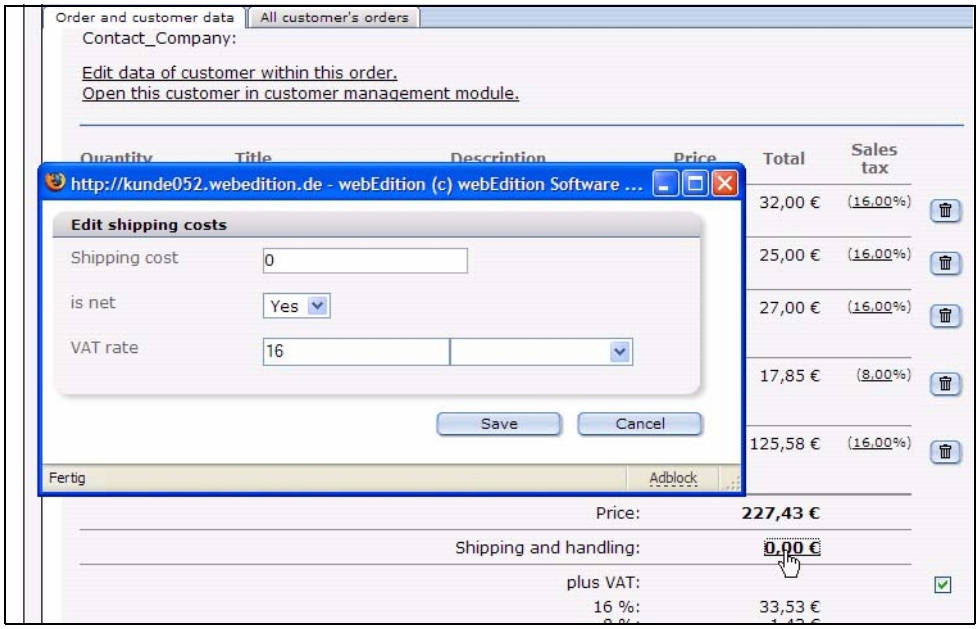
```
<tr>
  <th>Preis</th>
  <td><we:sum name="totalPrice" num_format="german" /> €</td>
</tr>
<tr>
  <th>Versandkosten</th>
  <td><we:shipping sum="totalPrice" num_format="german" /> €</td>
    <we:calculate print="false" sum="totalVat1">
      <we:shipping sum="totalPrice" />/100*<we:shopVat id="1" />
    </we:calculate>
</tr>
<tr>
  <th>zzgl. MwSt.</th>
  <td><we:sum name="totalVat1" num_format="german" /> €</td>
</tr>
<tr>
  <th>Gesamtpreis</th>
  <td>
    <we:calculate num_format="german">
      <we:shipping sum="totalPrice" num_format="german" />
      +  <we:sum name="totalVat1" />
      +  <we:sum name="totalPrice" />
    </we:calculate> €
  </td>
</tr>
```

In this example the forwarding expenses are displayed and added to the total expenses. In addition, the value added tax of the forwarding expenses is calculated, added up with the other value added tax shares and likewise opened on the whole price.

2.5.2 Forwarding expenses with orders

With the order process the valid forwarding expenses and all accompanying information are determined and saved with the order. Then within the Shop Module the due forwarding expenses are indicated and added for the whole price. In addition, these postage costs can be edited afterwards, in addition one simply clicks the postage costs and then can edit the postage relevant data for this order.

Figure 28 Change shipping cost in order



Note: Tag from Version 3.5: <*we:shipping*>: With we:shipping tag the forwarding expenses entered in the Shop Module can be accessed. With the help of the order value and the origin land of the logged in customer the suitable forwarding expenses are determined and returned. Besides, the imperative parameter sum = "" passes the name we:sum which contains the goods value of the order. The result from we:shipping can be processed in the Template and be added, e.g., with we:calculate tag to the shopping cart.

3 Linking to a payment provider

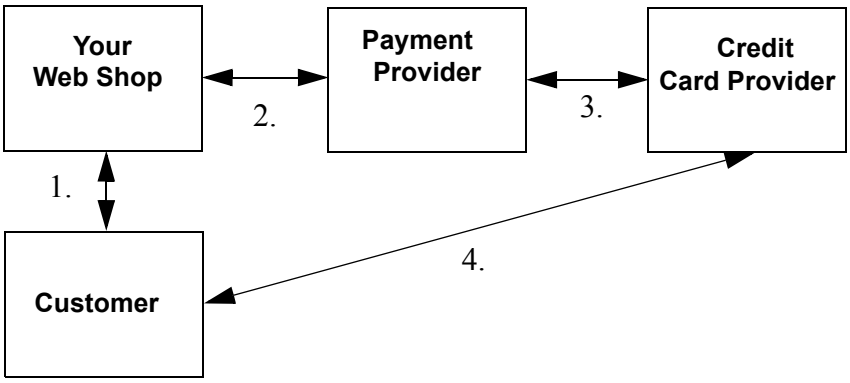
The following chapter discussed some guidlines that you can use for linking your Web site to a payment provider.

> ***Note:*** Every payment provider has its own interface and requirements for the transfer of payment data. For this reason, webEdition does not provide an interface for payment providers.

3.1 Payment provider transaction model

Figure 29 shows a typical transaction arrangement.

Figure 29 On-line shop transactions



By way of example, we have described two possible ways to link your shop to a payment provider.

- transfer the total price
- transmit every item individually

3.2 Integrating PayPal

The settings dialogue for PayPal can be accessed with *Edit> Payment Provider*, the Quckickstart button of the same name or the Payment provider button. The Payment provider window is opened.

Figure 30 Payment Provider settings: PayPal

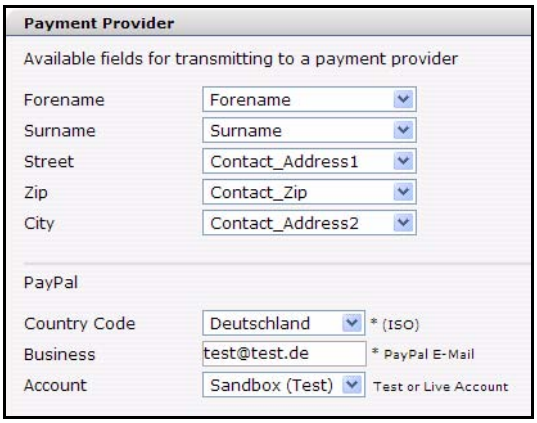


Figure 31 Form for personal entries

To be able to form comfortably a payment winding up with PayPal, some personal data are required . Address entries also belong to it. Those data which are available in this dialog about the select boxes (These data are administered in the customer management) and are selected, must be taken into consideration with the creation of a shop form for the purpose of the registration with personal data. If you select, e.g., for the transmission of the given name the option Forename in the selectbox, this must be also taken into consideration in the source program of the form in the shop with the field according to given name.

For the form, the following XHTML code was used:

```
<we:form id="self" pass_id="customerData" name="userform">
<we:sessionField name="ID" type="hidden" />
<we:sessionField name="UserGroup" type="hidden"
value="webCustomer" />
<we:sessionField name="Username" type="hidden" autofill="true" />
<we:sessionField name="Password" type="hidden" autofill="true" />
   <fieldset>
      <legend> Billing address</legend>
       <p>
        <label for="s[Salutation_Salutation]"> Salutation: </label>
     <we:sessionField name="Salutation_Salutation" type="textinput"
class="select" choice="on" options="Mr,Mrs"
id="s[Salutation_Salutation]" />
   </p>
   <p>
   <label for="s[Forename]"> Forename*: </label>
  <we:sessionField name="Forename" type="textinput" class="inputs"
id="s[Forename]" />
   </p>
   <p>
   <label for="s[Surname]"> Surname*: </label>
```

```
  <we:sessionField name="Surname" type="textinput" class="inputs"
id="s[Surname]" />
  </p>
  <p>
   <label for="s[Contact_Company]"> Company: </label>
  <we:sessionField name="Contact_Company" type="textinput"
class="inputs" id="s[Contact_Company]" />
  </p>
  <p>
   <label for="s[ustid]"> UST.-ID: </label>
  <we:sessionField name="ustid" type="textinput" class="inputs"
id="s[ustid]" />
  </p>
  <p>
  <label for="s[Contact_Address1]"> Street*: </label>
 <we:sessionField name="Contact_Address1" type="textinput"
class="inputs" id="s[Contact_Address1]" />
  </p>
  <p>
  <label for="s[Contact_Zip]"> Zip-Code*: </label>
 <we:sessionField name="Contact_Zip" type="textinput" class="zip"
id="s[Contact_Zip]" />
  </p>
  <p>
  <label for="s[Contact_Address2]"> City*: </label>
 <we:sessionField name="Contact_Address2" type="textinput"
class="inputs" id="s[Contact_Address2]" />
  </p>
  <p>
  <label for="s[Contact_Country]"> Country: </label>
 <we:sessionField name="Contact_Country" type="select"
class="select"
values="Deutschland,Schweiz,Österreich,Frankreich,Anderes"
id="s[Contact_Country]" />
  </p>
  <p>
  <label for="s[Contact_Email]"> Email: </label>
 <we:sessionField name="Contact_Email" type="textinput"
class="inputs" id="s[Contact_Email]" />
  </p>
  <p>
  <label for="s[Contact_Tel1]"> Phone: </label>
 <we:sessionField name="Contact_Tel1" type="textinput"
class="inputs" id="s[Contact_Tel1]" />
  </p>
  <p>
  <label for="s[Attention]">Attention by?:</label>
 <we:sessionField name="Attention" type="textinput" choice="true"
options="Bitte auswählen,Zeitschrift,Empfehlung eines
Bekannten,Empfehlung auf einer Website,Sonstiges" id="s[Attention]"
/>
  </p>
  <p>
  <label for="s[Exacting]"> How exactly?: </label>
  <we:sessionField name="Exacting" type="textinput" class="inputs"
id="s[Exacting]" />
   </p>
   <p>
   <label> </label>
  <input type="submit" class="inputButton" name="order"
value="Weiter" />
   </p>
   <p>
   (*) Mandatory fields
   </p>
  </fieldset>
</we:form>
```

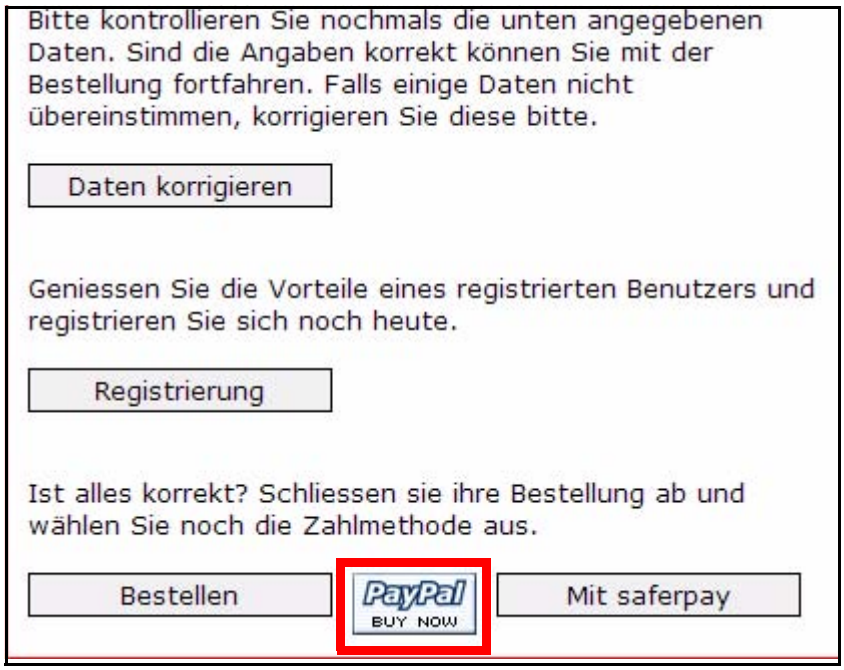
A correct entry leads to the fact that everything, also personal data, are validated and transmitted to PayPal.

