

# Kentico CMS 5.5 R2 E-commerce Guide



# Table of Contents

<b>E-commerce Guide</b>	<b>7</b>
..Kentico CMS E-commerce module.....	7
<b>Getting Started</b>	<b>9</b>
..Making your first purchase.....	9
...Adding new products.....	14
...Adding product options.....	18
...Customizing product categories.....	22
...Adding product to multiple categories.....	26
...Displaying featured products.....	30
...Sorting and paging products.....	33
...Changing company details.....	37
<b>Customizing the product fields</b>	<b>44</b>
...Defining a new product type.....	44
...Customizing product design.....	51
...Adding images to product gallery.....	54
...Adding product custom fields.....	57
<b>Customizing web site design</b>	<b>62</b>
...Changing the header and menu (master page).....	62
...Changing colors using CSS styles.....	71
<b>Integration with your existing Kentico CMS web site</b>	<b>75</b>
...Overview.....	75
...Web parts.....	76
My account .....	76
Random products .....	78
Shopping cart .....	82
Shopping cart preview .....	85
Similar products by sale .....	87
Top N products by sale .....	88
Wishlist .....	90
...Displaying and resizing images.....	92
Overview .....	92
Displaying images .....	92
Storing images .....	96
Resizing images .....	96
...Tools.....	99

Overview .....	99
BizForms .....	100
Blogs .....	101
Booking system .....	102
Content staging .....	103
Event calendar .....	104
File import .....	105
Forums .....	106
Geo mapping .....	108
Image Gallery .....	108
Messaging .....	109
Newsletters .....	110
Polls .....	111
Reporting .....	112
RSS Feeds .....	113
User Contributions .....	113
Web Analytics .....	114
Web Farm synchronization .....	115
.....Adding items to the shopping cart.....	116
.....Adding items to the wish list.....	119
.....Displaying product price.....	120
.....Getting product URL.....	121
.....Searching.....	122
<b>Configuration settings</b> .....	<b>125</b>
.....Configuration overview.....	125
.....Orders.....	126
.....Customers.....	127
.....Products.....	129
.....Product options.....	131
.....Manufacturers.....	132
.....Suppliers.....	132
.....Discount coupons.....	132
.....Discount levels.....	133
.....Reports.....	133
.....Configuration.....	134
Store settings .....	134
Departments .....	135
Shipping options .....	135
Payment methods .....	136
Tax classes .....	137
Currencies .....	137
Exchange rates .....	138
Order status .....	138
Public status .....	139
Internal status .....	139
Invoice .....	139

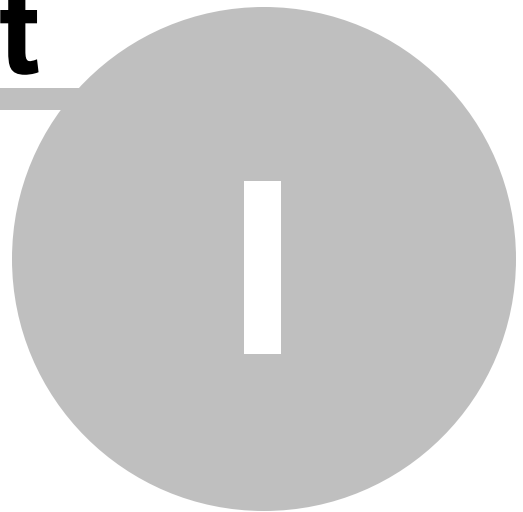
.....Enabling the e-commerce module.....	140
.....Security configuration.....	142
.....Countries and states.....	142
.....E-commerce and multi-site configuration.....	143
.....Mapping document fields to product properties.....	144
.....Sitemanager settings.....	144
.....Web.config settings.....	145
.....Localization settings.....	147
.....Customizing invoice and e-mail templates.....	149
.....Using product inventory.....	155
.....Discounts overview.....	160
.....Deleting old shopping carts.....	166
<b>Purchase process and payment gateways</b>	<b>168</b>
.....Purchase process overview.....	168
.....Customizing the purchase process.....	169
.....Developing custom dialogs for the checkout process.....	172
.....Payment gateways.....	178
.....Authorize.NET configuration.....	179
.....PayPal configuration.....	180
.....Developing custom payment gateways.....	183
.....Payment results.....	189
.....Customer credit.....	191
<b>Developing custom providers</b>	<b>194</b>
.....Overview.....	194
.....Using custom providers.....	195
.....CustomAddressInfoProvider.....	197
.....CustomCurrencyInfoProvider.....	197
.....CustomCustomerInfoProvider.....	199
.....CustomDepartmentInfoProvider.....	199
.....CustomDiscountCouponInfoProvider.....	200
.....CustomDiscountLevelInfoProvider.....	201
.....CustomExchangeTableInfoProvider.....	202
.....CustomInternalStatusInfoProvider.....	203
.....CustomManufacturerInfoProvider.....	203
.....CustomOptionCategoryInfoProvider.....	204
.....CustomOrderInfoProvider.....	205
.....CustomOrderItemInfoProvider.....	206
.....CustomOrderStatusInfoProvider.....	207

---

.....CustomOrderStatusUserInfoProvider.....	207
.....CustomPaymentOptionInfoProvider.....	208
.....CustomPublicStatusInfoProvider.....	209
.....CustomShippingOptionInfoProvider.....	210
.....CustomShoppingCartInfoProvider.....	210
.....CustomShoppingCartItemInfoProvider.....	212
.....CustomSKUInfoProvider.....	212
.....CustomSupplierInfoProvider.....	214
.....CustomTaxClassInfoProvider.....	214
.....CustomVolumeDiscountInfoProvider.....	216
.....Using the Product datalist web part.....	216
<b>Figure A: Tax calculation process</b>	<b>218</b>
<b>Figure B: Discount calculation</b>	<b>220</b>
<b>Figure C: Shopping cart price calculation</b>	<b>222</b>
<b>Figure D: Shopping cart retrieval</b>	<b>224</b>

**Part**

---



**E-commerce Guide**

---

# 1 E-commerce Guide

## 1.1 Kentico CMS E-commerce module

The Kentico CMS E-commerce module provides you with a range of out-of-box solutions for your E-business. The task of developing and managing the e-business website, once painfully slow and complex, has been diminished to little more than few clicks.

For customers, the built-in E-commerce module offers a possibility of making purchase online through the integrated shopping cart, checking orders status, subscribing to a newsletter etc.

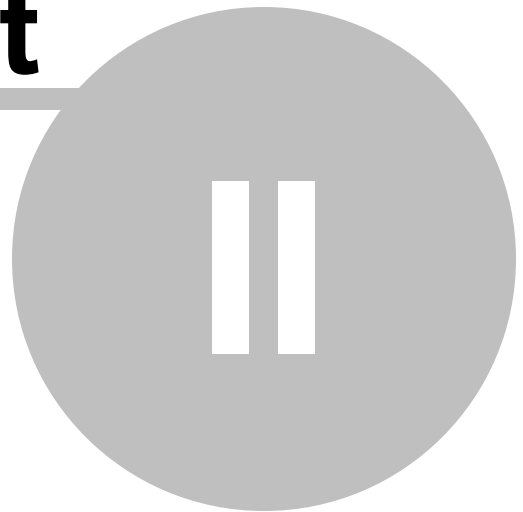
For site owners, the module offers tools for managing orders, shipping and payment options, products or manufactures lists and much more.

It supports:

- Product management
- Product options management
- Custom product types
- Product image gallery
- Product inventory
- Order management
- Customer management
- Multiple currencies
- Configurable tax calculation based on country and state
- Discount coupons
- Discount levels
- Volume discounts
- Wish list
- Custom providers for alternative shipping and tax calculations
- Custom checkout process
- Built-in payment processors, such as PayPal, Authorize.NET
- Custom payment processors
- Reports and statistics

# Part

---



**Getting Started**

---



## 2 Getting Started

### 2.1 Making your first purchase

For start, you will learn how to buy a product on your website. This experience will help you in revealing any possible pitfalls which might be awaiting your customers on your website.

1. Open your favorite internet browser and go to your website. Go to the **Products** tab and click the **Add to cart** button below **Nokia N82**.

The screenshot shows an e-commerce website interface. At the top, there's a navigation bar with 'Home', 'News', 'Products' (highlighted with a red box), 'How to buy', 'Company', and 'Silverlight'. Below the navigation bar, there's a search bar and a 'Search' button. The main content area is titled 'Electronics for you' and displays a list of products. The 'Products' section includes filters for Status, Manufacturer (Nokia), and Only in stock. Below the filters, there are two product listings: 'Nokia 6300' for \$129.00 and 'Nokia N82' for \$490.00. The 'Add to cart' button for the Nokia N82 is highlighted with a red box. The page also features a sidebar with 'Products' categories, a 'Poll' section, and 'Quick links'. At the bottom, there's a 'POWERED BY Kentico' logo.

2. Now specify your order. Choose **Black** as **Color** and enter **4** into the text box next the **Add to cart** button. You are about to add 4 pieces of Nokia N82 in black color to your shopping cart. Click **Add to cart**.

## Nokia N82



### Parameters:

Width: 112  
 Height: 50  
 Weight(g): 114  
 Display type: TFT 16M colors  
 Display resolution: 240 x 320  
 Bluetooth: yes  
 IrDA: no  
 GPRS: yes  
 EDGE: yes  
 HSCSD: yes  
 3G: yes  
 Wi-Fi: yes  
 Java: yes  
 Built-in camera: yes  
 MP3 player: yes

Our price: \$563.50  
 Without tax: \$490.00

#### Nokia N82 - Color

- Silver (+ \$0.00)  
 Black (+ \$10.00)

Total price (without tax): \$500.00

Add to wishlist

4 Add to cart

3. You have been redirected to the shopping cart. In the **Step 1**, click the first **Price detail** button.

Step 1 of 6 - Add some products to the shopping cart

### Shopping cart

Currency: U.S. Dollar

Remove	Product name	Units	Unit price	Unit discount	Tax	Subtotal
<input type="checkbox"/>	<a href="#">Nokia N82</a>	4	490.00	29.40	276.36	2118.76
	- Black	4	10.00	0.00	6.00	46.00

If you have a coupon code, please enter it here:

Total shipping: \$0.00  
**Total price: \$2164.76**

4. Now you can see detailed description of the price of your order including the volume discount and taxes. Close the window.

**Product price detail**

**Nokia N82**

Unit price without tax	\$490.00
------------------------	----------

Discounts	Per unit
- Volume discount for minimal amount of 3 units	-6%    -\$29.40
Price after discount	\$460.60

Taxes	Per unit
- Sales tax	+15%    +\$69.09
Price with tax	\$529.69

Total	
Units	4
Total price	\$2118.76

Close

http://localhost/Manuals/C Local intranet | Protected Mode: Off

5. Now click the **Check out** button at the bottom.

Step 1 of 6 - Add some products to the shopping cart

Shopping cart

Currency: U.S. Dollar

Remove	Product name	Units	Unit price	Unit discount	Tax	Subtotal
<input type="checkbox"/>	<a href="#">Nokia N82</a>	4	490.00	29.40	276.36	2118.76
	- Black	4	10.00	0.00	6.00	46.00

If you have a coupon code, please enter it here:

Total shipping: \$0.00  
Total price: **\$2164.76**

Empty


Continue shopping

6. In the **Step 2**, choose **Create a new account**. Now enter information for your new account:

- **First name:** David
- **Last name:** Simons
- **E-mail(user name):** DavidSimons@example.com

Click **Next**.

Step 2 of 6 - Registration check



**User registration**

Sign in using your existing account

Create a new account

First name: David

Last name: Simons

E-mail (user name): DavidSimons@example.com

Company account:

Password: ●●●●

Confirm password: ●●●●

Continue as anonymous customer


Back Next

7. In the **Step 3**, enter the new billing address:

- **Name (Company or personal):** Development Soft
- **Address Line:** 1020 Blueberry Ln.
- **City:** Tucson
- **ZIP:** 85754

Please note that you can choose the shipping or company address different from the billing address by checking one of the check boxes at the bottom. For now, just click **Next**, though.

Step 3 of 6 - Select billing and shipping address



**Billing address**

Billing address: (new) ▼

Name (company or personal): Development Soft

Address lines: 1020 Blueberry Ln.

City: Tuscon

ZIP: 85754

Country: USA ▼

Alaska ▼


Phone number:

**Shipping address**

My shipping address is different from the billing address.

8. In the **Step 4**, you can choose the shipping and payment method for your delivery. Click **Next**.

Step 4 of 6 - Select payment and shipping methods




**Shipping and payment methods**

Shipping: DHL ▼

Payment: Cash on delivery ▼

9. In the **Step 5**, make sure that you all the order information are correct. If everything looks right, click **Order now**.

Step 5 of 6 - Order preview



**Order preview**

Billing address  
Development Soft  
1020 Blueberry Ln.  
Tuscon  
85754  
USA, Alaska

Shipping address  
Development Soft  
1020 Blueberry Ln.  
Tuscon  
85754  
USA, Alaska

Payment method: Cash on delivery      Shipping option: DHL

Product name	Units	Unit price	Unit discount	Tax	Subtotal
Nokia N82	4	490.00	29.40	276.36	2118.76
- Black	4	10.00	0.00	6.00	46.00

Shipping: \$0.00  
**Total price: \$2164.76**

Tax name	Tax summary
Sales tax	282.36

Order note:

If everything went well, you can see the confirmation text.

## Thank you for your order

We have received your order and we will process it within 24 hours.

*This is a sample page showing the confirmation text. You can configure your payment methods in **Tools -> E-commerce -> Configuration -> Payment methods** so that the user is redirected to some third-party on-line payment gateway, such as Authorize.NET, PayPal and others.*

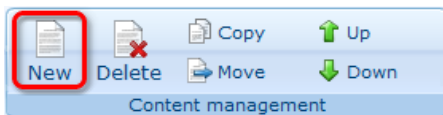
Congratulations, you've just made your first order.

## 2.2 Adding new products


1. Sign in to **CMS Desk**.
2. Go to **CMS Desk -> Products -> PDAs**.

The screenshot shows the Kentico CMS Desk interface for an 'E-commerce starter site'. The top navigation bar includes 'Content', 'My desk', 'Tools', and 'Administration'. The document action toolbar at the top left contains buttons for 'New', 'Delete', 'Move', 'Up', and 'Down', with the 'New' button highlighted by a red box. The left sidebar shows a tree view of the site structure, with 'PDAs' selected under the 'Products' category. The main content area displays a product list for 'PDAs' with filters for status, manufacturer, and pagination. Three products are listed: 'Asus A639' (\$470.00), 'HP IPAQ 114' (\$389.00), and 'HP IPAQ hx2795 Pocket PC' (\$415.00). Each product has an 'Add to cart' button and 'Edit'/'Delete' options. A shopping cart notification in the top right corner indicates 'Your shopping cart is empty'.




















3. Click  **New** at the document action toolbar.




4. Choose the **Product - PDA** document type.

 **New document**

Please select new document type:


-  [Page \(menu item\)](#)
-  [Article](#)
-  [Blog](#)
-  [Event \(booking system\)](#)
-  [FAQ](#)
-  [File](#)
-  [Folder](#)
-  [Image gallery](#)
-  [Job opening](#)
-  [News](#)
-  [Office](#)
-  [Press release](#)
-  [Product - Camera](#)
-  [Product - Cell phone](#)
-  [Product - Computer](#)
-  [Product - Laptop](#)
-  [Product - MP3 & MediaPlayer](#)
-  [Product - PDA](#)
-  [Simple article](#)

---

 [Link an existing document](#)

5. Adding new product is two step process. Firstly, you enter attributes for the new document. Secondly, you enter attributes for the new product. For explanation about the difference between these two please see the box at the end of the [Products](#) chapter. Now enter attributes for the new document:

- **Name:** HTC Touch Cruise
- **Battery:** Lithium ion
- **Display type:** TFT LCD
- **Resolution:** 240x320
- **RAM(MB):** 128
- **Processor(MHZ):** 400
- **Weight(g):** 200
- **Operating system:** Microsoft Windows Mobile 6

Now enter attributes for product itself. Check the **Create a new product** checkbox and enter 900 as **Price**. Click  **Save** at the top to save your new product.



---

Name:   
 Battery:   
 Display type:   
 Resolution:   
 RAM (MB):   
 Processor (MHz):   
 Weight(g):   
 Operating system:   
 Publish from:    
 Publish to:

Create a new product  
 Price:   
 Product image: Upload:    
 Description: 

B I [List Icons]

Department:

The new camera has been added to your product list.

## ■ HTC Touch Cruise



No image

### Parameters:

Battery: Lithium ion  
 Display type: TFT LCD  
 Resolution: 240×320  
 RAM (MB): 128  
 Processor (MHz): 400  
 Weight(g): 200  
 Operating system: Microsoft Windows Mobile 6

**Our price: \$900.00**  
 Without tax: \$900.00

1



### Creating product from existing document

This chapter describes how to create a new product while creating new document. In **Kentico CMS**, you can also choose to create product from an already existing document, though.

All you have to do is to select a given document in the content tree and switch to the **Product** tab. Then check **Mark document as product**. By choosing **Select an existing product**, you can use an existing product as a template for a new product. By choosing **Create a new product**, you can create a new one from scratch.

## 2.3 Adding product options

In Kentico CMS, product options can be defined for each product. With specified product option for the given product, you can offer your customers greater variability in choosing the right product and boost your sales as well. In this chapter, you will learn how to set up a new product option for the product which hasn't had any product option specified so far.

1. Go to **CMS Desk -> Tools -> E-commerce -> Product options**.

Actions	Name	Selection type	Enabled
	Cell phones accessories	Checkboxes - horizontal	Yes
	iPod Shuffle - Colors	Radiobuttons - vertical	Yes
	Nokia 6300 - Color	Dropdown list	Yes
	Nokia N82 - Color	Radiobuttons - vertical	Yes
	Notebooks - AMD CPU	Dropdown list	Yes
	Notebooks - DDR2 modules	Dropdown list	Yes
	Notebooks - HDD	Dropdown list	Yes
	Notebooks - Intel CPU	Dropdown list	Yes
	Operating systems	Dropdown list	Yes
	PCs - Cases	Dropdown list	Yes
	PCs - CD/DVD ROM/RW, BlueRay	Dropdown list	Yes
	PCs - CPU Intel	Dropdown list	Yes
	PCs - Graphics card	Dropdown list	Yes
	PCs - HDD	Dropdown list	Yes
	PCs - Memory modules	Dropdown list	Yes
	PCs - Monitors	Dropdown list	Yes
	PCs - Motherboards	Dropdown list	Yes
	PCs - Sound cards	Dropdown list	Yes
	Samsung SGH i620 - Color	Dropdown list	Yes
	Special PC accessories	Checkboxes - horizontal	Yes

2. Click **New option category**.
3. Enter *Color* into the **Display name** and **Code name** text boxes and click **OK**.

[Option categories](#) ▶ New option category

Display name:

Code name:

4. Select **RadioButtons in vertical layout** from the **Selection type** drop-down menu.

[Option categories](#) ▶ Color

[General](#) [Options](#)

Display name:

Code name:

**Selection type:**

Display price:

Default option(s):

- Black (+ \$0.00)
- Silver (+ \$10.00)
- Green (- \$10.00)

Description:

Default record text:

Enabled:

5. Switch to the **Options** tab and click **New product option**.

[Option categories](#) ▶ Color

[General](#) [Options](#)

6. Enter *Black* into the **Product name** text box, *0* into the **Price** text box and choose **General** from the **Department** drop-down menu. Then click **OK** at the bottom.

[Product options](#) ▶ New product option

Product name:	<input type="text" value="Black"/>
Enabled:	<input checked="" type="checkbox"/>
Product number:	<input type="text"/>
Description:	<div style="border: 1px solid gray; padding: 5px; min-height: 150px;">[Rich text editor area]</div>
Price:	<input type="text" value="0"/>
Department:	<input type="text" value="General"/>
Manufacturer:	<input type="text" value="(none)"/>
Supplier:	<input type="text" value="(none)"/>
Public status:	<input type="text" value="(none)"/>
Internal status:	<input type="text" value="(none)"/>
Availability (days):	<input type="text"/>
Product image URL:	Upload: <input type="text"/> <input type="button" value="Browse..."/>
<b>Package</b>	
Package weight:	<input type="text"/>
Package height:	<input type="text"/>
Package width:	<input type="text"/>
Package depth:	<input type="text"/>
<b>Inventory</b>	
Available items:	<input type="text"/>
Sell only if items are available:	<input type="checkbox"/>
<input type="button" value="OK"/>	

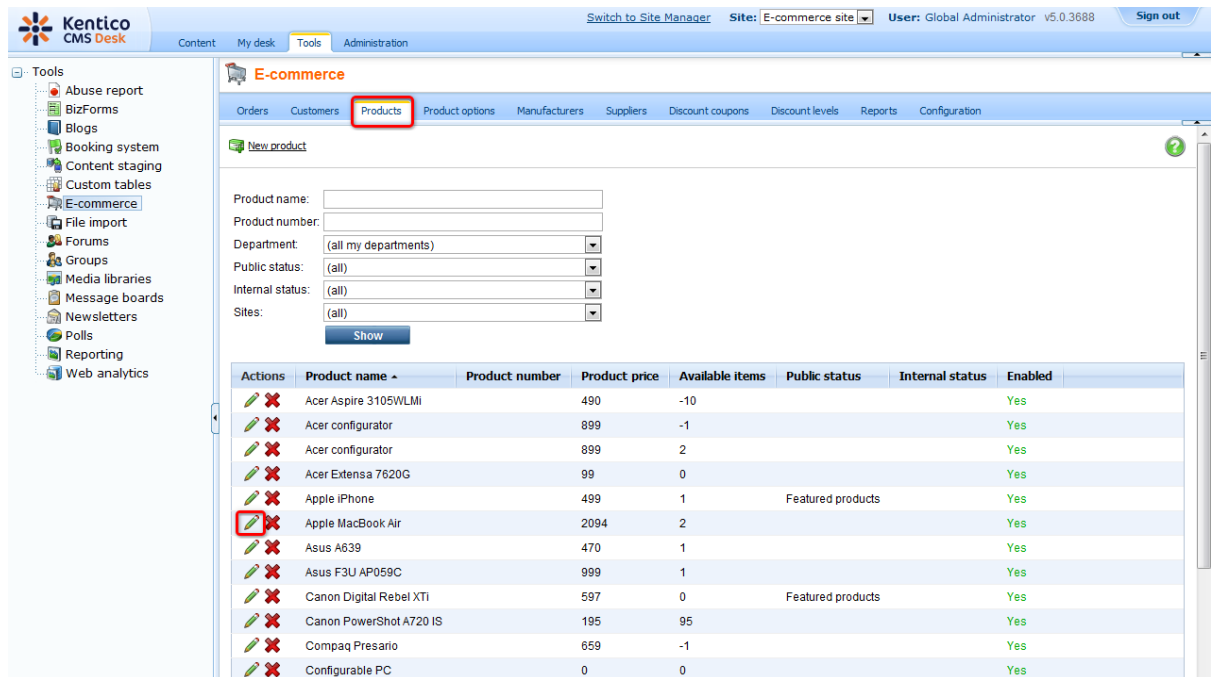
6. Click the **Options** tab and then click **New product option** again.