PayPal needs exactly 2 entries for the correct payment winding up. This are the PayPal business address (email) as well as the ISO-country code for the correct winding up of the payment with regard to accordingly correct currency and tax. By means of a selectbox the suitable ISO-Country code is selected and the business address is given what is actually the PayPal email address. Optional you may choose in another Selectbox between the PayPal shop (live shop) and the PayPal sand box (PayPal integrated developing environment for testing). Further information about the sand box can be found in the PayPal documentation.

3.3 Using PayPal on the Website

For the data transmission to PayPal the PayPal tag can be used, if the neccessary data has been enetered in the backend.First of all there should be, e.g., a PayPal Buy Now button which links on a special PayPal page. This should be placed within the data-transmitting form.

Figure 32 PayPal Buy Now Button (german example)



the corresponding XHTML source code:

```
<we:a id="113"><img
src="http://images.paypal.com/images/x-click-but01.gif" border="0"
width="62" height="31" alt="Make Payments with PayPal - it's fast,
free and secure!" title="Make Payments with PayPal - it's fast, free
and secure!"></we:a>
```

In this example the id 113 refers to the document paypal.php.

In this file merely the paypal tag will be interpreted called with the names for the shop as well as price and whether the price as Net and taxshould be used. More information about this tag can be found in the description of the *<we:paypal>* tag.

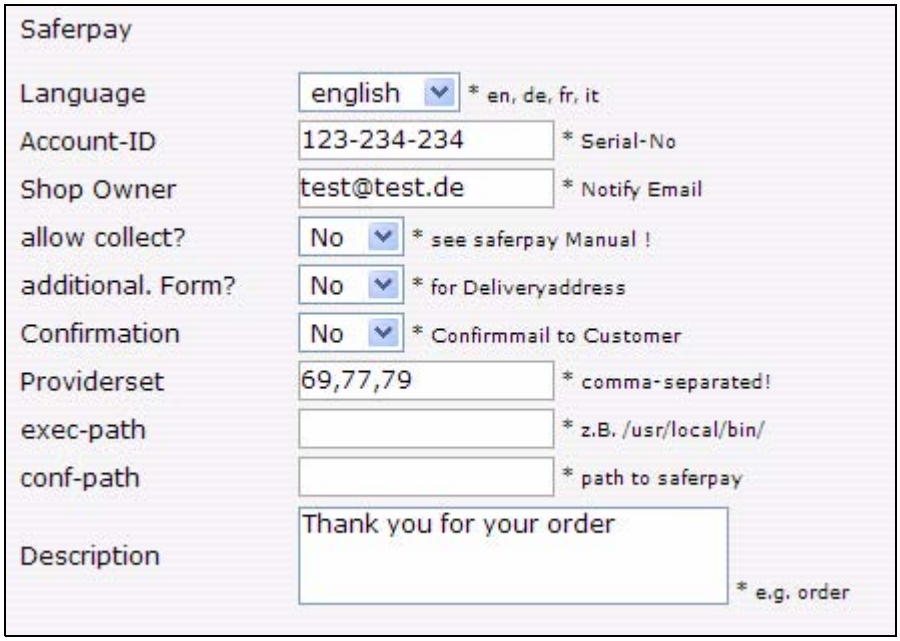
The source code of the file paypal.php resp. paypal.tmpl:

```
<we:sessionStart /><we:createShop shopname="demoshop"
/><we:addDelShopItem shopname="demoshop" /><!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <we:title></we:title>
  <we:description/>
  <we:keywords/>
  <we:charset defined="ISO-8859-1">ISO-8859-1</we:charset>
  <we:css id="10" />
</head>
<body>
<div class="content">
  <div class="header">
    <we:include id="46" />
  </div>
  <div class="leftNavigation">
    <we:include id="11" />
    <we:include id="3" />
  </div>
  <div class="mainContent">

    <div class="fullWidth">
      <div class="contentFirstDiv">
        <h1>Pay Pal</h1>
        <h2>powered by webEdition</h2>
        <div class="divWithPadding">
         Der Text sollte noch geschrieben werden.
        </div>
      </div>
    </div>

    <div class="fullWidth">
      <div class="border">
        <div class="divWithPadding">
 <we:paypal shopname="demoshop" pricename="price" netprices="true"
usevat="true" />
        </div>
      </div>
    </div>

  </div>
</div>
<we:ifVar name="action" match="success" type="request">
<we:deleteShop shopname="demoshop" />
</we:ifVar>
</body>
</html>
```

3.4 Integrating Saferpay

The settings dialogue for Saferpay can be accessed with *Edit> Payment Provider*, the Quckickstart button of the same name or the Payment provider button. The Payment provider window is opened.

Figure 33 Payment Provider settings: Saferpay



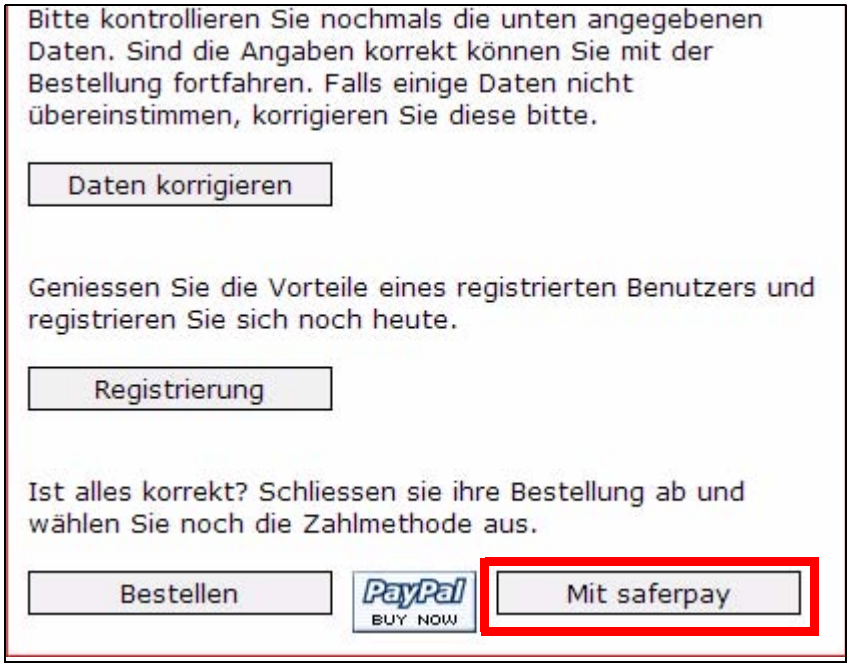
- *Language*: Saferpay accepts the language variations specified here. With this setting you can influence the appearance of the Saferpay terminal.
- *Account-ID*: In this field must be deposited the supplied Account-ID by saferpay.
- *Shop Owner*: The email address deposited in this field is used by Saferpay as a Notify email. Message is dispatched to this email in case of an order.
- *allow collect?*: Here can be selected whether Saferpay should permit the collecting of several orders before the realization of the order process with the total sum. Standard is a no.
- *additional. Form?*: Here can be selected whether Saferpay should generate an additional form to the registration of an address of delivery during the realization of the order process in the terminal. Standard is a no
- *Providerset*: A comma-separated list, consisting of Provider-IDs to limit the Card types to be used. To find out the suitable Provider-ID, please rightclick in the Saferpay select terminal (VT) on the suitable Card logo and select "Properties". Precise information can be gathered from the saferpay documentation.
- *exec-path*: Path to executable command line of saferpay
- *conf-path*: Path to the configuration files of saferpay, e.g., the directory "saferpay" in the root directory of the web server.
- *Description*: A description of the offer which should appear in the Saferpay terminal (VT).

Further informations about Saferpay and its configuration can be found in the Saferpay documentation.

3.5 Using Saferpay on your website

First of all, create a Saferpay button which links on a special Saferpay page. This should be placed within the data-transmitting form.

Figure 34 Saferpay Button in Form (german example)



The code:

```
<we:a class="inputButton" id="123"> With saferpay </we:a>
```

In this example the id 123 refers to the document saferpay.php. In this file merely the saferpay tag will be interpreted after the implementing of the *OpenSaferpayScript* file called with the names for the shop as well as price and whether the price as Net and tax should be used. In addition, in the tag the IDs of the sequence pages (in each case for onsuccess, onfailure or onabortion) can be defined. Further} information on the *<we:saferpay>* tag can be found in the tag-reference.

The code for saferpay.php resp. saferpay.tmpl:

```
<we:sessionStart /><we:createShop shopname="demoshop"
/><we:addDelShopItem shopname="demoshop" /><!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <we:title></we:title>
  <we:description/>
  <we:keywords/>
  <we:charset defined="ISO-8859-1">ISO-8859-1</we:charset>
  <we:css id="10" />
</head>
<body>
<div class="content">
  <div class="header">
    <we:include id="46" />
  </div>
  <div class="leftNavigation">
    <we:include id="11" />
    <we:include id="3" />
  </div>
  <div class="mainContent">

    <div class="fullWidth">
      <div class="contentFirstDiv">
        <h1>saferpay</h1>
```

```
            <h2>powered by webEdition</h2>
            <div class="divWithPadding">
             Der Text sollte noch geschrieben werden.
            </div>
          </div>
        </div>

        <div class="fullWidth">
          <div class="border">
            <div class="divWithPadding">
<script
src="http://www.saferpay.com/OpenSaferpayScript.js"></script>
<we:saferpay shopname="demoshop" pricename="price"
netprices="true" usevat="true" onsuccess="125" onfailure="126"
onabortion="127" />
            </div>
          </div>
        </div>

      </div>
    </div>
    <we:ifVar name="action" match="success" type="request">
    <we:deleteShop shopname="demoshop" />
    </we:ifVar>
    </body>
    </html>
```

3.6 New or extended tags in version 3.5

For Version 3.5 of webEdition, new tags have been added as well as existing tags extended to better support the imporived shop functions.

3.6.1 we:paypal

With <we:paypal> module built in PayPal variables from the shop can be accessed, like business address (shop email), country password (Country code ISO) as well as on the setting whether a test account or live shop should be used. In addition, the settings used in a form and in PayPal for transmitting personal data are set. The mandatory parameter *shopname = ""* passes the name of the current shop, *pricename = ""* passes the name of the current prize field. *netprice = ""* determines whether the prices are Net or Gross (TRUE or FALSE) and *usevat = ""* investigates whether a value added tax should be used, or not (TRUE or FALSE).

3.6.2 we:saferpay

With <we:saferpay> you can access the Saferpay variables eneterd in the shop module (see Section 3.4). The settings used in a form and for transmitting personal data to saferpay are also set. The mandatory parameter shopname = "" passes the name of the current shop, pricename = "" passes the name of the current prize field. netprice = "" determines whether the prices are Net or Gross (TRUE or FALSE) and usevat = "" investigates whether a value added tax should be used, or not (TRUE or FALSE). onsuccess = "" links on the ID of the web edition-page in case of a successful order. onfailure = "" links on the ID of the web edition-page in case of a not successful order if an error should have appeared. onabortion = "" links on the ID of the web edition-page in case of an abnormal termination of the order.

3.6.3 we:sessionField /addition: autofill=true

This tag was extended with the attribute "autofill". This attribute will come hany especially while implementing a form. If a <we:sessionField> is provides with the addition autofill=true, a Script is called in the background which autofills a variable

with a value. When using this addition, e.g., within an input field to the entry of an username and/or password, webEdition automatically generates a value and needs and e.g., the user no longer has to log in to be able to order in the shop. The XHTML code for such a form could look like this:

```
<we:form id="self" pass_id="customerData" name="userform">
<we:sessionField name="ID" type="hidden" />
<we:sessionField name="UserGroup" type="hidden"
value="webCustomer" />
<we:sessionField name="Username" type="hidden" autofill="true" />
<we:sessionField name="Password" type="hidden" autofill="true" />
   <fieldset>
      <legend> Rechnungsadresse </legend>
       <p>
        <label for="s[Salutation_Salutation]"> Anrede: </label>
     <we:sessionField name="Salutation_Salutation" type="textinput"
class="select" choice="on" options="Herr,Frau"
id="s[Salutation_Salutation]" />
   </p>
   <p>
   <label for="s[Forename]"> Vorname*: </label>
  <we:sessionField name="Forename" type="textinput" class="inputs"
id="s[Forename]" />
   </p>
   <p>
   <label for="s[Surname]"> Nachname*: </label>
  <we:sessionField name="Surname" type="textinput" class="inputs"
id="s[Surname]" />
  </p>
  <p>
   <label for="s[Contact_Company]"> Firma: </label>
  <we:sessionField name="Contact_Company" type="textinput"
class="inputs" id="s[Contact_Company]" />
  </p>
  <p>
   <label for="s[ustid]"> UST.-ID: </label>
  <we:sessionField name="ustid" type="textinput" class="inputs"
id="s[ustid]" />
  </p>
  <p>
  <label for="s[Contact_Address1]"> Strasse*: </label>
 <we:sessionField name="Contact_Address1" type="textinput"
class="inputs" id="s[Contact_Address1]" />
  </p>
  <p>
  <label for="s[Contact_Zip]"> PLZ*: </label>
 <we:sessionField name="Contact_Zip" type="textinput" class="zip"
id="s[Contact_Zip]" />
  </p>
  <p>
  <label for="s[Contact_Address2]"> Ort*: </label>
 <we:sessionField name="Contact_Address2" type="textinput"
class="inputs" id="s[Contact_Address2]" />
  </p>
  <p>
  <label for="s[Contact_Country]"> Land: </label>
 <we:sessionField name="Contact_Country" type="select"
class="select"
values="Deutschland,Schweiz,Österreich,Frankreich,Anderes"
id="s[Contact_Country]" />
  </p>
  <p>
  <label for="s[Contact_Email]"> Email: </label>
 <we:sessionField name="Contact_Email" type="textinput"
class="inputs" id="s[Contact_Email]" />
  </p>
```

```
  <p>
  <label for="s[Contact_Tel1]"> Telefon: </label>
 <we:sessionField name="Contact_Tel1" type="textinput"
class="inputs" id="s[Contact_Tel1]" />
  </p>
  <p>
  <label for="s[Attention]">Aufmerksam durch?:</label>
 <we:sessionField name="Attention" type="textinput" choice="true"
options="Bitte auswählen,Zeitschrift,Empfehlung eines
Bekannten,Empfehlung auf einer Website,Sonstiges" id="s[Attention]"
/>
  </p>
  <p>
  <label for="s[Exacting]"> Wodurch genau?: </label>
  <we:sessionField name="Exacting" type="textinput" class="inputs"
id="s[Exacting]" />
   </p>
   <p>
   <label> </label>
  <input type="submit" class="inputButton" name="order"
value="Weiter" />
   </p>
   <p>
   (*) Pflichtfelder
   </p>
  </fieldset>
</we:form>
```

4 Designing a template for the Shop Module

This chapter shows you how to use we:tags to create templates for your on-line shop. This following functions are described in this chapter:

4.1 Creating a detailed view of an item

The process of creating a template for an item summary is performed in a similar fashion to other webEdition functions (see "Creating Templates" in the *webEdition User Manual*). You can use any design for the detail page of an item. The only exceptions are the following we:tags, which must be on a Shop Detail page as there are fields in the Shop Module that correspond to these fields:

- A <we:input> or <we:textarea> with the name "shoptitle"
- A <we:input> or <we:textarea> with the name "shopdescription"
- A <we:input> with the price of the product. The name of this entry field is up to you, but it is very important because this field name must be provided later on as a price variable together with the <we:writeshopdata> tag.

  Note 1: **Important!** The article category has to be named with a starting *shop.* The category can also be named like *shop_1, shop_new, shop_online* or the likes. Else, no articles can be installed in the shop.

  Note 2: The price variable has no fixed name, since you can also work with various prices. The price written to the database is the one provided in the *pricename* variable in the <we:writeShopData> tag.

Note: Please also note that all detailed views of items must be assigned as a uniform document type and/or a uniform category, since item summaries are generated with

webEdition's listview function. Ideally you will have a uniform document type (e.g. shop_article or the like) for all detailed views in your Shop, and will distinguish between the item groups according to various categories (shop_article_dvd, shop_article_cd, etc.), so that they generate both a general summary of all items (only via document types) and summaries of all item groups (document types and categories).

4.2 Creating item summaries

The summaries of items or item groups are generated as normal listviews. In the following code example, assume that you have named a document type "shop", and that there are two item groups, "shop_dvd" and "shop_cd":

"shop_dvd" sample item group summary:

```
<we:listview rows="6" doctype="shop" categories="shop_dvd">
  <table border="0" cellpadding="0" cellspacing="0" width="500">
  <we:repeat>
  <tr>
    <td class="normal">
      <b><we:field name="shoptitle" alt="we_path"
hyperlink="on"/></b><br>
     <we:field name="shopdescription" alt="we_text" max="200"/><br>
      Price: <we:field name="prize"> Euro <we:a
id="id_des_shopping_cartes"
shop="on">[In den Shopping_cart]</we:a>
    </td>
  </tr>
  <tr>
    <td><img src="/root/layout_images/pixels.gif" width="500"
height="6"
border="0"></td>
  </tr>
  </we:repeat>
  </table>
</we:listview>
```

All items:

```
<we:listview rows="6" doctype="shop">
```

4.3 Ordering items

In order to allow an on-line shopper to put an item into the shopping cart, the following function must be available in the detailed item view or the summary:

```
<we:a id="id_of_the_following_page" shop="on">[Order]</we:a>
```

The shop attribute was added to the <we:a> tag. This function transfers the item information into the shopping cart. This is only possible if the following we:tags are on the page specified in the id attribute of the <we:a> tag:

```
<we:createShop shopname="shopname"/><we:addShopitem
shopname="shopname"/>
```

Only when this function is called up will the item be added to the order or put into the shopping cart. It is essential that you ensure that every order request with the <we:a> tag and the shop="on" attribute is followed by a page with <we:createShop/> and <we:addShopItem/> tags. The next page can be any page (even the calling page itself). Ideally it is the shopping cart which immediately displays the ordered item. If it is not the shopping cart, the ordered item is saved and displayed the next time the shopping cart is called.

4.4 Making a shopping cart

In order to make a shopping cart for your Web site, you must use a webEdition list with the following we:tag:

```
<we:repeatShopItem
shopname="shopname">.........</we:repeatShopItem>
```

Below is an extended example:

```
<we:createShop shopname="shopers"/><we:addDelShopItem
shopname="shopers"/>
<!—Begin header -->
<table width="100%" border="0">
<tr bgcolor="silver">
<td>Item</td>
<td width="50">Number</td>
<td>Price</td>
<td>Total price</td>
</tr>
<!-- End header -->
<!-- Begin article listing -->
<we:repeatShopItem shopname="shopers">
<tr>
<td class="normal" bgcolor="white"><table
border="0"><tr><td><we:field
name="Bild" type="img" hyperlink="on" border="0" height="30"
width="30"
align="top"/>
</td><td><b><we:field name="Title" alt="we_path"
hyperlink="on"/></b><br><we:field name="Description" alt="we_text"
max="200"/></td>
</tr>
</table>
<!-- Begin possible orders -->
<td align="center"><we:showShopItemNumber
shopname="shopers"/><br>[
<we:a id="148" shop="on" amount="1">+1</we:a>
|<we:a id="148" shop="on" amount="-1">-1</we:a>]</td>
<td></td>
<td  align="right">EURO <we:calculate
sum="waren"><we:showShopItemNumber
shopname="shopers">*<we:field name="Price"/></we:calculate></td>
</td>
<!-- End possible orders -->
</tr>
</we:repeatShopitem>
<!—End article listing -->
<tr><td colspan="4"></td></tr>
<tr bgcolor="silver"><td colspan="3">total:</td><td align="right">
 EURO <we:sum name="waren"/>
</td></tr>
</table>
```

The amount attribute is a further enhancement of the <we:a> tag. Here we are using it in the form:

```
<we:a id="148" shop="on" amount="5">+five</we:a>
```

The ID listed is the next page is the same as the calling page in this example. The shop attribute specifies that this is a Shop function. The amount attribute provides a link that increases the item number (we:shopItemNumber) by five. Another way to allow shoppers to change the item quantity is by including select boxes, as shown in the following example:

```
<!-- Begin possible orders -->
```

```
<td align="center"><we:showShopItemNumber shopname="shopers"/>
<we:form id="id_der_folgeseite" type="shopliste">
<td width="30" class="nav0" bgcolor="white">
      <select name="shop_anzahl" size="1" class="nav0">
        <option value="1">1</option>
        <option value="2">2</option>
        <option value="3">3</option>
        <option value="4">4</option>
        <option value="5">5</option>
        <option value="6">6</option>
        <option value="7">7</option>
        <option value="8">8</option>
        <option value="9">9</option>
        <option value="10">10</option>
      </select></td>
</we:form>
<!-- End possible orders -->
```

4.5 Special function of the <we:a> tag in the Shop Module

Within an item list in the Shop, the <we:a> tag takes on three new attributes:

- shop= "shows the tag that it is in a Shop"
- amount= "Number of items that are put into the shopping cart by clicking on the link created by the we:a tag"
- delarticle= "takes the corresponding item out of the shopping cart again"

The following code puts the corresponding item into the shopping cart:

```
<we:a id="id_der_folgeseite" shop="on" amount="1">Into the shopping
cart</we:a>
```

The following code takes the item out of the shopping cart again:

```
<we:a id="id_der_folgeseite" delarticle="on">Out of the shopping
cart</we:a>
```

The following code creates a link to delete the complete shopping cart:

```
<we:a id="id_der_folgeseite" delshop="on">Delete shopping
cart</we:a>
```

4.6 Special functions of the <we:form> tag in the Shop Module

The we:form tag has one additional attribute in the Shop Module:

```
type="shopliste"
```

This attribute lets you use entry fields or select boxes to provide the number of items for ordering. In the <we:form> tag, id="self" can be omitted as omitting information in the id attribute always signifies a link to itself. When it is linked to another page, id="id_of_next_page" must always be included. For example:

```
<!—Begin possible orders -->
<td align="center"><we:showShopItemNumber shopname="shopers"/>
<we:form  id="id_der_folgeseite"type="shopliste">
<input type="Text" name="shop_anzahl">
<input type="submit" value="abschicken">
</we:form>
<td  align="right">EURO <we:calculate
sum="waren"><we:showShopItemNumber
shopname="shopers">*<we:field name="Price"/></we:calculate></td>
</td>
<!-- End possible orders -->
```

4.7 Performing calculations

webEdition allows you to perform extensive calculations with the <we:calculate> and <we:sum> we:tags.

<we:calculate> offers you all functions that PHP generally offers:

- + Addition
- - Subtraction
- * Multiplication
- / Division
- () Nested calculations
- sqrt Square root

<we:sum name="your_sum"> adds all individual totals that are defined in the <we:calculate sum="your_sum"> tag.

The following is an example for <we:calculate>:

```
<we:calculate name="endpreis" num_format="german" print="on">
(<we:session
field name ="artikel_summe"/>*(1-<we:field
name="rabattstufe1"/>))*(1+(<we:field
name="mwst"/>/100))</we:calculate>
```

The following is a full example showing the use of <we:calculate> and <we:sum>:

```
<we:createShop shopname="shopers"/><we:addDelShopItem
shopname="shopers"/>
<table width="100%" border="0">
<tr bgcolor="silver">
<td>Item</td>
<td width="50">Number</td>
<td>Price</td>
<td>Total price</td>
</tr>
<we:repeatShopItem shopname="shopers">
<tr>
<td class="normal"  bgcolor="white"><table
border="0"><tr><td><we:field
name="Bild" type="img" hyperlink="on" border="0" height="30"
width="30"
align="top"/>
</td><td><b><we:field name="Artikelname" alt="we_path"
hyperlink="on"/></b><br><we:field name="Description" alt="we_text"
max="200"/></td>
</tr>
</table>
<td align="center"><we:showShopItemNumber
shopname="shopers"/><br>[<we:a
id="148" shop="on" Number="1">+1</we:a>|<we:a id="148" shop="on"
Number="-
1">-1</we:a>]</td>
<td><we:field name="Price"/></td>
<td  align="right">EURO
<we:calculate sum="warenkorb"
num_format="german"><we:showShopitemNumber
shopname="shopers">* <we:field name="Price"/></we:calculate></td>
</td>
</tr>
</we:repeatShopitem>
```

```
<tr><td colspan="4"></td></tr>
<tr bgcolor="silver"><td colspan="3">total:</td><td align="right">
 Euro <we:sum name="warenkorb" num_format="german"/>
</td></tr>
</table>
```

The code cited above produces the following output on a browser (see Figure 35):

Figure 35 Output to browser

5 Customer Management: Introduction

This introduction is intended to help you familiarize yourself with the webEdition Customer Management and Customer Management PRO Modules. This chapter treats what the modules do and how to install them. You can also find information here about the basic layout and command features for the Customer Management Module. These topics are treated in the following sections:

- Section 5.1, "What is the webEdition Customer Management Module?" on page 43
- Section 5.2, "Installation" on page 43
- Section 5.3, "General information and navigation" on page 44

The PRO Module builds upon the standard Customer Management Module and therefore the information in this book pertains to both webEdition applications. Supplementary material describing the particular features of the Customer Management PRO Module is found in 4, "The Customer Management PRO Module" on page 37.