7. Now enter *Silver* as **Product name**, *10* as **Price** and choose **General** as **Department**. Then click **OK**.

8. Click the **Options** tab and **New product option** again.

9. Enter *Green* as **Product name**, *-10* as **Price** and choose **General** as **Department**. Then click **OK**.

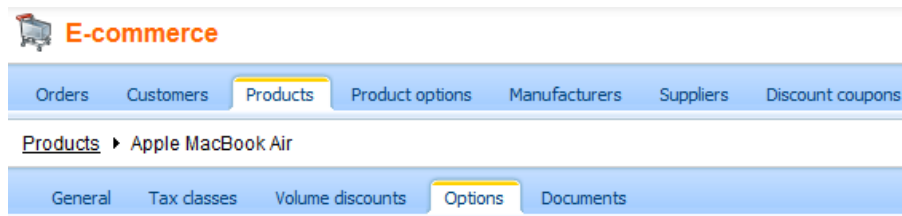
10. Now switch to **Products** tab and click the **Edit** button next to the **Apple MacBook Air** product name.



The screenshot shows the Kentico CMS Desk interface. The top navigation bar includes 'Content', 'My desk', 'Tools', and 'Administration'. The 'E-commerce' section is active, with sub-tabs for 'Orders', 'Customers', 'Products' (highlighted with a red box), 'Product options', 'Manufacturers', 'Suppliers', 'Discount coupons', 'Discount levels', 'Reports', and 'Configuration'. A 'New product' form is visible with fields for Product name, Product number, Department, Public status, Internal status, and Sites. Below the form is a table of products:

Actions	Product name	Product number	Product price	Available items	Public status	Internal status	Enabled
	Acer Aspire 3105WLMi		490	-10			Yes
	Acer configurator		899	-1			Yes
	Acer configurator		899	2			Yes
	Acer Extensa 7620G		99	0			Yes
	Apple iPhone		499	1	Featured products		Yes
	Apple MacBook Air		2094	2			Yes
	Asus A639		470	1			Yes
	Asus F3U AP059C		999	1			Yes
	Canon Digital Rebel XTi		597	0	Featured products		Yes
	Canon PowerShot A720 IS		195	95			Yes
	Compaq Presario		659	-1			Yes
	Configurable PC		0	0			Yes

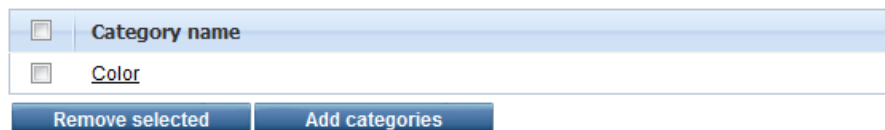
11. Switch to the **Options** tab and click the **Add categories** button. In the pop-up window, select **Color** product option category and click **OK**.



The screenshot shows the 'E-commerce' interface with the 'Products' tab selected. The breadcrumb path is 'Products > Apple MacBook Air'. The 'Options' sub-tab is active, showing tabs for 'General', 'Tax classes', 'Volume discounts', 'Options', and 'Documents'.

The changes were saved.

The following categories with product options are assigned to the product:



The pop-up window displays a list of assigned categories with checkboxes:

- Category name
- Color

Buttons at the bottom include 'Remove selected' and 'Add categories'.

The new product option has been added to the **Apple MacBook Air** product.

Sign in to [CMS Desk](#). Sign in to [CMS Site Manager](#). The default account is administrator with blank password.

E-commerce starter site

Electronics for you

Shopping cart | My account | My wishlist  
Your shopping cart is empty

Home News Products How to buy Company Silverflight Global Administrator Sign out

Products / Notebooks / 13" / Apple MacBook Air

Products

- » Cameras
- » Cell phones
- » MP3 Players
- » Notebooks
  - » 13"
  - » 14"
  - » 15"
  - » 17"
- » PDAs
- » PCs

Poll

How did you find this site?

Referred by a friend

Print advertising

Link from another web site

Search engine

Vote

Quick links

- » [How to buy](#)
- » [Company](#)
- » [News](#)
- » [Home](#)

Apple MacBook Air

Parameters:

Processor type: Intel Core 2 Duo processor with 4MB on-chip shared L2

Display type: 13.3-inch widescreen TFT

Resolution: 1280x800

Graphics card: Intel GMA X3100 144MB

Memory type: 667MHz DDR2 SDRAM onboard

Memory size: 2048

Optical drive:

Hard Drive: 80GB 4200-rpm Parallel ATA

Wireless LAN: yes

Bluetooth: yes

Infrared: no

Battery type: Integrated 37-watt-hour lithium-polymer battery

Weight(g): 1.36

Operating system: Mac OS

Accessories:

Our price: **\$2094.00**  
Without tax: \$2094.00

Color

Black (+ \$0.00)

Silver (+ \$10.00)

Green (- \$10.00)

Total price (without tax): **\$2094.00**

Add to wishlist

1 Add to cart

Latest news

2/13/2008

[Dear customers, our shop now supports payments through PayPal. You can use this feature from this day.](#)

2/13/2008

[Dear customers, our company has become Apple partner so you can buy Apple's great products at our shop.](#)

Newsletter subscription

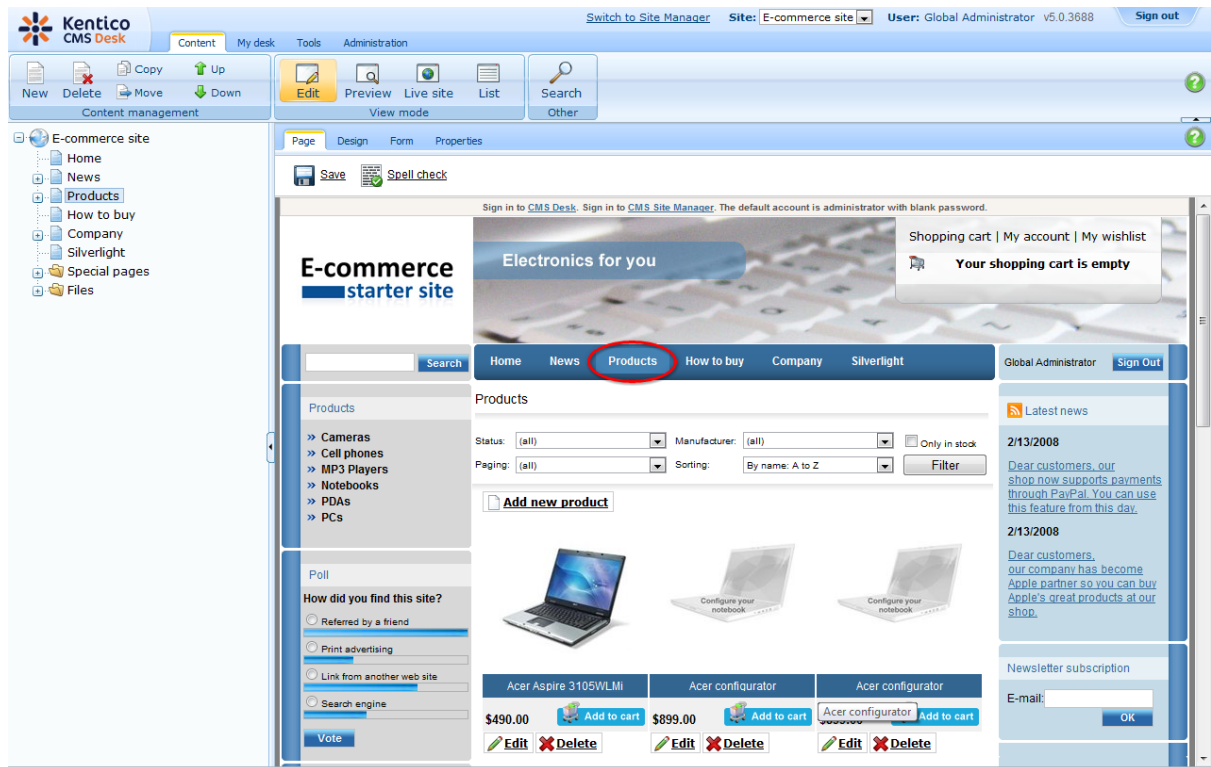
E-mail:

OK

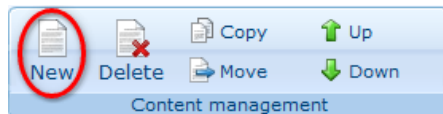
For information how to display product options in product detail please refer to the [Adding items to the shopping cart](#) chapter.

## 2.4 Customizing product categories


1. Go to **CMS Desk** -> **Content** and choose **Products** in the content tree.






















2. Click  **New** at the document action toolbar.




3. Choose the  **Page(menu item)** document type.

 **New document**

Please select new document type:

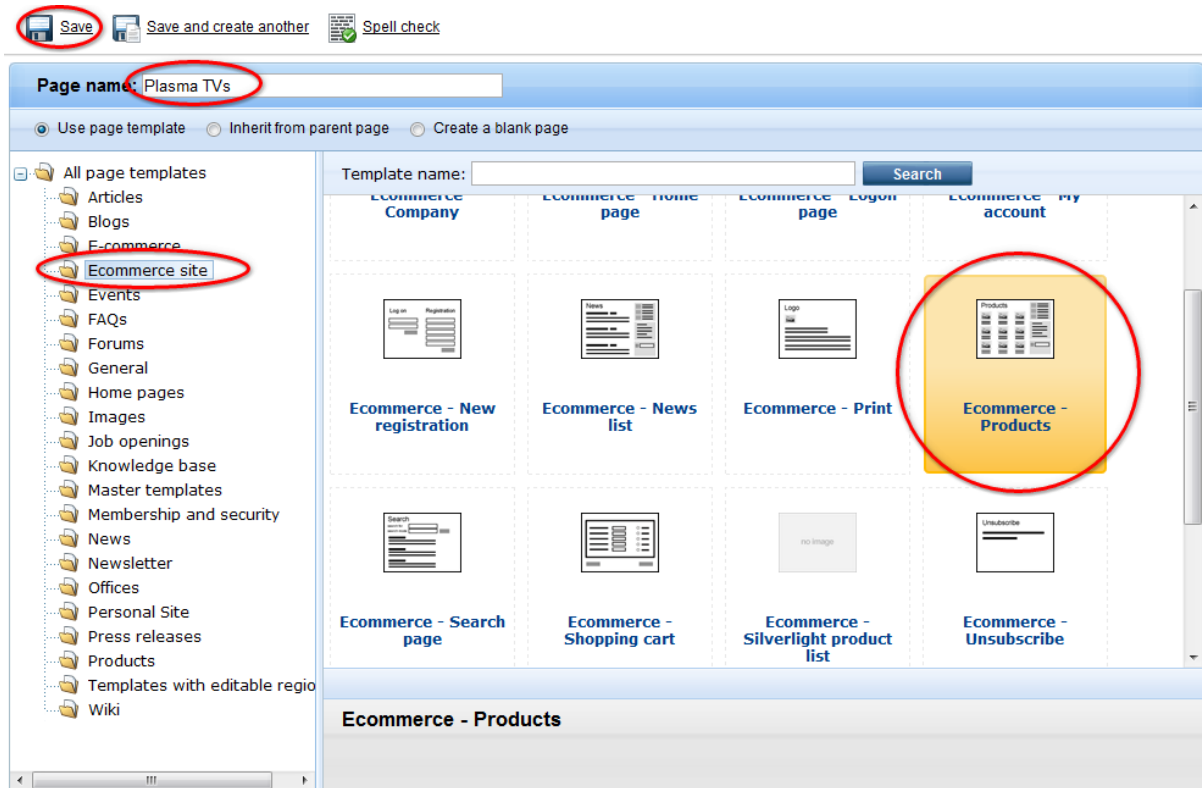
-  [Page \(menu item\)](#)
-  [Article](#)
-  [Blog](#)
-  [Event \(booking system\)](#)
-  [FAQ](#)
-  [File](#)
-  [Folder](#)
-  [Image gallery](#)
-  [Job opening](#)
-  [News](#)
-  [Office](#)
-  [Press release](#)
-  [Product - Camera](#)
-  [Product - Cell phone](#)
-  [Product - Computer](#)
-  [Product - Laptop](#)
-  [Product - MP3 & MediaPlayer](#)
-  [Product - PDA](#)
-  [Simple article](#)

---

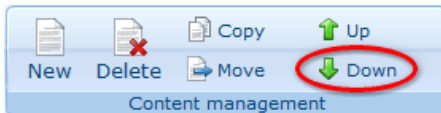
 [Link an existing document](#)

4. Enter *Plasma TVs* as the **Page name**, make sure the **Use page template** radio button is selected and choose **E-commerce site -> Ecommerce - Products** template. Click  **Save**.

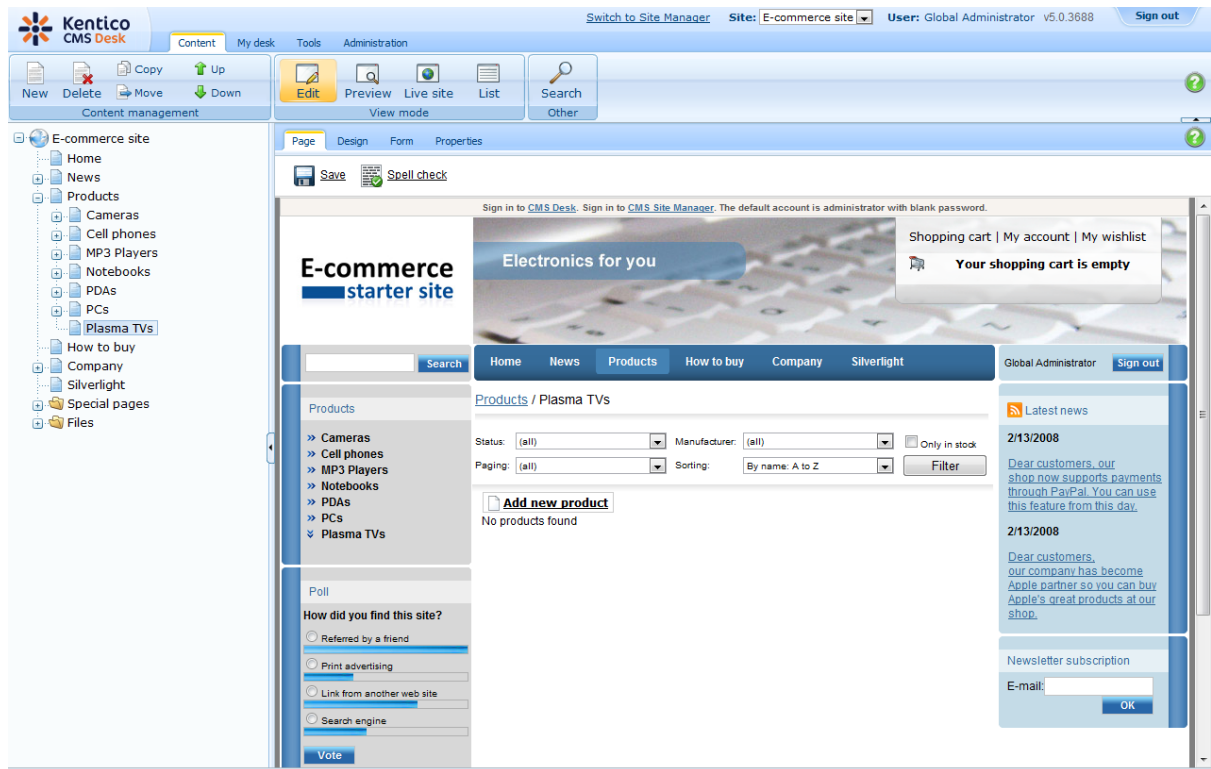




5. Your new page has been appended to the **Products** page. Now use the **Down** button at the document action toolbar to move it to the bottom of the section.



That's how you add a new category to your **Products** page.




## 2.5 Adding product to multiple categories

You might need to add some product into more than one category. This chapter will help you to learn how to add a product to multiple categories.




















1. Go to **CMS Desk** -> **Content** -> **Products** -> **PCs** and click  **New** at the document action toolbar.

The screenshot shows the Kentico CMS Desk interface for an e-commerce site. The top navigation bar includes 'Content', 'My desk', 'Tools', and 'Administration'. The 'Content management' toolbar has a 'New' button circled in red, along with 'Delete', 'Move', and 'Down' options. The main content area displays a product list for PCs, including Compaq Presario, Configurable PC, and Force PC, with prices and 'Add to cart' buttons. A poll and newsletter subscription form are also visible.


2. Click  Link an existing document at the bottom.

 **New document**

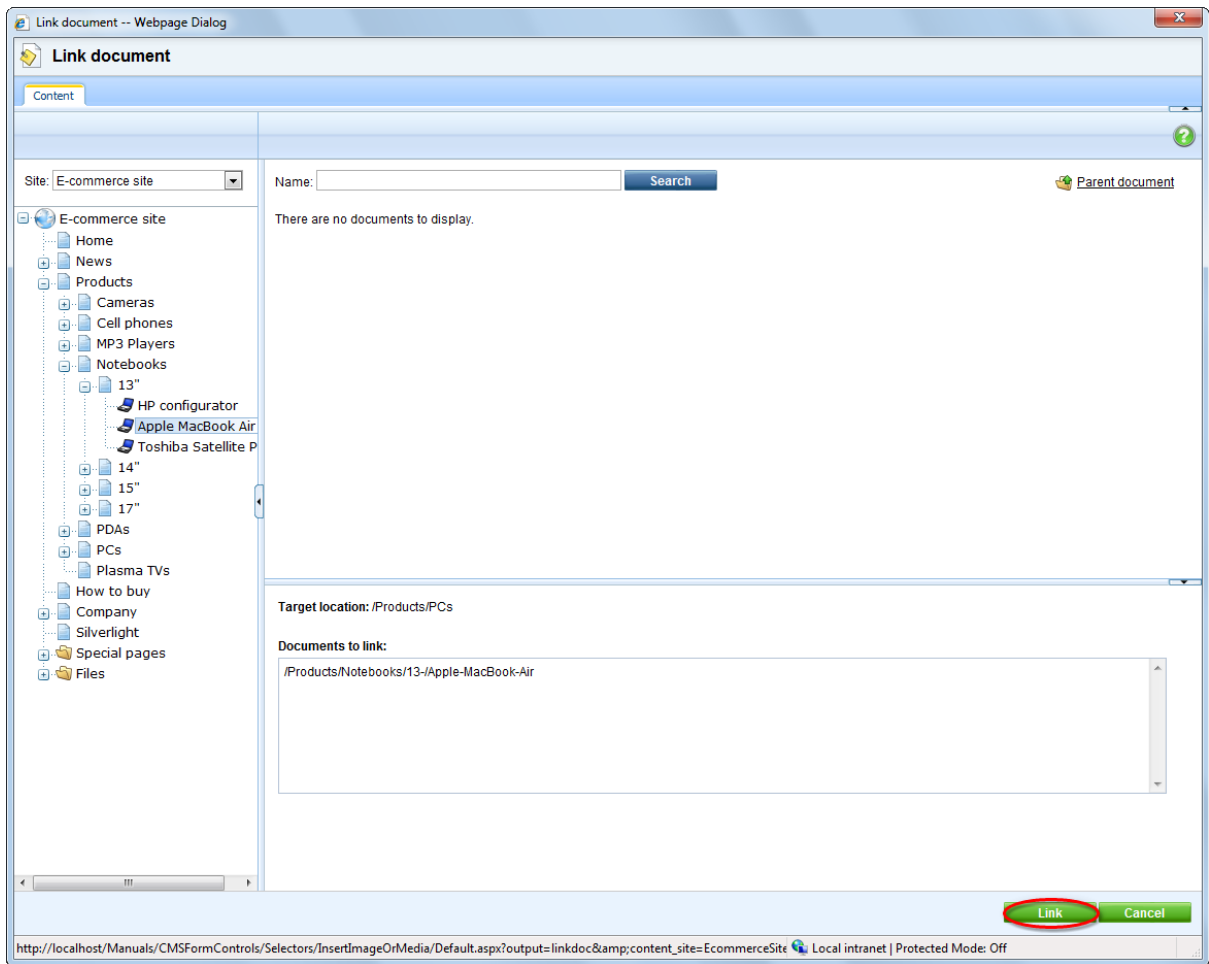
Please select new document type:

-  [Page \(menu item\)](#)
-  [Article](#)
-  [Blog](#)
-  [Event \(booking system\)](#)
-  [FAQ](#)
-  [File](#)
-  [Folder](#)
-  [Image gallery](#)
-  [Job opening](#)
-  [News](#)
-  [Office](#)
-  [Press release](#)
-  [Product - Camera](#)
-  [Product - Cell phone](#)
-  [Product - Computer](#)
-  [Product - Laptop](#)
-  [Product - MP3 & MediaPlayer](#)
-  [Product - PDA](#)
-  [Simple article](#)

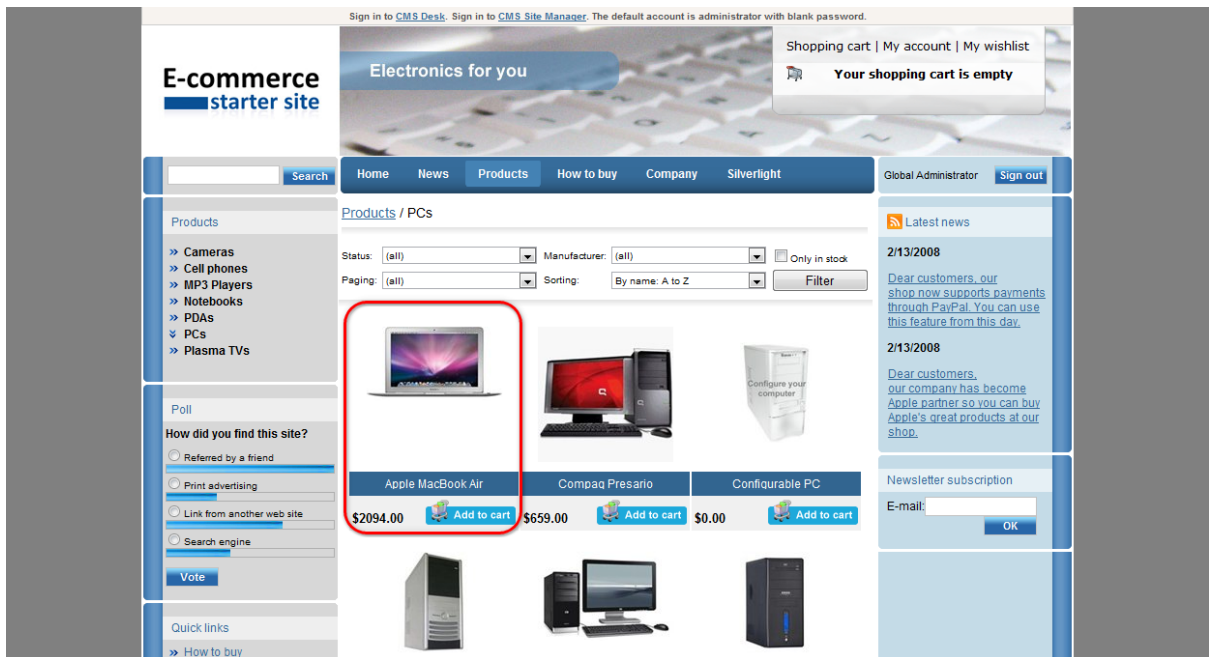
---

 [Link an existing document](#)

3. Now select **Apple MacBook Air** at **E-commerce Site -> Products -> Notebooks -> 13"** and click **Link**.



Apple MacBook Air has been added to the PCs category.



**Please note:**

- While copying a document, the same product (with the same SKU) that has been linked to the original document is linked to the new document as well.
- While creating link to an existing document, a product (SKU) is not directly related to the given link. The newly created link is an "image" of the existing document, so that the product is linked to the original document only.

## 2.6 Displaying featured products

1. Go to **CMS Desk** -> **Tools** -> **E-commerce** and choose **Products** tab.

The screenshot shows the Kentico CMS Desk interface. The top navigation bar includes 'Content', 'My desk', 'Tools', and 'Administration'. The 'Tools' menu is circled in red. Below it, the 'E-commerce' section is active, with 'Products' also circled in red. The 'New product' form is visible, and a table of products is shown below. The 'Apple iPhone' row is highlighted.

Actions	Product name	Product number	Product price	Available items	Public status	Internal status	Enabled
	Acer Aspire 3105WLMi		490	-10			Yes
	Acer configurator		899	-1			Yes
	Acer configurator		899	2			Yes
	Acer Extensa 7620G		99	0			Yes
	Apple iPhone		499	1	Featured products		Yes
	Apple MacBook Air		2094	2			Yes
	Asus A639		470	1			Yes
	Asus F3U AP059C		999	1			Yes
	Canon Digital Rebel XTi		597	0	Featured products		Yes
	Canon PowerShot A720 IS		195	95			Yes
	Compaq Presario		659	-1			Yes
	Configurable PC		0	0			Yes
	Creative ZEN		150	-6	Featured products		Yes
	Creative Zen Stone		50	0			Yes
	Emgeton M6 Cult		145	0			Yes

2. Click the **Edit** () button next to the product which you want to display as a featured product.

A close-up view of the product list from the screenshot. The 'Apple iPhone' row is highlighted, and the 'Edit' button (pencil icon) is circled in red.

	Acer Aspire 3105WLMi
	Acer configurator
	Acer configurator
	Acer Extensa 7620G
	Apple iPhone
	Apple MacBook Air
	Asus A639
	Asus F3U AP059C

3. From the **Public status** drop-down menu, choose **Featured products** and click **OK** at the bottom.

General Tax classes Volume discounts Options Documents

Product name:

Enabled:

Product number:

Description: 

iPhone is a revolutionary new mobile phone that allows you to make a call by simply tapping a name or number in your address book, a favorites list, or a call log. It also automatically syncs all your contacts from a Windows PC, Mac, or Internet service. And it lets you select and listen to voicemail messages in whatever order you want — just like email.

Price:

Department:

Manufacturer:

Supplier:

Public status:

Internal status:

Availability (days):

Product image URL:

**Package**

Package weight:

Package height:

Package width:

Package depth:

**Inventory**

Available items:

Sell only if items are available:

Your product has been added to the featured products.



Sign in to [CMS Desk](#). Sign in to [CMS Site Manager](#). The default account is administrator with blank password.

E-commerce starter site

Electronics for you

Shopping cart | My account | My wishlist  
Your shopping cart is empty

Search Home News Products How to buy Company Silverlight Global Administrator Sign Out

Welcome to E-commerce starter site

Special offer of notebooks

Our price: \$499.00  
Apple iPhone

Our price: \$597.00  
Canon Digital Rebel XT

Latest products

- Toshiba Satellite Pro \$1499.00
- Tiger Extreme PC \$999.00
- Sony VAIO VGN-CR320E \$1189.00

Latest news RSS

2/13/2008  
Dear customers, our shop now supports payments through PayPal. You can use this feature from this day.

2/13/2008  
Dear customers, our company has become Apple partner so you can buy Apple's great products at our shop.

Newsletter subscription

E-mail:  OK

On home page, by default there are displayed 4 products randomly chosen from all products with public status set to **Show as featured product**. Should you want set different filters for displaying products on you home page or should you need more information about the **Random products** web part, please refer to the [Random products](#) chapter.

## 2.7 Sorting and paging products

1. Go to **CMS Desk** -> **Products** and switch to the **Design** tab. Click the  **Configure** button of the **Product list** web part.

The screenshot shows the Kentico CMS 5.5 R2 E-commerce Guide interface. The 'Design' view is active, and the 'Products' section is highlighted in the left sidebar. The main content area displays a product list for 'Electronics for you' with filters for 'ProductFilter', 'Manufacturer', and 'Paging'. The 'Paging' section shows 'Paging: (all)' and 'Sorting: By name: A to Z'. The 'ProductList' section is also highlighted with a red circle. The page includes a search bar, navigation menu, and a shopping cart notification.

2. In the webpage dialog, find the **Content filter** section and enter *SKUPrice ASC* into the **ORDER BY expression** text box, then find the **Paging** section and enter *16* into the **Page size** checkbox. Now click **OK**.

The screenshot shows the 'Web part properties (Datalist)' dialog box in the CMS Desk. The 'General' tab is selected, and the 'ORDER BY expression' field is highlighted with a red box. The field contains the text 'SKUPrice ASC'. Other fields include 'Combine with default culture' (radio buttons for Yes, No, and Use site settings), 'Culture code' (text box with Select and Clear buttons), 'Maximum nesting level' (text box with -1), 'Select only published' (checkbox checked), 'Select top N documents' (text box), 'Site name' (text box with Select and Clear buttons), 'WHERE condition' (text box with (NodeSKUID IS NOT NULL) AND (SKUEnabled = 1), 'Columns' (text box), 'Filter out duplicate documents' (checkbox), and 'Filter name' (text box with ProductFilter). The 'Refresh content' checkbox is checked. The dialog has OK, Cancel, and Apply buttons at the bottom.

Web part properties (Datalist)

General Layout

Combine with default culture:  Yes  No  Use site settings

Culture code:  Select Clear

Maximum nesting level:

**ORDER BY expression:**

Select only published:

Select top N documents:

Site name:  Select Clear

WHERE condition:

Columns:

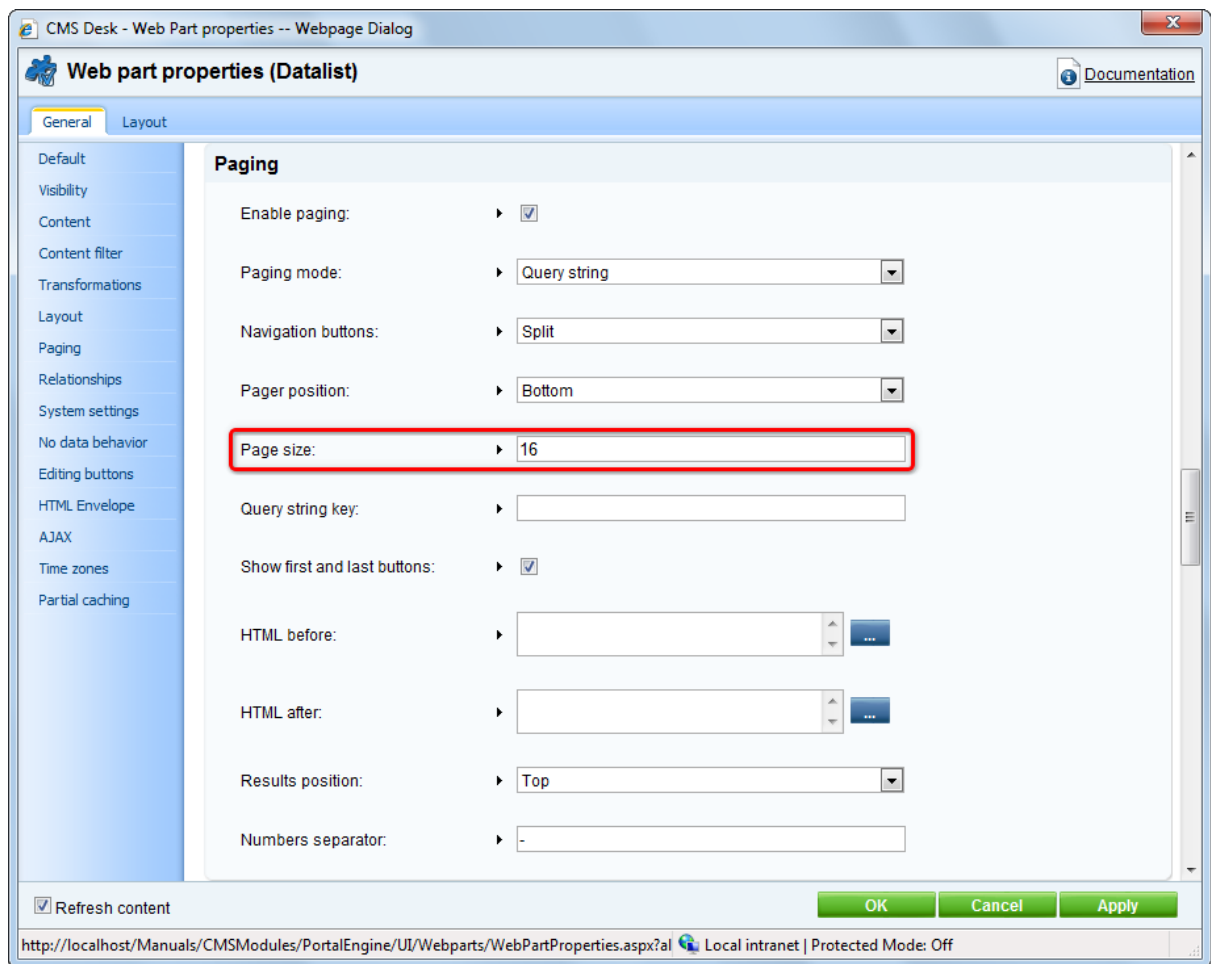
Filter out duplicate documents:

Filter name:

Refresh content

OK Cancel Apply

http://localhost/Manuals/CMSModules/PortalEngine/UI/Webparts/WebPartProperties.aspx?al Local intranet | Protected Mode: Off



3. Switch to the **Preview** viewing mode. You can see that the products are sorted from the cheapest to the most expensive and there are 16 of them.

Sign in to [CMS Desk](#). Sign in to [CMS Site Manager](#). The default account is administrator with blank password.

**E-commerce starter site**

Electronics for you

Shopping cart | My account | My wishlist  
Your shopping cart is empty

Home News Products How to buy Company Silverlight

Global Administrator [Sign out](#)

Search

Products

[» Cameras](#)  
[» Cell phones](#)  
[» MP3 Players](#)  
[» Notebooks](#)  
[» PDAs](#)  
[» PCs](#)  
[» Plasma TVs](#)

Products

Status: (all) Manufacturer: (all)  Only in stock

Paging: (all) Sorting: By price: low to high [Filter](#)

Configure your computer

ZEN

Configure your computer

Configurable PC \$0.00 [Add to cart](#)

Creative Zen Stone \$50.00 [Add to cart](#)

iPod Shuffle \$79.00 [Add to cart](#)

Palm Z22 Handheld \$93.00 [Add to cart](#)

Acer Extensa 7620G \$99.00 [Add to cart](#)

iRiver T60 \$117.00 [Add to cart](#)

Best selling products

Nokia N82 \$490.00

Latest news

2/13/2008

[Dear customers, our shop now supports payments through PayPal. You can use this feature from this day.](#)

2/13/2008

[Dear customers, our company has become Apple partner so you can buy Apple's great products at our shop.](#)

Newsletter subscription

E-mail:  [OK](#)

Quick links

[» How to buy](#)  
[» Company](#)  
[» News](#)  
[» Home](#)



### Sorting products

You may decide to sort products in a different way.

Should you want to sort products from the most expensive to the cheapest, you would enter *SKUPrice DESC* into the **ORDER BY expression** text box.

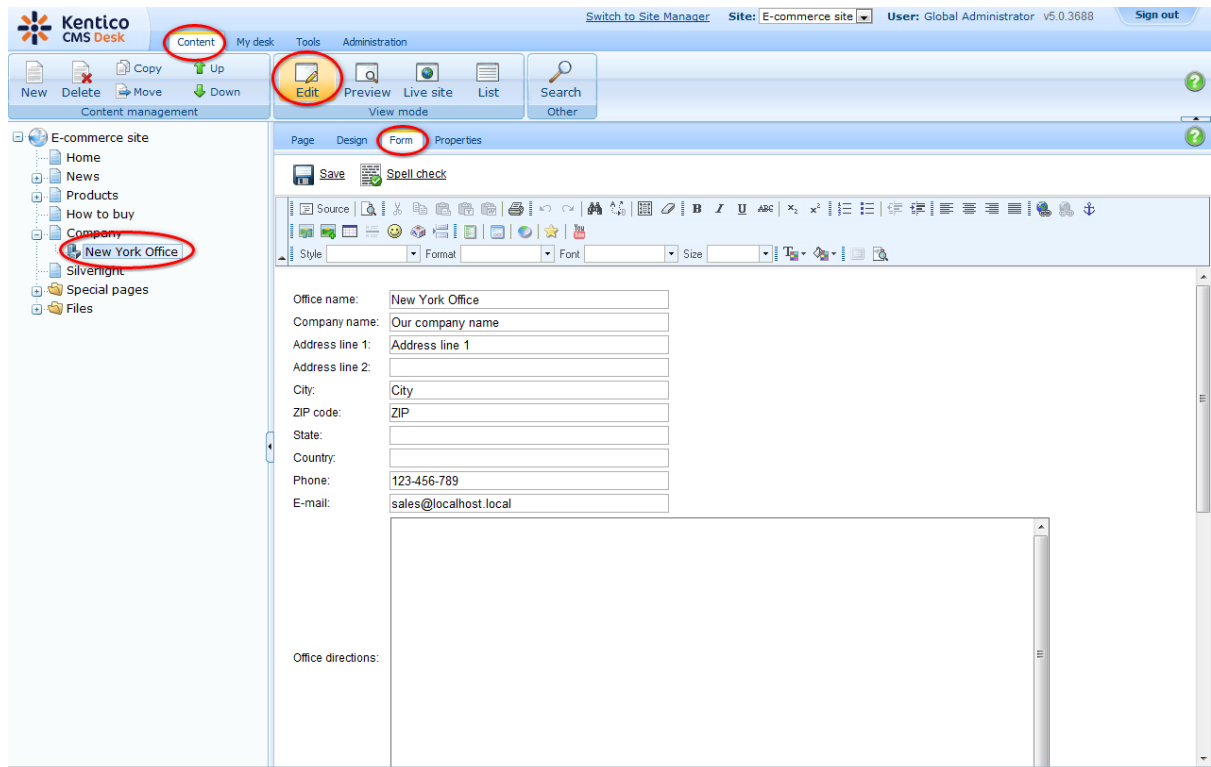
You can choose to sort products alphabetically as well. For sorting from A to Z, enter *NodeName ASC*. For sorting from Z to A, enter *NodeName DESC* into the **ORDER BY expression** text box.

## 2.8 Changing company details

In this chapter, you will find out how to change your company details and add a new office.

### Changing company details

1. Go to **CMS Desk -> Content -> E-commerce Site -> Company -> New York Office** and switch to the **Form** tab.



2. Enter your new company details:

- **Address line 1:** 316 W 49th St
- **City:** New York City
- **ZIP code:** 10019
- **State:** NY
- **Country:** United States

Then click **Save**.

Office name:

Company name:

Address line 1:

Address line 2:

City:

ZIP code:

State:

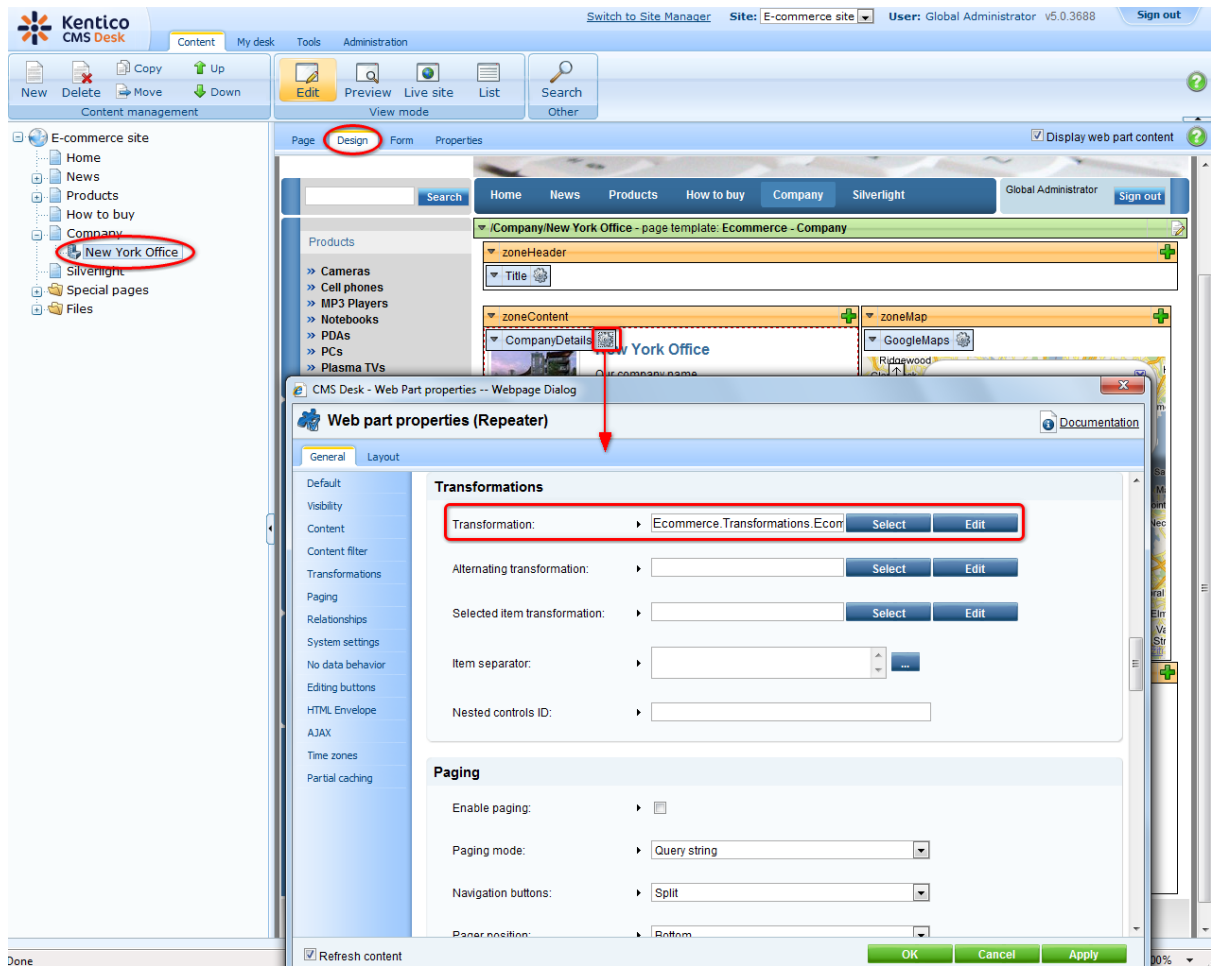
Country:

Phone:

E-mail:

Office directions:

**Please note:** You can change the transformation of the company detail at **CMS Desk -> Content -> E-commerce Site -> Company -> New York Office -> Design -> <edit CompanyDetails> -> Transformations -> Transformation.**



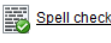









































## Creating new office

1. Go to **CMS Desk** -> **Content** -> **E-commerce Site** -> **Company** and click the **New** button from the document action toolbar.
2. Choose **Office** as the new document type.
3. Now enter details for your new office:
  - **Office name:** San Francisco Office
  - **Address line 1:** 23 New Montgomery St
  - **City:** San Francisco
  - **ZIP code:** 94105
  - **State:** CA
  - **Country:** United States

Then scroll down a little bit and click **Browse** to select an image as **Office photo**. Then click **Save**.