5.1 What is the webEdition Customer Management Module?

The webEdition Customer Management Module is a database that allows you to register and manage external visitors to your web site. (These visitors are called "customers" in this document.) The Customer Management Module can manage customer data in two ways:

- You can enter customer data manually yourself.
- With the help of a registration form that you have designed for your Web site, customers can enter their data themselves. In this case, customers will log in by entering their user name and password. This feature allows you to offer registered customers an extended selection of information or give them access to areas from which non-registered users are barred.

The Customer Management PRO Module allows you to perform all of the same functions as the Customer Management Module. In addition to the basic functions, the PRO module gives you enhanced abilities to manage your database by allowing you to:

- Sort your customers
- Search for customers

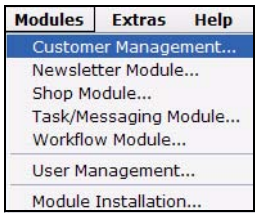
5.2 Installation

The installation procedure for all modules is described in The webEdition Installation, Update and Backup Procedures. A .pdf version of this guide is available at the following URL: http://www.living-e.de

5.3 General information and navigation

After installation, you will find a new menu item in the main menu called *Modules*, which contains all the modules in your installation of webEdition (see Figure 36).

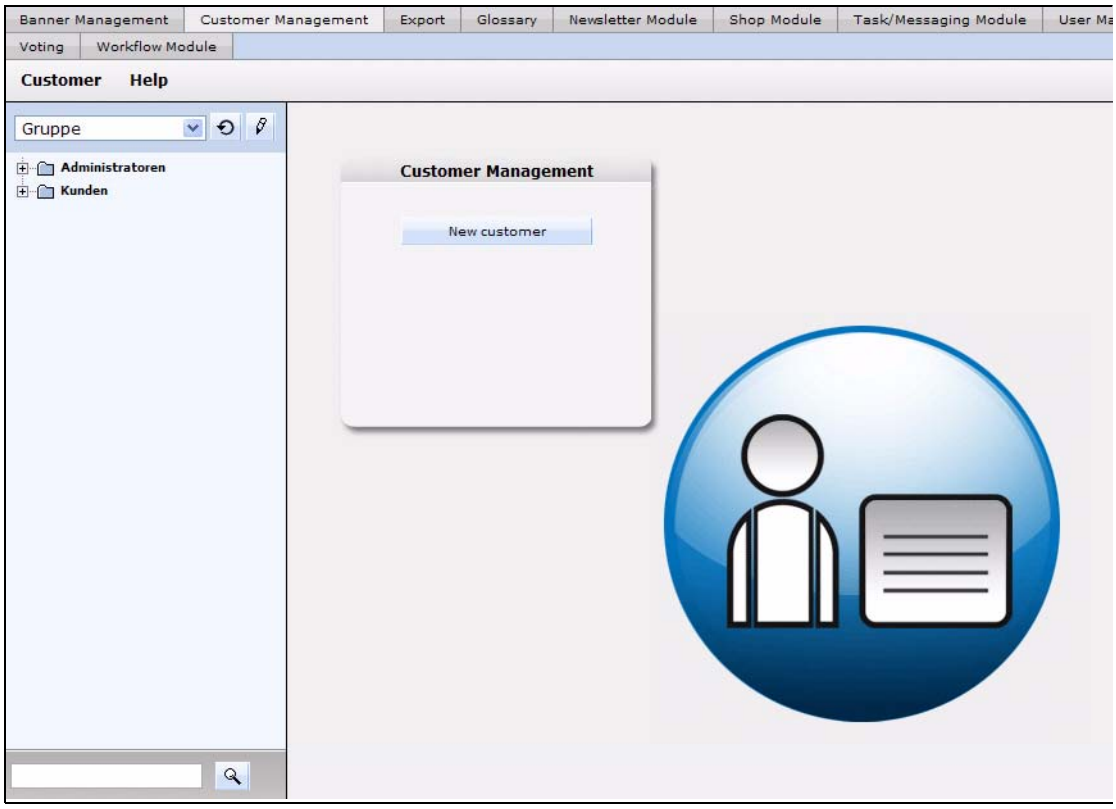
Figure 36 Modules drop-down menu



5.3.1 Opening the module

To open the module, select *Customer Management* from the *Modules* drop-down menu on the webEdition main screen. The Customer Management Quickstart screen opens as seen in Figure 37.

Figure 37 Customer Management Module: mainscreen with Quickstart option

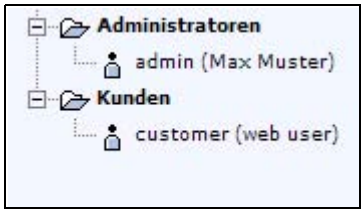


Using the *Quickstart* screen, you have the option of going directly to a new customer configuration screen.

5.3.1.1 Customer Management explorer menu

The explorer menu displays a list of icons representing the customer data you have in your customer database (see Figure 38). You can access and modify the profile for a particular customer by double clicking on the icon for that customer.

Figure 38 Customer Management explorer menu



5.3.1.2 Permissions to modify the customer database

The default user has all of the permissions to create, modify and delete the data and data fields in the Customer Management database. Additional flexibility is available with the User Management Modules.

5.3.1.2.1 Interaction with the User Management Modules If the Standard User Management Module is installed, permission to set up new customers (registered visitors to your site), or to modify the fields in the customer database are limited to the administrator. If you have installed the User Management PRO Module, the administrator can assign to any user the permission to create, delete, edit customer or edit customer fields. For more information on the User Management Module, see the *User Management Module User Guide* from the webEdition web site.

5.3.2 Customer Management PRO explorer menu

Like the standard Customer Management Module, the explorer menu in the PRO Module displays a list of icons representing the customer data you have in your customer database. There are two additional features visible in the PRO Module explorer menu (see Figure 39):

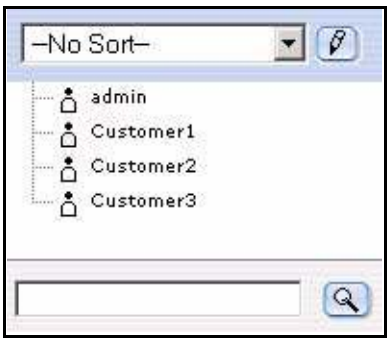


Figure 39 Customer Management PRO explorer menu

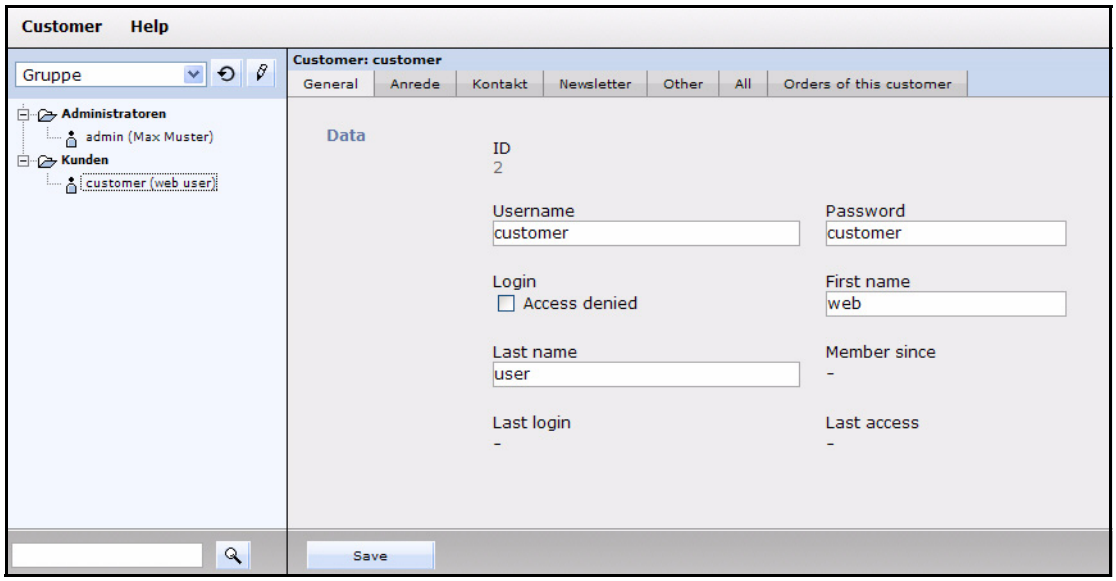
•Sort feature. At the top of the explorer menu is the sort feature. The sort feature has a select box where you can choose from your defined sort profiles, and a pencil icon where you can create and/or edit the your parameters for your existing sort profiles.

•Search feature. At the bottom of the explorer menu is the search feature which consists of a search field, and a search icon.

5.3.3 The *Customer* drop-down menu

Click on *Modules* > *Customer management* to open the Customer Management Module main screen (see Figure 40).

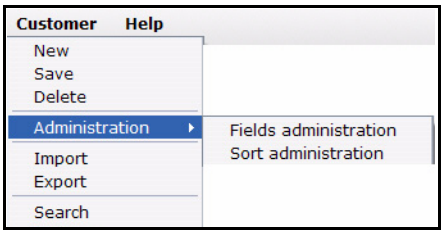
Figure 40 Customer Management main screen



The *Customer* menu in the PRO Module has the following items (see Figure 41):

- *Administration*. The PRO Module offers two administrative features.
  - — *Fields administration*.
  - — *Sort administration*.
- *Search*. Use this item to access the module's search engine
- *Settings*. This item allows you to set your default sort view and your Start year for date sorting.

Figure 41 *Customer* drop-down menu in the PRO Module



5.3.4 Using the Search function

The Customer Management PRO Module offers advanced search capabilities, allowing you to find and manage the data for your customers.

You can access the search feature in the following ways:

- Click on *Customer* > *Search*
- Use the simple search option at the bottom of the explorer menu.

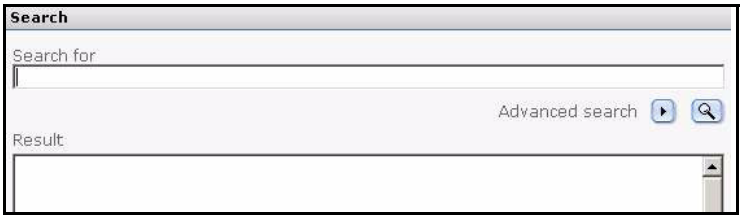
Procedure 1 Using the search feature

In the Customer Management Module main screen

1 Click on *Customer* > *Search* or enter a name for a customer in the search field at the bottom of the explorer menu and click the search icon (represented by a magnifying glass).

The Search window opens (see Figure 42).

Figure 42 Search window



2 Two types of searches are possible. Do one of the following steps:

a *Simple search*.

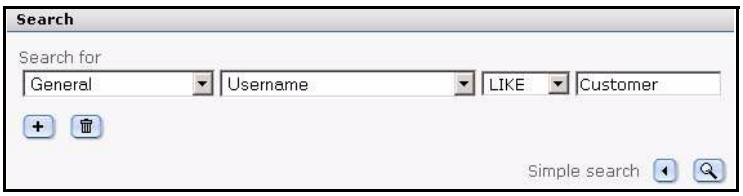
To perform a simple search, enter the text string in the *Search for* field.

b *Advanced search*.

i Click the right arrow icon beside *Advanced search*.

The Advanced search fields appear as seen in Figure 43.

Figure 43 Advanced search



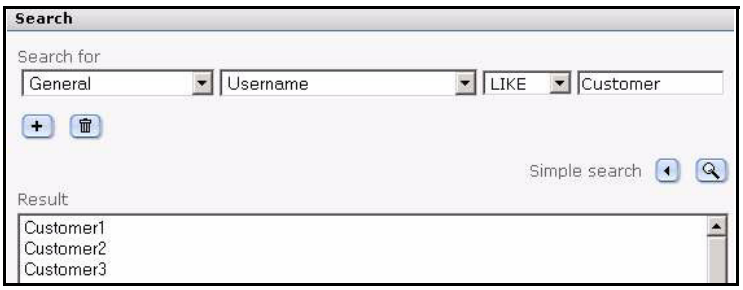
ii Select your search parameters from the select boxes.

- In the first select box, choose one of the views that you have defined.
- In the second select box, choose from the fields that you have defined within the view.
- In the third select box, choose a search operator.
- Enter a parameter in the text field.

3 Click the search icon (magnifying glass).

If the system finds a match for your search, the item will appear in the Result box (see Figure 44).

Figure 44 The Search Result box



4 To access or edit the data for an item found in the *Search Result* box, double click on the item.

webEdition opens the editing area for the customer, where you can modify the data in the General, Other and All views, and also those additional views (such as Salutation or Contact) that you have created.

5 You have completed this procedure.

5.3.5 Using the *Sort* function

The Customer Management PRO Module offers advanced sorting capabilities, allowing you to create and save multiple sort profiles so that you can manage the data for your customers.

You access the sort feature by selecting an item from the sort select box at the top of the explorer menu. You can edit your sort parameters by clicking the edit (pencil) icon adjacecnt to the sort select box, or by clicking *Customer* > *Administration* > *Sort administration*.

> ***Note:*** The first time you log in to the system, there are no sort parameters defined. The select box has —*No Sort*—.

Procedure 2 Using the sort feature

In the Customer Management Module main screen

1 Click *Customer* > *Administration* > *Sort administration*.

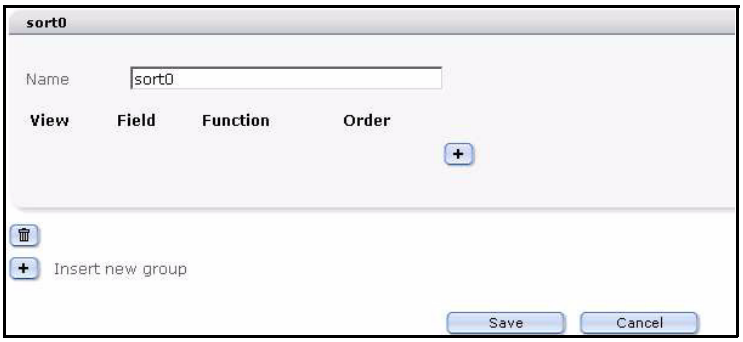
The Sort administration screen opens. The first time that you use this feature, the system displays a plus icon.

Figure 45 Sort administration screen

2 Click on the plus (+) icon to activate *Insert new group*.

The Sort area opens (see Figure 46).

Figure 46 Sort area



3 Type a name to identify your sort group in the *Name* field.

4 Click the plus (+) icon to add your sort parameters. Choose the sort parameters in accordance with the way you wish to arrange and access your customer data.There are four areas, each is represented by a select box. The views, fields and functions that are available are determined by the views and fields that you have defined for your Customer Management database (see Section 6.4.1, "Creating a new view" on page 57).

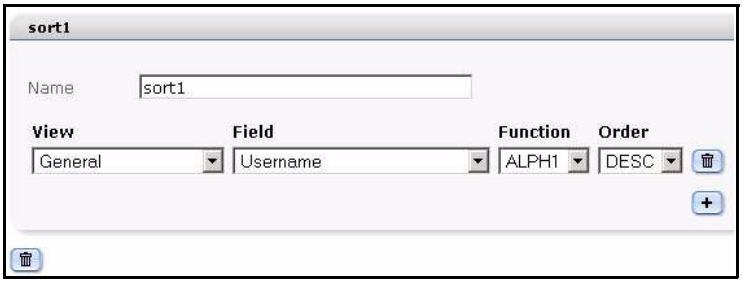
a *View*. Sort according to view.

b *Field*. The available fields are determined by the view you have chosen.

c *Function*. The available functions vary depending on the field type that characterises the field you have chosen. For example text, can be sorted by alphabetical order; dates by day, moth, year (etc.).

d *Order*. Use this command to choose to display your sort results in ascending (*ASC*) and descending (*DESC*) order.

Figure 47 Sort parameters selected



5 Click *Save*.

The system shows you a confirmation message.

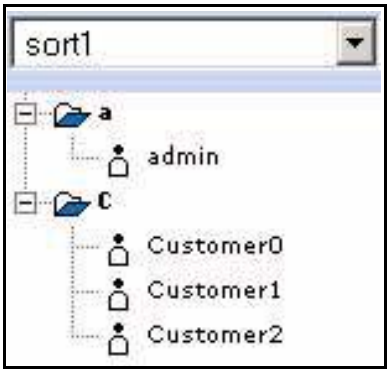
6 Click *OK*.

The sort group you have created appears in the sort select box in the explorer menu.

7 Click on your newly created sort group.

Your customers appear in folders that coform to the parametes you have set.

Figure 48 New sort group



8 You have completed this procedure.

5.3.6 About fields and views in the customer database

You can use the customer database to store and maintain all of your data about your customers. Typically such a database would include information under such categories as customer name, E-mail address, street address, telephone number, etc.

5.3.6.1 About views

The webEdition Customer Management Module software is pre-programmed to provide you with the following default categories, which are displayed in the module's interface in the following views: *General*, *Other*, and *All*. You are not limited to the foregoing default settings. You can modify the database to meet your own needs by adding new types of information. You can add, for example, a view for a salutation or contact information (see Section 6.4, "Working with customized views" on page 57). In this manner you are free to customize the database to meet the needs of your business or organization.

5.3.6.1.1 The *General* view The *General* view provides the basic interface to your customer database. You use this view to enter user data about a new customer.

Select *Customer* > *New* from the Customer Management Module menu to access the default *General* view (see Figure 49):

The *General* view has the following fields, each of which allows you to enter a customer's basic data:

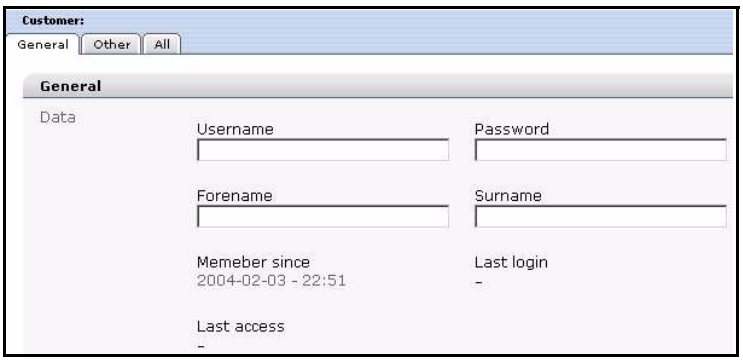


Figure 49 *General* view

- *Username*
- *Password*
- *Forename*
- *Surname*
- *Member since*. This is a read only-field which tells you the date and time the user was created in the system.
- *Last login*. This is a read only-field which tells you the date and time the user was last logged into the system.
- *Last access*. This is a read only-field which tells you the date and time the user last accessed data from the system.

5.3.6.1.2 Other view This view is included so that you can add miscellaneous field types to a default view without having to create additional views.

5.3.6.1.3 *All* view This view displays all your views in succession.

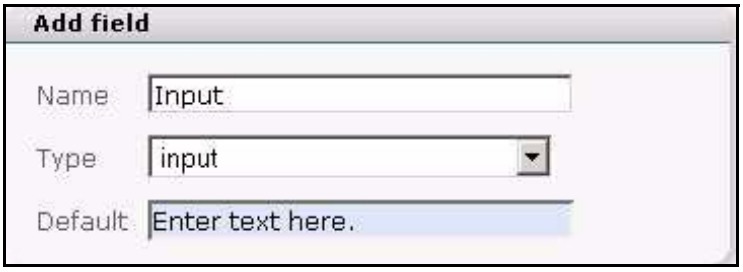
5.3.6.2 About fields

Within each of these categories (views) are the data fields which contain specific information about your customer. You can customize these fields to meet your own needs. You have the option of expanding all fields and views (with the exception of the *General* data). You can give names to your field, define the field type, and assign default values to those fields. (These activities can be performed using the *Fields administration* command, which is discussed in detail in Section 6.3, "Working with fields in the customer database" on page 54).

5.3.6.2.1 Field types There are five default field types from which you can choose. Each can be assigned default values:

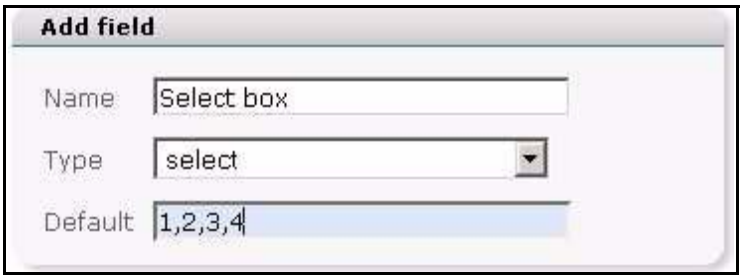
- *input*. This field type creates a text input field suitable for short text entries. You can enter a default value in the *Default* field.

Figure 50 Input text field



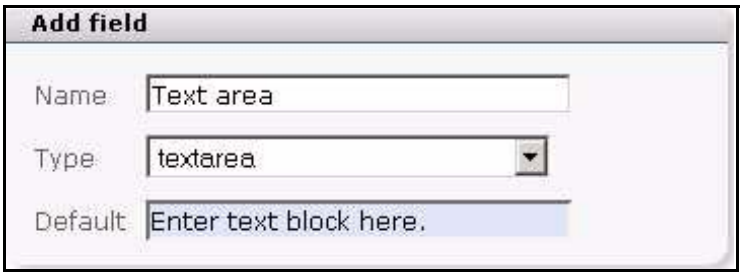
- *select*. This field type creates a select box. The default values that you enter in the *Default* field must be separated by a comma. Each of the comma-separated values will appear as an item in your select list.

Figure 51 select field type



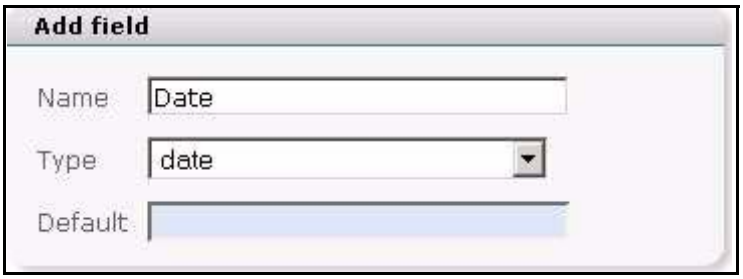
- *textarea*. This field type creates a box in which text can be written.

Figure 52 text area field type



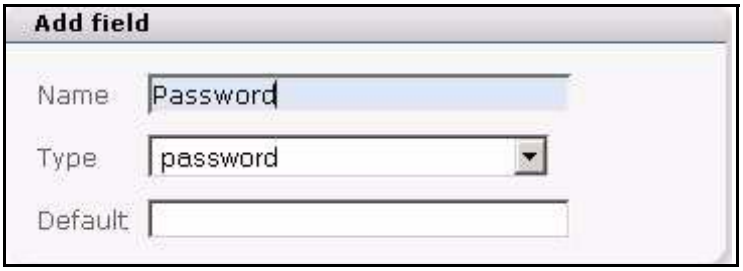
- *date*. This field type creates date and time select boxes.