 Save  Save and create another  Spell check

Source |                                          

Sign in to [CMS Desk](#). Sign in to [CMS Site Manager](#). The default account is administrator with blank password.

E-commerce starter site

Electronics for you

Shopping cart | My account | My wishlist  
Your shopping cart is empty

Home News Products How to buy Company Silverlight Global Administrator Sign out

Search

Products

- » Cameras
- » Cell phones
- » MP3 Players
- » Notebooks
- » PDAs
- » PCs
- » Plasma TVs

Poll

How did you find this site?

Referred by a friend

Print advertising


Link from another web site

Search engine

Quick links

» [How to buy](#)


Our company

 **New York Office**

Our company name  
316 W 49th St  
New York City, NY 10019  
United States

Phone: 123-456-789  
E-mail: [sales@localhost.local](mailto:sales@localhost.local)

Opening hours: Monday to Thursday 9.00am - 6.00pm  
Friday 9.00am - 4.00pm

 **San Francisco Office**

New Montgomery St  
San Francisco, CA 94105  
United States

Phone:  
E-mail:

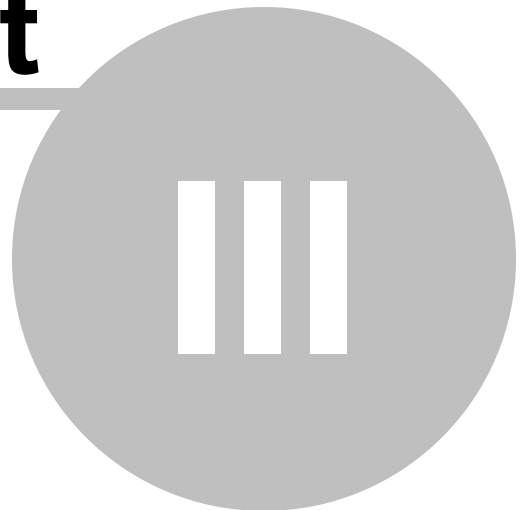
**Contact Us**

If you wish to contact us, please use this form:

**Please note:** In the **Form** tab, you can use the **Office description** text box to enter additional information about your office (e.g. opening hours). Should you feel this text box is not convenient for you (for instance you want to edit displayed text in the **Page** tab), you can always leave Office description blank and add the Text -> Editable text web part on the page instead.

# Part

---



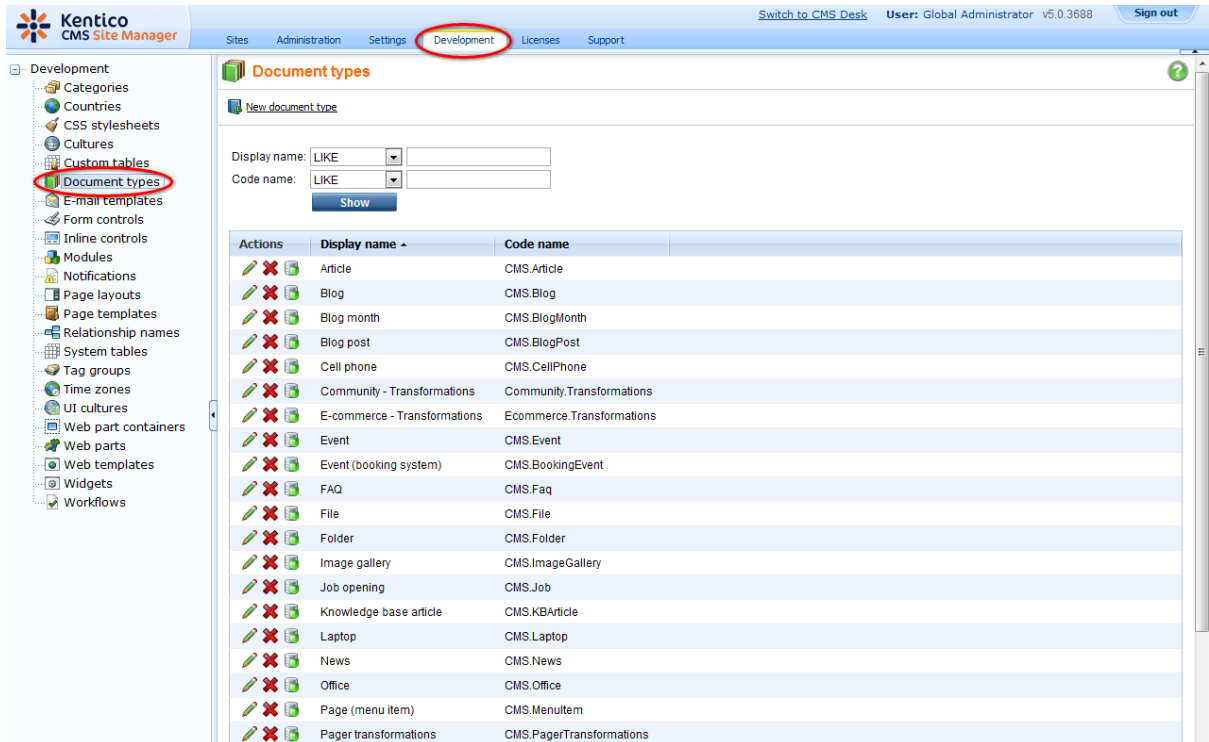
**Customizing the product fields**

---

## 3 Customizing the product fields

### 3.1 Defining a new product type

1. Go to Site Manager -> Development -> Document Types.



The screenshot shows the Kentico CMS Site Manager interface. The 'Development' menu item is circled in red. The 'Document types' menu item is also circled in red. The main content area shows the 'Document types' section with a 'New document type' link and a form for creating a new document type. The form has 'Display name' and 'Code name' fields, both with a dropdown menu set to 'LIKE' and a 'Show' button. Below the form is a table of existing document types.

Actions	Display name	Code name
	Article	CMS.Article
	Blog	CMS.Blog
	Blog month	CMS.BlogMonth
	Blog post	CMS.BlogPost
	Cell phone	CMS.CellPhone
	Community - Transformations	Community.Transformations
	E-commerce - Transformations	Ecommerce.Transformations
	Event	CMS.Event
	Event (booking system)	CMS.BookingEvent
	FAQ	CMS.Faq
	File	CMS.File
	Folder	CMS.Folder
	Image gallery	CMS.ImageGallery
	Job opening	CMS.Job
	Knowledge base article	CMS.KBArticle
	Laptop	CMS.Laptop
	News	CMS.News
	Office	CMS.Office
	Page (menu item)	CMS.Menuitem
	Pager transformations	CMS.PagerTransformations

2. Click the **New Document Type** link.

3. You have been redirected the **New document type** wizard.

In the **Step 1**, enter:

- **Document type display name:** Plasma TV
- **Document type code name:** CustomProduct (**namespace**), television (**document type**)

Click **Next**.

**Step 1** | **General**  
Please enter document type display name (for users) and code name (it will be used in your code when necessary).

Document type display name:

Document type code name:

[Next >](#)

4. In the **Step 2**, leave the default values and click **Next** again.

**Step 2** | **Data type**  
Please choose document data type. If you choose a document type with custom attributes you will also need to supply names of the new database table and its primary key.

The document type has custom fields

Table name:

Primary key name:

The document type is only a container without custom fields

[Next >](#)

5. In the **Step 3**, you are asked to define attributes for the new document type. Click **New Attribute (+)** and enter:

- **Attribute name:** TVName
- **Attribute size:** 100
- **Field caption:** Product name

From the **Field type** drop-down list, choose **Textbox** and click **OK**.

**Step 3** | **Fields**  
Please define custom attributes of the document type and their appearance in the editing form. You can define attributes, such as product number, product weight, press release text, etc.

The screenshot shows the 'Fields' configuration interface. On the left, a list of attributes is shown, with 'New attribute' selected. The main configuration area is divided into two sections: 'Database' and 'Field'. The 'Database' section contains the following fields: 'Attribute name' (TVName), 'Attribute type' (Text), and 'Attribute size' (100). The 'Field' section contains the following fields: 'Field caption' (Product name) and 'Field type' (Text box). A green 'OK' button is highlighted with a red circle at the bottom right of the configuration panel. A 'Next >' button is located at the bottom right of the entire interface.

6. Click **New Attribute** again. Enter:

- **Attribute name:** TVScreenType
- **Attribute size:** 100
- **Field caption:** Screen type

From the **Field type** drop-down list, choose **Textbox** and click **OK**.

**Step 3** | **Fields**  
Please define custom attributes of the document type and their appearance in the editing form. You can define attributes, such as product number, product weight, press release text, etc.

TelevisionID  
TVName  
New attribute

**Database**

Attribute name: TVScreenType

Attribute type: Text

Attribute size: 100

Allow empty value:

Attribute default value:

Display attribute in the editing form

**Field**

Field caption: Screen type

Field type: Text box

Field description:

OK

Next >

7. Now click **New Attribute** again and enter information for the last attribute:

- **Attribute name:** TVScreenSize
- **Attribute type:** Integer number
- **Field caption:** Screen size in inches

From the **Field type** drop-down list, choose **Textbox** and click **OK**.

**Step 3** | **Fields**  
Please define custom attributes of the document type and their appearance in the editing form. You can define attributes, such as product number, product weight, press release text, etc.

TelevisionID  
TVName  
TVScreenType  
New attribute

**Database**  
Attribute name: TVScreenSize  
Attribute type: Integer number  
Attribute size:  
Allow empty value:   
Attribute default value:

Display attribute in the editing form

**Field**  
Field caption: Screen size in inches  
Field type: Text box  
Field description:

OK

Next >

8. In the **Step 4**, choose *TVName* in the **Document name source** drop-down list. Click **Next**.

**Step 4** | **Additional settings**  
Please choose the source field that will be used as a document name. You can choose either one of the custom fields or you can choose to use document name as a separate field.

Document name source: TVName

Next >

9. In the **Step 5**, click the **Add document types** button and select the **Page (menu item)** document type. Click **Next**.



**Step 5** | **Parent types**  
Please select document types under which this document template can be placed.

<input type="checkbox"/>	Document type name
<input type="checkbox"/>	Page (menu item) (CMS.Menuitem)

10. Now make sure that you have assigned the document type to the Ecommerce site and click **Next**.

**Step 6** | **Sites**  
Please select sites where this document type can be used:

<input type="checkbox"/>	Site name
<input type="checkbox"/>	E-commerce site

11. In the **Step 7**, you are asked to specify how documents of this type will be indexed and displayed in the search results. For more information on these settings, please refer to the [Settings for particular object types](#) topic of Kentico CMS Developer's Guide. Make your choice and click **Next**.

**Step 7** | **Search options**  
Please set search fields for Smart search module.

Title field:

Content field:

Image field:

Date field:

Set automatically

Field name	Content	Searchable	Tokenized	Custom search name
TelevisionID	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
TVName	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text"/>
TVScreenType	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text"/>
TVScreenSize	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

**Next >**

12. The wizard has finished the configuration of the new document type. It has automatically created not only the database table, but also the SQL queries for SELECT, INSERT, UPDATE, DELETE operations and a default transformation.

Click **Finish**.

**Step 8** | **The wizard has finished**

The setup has finished the following steps:

- The new document type was created.
- The new editing form was created.
- The document types were added among allowed child types of the new document type.
- The sites were selected where this document type can be used.
- The default queries were created.
- The default ASCX transformations were created.
- The default permission names were created.
- The default icon was created.
- Document smart search specification was created.

**Finish**

12. You will be redirected to the General tab of the product type editing interface. Enable the **Document is product type** option and click **OK**.

Document types ▶ Plasma TV

General Fields Form Transformations Queries Child types Sites E-commerce Alternative forms Search fields Documents

Document type display name: Plasma TV

Document type code name: CustomProduct namespace television document type

Table name: CustomProduct\_television

New page:

Editing page:

Editing form:

Preview page:

List page:

Use publish from/publish to:

Show template selection:

Default page template:

Behaves as Page (menu item) type:

Document is product type:

13. Now go to **CMS Desk -> Products -> Plasma TV** and click **New** at the document action toolbar. You'll see that you can choose **Plasma TV** as the new document type.



### Adding attributes

This chapter explains how to add an attribute to specific document type. (e.g. product-camera, product - cell phones).

For adding attribute common to all products, please refer to the [Adding product custom fields](#) chapter.



### Changing document type icon

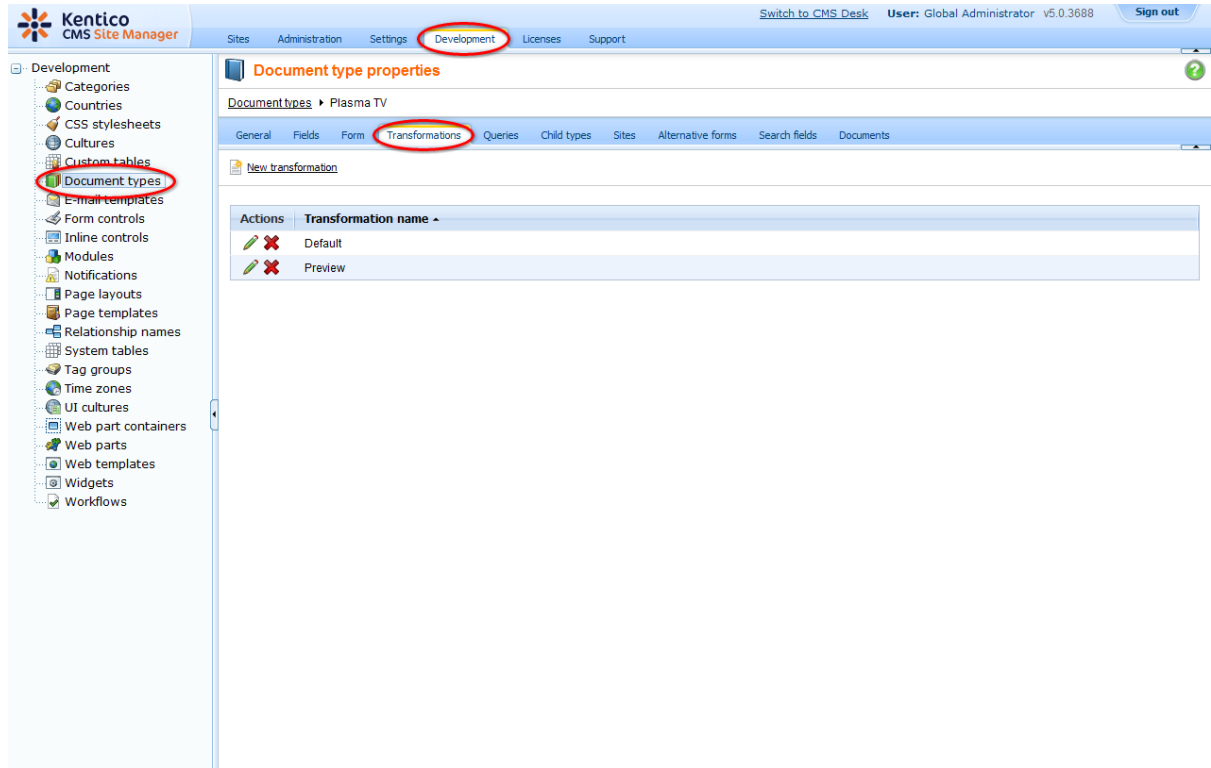
The default icon is created with the new document type. Should you want to replace this icon with your own, go to **<your web project folder> \App\_Themes\Default\Images\CMSDesk\Icons\** and find the file named **<namespace>\_<your new document type>.gif** (in our example, this is going to be **CustomProduct\_television.gif**). All you have to do is replace this file with the image file of your new icon. Please note that the new file has to be named the same as the one being replaced.

## 3.2 Customizing product design

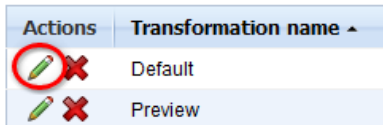
### Editing existing transformation

1. Go to **CMS Site Manager -> Development -> Document types**, choose the **Plasma TV** document

type and switch to the **Transformation** tab.



2. Click the **Edit transformation** button next to the **Default** transformation name.



3. Delete the content of the **Code** text box and insert the code from the gray box bellow. This code will change the transformation to display only the name of a TV and its screen type. Click **Save** at the top.

```
<table>
  <tr>
    <td>Product name:</td>
    <td><%# Eval("TVName") %></td>
  </tr>
  <tr>
    <td>Screen type:</td>
    <td><%# Eval("TVScreenType") %></td>
  </tr>
</table>
```


[Check out to file](#)

You can check out the transformation to file `c:\inetpub\wwwroot\Manuals\CMSTransformations\customproduct\television\default.ascx` to edit transformation externally.

Transformation name:

Transformation type:

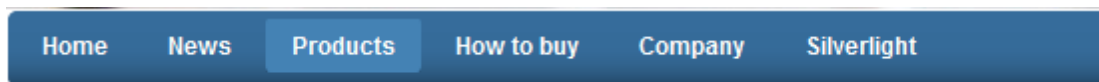
Code:

[Generate default transformation](#)

```
<%@ Control Language="C#" AutoEventWireup="true" Inherits="CMS.Controls.CMSAbstractTransformation" %>
<%@ Register TagPrefix="cc1" Namespace="CMS.Controls" Assembly="CMS.Controls" %>
<table>
  <tr>
    <td>Product name:</td>
    <td><%= Eval("TVName") %></td>
  </tr>
  <tr>
    <td>Screen type:</td>
    <td><%= Eval("TVScreenType") %></td>
  </tr>
</table>
```

[Click here to view list of transformations examples.](#)

That's how you change default transformation for your product so that the product name and screen type are displayed.



[Products](#) / [Plasma TVs](#) / [LG plasma TV 42PC51](#)

Product name: LG plasma TV 42PC51

Screen type: Plasma

**Please note:** You can modify the design of the **product list** web part by changing the **CMS.Root.EcommerceProductsList** transformation in the manner similar to the given example.



### Adding namespaces

You can add additional namespaces into the **web.config** file in the following location:

```
<system.web>
  <pages validateRequest="false">
    <namespaces>
      <add namespace="CMS.CMSHelper" />
      <add namespace="CMS.GlobalHelper" />
    </namespaces>
  </pages>
</system.web>
```

In case you add namespace into the **web.config** file, you don't have to specify it when calling its objects in transformations. Therefore, instead of calling **CMS.GlobalHelper.ResHelper.GetString("MyCustomString")**, you could call just **ResHelper.GetString("MyCustomString")**.



### Print page

Kentico CMS allows you to add a link button to your web page that will create print version of the given document. Please refer to the **Print page** chapter in **Kentico CMS Developer's Guide** for more information.

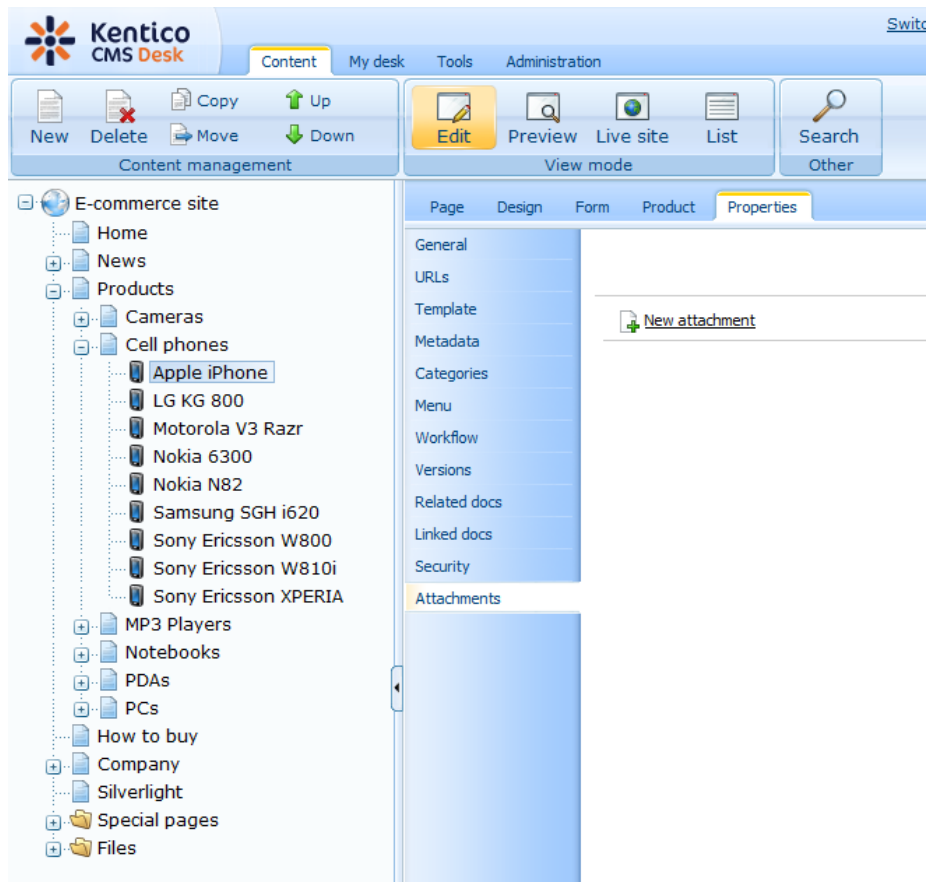
## 3.3 Adding images to product gallery

Now you will learn how to add images to product gallery.

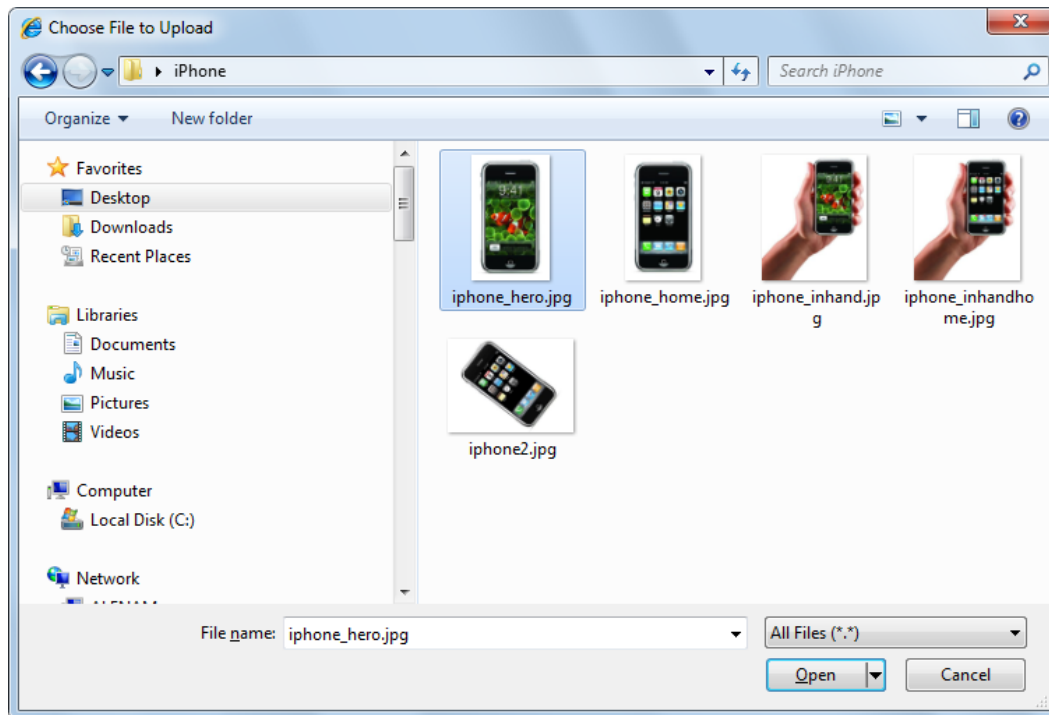
The images in the product gallery are displayed using the **Attachments image gallery** web part. Therefore, it is essential that you have placed this web part into your product page template.



The images of a product are stored as attachments to the document. To add images to the product, go to **CMS Desk -> Content -> Edit -> Properties -> Attachments** tab and select the product you want to add the images to.



By clicking the [New attachment](#) link, a dialog for uploading files appears where you can add images. Just select the desired image and click **Open**. Repeat the procedure for all images you want to attach.



The images are stored as attachments of the product.

The screenshot shows the Kentico CMS Desk interface. The left sidebar displays a tree view of the 'E-commerce site' structure, including 'Home', 'News', 'Products', 'Cameras', 'Cell phones', 'MP3 Players', 'Notebooks', 'PDAs', 'PCs', 'How to buy', 'Company', 'Silverlight', 'Special pages', and 'Files'. The 'Cell phones' folder is expanded to show 'Apple iPhone', which contains several phone models: 'LG KG 800', 'Motorola V3 Razr', 'Nokia 6300', 'Nokia N82', 'Samsung SGH i620', 'Sony Ericsson W800', 'Sony Ericsson W810i', and 'Sony Ericsson XPERIA'. The main content area shows the 'Attachments' tab for a product, displaying a table of five image attachments.

Actions	Update	Name
		iphone_hero.jpg
		iphone_home.jpg
		iphone_inhand.jpg
		iphone_inhandhome.jpg
		iphone2.jpg



You can arrange the images in the attachment using the **Up** (↑) and **Down** (↓) icons. Images can be edited via the **Edit** (✎) icon, or deleted by clicking the **Delete** (✖) icon.

When the images are attached to the product, you can display them on the live site within the **Attachments image gallery** web part.

The screenshot shows the Kentico CMS Desk interface. The top navigation bar includes 'Content', 'My desk', 'Tools', and 'Administration'. The 'Content management' toolbar contains icons for 'New', 'Delete', 'Move', 'Copy', 'Up', 'Down', 'Edit', 'Preview', 'Live site', 'List', and 'Search'. The left sidebar shows a tree view of the site structure, with 'E-commerce site' expanded to show 'Products' > 'Cell phones' > 'Apple iPhone'. The main content area displays a product page for an iPhone. It includes a price of '\$499.00', a 'Cell phones accessories' section with options for 'Car charger (+ \$5.00)', 'Headset (+ \$70.00)', and 'Leather case (+ \$10.00)', a 'Total price (without tax): \$499.00', an 'Add to wishlist' button, a quantity selector set to '1', and an 'Add to cart' button. Below this is a 'Description' section and a 'Product gallery' section with five images of the iPhone.

### 3.4 Adding product custom fields

1. Go to **CMS Site Manager** -> **Development** -> **System tables**.

Actions	Class display name ^	Class name	Table name
	Ecommerce - Customer	ecommerce.customer	COM_Customer
	Ecommerce - Order	ecommerce.order	COM_Order
	Ecommerce - Order item	ecommerce.orderitem	COM_OrderItem
	Ecommerce - Shopping cart	ecommerce.shoppingcart	COM_ShoppingCart
	Ecommerce - Shopping cart item	ecommerce.shoppingcartitem	COM_ShoppingCartSKU
	Ecommerce - SKU	ecommerce.sku	COM_SKU
	Group	Community.Group	Community_Group
	Media file	media.file	Media_File
	Newsletter - Subscriber	newsletter.subscriber	Newsletter_Subscriber
	User	cms.user	CMS_User
	User - Settings	cms.usersettings	CMS_UserSettings

2. Click **Edit** next to the **Ecommerce - SKU**.

Actions	Class display name ^	Class name	Table name
	Ecommerce - Customer	ecommerce.customer	COM_Customer
	Ecommerce - Order	ecommerce.order	COM_Order
	Ecommerce - Order item	ecommerce.orderitem	COM_OrderItem
	Ecommerce - Shopping cart	ecommerce.shoppingcart	COM_ShoppingCart
	Ecommerce - Shopping cart item	ecommerce.shoppingcartitem	COM_ShoppingCartSKU
	Ecommerce - SKU	ecommerce.sku	COM_SKU
	Group	Community.Group	Community_Group
	Media file	media.file	Media_File
	Newsletter - Subscriber	newsletter.subscriber	Newsletter_Subscriber
	User	cms.user	CMS_User
	User - Settings	cms.usersettings	CMS_UserSettings

3. Click **New Attribute** (+).

4. Into the **Attribute name** text box, enter *SKUColor*. Choose **Text** as **Attribute type**, enter *100* as **Attribute size** and *Product color* as **Field Caption**. Choose **Text box** as **Field Type**. Click **OK**.

**Database**

Attribute name:

Attribute type:

Attribute size:

Allow empty value:

Attribute default value:

Display attribute in the editing form

**Field**

Field caption:

Field type:

5. Now switch to **CMS Desk** and go to **Tools -> E-commerce -> Products**. Click **Edit** next to **Acer Aspire 3105WLMi**.

Actions	Product name
	Acer Aspire 3105WLMi
	Acer configurator
	Acer configurator
	Acer Extensa 7620G
	Apple iPhone

6. Switch to the **Custom fields** tab and enter *Blue* into the **Product color** text box. Then click **OK**.

**E-commerce**

Orders Customers **Products** Product options Manufacturers

Products ▶ Acer Aspire 3105WLMi

General **Custom fields** Tax classes Volume discounts Options

Product color:

That's how you create a custom field for all products and how you set its value for specific product.

## Product custom fields in transformations

For displaying values of the given custom field in transformation, enter `Eval("<custom field name>")`. For instance, for the example given in this chapter, you would enter `Eval("SKUColor")` (please see the [Customizing product design](#) chapter for more details).

In versions prior to 4.0, you had to update the following database views manually with the name of the new custom field in order for the `Eval("<custom field name>")` to work correctly. Now it is done **automatically** so that you **don't need to modify the database** at all.

1. `View_COM_SKU`
2. `View_CMS_Tree_Joined`
3. `View_CMS_Tree_Joined_Versions`

## Product custom fields in code

In code, you can get and set the given value in the following way:

**[C#]**

```
using System;
using CMS.Ecommerce;

// How to set value of the custom field
SKUInfo product = new SKUInfo();
product.SetValue("SKUColor", "green");

// How to get value of the custom field
string color = Convert.ToString(product.GetValue("SKUColor"));
```

**Part**

---

**IV**

**Customizing web site design**

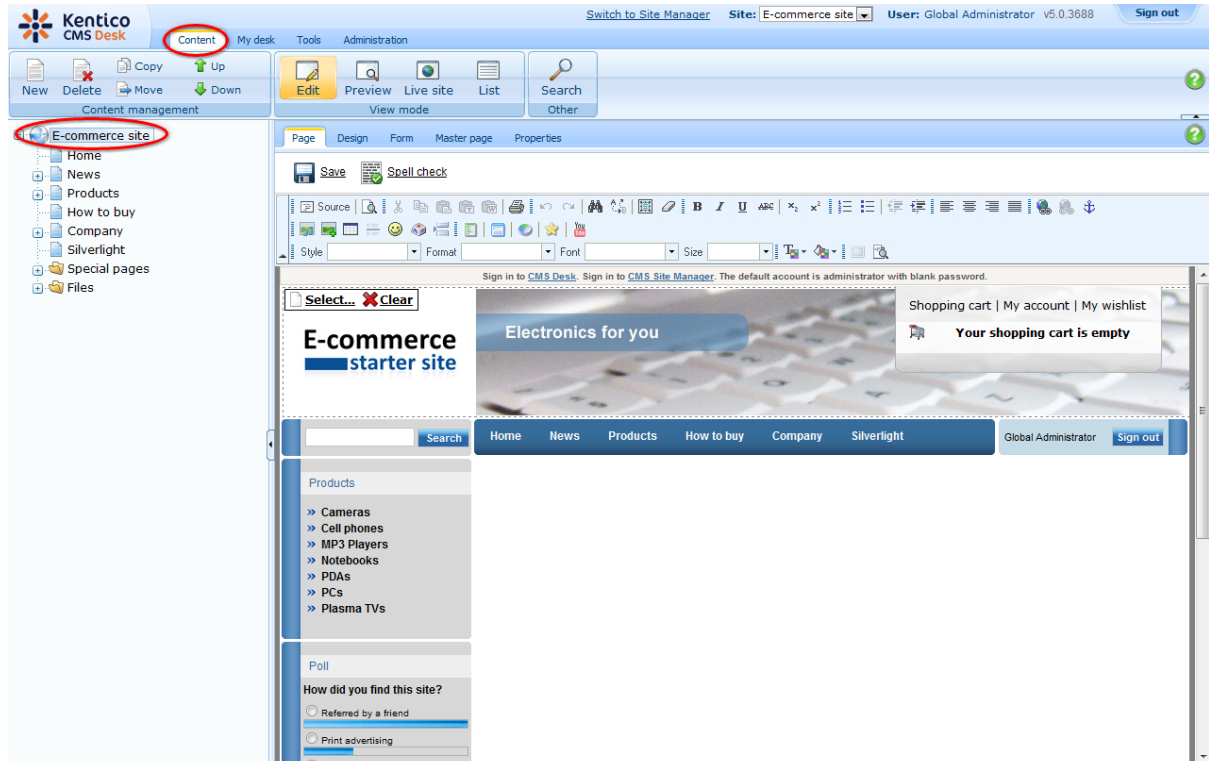
---

## 4 Customizing web site design

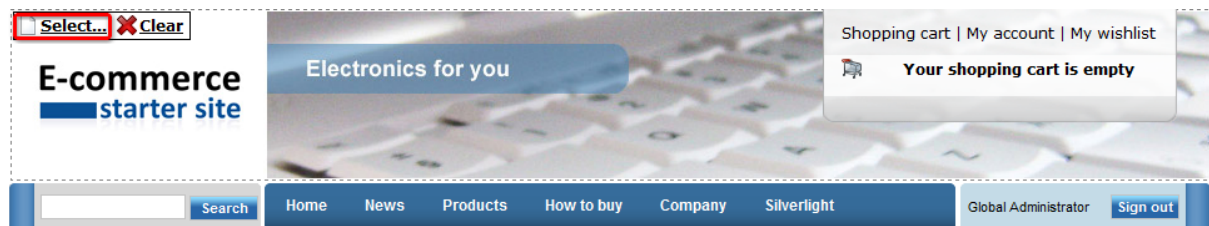
### 4.1 Changing the header and menu (master page)

#### Changing a logo in the header

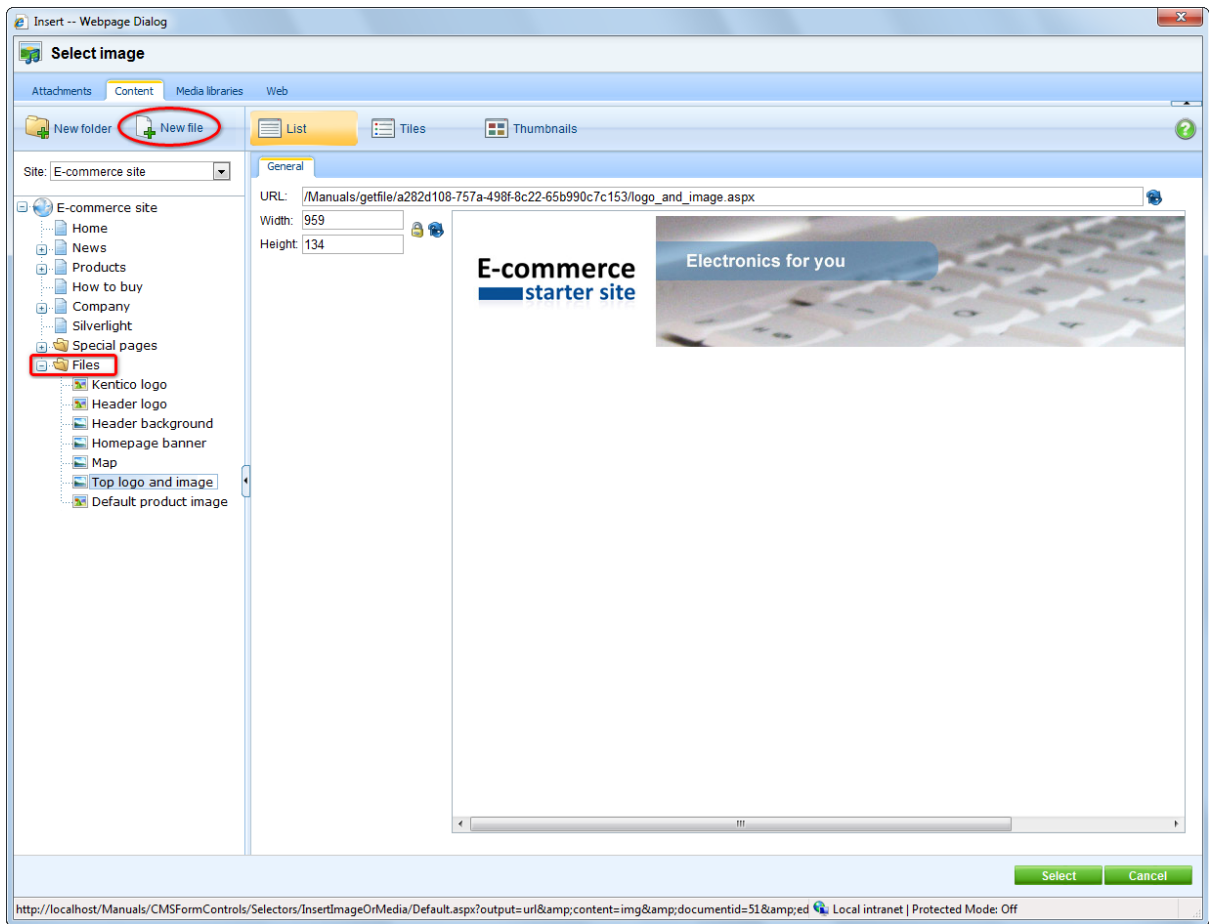
1. Go to **CMS Desk** -> **Content** -> **Ecommerce site**.



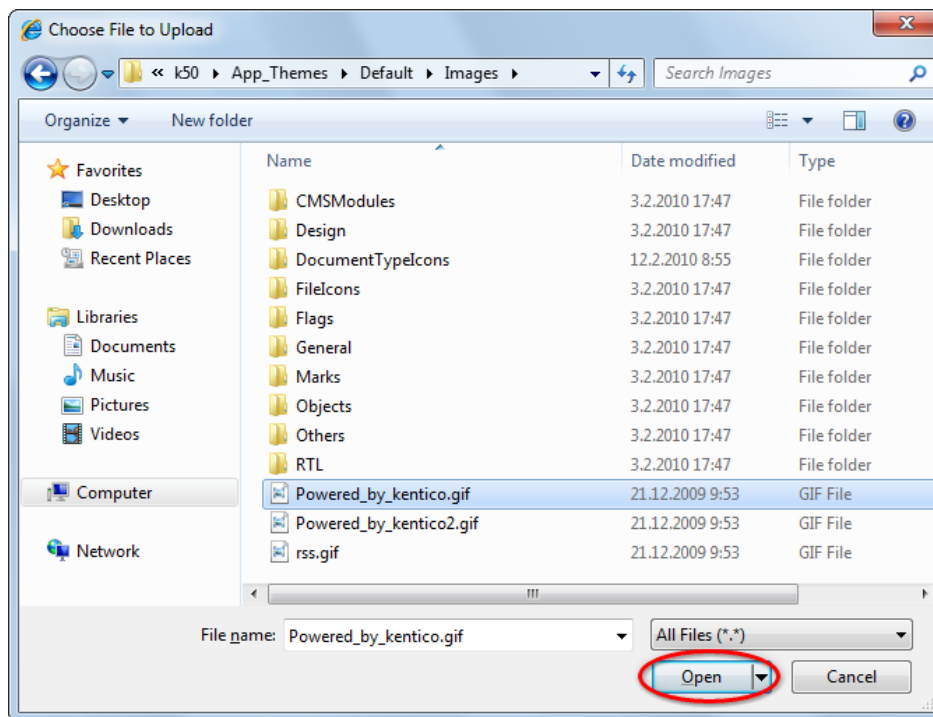
2. Click **Select ...** above the logo.