Figure 53 Date field type



- *password*. This field type creates a text input box. For reasons of security, characters typed by a user in this field are masked.

Figure 54 Password field type



6 Using the customer database

This chapter describes how to use the Customer Management and Customer Management PRO Modules to create and manage a customer database. This chapter describes how to modify the database to meet the needs of your business or organization by adding new fields and views to accomodate different types of customer information.

This chapter describes how to work with the customer database under the following sections:

6.1 Creating a new customer using the Customer Management Module

Use the following procedure to create a new customer.

Procedure 3 Creating a new customer

In the Customer Management Module main screen

1 Click *Customer > New*.

The system opens the General view , where you can enter basic data about your customer (see Figure 55).

Figure 55 New customer: *General* view

2 Enter all the customer data in the prescribed fields.

3 Save the data by clicking on the *Save* button at the bottom of the page.

webEdition confirms that the customer data was saved. Click OK. An icon representing the newly created customer appears in the explorer menu. The customer can now log onto the home page using the login you have assigned.

Figure 56 New customer icon in the explorer menu

Note: To find out how to create the login mask, go to our documentation page at http://www.living-e.de, and consult the publication titled *The webEdition User Guide*.

4 You have completed this procedure.

6.2 Modifying customer data

You can modify the data for a customer by double clicking the icon listed in the explorer menu that signifies your customer. Thereafter, you can toggle among the tabs on the main screen to add or access information about the customer.

6.3 Working with fields in the customer database

The Customer Management Module allows you to change fields using the *Fields administration* feature. The following sections describe procedures for working with fields.

6.3.1 Adding fields to the customer database

The following procedure shows you how to add new field names to the default customer management database so that you can customize it for your own needs.

ATTENTION

Once you insert (or delete) field names, they will be accessible (or deleted) for all customers.

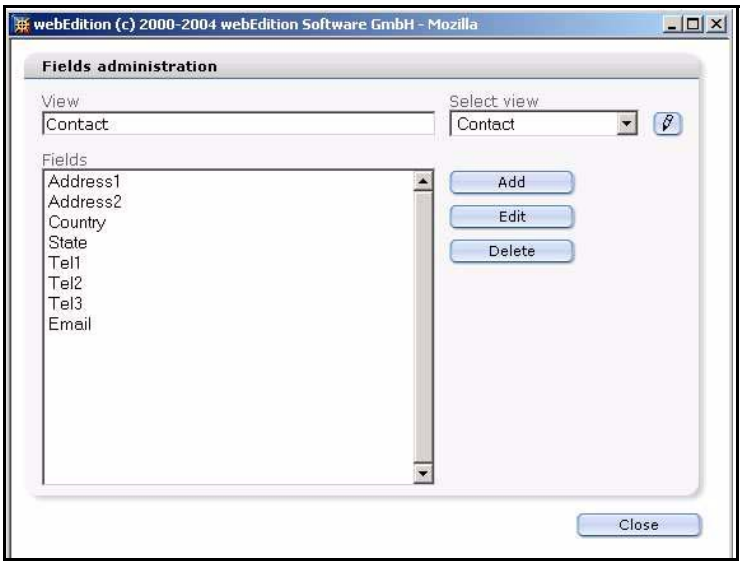
Procedure 4 Adding field names to the default customer database

In the Customer Management Module main screen

1 Click *Customer* > *Administration* > *Fields administration*.

The Fields administration screen opens (see Figure 57).

Figure 57 Fields administration screen

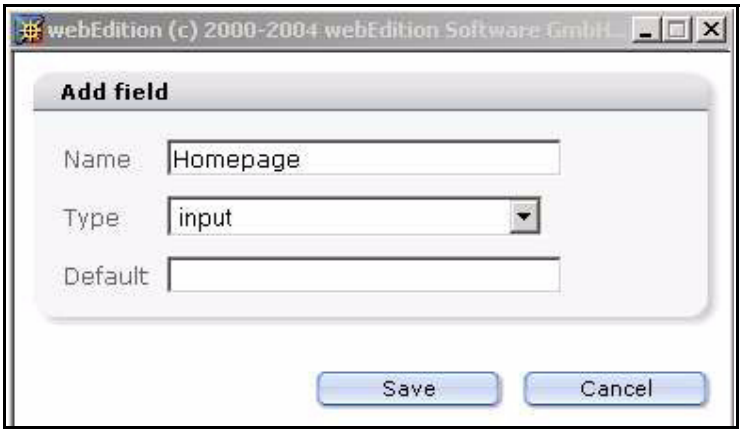


2 You must associate a field with an existing view. Use the *Select view* select box to choose a view. (If you wish to create a new view first, see Section 6.4, "Working with customized views" on page 57).

3 Click the *Add* button adjacent to the *Fields* area.

The Add field dialogue box appears.

Figure 58 *Add field* dialogue box

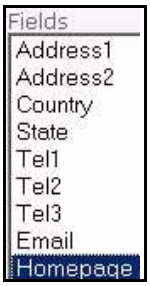


4 Enter a field name, select a field type from the select box, and enter a default value. (For information about field types, see Section 5.3.6.2.1, "Field types" on page 51).

5 Click on the *Save* button.

The new field appears in the Fields list on the Fields administration screen.

Figure 59 New customer field



6 You have completed this procedure.

6.3.2 Modifying fields in the customer database

Procedure 5 Changing the properties of a field

In the Customer Management Module main screen

1 Click *Customer* > *Administration* > *Fields administration*.

The Fields administration screen opens (see Figure 57, "Fields administration screen" on page 55).

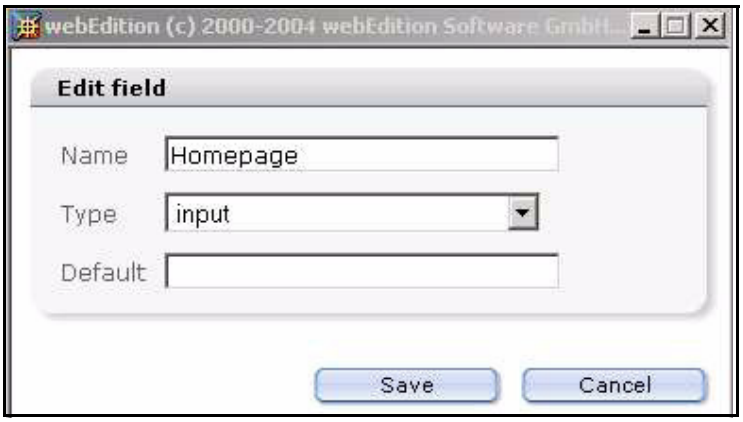
2 Use the *Select view* select box to choose the view which has the field you wish to change.

3 In the *Fields* area, highlight the name of the field you wish to change. See, for example, Figure 59, "New customer field" on page 55.

4 Click the *Edit* button adjacent to the *Fields* area.

The Edit field dialogue box appears.

Figure 60 *Edit field* dialogue box

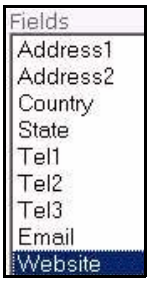


5 Change a field name, field type or default value.

6 Click on the *Save* button.

The modified field appears in the Fields area on the Fields administration screen.

Figure 61 Edited customer field



7 You have completed this procedure.

6.3.3 Deleting a field

Procedure 6 Deleting the name of a view

In the Customer Management Module main screen

1 Click *Customer* > *Administration* > *Fields administration*.

The Fields administration screen opens (see Figure 57, "Fields administration screen" on page 55).

2 Use the *Select view* select box to choose the view which has the field you wish to change.

3 In the *Fields* area, highlight the name of the field you wish to delete.

4 Click the *Delete* button.

webEdition asks you if you want to delete the chosen field.

5 Click the *OK* button to delete the field(s).

The system confirms the action. Click the OK button again.

Note: *If you delete the last field in a selected view, you will also delete the view.*

6 Click on the *Close* button.

7 You have completed this procedure.

6.4 Working with customized views

The Customer Management Module allows you to manage field types arranged in views. Using the *Fields administration* feature, you can create new views and new fields for each view. The following views are predefined and cannot be changed or deleted: *General*, *Other*, *All*.

6.4.1 Creating a new view

Procedure 7 Creating a new view

In the Customer Management Module main screen

1 Click *Customer* > *Administration* > *Fields administration*.

The Fields administration screen opens (see Figure 57, "Fields administration screen" on page 55).

2 To create a new view, enter a name for the new view in the *View* field.

3 Click the *Add* button adjacent to the *Fields* area.

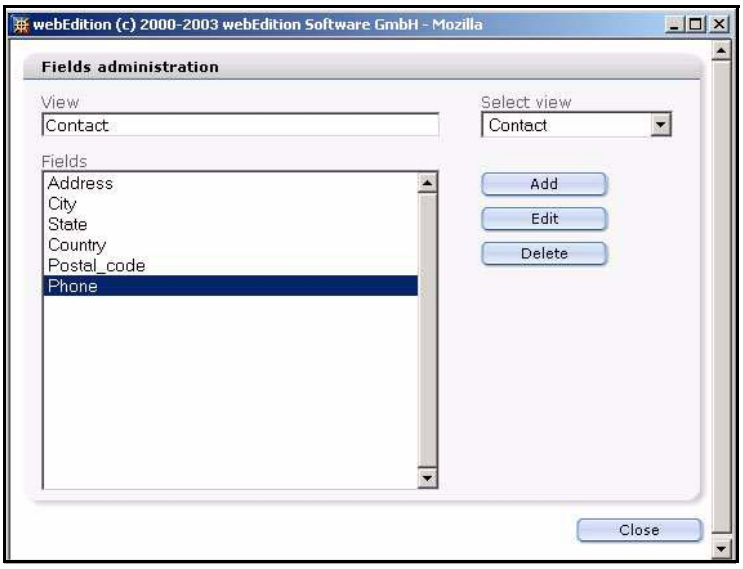
The Add field dialogue box appears.

4 Enter a field name, field type and default value and click *OK.*

Note: You must create at least one field for your view.

A new view has been created. The new view name appears in the View field and also in the Select view select box (see Figure 62).

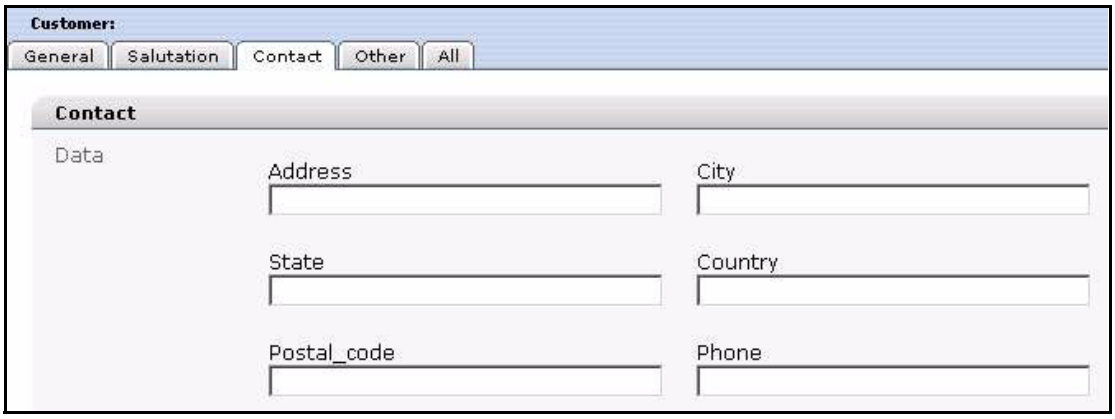
Figure 62 Creating a new view



5 Click on the *Close* button.

A new tab for the view appears on the Customer Management main screen.