3. In the web page dialog, select **Files** in the content tree and click the **New file** button at the top-left.

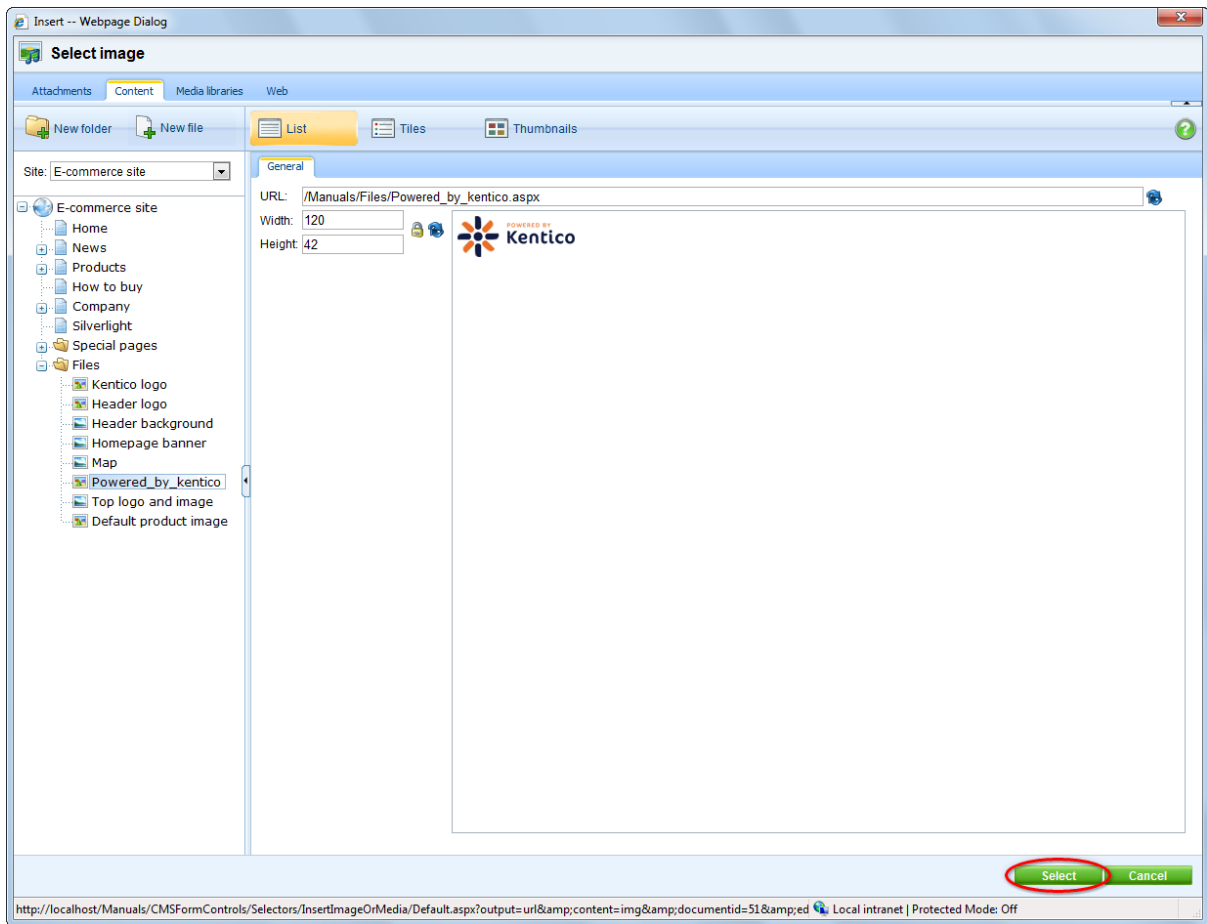


4. Find the image you want to upload and click **Open**.

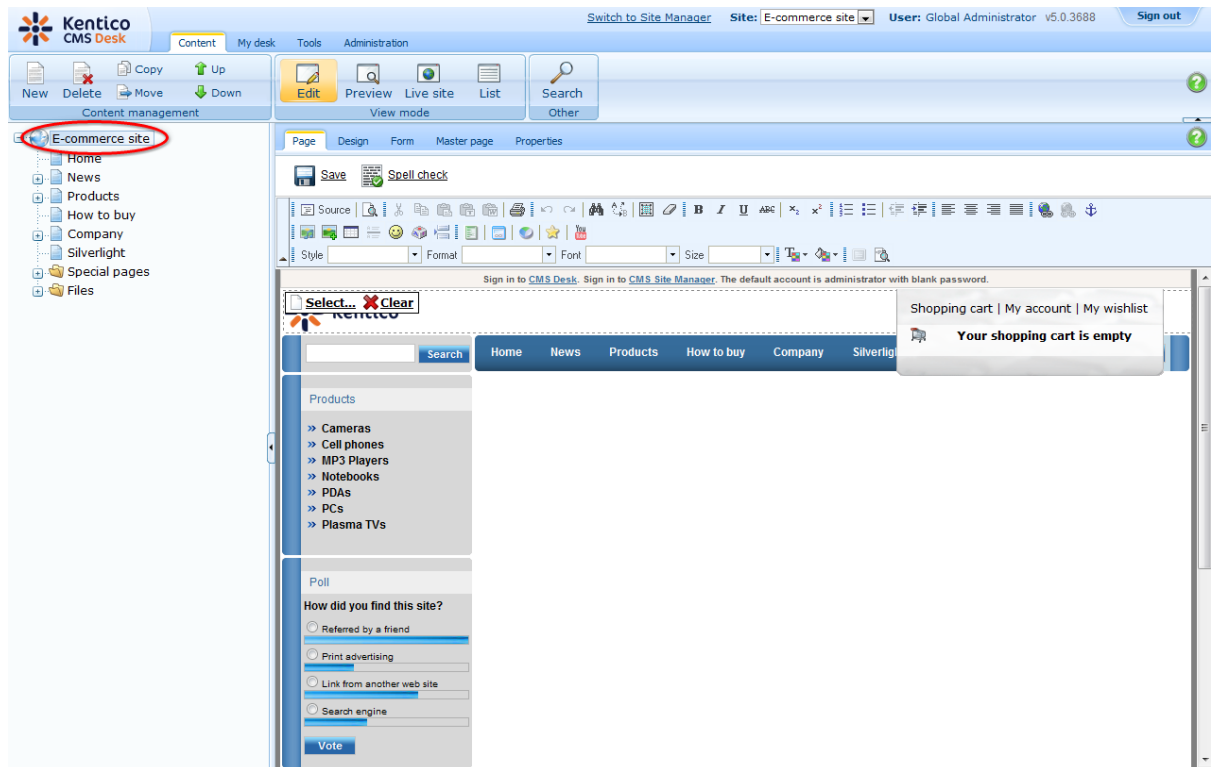


5. Click **Select**.





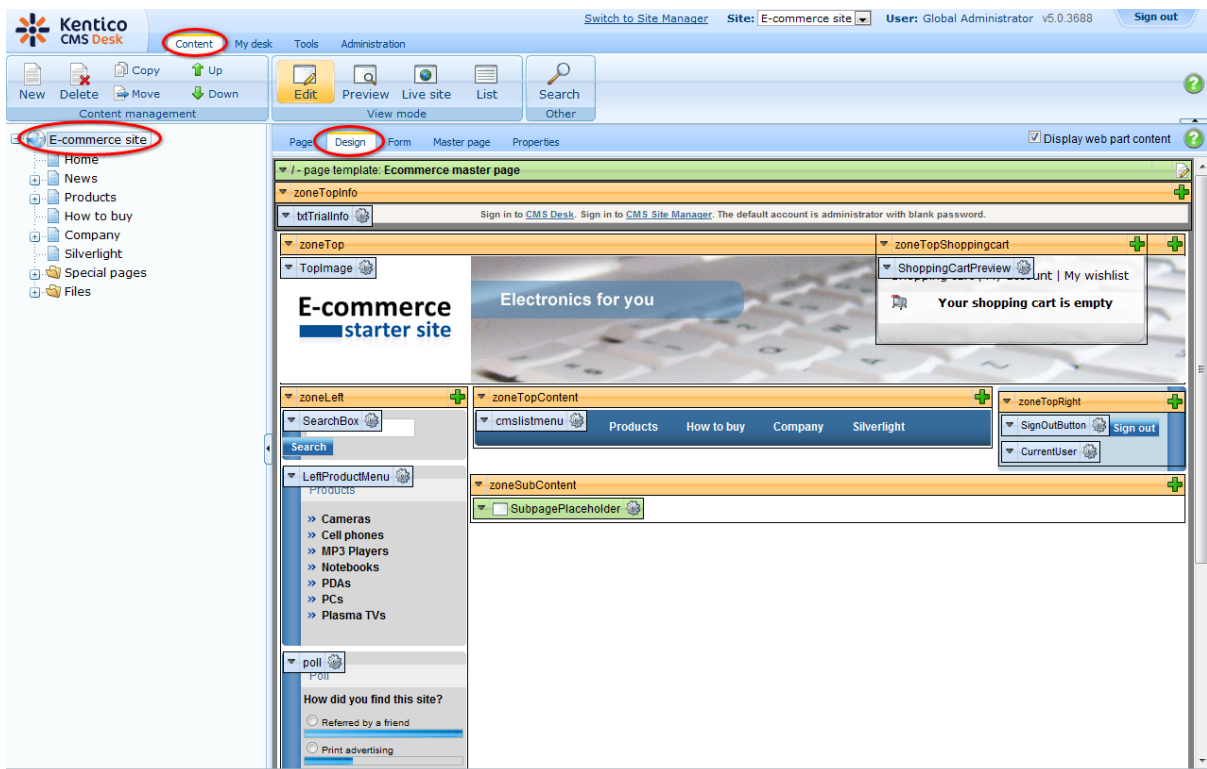
6. Now click **Save**.



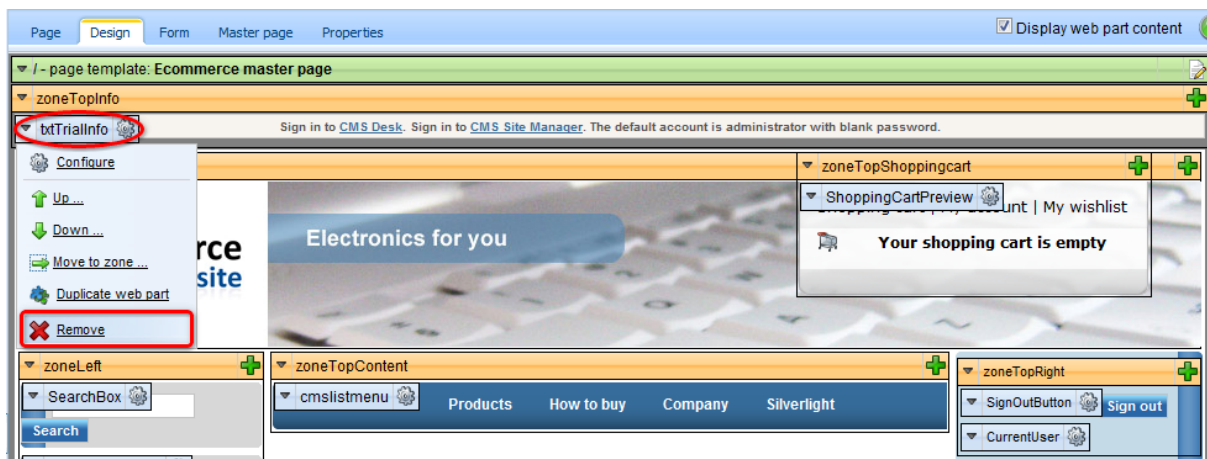
You've just publish a new logo on your website.

## Removing the log-on bar

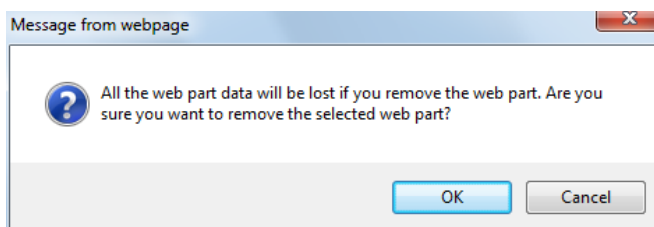
1. Go to **CMS Desk** -> **Content** -> **Ecommerce site** and switch to the **Design** tab.



2. Right-clicked the **txtTrialInfo** webpart at the top-left and choose **Remove**.



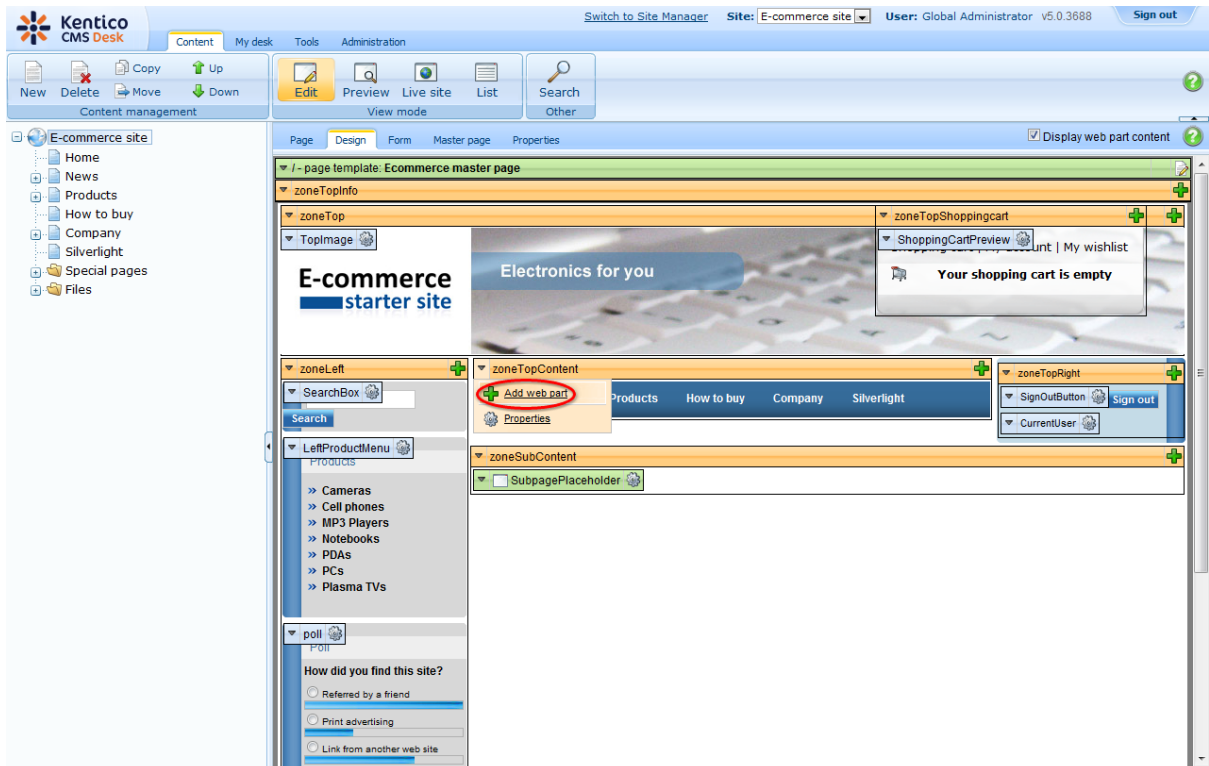
3. Click **OK** to remove the bar from your website.



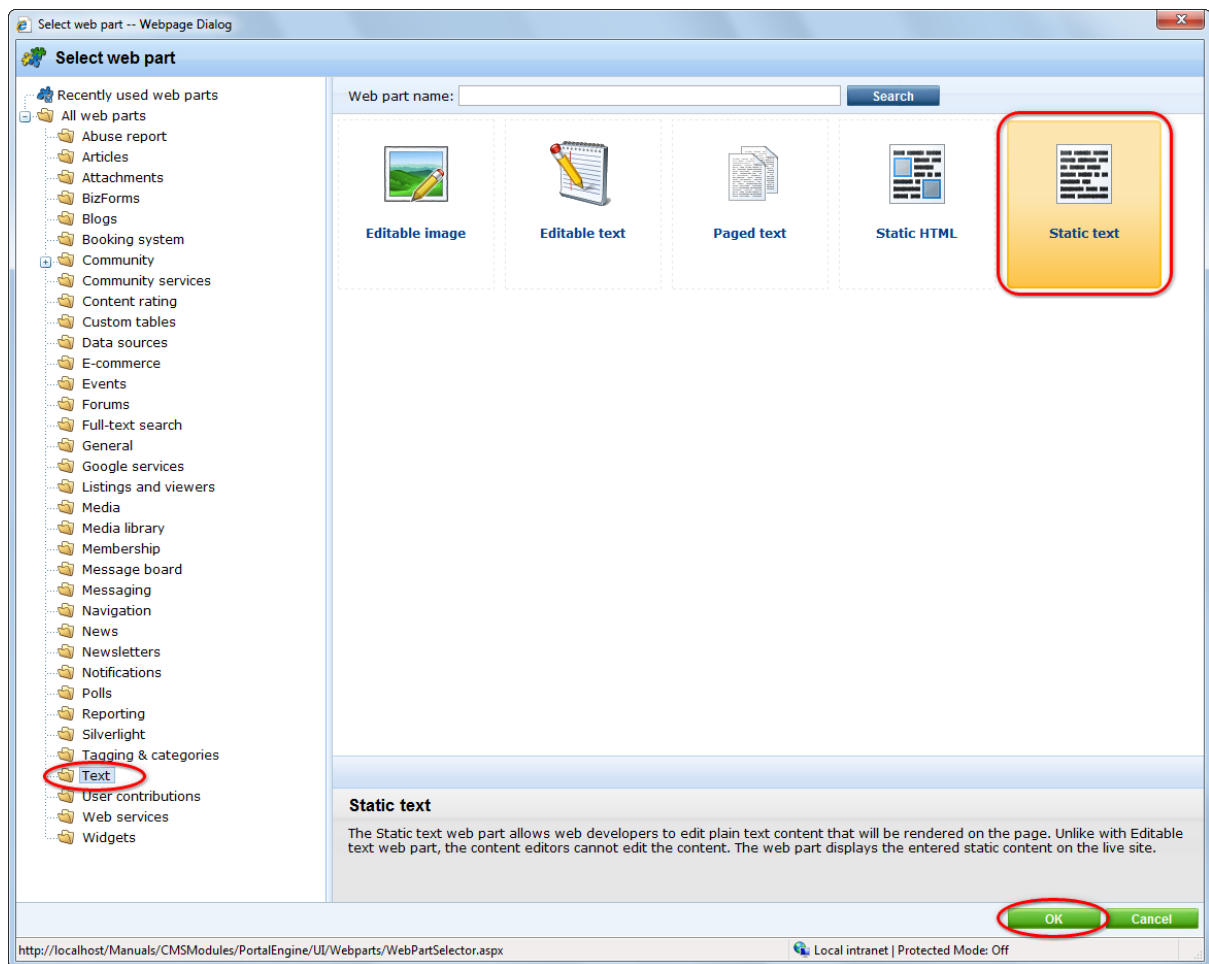
That's how you remove the signing-in bar from your website.

## Changing content of the menu

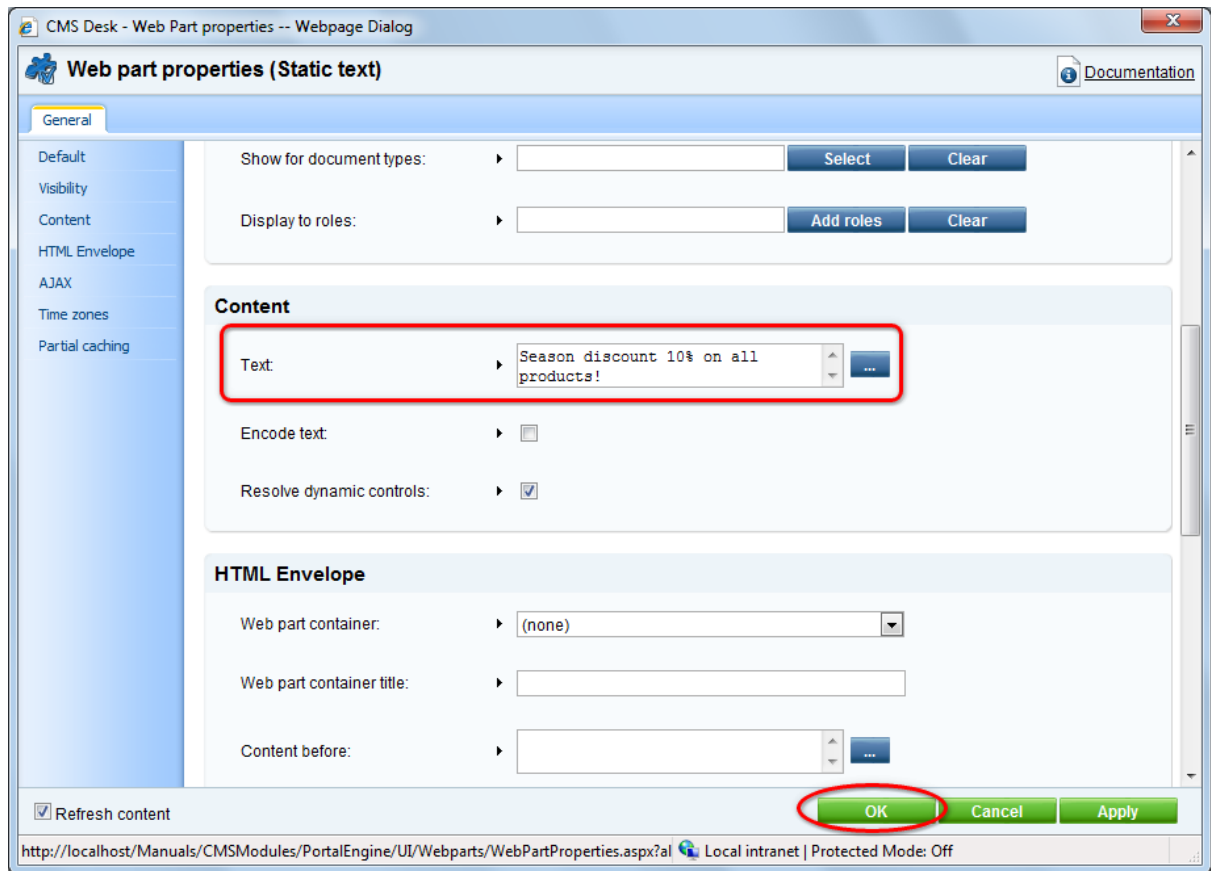
1. Go to **CMS Desk -> Content -> Ecommerce site** and switch to the **Design** tab.
2. Click the **Add web part** button from the drop-down menu in **zoneTopContent**.



3. In the web page dialog, choose **Text -> Static text** and click **OK**.



4. Enter *Season discount 10% on all products!* as **Text** and click **OK**.



5. Now switch to the **Live site** viewing mode. You can see the new text below the main menu.

POWERED BY  
**Kentico**

Shopping cart | My account | My wishlist  
Your shopping cart is empty

Home News Products How to buy Company Silverlight Global Administrator Sign out

Season discount 10% on all products!

Welcome to E-commerce starter site

Special offer of notebooks

Latest products

- Toshiba Satellite Pro \$1499.00
- Tiger Extreme PC \$999.00
- Sony VAIO VGN-CR320E \$1189.00

Latest news RSS

2/13/2008  
Dear customers, our shop now supports payments through PayPal. You can use this feature from this day.

2/13/2008  
Dear customers, our company has become Apple partner so you can buy Apple's great products at our shop.

Products

- » Cameras
- » Cell phones
- » MP3 Players
- » Notebooks
- » PDAs
- » PCs
- » Plasma TVs

Poll

How did you find this site?

- Referred by a friend
- Print advertising
- Link from another web site
- Search engine

Vote

Quick links

- » [How to buy](#)
- » [Company](#)
- » [News](#)
- » [Home](#)





















### More information

Should you need more information about customizing your website design, please refer to **Kentico CMS Developer's Guide**.













## 4.2 Changing colors using CSS styles

1. Go to **CMS Site Manager** -> **Development** -> **CSS stylesheets**.

The screenshot shows the Kentico CMS Site Manager interface. The top navigation bar includes 'Sites', 'Administration', 'Settings', 'Development' (highlighted with a red circle), 'Licenses', and 'Support'. The left sidebar shows a tree view of development options, with 'CSS stylesheets' also highlighted with a red circle. The main content area displays the 'CSS stylesheets' section, featuring a table with columns for 'Actions', 'Display name', and 'Code name'. The table lists several stylesheets, including 'Ecommerce site', 'Ecommerce site printer styles', 'Personal Site', 'Personal Site - Blue', 'Personal Site - Green', and 'Personal Site - Red'. Each row has an 'Edit' icon (pencil) and a 'Delete' icon (X).

Actions	Display name	Code name
  	Ecommerce site	EcommerceSite
  	Ecommerce site printer styles	EcommerceSitePrinter
  	Personal Site	PersonalSite
  	Personal Site - Blue	PersonalSiteBlue
  	Personal Site - Green	PersonalSiteGreen
  	Personal Site - Red	PersonalSiteRed

2. Click the **Edit** button next to the **Ecommerce site**.

Actions	Display name	Code name
  	Ecommerce site	EcommerceSite
  	Ecommerce site printer styles	EcommerceSitePrinter
  	Personal Site	PersonalSite
  	Personal Site - Blue	PersonalSiteBlue

3. In the **Stylesheet text** text box change background color of **body.LTR**, **body.RTL** from gray to yellow. Click **Save** at the top.





Check out the stylesheet to file c:\inetpub\wwwroot\Manuals\CMSCSS\Stylesheets\EcommerceSite.css to edit the stylesheet externally.

Stylesheet display name:

Stylesheet code name:

Stylesheet text:

```

/*#Main styles#*/
body
{
    font-family: Arial;
    font-size: 12px;
}

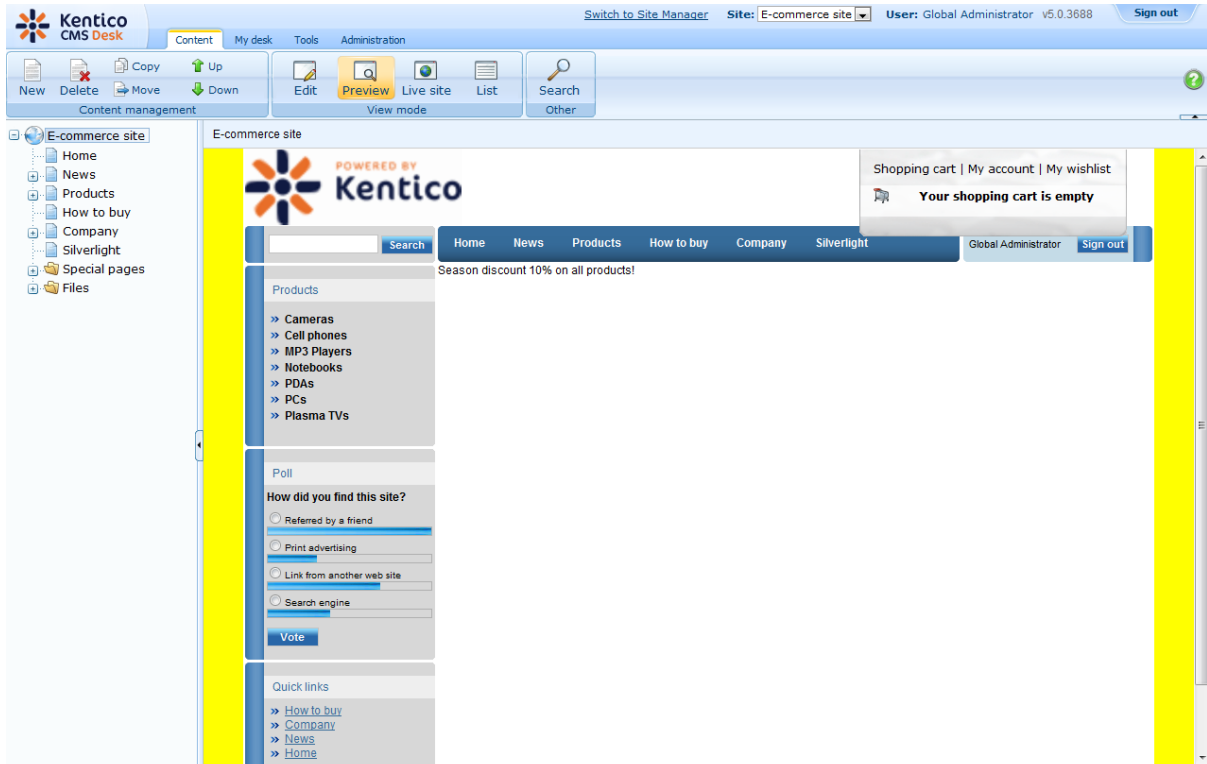
body.LTR, body.RTL
{
    background: yellow;
    padding: 0px;
    margin: 0px;
}

legend
{
    font-size: 12px;
}

a
{
    color: #356B99;
}
    
```

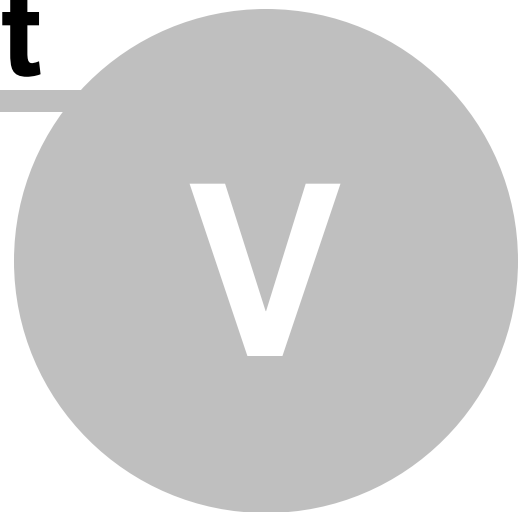
- Boxes
  - Blue box left
  - Blue box right
- Buttons
- Company
- Content rating
- Main styles
- Menus
  - Left menu
  - Top menu
- Modal popup
- News
- Newsletter subscription
- Other styles
- Polls
- Products
  - Filter
  - Forum
- Random product
- Random product right column
- Random product with status
- Search
- Search box
- Send to friend
- Shopping cart item selector
- Shopping cart preview

4. Go to **CMS Desk** and switch to the **Preview** mode. You can see that the background of the E-commerce website is yellow.



**Part**

---



**Integration with your existing Kentico CMS web site**

## 5 Integration with your existing Kentico CMS web site

### 5.1 Overview

The e-commerce module provides several web parts that allow you to integrate it into your website:

- **Shopping cart** - displays the shopping cart content and ensures the check-out process. For more info please see the [Shopping cart](#) chapter.
- **Shopping cart preview** - displays the links to the shopping cart, to my account and to my wishlist and the total value of the shopping cart content. For more info please see the [Shopping cart preview](#) chapter.
- **My account** - displays the details of the current user, such as personal settings, addresses, orders, invoices and allows user to change the password. For more info please see the [My account](#) chapter.
- **Wishlist** - displays the wishlist of the currently logged-on user. For more info please see the [Wishlist](#) chapter.
- **Similar products by sale** - displays products which were purchased together with the current product by previous customers. For more info please see the [Similar products by sale](#) chapter.
- **Random products** - displays random N products that correspond to the criteria specify in the content filter. For more info please see the [Random products](#) chapter.
- **Product datalist** - display products based on their e-commerce product (SKU) properties instead of displaying the standard CMS documents.
- **Top N products by sales** - displays N best-selling products. For more info please see the [Top N products by sales](#) chapter.

These web parts can be used also as independent user controls placed on the ASPX pages.

### How to organize the pages

A typical content tree structure of an e-commerce-enabled website looks like this:

- **Home** (page)
- **Products** (page)
  - **Category 1** (page; contains Repeater or DataList web parts displaying products with links to shopping cart)
    - **Product 1** (product specification document; it's marked as a product)
    - **Product 2** (product specification document; it's marked as a product)
  - **Category 2** (page; contains Repeater or DataList web parts displaying products with links to shopping cart)
    - **Sub category 1** (page; contains Repeater or DataList web parts displaying products with links to shopping cart)
      - **Product 3** (product specification document; it's marked as a product)
      - **Product 4** (product specification document; it's marked as a product)
- **News** (page)
- **Special pages**
  - **Shopping Cart** (page; it contains the Shopping cart web part and can be configured to be hidden in the navigation and site map)
  - **My Account** (page; it contains the My account web part and can be configured to be hidden in the navigation and site map)

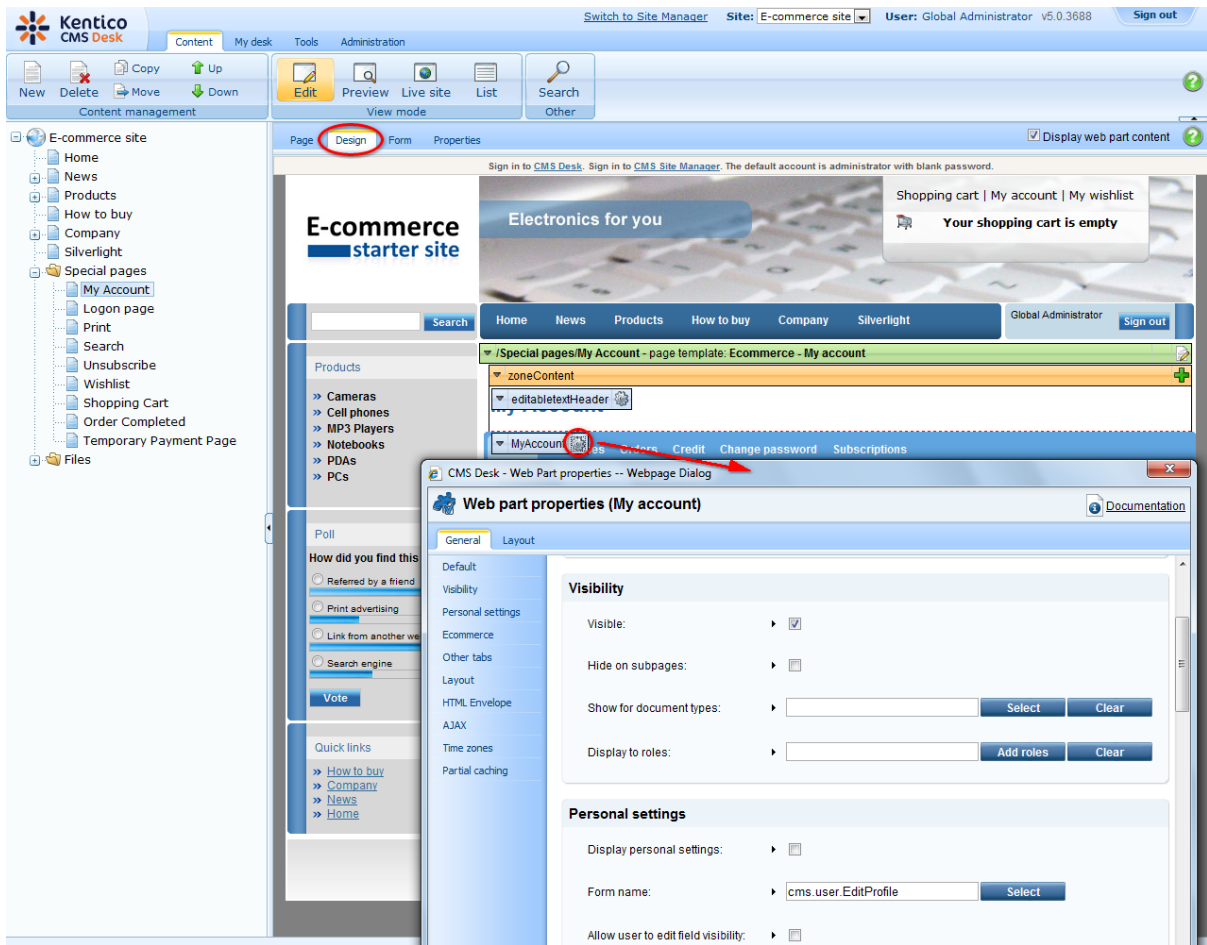
## 5.2 Web parts

### 5.2.1 My account

The **My account** web part displays the details of the current user, such as personal settings, addresses, orders, invoices and allows user to change the password.

The screenshot shows a web page for an e-commerce site. At the top, there is a navigation bar with links for Home, News, Products, How to buy, Company, Silverlight, and a user profile section for 'Global Administrator' with a 'Sign out' button. A shopping cart notification in the top right corner shows 'Shopping cart', 'My account' (circled in red), and 'My wishlist', with a message 'Your shopping cart is empty'. The main content area is titled 'My Account' and has tabs for 'Details', 'Addresses', 'Orders', 'Credit', 'Change password', and 'Subscriptions'. The 'Details' tab is active, showing a form with the following fields: First name (Global), Last name (Administrator), Company account (checkbox), E-mail (administrator@localhost.local), Phone, Fax, Preferred currency (U.S. Dollar), Preferred shipping option (none), Preferred payment method (none), and Country/state (none). An 'OK' button is at the bottom of the form. The left sidebar contains a search bar, a 'Products' menu with categories like Cameras, Cell phones, MP3 Players, Notebooks, PDAs, and PCs, a poll titled 'How did you find this site?' with radio buttons for 'Referred by a friend', 'Print advertising', 'Link from another web site', and 'Search engine', and a 'Vote' button. Below the poll are 'Quick links' for 'How to buy', 'Company', 'News', and 'Home'. The footer features the 'POWERED BY Kentico' logo.

You can modify the functionality of **My account** by setting following properties in the **Web Part properties** dialog:



### Display properties

<b>Display my details</b>	Indicates if <b>My Details</b> should be displayed to the user.
<b>Display my addresses</b>	Indicates if <b>My Addresses</b> should be displayed to the user.
<b>Display my orders</b>	Indicates if <b>My Orders</b> should be displayed to the user.
<b>Display my credit</b>	Indicates if <b>My credit</b> should be displayed to the user.
<b>Display change password</b>	Indicates if dialog for password reset should be displayed to the user.
<b>Display my subscriptions</b>	Indicates if <b>My Subscriptions</b> should be displayed to the user.
<b>Display my messages</b>	Indicates if <b>My Messages</b> should be displayed to the user.
<b>Display my personal settings</b>	Indicates if <b>Personal settings</b> should be displayed to the user.

### Settings properties

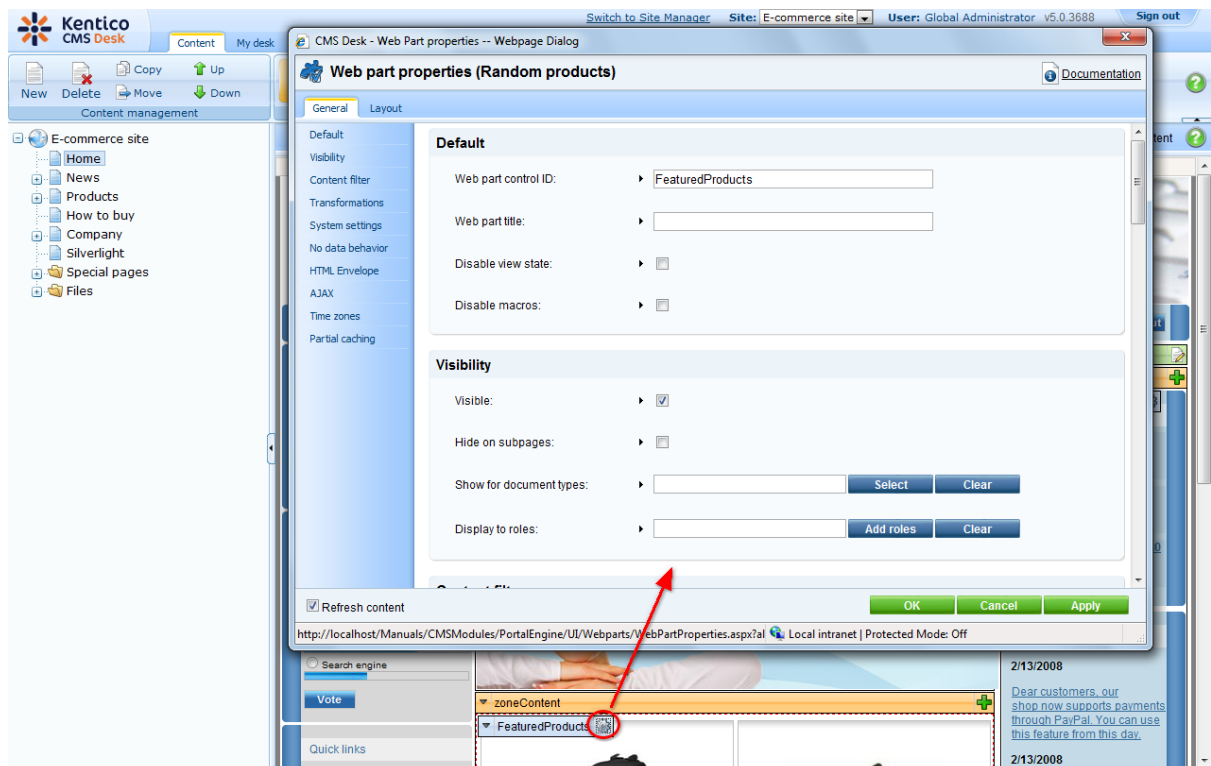
<b>Show order tracking</b>	Indicates if order tracking number should be visible in the order list.
----------------------------	---

<b>number</b>	
<b>Allow empty password</b>	Indicates if empty password is allowed when changing the user password.
<b>Tabs css class</b>	CSS class used for the tabs.

## 5.2.2 Random products

The **Random products** web part displays products that correspond to the criteria specified in the content filter.

You can modify the functionality of **Random products** by setting properties in the **Web Part properties** dialog as you can see on the following image:



### Content filter main properties

<b>Product public status</b>	Public status of products which should be displayed.
<b>Product internal status</b>	Internal status of products which should be displayed.
<b>Product department</b>	Department of products which should be displayed.
<b>Only random N products</b>	Indicates how many random products should be displayed. Don't put any value, if you want display all products.
<b>Top N</b>	The first N products selected according to the chosen criteria.

<b>Path</b>	Path of the documents to be displayed.
<b>Document Types</b>	Indicates from which document types should be products chosen.
<b>WHERE Condition</b>	WHERE part of the SELECT query.
<b>ORDER BY expression</b>	ORDER BY part of the SELECT query.

The **Random products** web part can be used in many different ways. All you have to do is to set the content filter accordingly to get rights products displayed.

## 1. Featured products on Home page

The screenshot shows a web page with a navigation menu (Home, News, Products, How to buy, Company, Silverlight) and a search bar. The main content area features a 'Special offer of notebooks' banner with a woman and a laptop. Below the banner, four products are displayed in a grid, each with an image, price, and name. A red box highlights these four products: Apple iPhone (\$499.00), Canon Digital Rebel XT (\$597.00), Emgeton M7 Cult (\$215.00), and iPod Nano (\$199.00). The right sidebar contains sections for 'Latest products' (listing Toshiba Satellite Pro, Tiger Extreme PC, and Sony VAIO VGN-CR320E), 'Latest news RSS' (with two news items dated 2/13/2008), and a 'Newsletter subscription' form.

On the **Home** page, **Random products** is set to display the maximum of 4 random products that have the public status set to **Show as featured product**:

**Only random N products:** 4

The maximum of 4 random products is displayed.

**Product public status:** Show as featured product

Only products with the public status set to **Show as featured product** are displayed.

**Transformation name:** ecommerce.transformations.EcommerceSite\_FeaturedProducts

The ecommerce.transformations.EcommerceSite\_FeaturedProducts transformation used in the list view mode.

For description of how to display the given product as the featured product, please refer to the [Displaying featured products](#) chapter.

## 2. Similar products at the product detail

At the product detail, the given web part is set to display the maximum of 4 random products which have price similar to the currently chosen product and which are of the same type.

**Show for document types:** CMSProduct.Camera;CMSProduct.CellPhone;CMSProduct.Computer; CMSProduct.Laptop;CMSProduct.MediaPlayer;CMSProduct.PDA

**Random product** is displayed only in the given document types, therefore, at product detail. If a user switch to the product list, the given web part is hidden. If you add a new product type and you want it to be displayed with the **Random product** web part, you need to add your new product type into the **Show for document types** text box.

**Only random N products:** 4

The web part displays 4 random products at most, according to the filter conditions.



**WHERE condition:** (SKUPrice > {%SKUPrice%} \* 0.6) AND (SKUPrice < {%SKUPrice%} \* 1.4) AND (NodeSKUID <> {%NodeSKUID%})

Ensures that only products with the price similar to the currently chosen product are displayed.

**Document types:** {%ClassName%}

Ensures that only the products of the same type as the currently chosen product are displayed.

For more details about the context macros please refer to the **Appendix A - Macro expressions** chapter in **Kentico CMS Developer's Guide**.

**Transformation name:** Ecommerce.Transformations.EcommerceSite\_RandomProducts

The Ecommerce.Transformations.EcommerceSite\_RandomProducts transformation used in the list view mode.

For instance, if the currently chosen product is type of **cmsproduct.cellphone** with **SKUPrice** set to 100, the **Random product** web part displays the maximum of 4 products, type of **cmsproduct.cellphone** with **SKUPrice** between 60 and 140.

### 3. Recommended products on News page and Home page

On the **News** page, the given web part is set to display 4 random products, regardless of their type.