Figure 63 New customer tab and view



6 You have completed this procedure.

6.4.2 Editing a view

Procedure 8 Editing the name of a view

In the Customer Management Module main screen

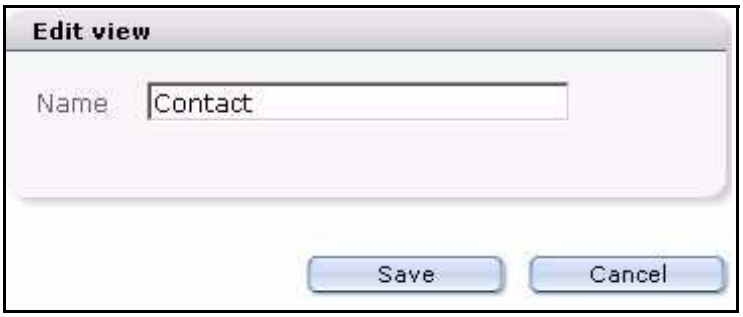
1 Click *Customer* > *Administration* > *Fields administration*.

The Fields administration screen opens (see Figure 57, "Fields administration screen" on page 55).

2 Click the pencil icon adjacent to the *Select view* select box.

The Edit view dialogue box appears.

Figure 64 *Edit view* dialogue box

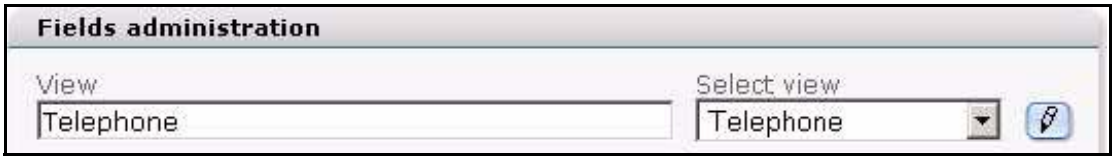


3 Enter a new name in the *Name* field.

4 Click the *Save* button.

A new name for the view appears in the View field and in the Select view select box.

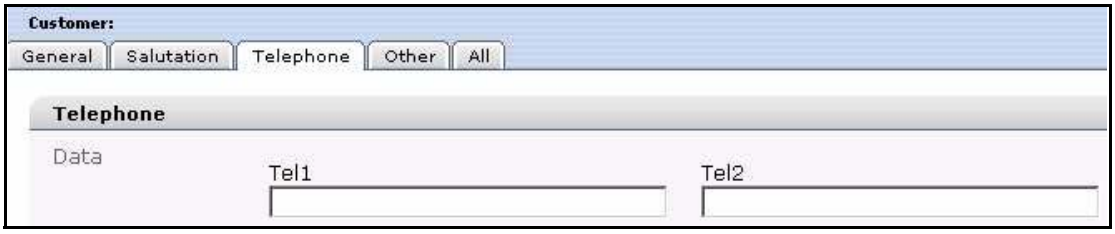
Figure 65 Edited customer view tab



5 Click on the *Close* button.

The newly-named tab for the view appears on the Customer Management main screen.

Figure 66 Renamed customer view tab



6 You have completed this procedure.

6.4.3 Deleting a view

Procedure 9 Deleting a view

In the Customer Management Module main screen

1 Click *Customer* > *Administration* > *Fields administration*.

The Fields administration screen opens (see Figure 57, "Fields administration screen" on page 55).

2 Highlight each of the fields in the *Fields* area.

3 Click the *Delete* button.

webEdition asks you if you want to delete the chosen field.

4 Click the *OK* button to delete the field(s).

The system confirms the action. Click the OK button again.

5 Click on the *Close* button.

Once you have removed all fields associate with the view, the view and its tab disappear from the Customer Management main screen.

6 You have completed this procedure.

7 Designing templates for the Customer Management Module

In order to make your website accessible to registered users (or customers as we refer to them), you must insert the appropriate we:tags into your templates. This chapter shows you how to use webEdition tags (we:tags) to do the following actions:

- create registration forms so that external users can register as "customers"
- create login areas for your customers on your web site
- make information accessible only to logged-in users or customers.

The we:tags discussed in this chapter pertain to both versions of the Customer Management Module, hence this chapter is intended for users of both the Customer Management and the Customer Management PRO modules.

Those using this chapter are expected to have a knowledge of template development and we:tags. Moreover, webEdition users must have either administrative priveleges, or have the permissins to create and deploy templates in order to manage customer access.

7.1 Creating registration forms

You use a registration form to create registered users (or customers) on your web site. By registering visitors to your web site, you can make information available to them that unregistered users cannot see or access.

To create a registration form, you use the <we:sessionField/> tag. This we:tag works closely with the Customer Management Module. The <we:sessionField> tag draws fields from the Customer Management Module and places them on a Web page.

For example, in your Customer database, if you have the field "Contact_Email", which is designed for the E-mail addresses of your customers, you can display this field value with the <we:sessionField tag> by adding the following variables: <we:sessionField name="Contact_Email" type="textinput"/>. This tag will produce one input box for the customer E-mail on the page. All field names that are in the Customer Management database can be used as variables with the <we:sessionField/> tag. These variables can be initially entered, changed or deleted by using a form containing the <we:sessionField..../> tag with the corresponding variable(s).

Procedure 10 Creating a registration form

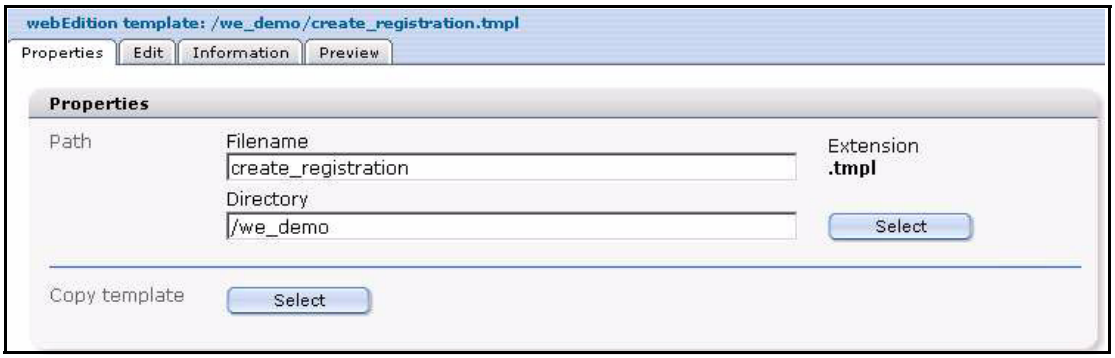
In the Templates view of the webEdition main screen

1 If you are not already in the *Templates* view, go there by clicking on the *Templates* tab.

2 Open a new template by selecting *File* > *New* > *Template*.

3 Go to the *Properties* view by clicking on the *Properties* tab (see Figure 67).

Figure 67 The *Properties* view



4 In the *Path* area, enter a name for the file in the *Filename* field, and a select a directory.

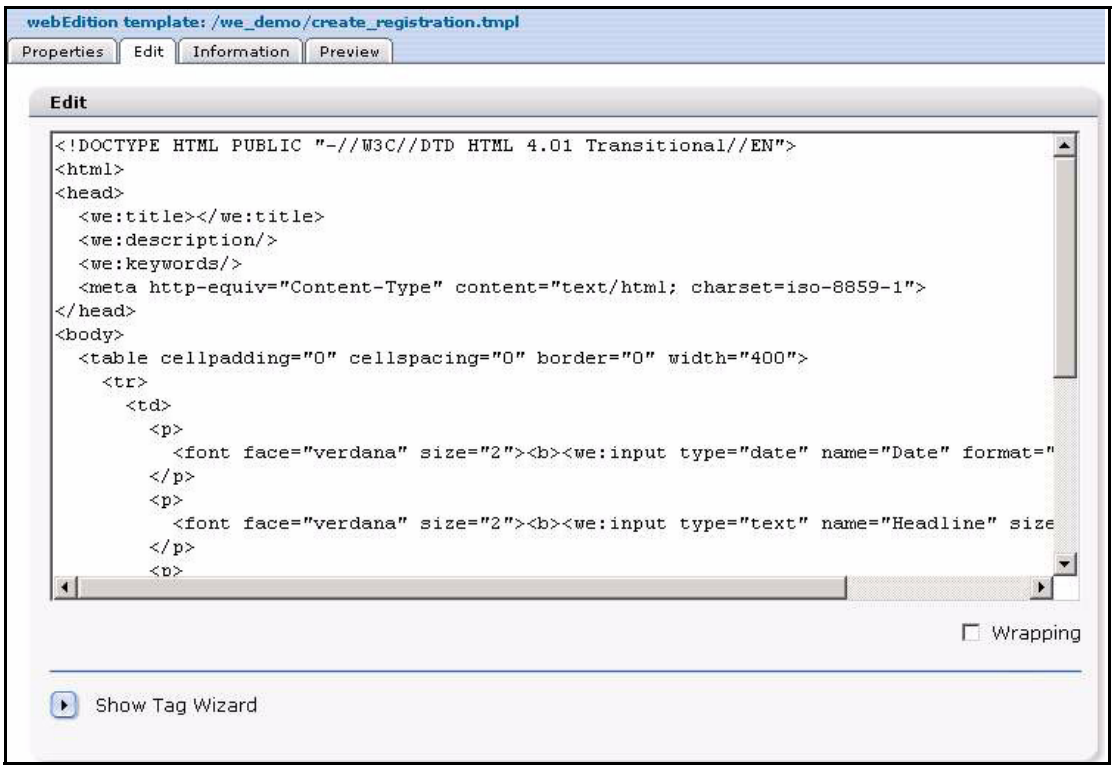
5 Click *Save*.

Your new template appears in the explorer menu.

Note: *Your template will have a .tmpl file extension.*

6 Click the *Edit* tab to open the *Templates Edit* view (see Figure 68).

Figure 68 Template *Edit* view



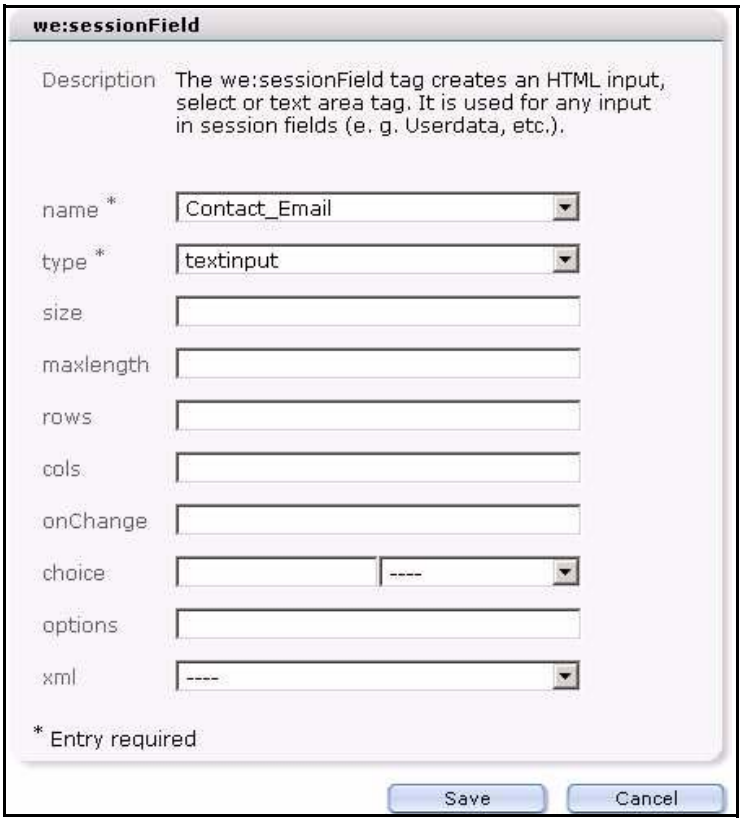
7 Click on the *Show Tag Wizard* arrow. In the *Tag Wizard* area, use the scroll bar to find *sessionField* tag (see Figure 69). Click the *sessionField* tag to highlight it.

Figure 69 we:Tag Wizard: sessionField

8 Click the arrow to the right of the select box to open a dialogue box for the <we:sessionField> tag.

A "we:sessionField" dialogue box opens (see Figure 70).

Figure 70 we:sessionField dialogue box



9 Choose a value from the *name* attribute select box. You will see all the default field names (and any that you have added in the Customer Management Module) that are available for your form (see Figure 71).

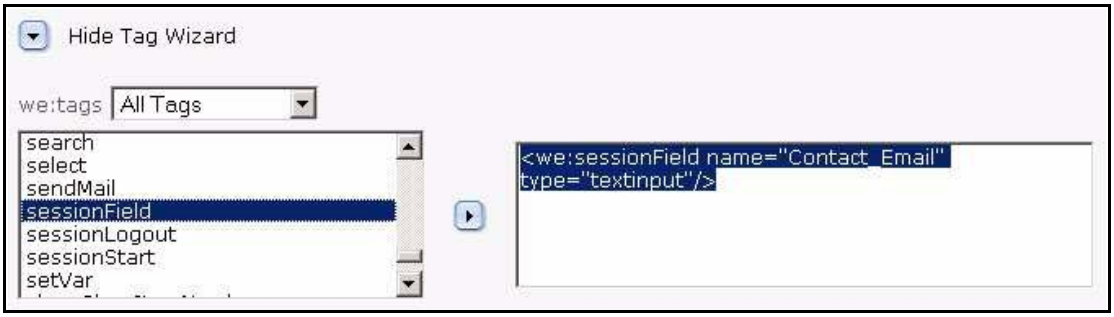
Figure 71 Available field names for your form

10 Choose the type of input field from the *type* select box.

11 Click *Save*.

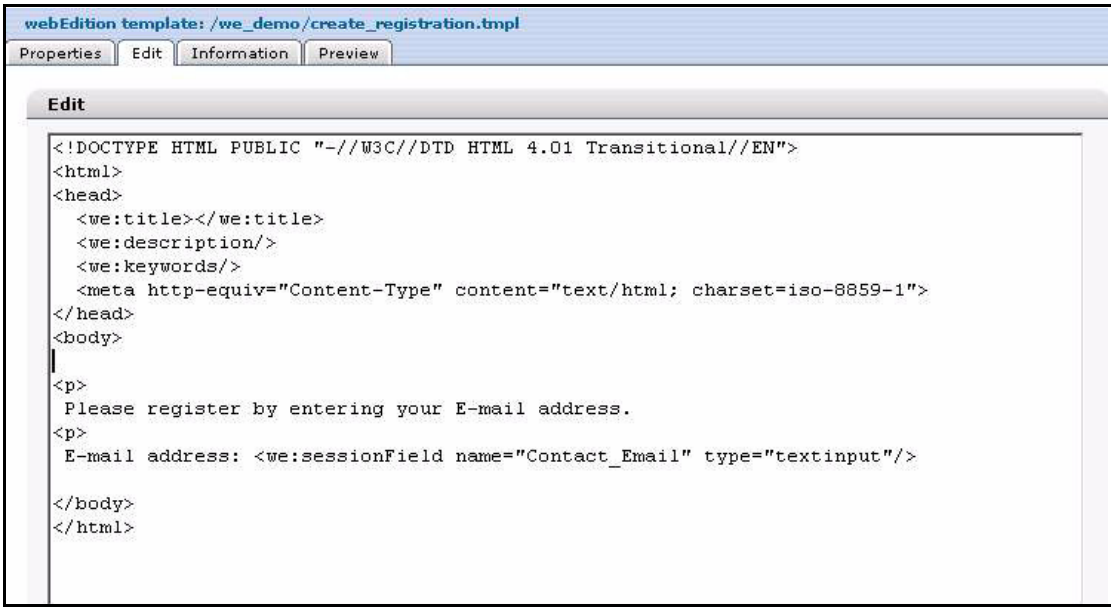
The dialogue box closes and you are returned to the Edit view. The <we:session> tag with your newly defined attribute appears in the right -hand portion of the Tag Wizard (see Figure 72).

Figure 72 The sessionField tag with defined attributes



12 Using your mouse, cut the newly defined tag from the right side of the Tag Wizard and paste it into your template. (You can also edit and add any accompanying text to the template at this time.)

Figure 73 The sessionField tag used in a template



13 Click *Save*.

14 Click on the *Preview* tab to view the changes to your template.

Figure 74 Preview of

15 You have completed this procedure.

7.2 Creating a form to register a new customer

To allow visitors to your site to register as a new customer, you need to create a link on your Web site that points to a form where all required fields (or all fields listed in the customer management database) are called up with the <we:sessionField/> tag. Once the visitor has entered all the required information, the form can be saved so that all data will be written to the database. Thereafter the visitor becomes a “customer” and can log in with a user name and password. From this point on, you can view and change the data associated with this customer using the Customer Management Module.

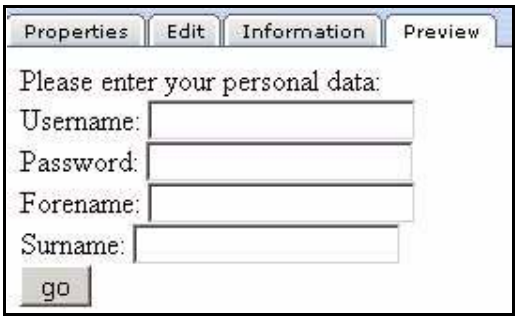
7.2.1 Designing registration forms: code example

You can design registration forms with the following we:tags:

```
<we:sessionStart/>
<we:saveRegisteredUser/>
<html>
<head>
  <we:title>webEdition Default-template</we:title>
  <we:description>Form</we:description>
  <we:keywords>webEdition, cms, </we:keywords>
</head>
<body>
Please enter your personal data:<br>
<we:form id="self">
  Username: <we:sessionField name="Username" type="textinput"/><br>
  Password: <we:sessionField name="Password" type="password"/><br>
  Forename: <we:sessionField name="Forename" type="textinput"/><br>
  Surname: <we:sessionField name="Surname" type="textinput"/><br>
  <we:sessionField name="ID" type="hidden"/>
  <input type="submit" value="go">
</we:form>
</body>
</html>
```

Figure 75 shows the output from the code noted above.

Figure 75 Example of form output



7.2.2 Explanation of code

It is essential that forms in which changes are to be made to the data always contain the following type of hidden field: <we:sessionField Last name="ID" type="hidden"> This field ensures that the session created by the <we:sessionStart/> tag is also transmitted and the customer is uniquely identified. The data cannot be assigned without this session. The following page must begin with: <we:sessionStart/><we:saveRegisteredUser/>. Aside from these tag conventions, you have complete freedom with the layout of your form.

7.3 Creating a login area for your customers

The webEdition Customer Management Module gives you the capability to create a section on your Web site where your customers can log in to your site. By doing so, you can make whole pages or entire sections of your site accessible only to registered users.

The following sub-sections describe how to

- generate a login section where a user can log in
- make information accessible only to logged-in users

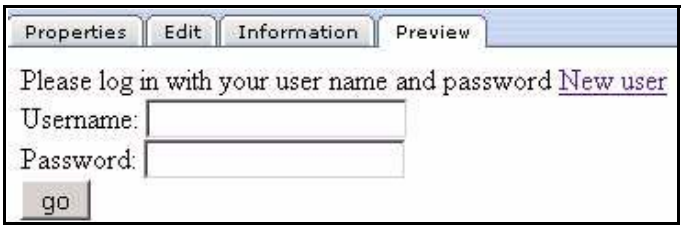
7.3.1 Generating a login section: code example

The following code is an example of how to design a template to allow customers to log in to your Web site:

```
<we:sessionStart/>
Please log in with your user name and password
<we:a id="self">New user</we:a>
<br>
<we:form id="self">
Username: <we:sessionField name="Username" type="textinput"/><br>
Password: <we:sessionField name="Password" type="password"/><br>
<input type="submit" value="go">
</we:form>
```

Figure 76 shows the output from the code noted above.

Figure 76 Example of login form



7.3.2 Making information accessible only to logged-in users: code example

You can make whole pages or entire sections accessible only to registered users. You can implement this function by using the <we:ifRegisteredUser/> tag:

```
<we:sessionStart/>
<we:ifRegisteredUser>
Hello: <we:sessionField Last name="user name" type="print"/><br>
Logged in
</we:ifRegisteredUser>
```

7.3.2.1 Explanation of code

For all functions that you wish to control from the Customer Management Module, you must always include the <we:sessionStart/> tag. This tag must appear at the very beginning of every page that has anything to do with the Customer Management Module, before any other (html) tag (e.g.<html> or <body>).

The code queries whether a user is registered. If so, the user will be greeted with "Hello, user name". User name is a placeholder for the visitor’s actual user name as entered in the system.

You can decide what is written between the <we:ifRegisteredUser> and </we:ifRegisteredUser> tags. It can be individual information, features, links to other pages, navigation or any other functions.

7.4 Creating a form to allow a customer to modify their existing registration data

In order to allows a customer to modify their existing registration data, you need to create a form that is identical to the one described in Section 7.2, "Creating a form to register a new customer" on page 65, but which is located in an area of your site that can only be accessed by registered customers. Because this user is already registered, the registration information will always be called up via the <we:sessionStart/> tag. When the customer accesses the form, the existing data in the database will automatically be written to the form fields. Thereafter the customer can modify and save his/her registration information.

Index

A

C

S

T

U

V

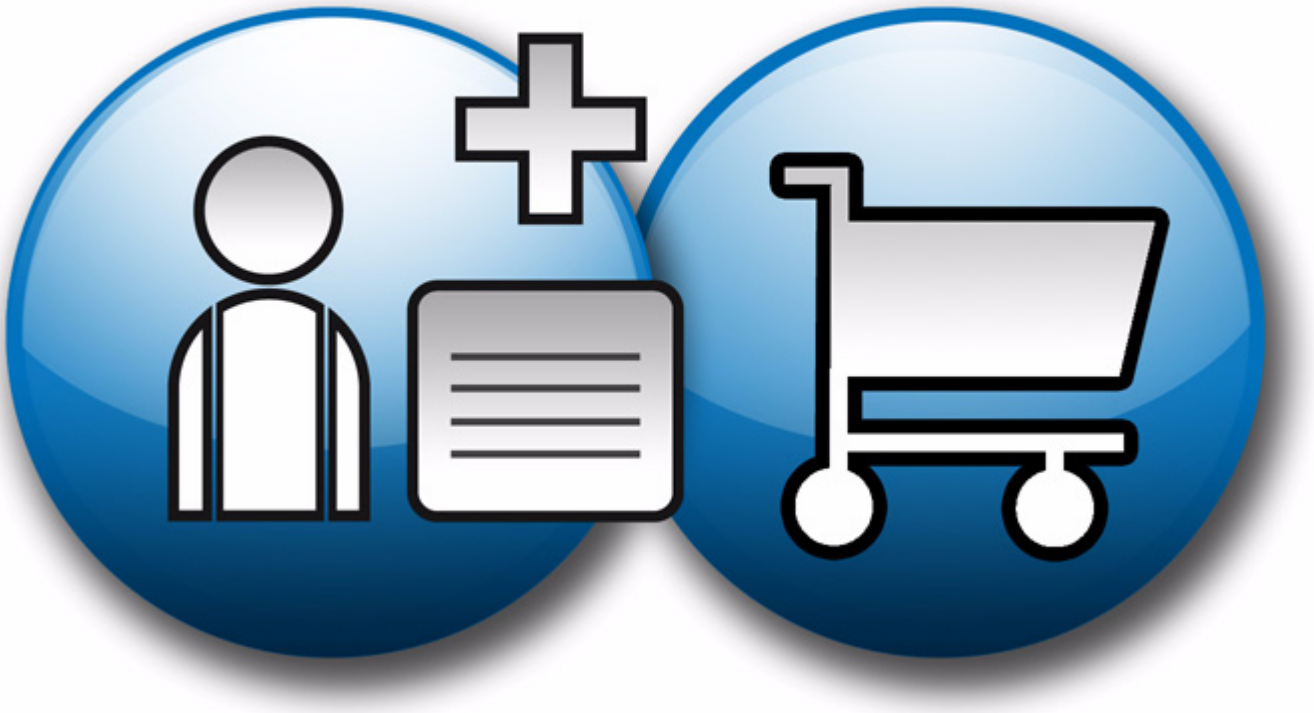
W

Y

Customer Management (PRO) and Shop Modules

User Guide

Standard 5.0.1
June 2007
Printed in Germany



living-e AG
Karlsburgstr. 2
76227 Karlsruhe
Germany

Internet: http://www.living-e.de
E-mail: info@living-e.de