The screenshot shows an e-commerce website interface. At the top, there's a navigation bar with 'Home', 'News', 'Products', 'How to buy', 'Company', and 'Silverlight'. A search bar is on the left. The main content area is titled 'News' and contains three news items: 'PayPal payment support', 'Apple products', and 'Our new e-shop launched'. On the right side, there's a 'Recommended products' section with four items: 'Acer Extensa 7620G' (\$99.00), 'Creative ZEN' (\$150.00), 'Nikon D40' (\$490.00), and 'Samsung SGH i620' (\$650.00). This section is highlighted with a red box. Below it is an 'RSS' feed for 'Latest news RSS' dated 2/13/2008.

In this case, the **Only random N products** property is set to **4** and the **Transformation name** property is set to **ecommerce.transformations.Ecommerce\_RandomProducts**.

On the **Home** page, the given web part is set to display 1 random product, regardless of its type.



In this case, the **Only random N products** property is set to **1**. The **Transformation name** property is set to `ecommerce.transformations.Ecommerce_RandomProduct`.

### 5.2.3 Shopping cart

The **Shopping cart** web part displays the shopping cart content and ensures the check-out process.

Sign in to [CMS Desk](#). Sign in to [CMS Site Manager](#). The default account is administrator with blank password.

**E-commerce starter site**

Electronics for you

Shopping cart | My account | My wishlist

Total price: \$172.50

Home News Products How to buy Company Silverlight Global Administrator Sign out

Products

- » Cameras
- » Cell phones
- » MP3 Players
- » Notebooks
- » PDAs
- » PCs

Poll

How did you find this site?

Referred by a friend

Print advertising

Link from another web site

Search engine

Vote

Quick links

- » [How to buy](#)
- » [Company](#)
- » [News](#)
- » [Home](#)

Step 1 of 6 - Add some products to the shopping cart

Shopping cart

Currency: U.S. Dollar

Remove	Product name	Units	Unit price	Tax	Subtotal
<input type="checkbox"/>	<a href="#">Creative ZEN</a>	1	150.00	22.50	172.50

If you have a coupon code, please enter it here:

Total shipping: \$0.00

Total price: \$172.50

Empty

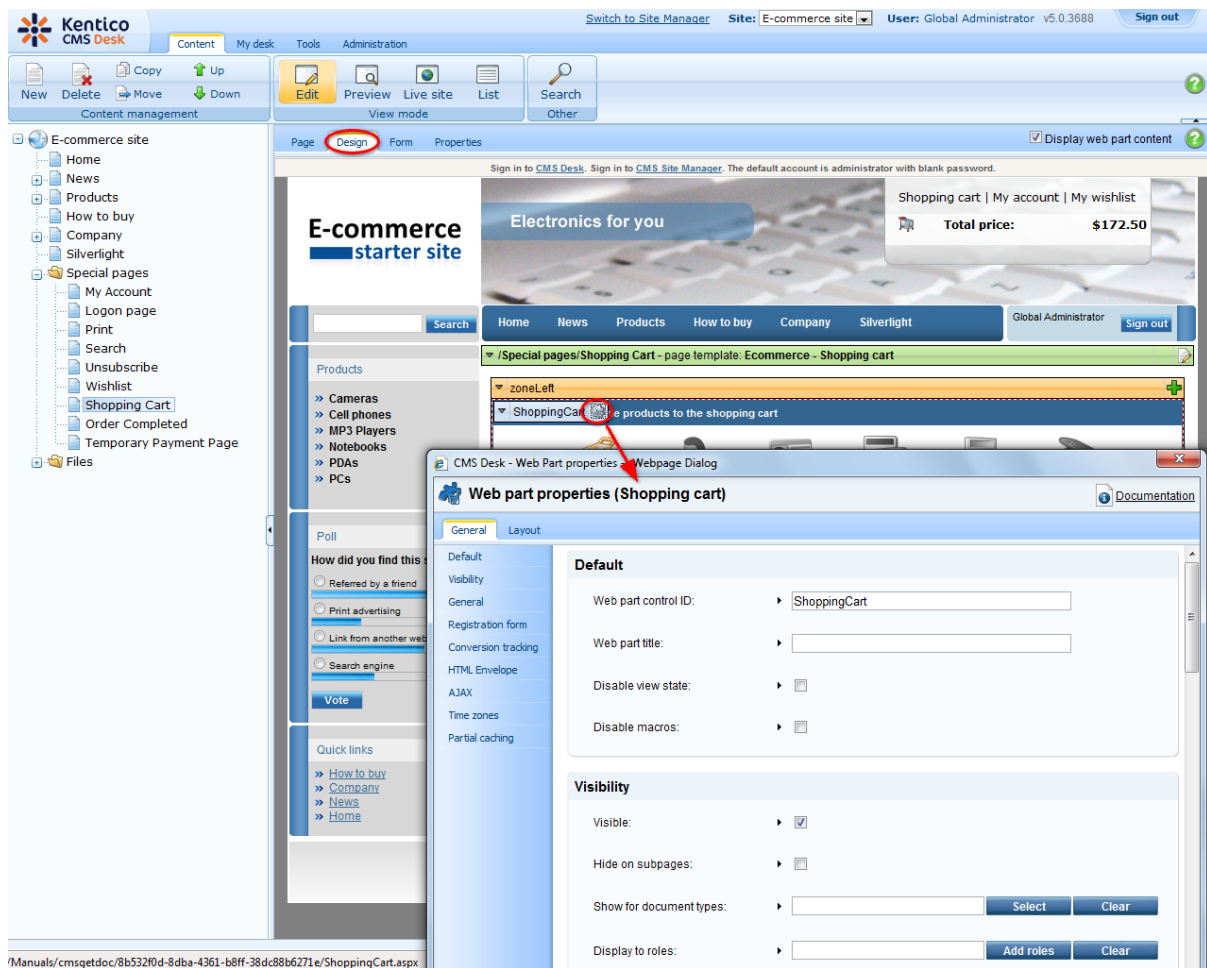
Update

Continue shopping

Check out

POWERED BY Kentico

You can modify the functionality of **Shopping cart** by setting following properties in the **Web Part properties** dialog:



## General properties

<b>Default URL after purchase</b>	Default page where the user should be redirected if no URL is specified for the given payment option.
<b>Allow forgotten password retrieval</b>	Indicates if the forgotten password can be retrieved in the <b>Shopping Cart sign-in</b> dialog.
<b>Display step images</b>	Indicates if the images should be displayed during the order process
<b>Image step separator</b>	The separator displayed between shopping progress images.
<b>Enable product price detail</b>	Indicates if the link to the price detail page should be displayed.
<b>Required fields mark</b>	HTML code for the required fields mark, e.g. asterisk (*)

## Registration form properties

<b>Assign user to role</b>	If you enter some role to this field, the user will be automatically assigned to it after registration.
----------------------------	---

<b>Notify administrator about new registrations to e-mail address</b>	Enter administrator's e-mail address if you want to send registration notification message.
---	---

## Conversion tracking properties

<b>Registration conversion name</b>	Name of the registration conversion used in web analytics.
<b>Order conversion name</b>	Name of the order conversion used in web analytics.

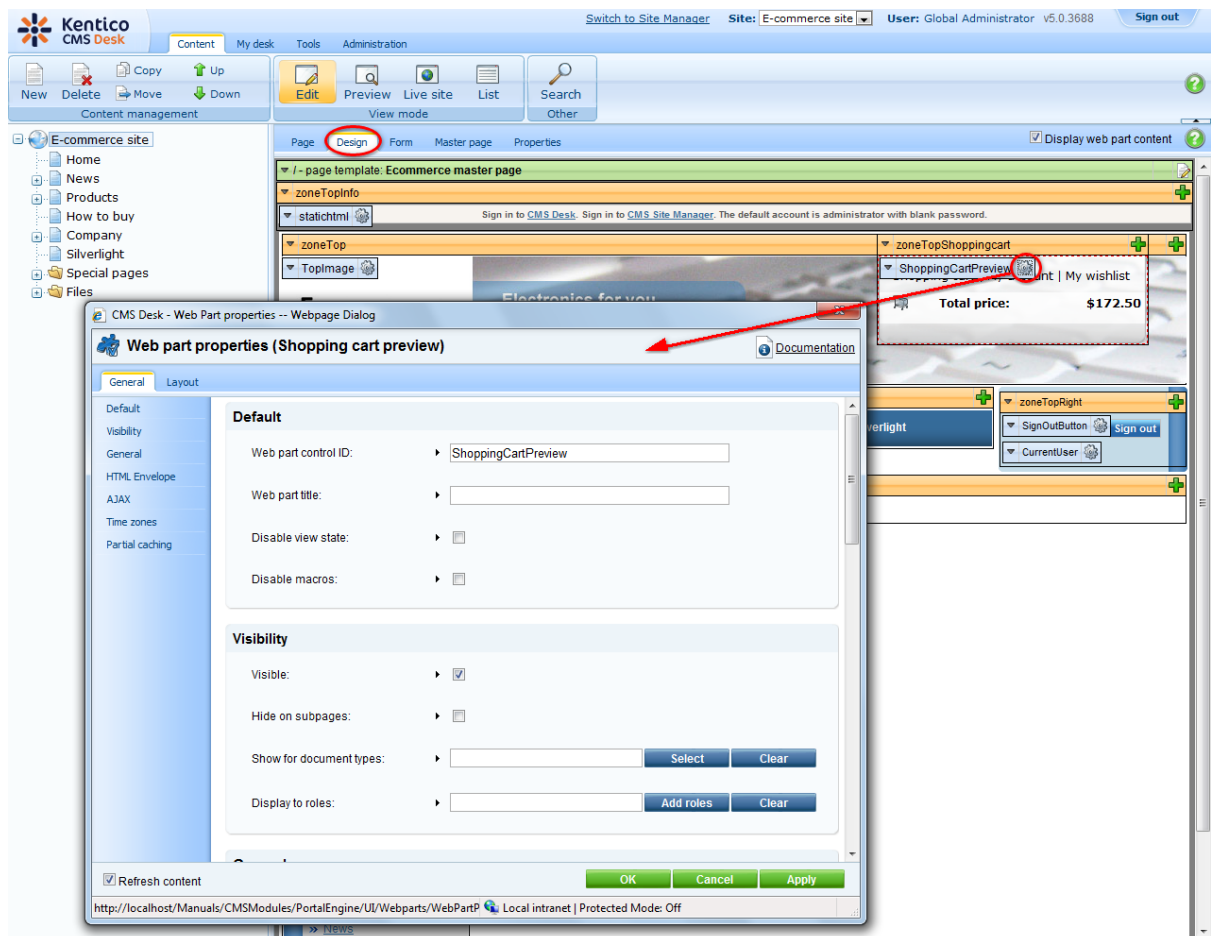
### 5.2.4 Shopping cart preview

The **Shopping cart preview** web part displays the links to the shopping cart, to my account and to my wishlist and the total value of the shopping cart content.

The screenshot displays an E-commerce starter site with the following elements:

- Header:** "E-commerce starter site" logo, "Electronics for you" banner, and a shopping cart preview box (highlighted in red) showing "Shopping cart | My account | My wishlist" and "Total price: \$172.50".
- Navigation:** Home, News, Products, How to buy, Company, Silverlight, Global Administrator, and Sign out.
- Left Sidebar:** Search bar, Products list (Cameras, Cell phones, MP3 Players, Notebooks, PDAs, PCs), Poll ("How did you find this site?"), and Quick links (How to buy, Company, News, Home).
- Main Content:** "Welcome to E-commerce starter site", "Special offer of notebooks" banner with a laptop image, and two product cards: "Apple iPhone" (Our price: \$499.00) and "iPod Nano" (Our price: \$199.00).
- Right Sidebar:** Recommended products (Toshiba Satellite Pro \$1499.00), Latest news RSS, and Newsletter subscription form.

You can modify the functionality of **Shopping cart preview** by setting following properties in the **Web Part properties** dialog:



## General properties

<b>Shopping cart link URL</b>	URL of the page with <b>Shopping cart</b> web part. If not set, the default URL from the <b>Site Manager -&gt; Manager -&gt; Settings -&gt; E-commerce -&gt; Shopping cart URL</b> settings is used.
<b>Shopping cart link text</b>	Text of the link to the <b>Shopping cart</b> page.
<b>Wishlist link URL</b>	URL of the page with <b>Wishlist</b> web part. If not set, the default URL from the <b>Site Manager -&gt; Settings -&gt; E-commerce -&gt; Wishlist URL</b> settings is used.
<b>Wishlist link text</b>	The link text for the <b>Wishlist</b> URL.
<b>My Account link URL</b>	URL of the page with <b>My Account</b> web part. If not set, the default URL from the <b>Site Manager -&gt; Settings -&gt; E-commerce -&gt; My account URL</b> settings is used.
<b>MyAccount link text</b>	Text of the link to the <b>My Account</b> page.
<b>Total price text</b>	Text displayed next to the total price.
<b>Show total price row</b>	Indicates if total price row should be displayed.

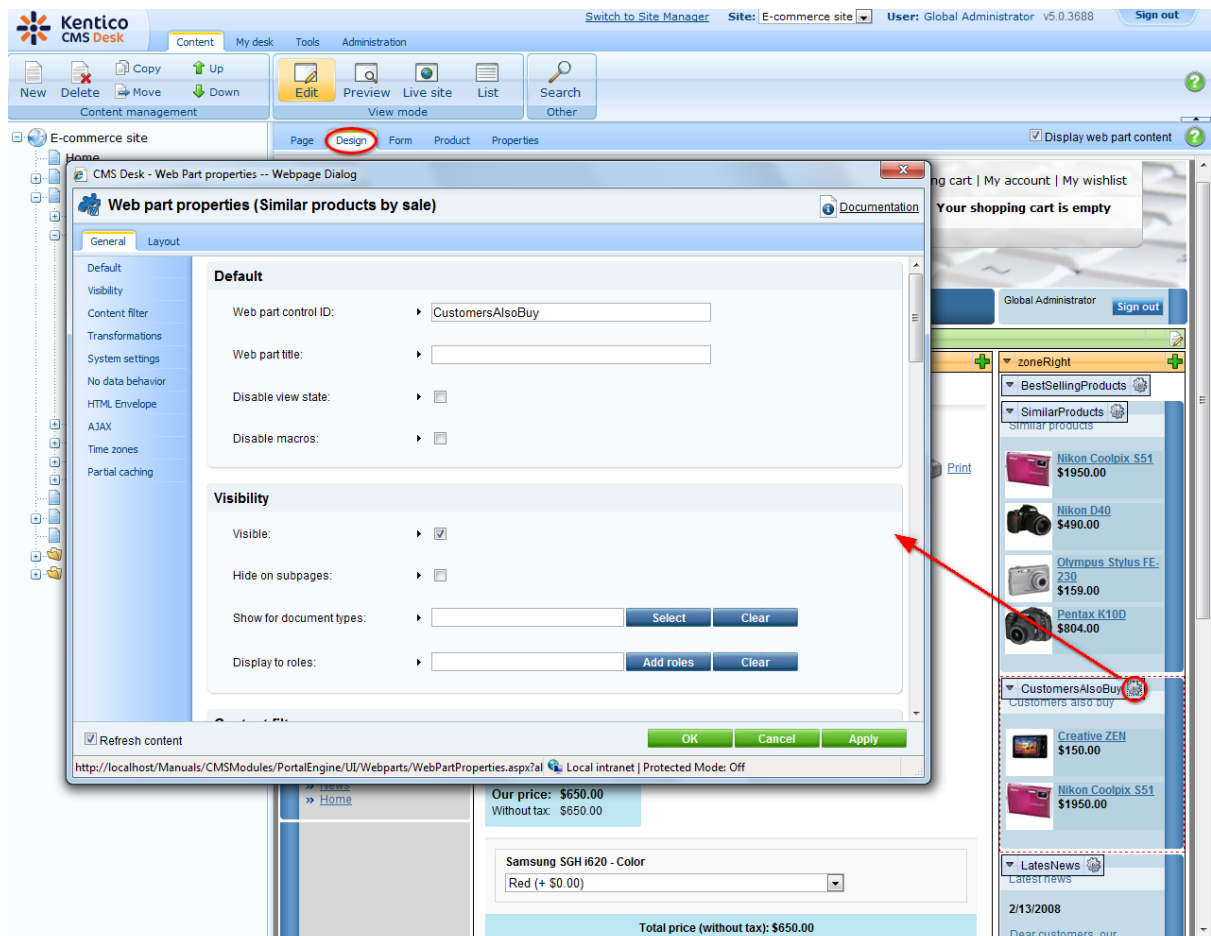
## 5.2.5 Similar products by sale

The **Similar products by sale** web part displays N products which other customers buy with the chosen product most commonly. By default, four products are displayed at the most.

For instance, if a customer has bought product A together with products B and C, some other customer who is viewing the details of product A is being displayed also with products B and C. This is because products B and C were purchased together with product A by the first customer.

The screenshot shows an e-commerce website for an "E-commerce starter site". The main navigation includes Home, News, Products, How to buy, Company, and Silverlight. The current page is "Products / Cell phones / Samsung SGH i620". The product details for the Samsung SGH i620 are displayed, including a product image, a list of parameters (Width: 94.9, Height: 59.3, Weight: 126g, etc.), and a price of \$650.00. A color selection dropdown is set to "Red (+ \$0.00)". The total price is \$650.00. On the right sidebar, the "Customers also buy" section is highlighted with a red box, showing two products: Creative ZEN for \$150.00 and Nikon Coolpix S51 for \$1950.00.

You can modify the functionality of **Similar products by sale** by setting following properties in the **Web Part properties** dialog:



## Content filter main properties

<b>Path</b>	Path of the documents to be displayed.
<b>Document types</b>	Types of documents that should be displayed, separated with a semicolon (;).
<b>Select top N products</b>	Selects only top N products. If blank, all items are selected..
<b>WHERE condition</b>	WHERE part of the SELECT query.
<b>ORDER BY expression</b>	ORDER BY part of the SELECT query.

## Transformations main properties

<b>Transformation name</b>	Transformation used in the list view mode.
----------------------------	--

### 5.2.6 Top N products by sale

The **Top N products by sale** web part displays the top N best-selling products. The best-selling products are chosen according to the frequency of their occurrence in customers' orders, not according



to the total volume of sales.

For instance, if a customer buys 10 items of the product A and 2 items of the product B and a second customer buys 4 items of the product B, the product B is evaluated as the best-selling.

You can modify the functionality of **Top N products by sale** by setting following properties in the **Web Part properties** dialog:

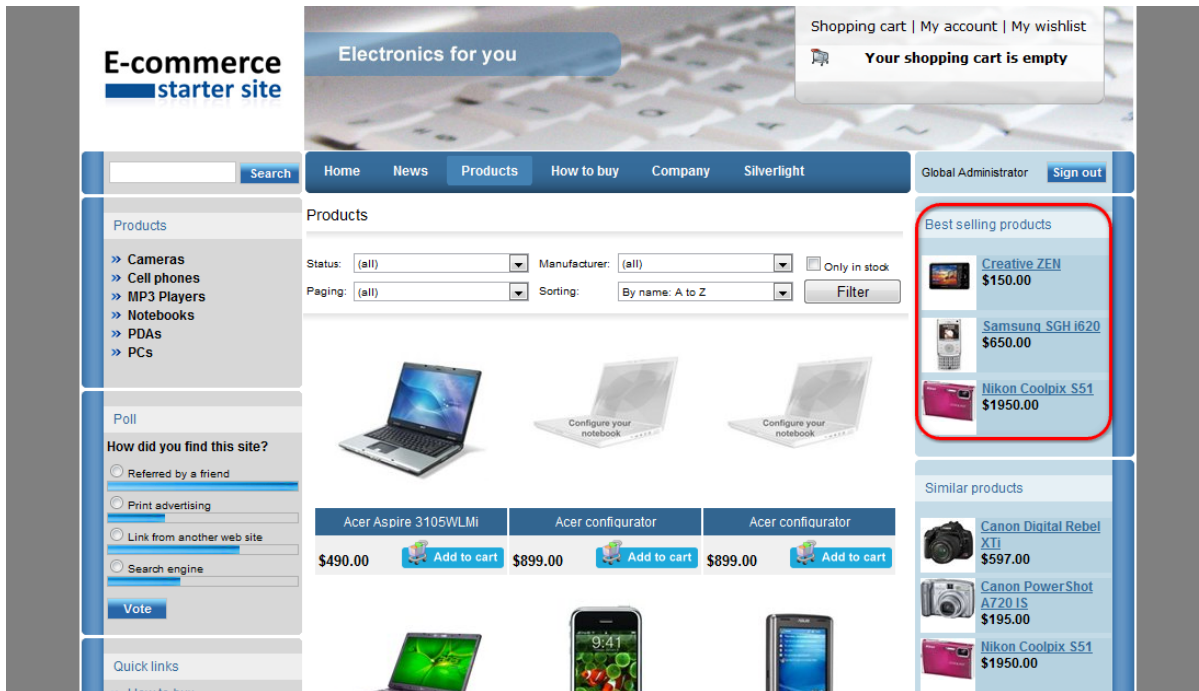
## Content filter main properties

<b>Path</b>	Path of the documents to be displayed.
<b>Document types</b>	Types of documents that should be displayed, separated with a semicolon (;).
<b>Select top N products</b>	Indicates how many best-selling products should be displayed.
<b>WHERE condition</b>	WHERE part of the SELECT query.
<b>ORDER BY expression</b>	ORDER BY part of the SELECT query.

## Transformations main properties

Transformation name

Transformation used in the list view mode.



On the sample E-commerce site at the **Products** page, the **Top N products by sale** web part is configured to display 4 products at most from the currently chosen products category. Its properties are set to the following values:

**Show for document types:** CMS.Menuitem

The given web part is displayed only in **CMS.Menuitem** documents. On the sample **E-commerce site**, the given web part is therefore displayed only in the list of products of a chosen category and is hidden in the product detail.

**Path:** ./%

Only best-selling products from the currently selected category are displayed. For more information about the path expressions please refer to Appendix B - Path expressions chapter in **Kentico CMS Developer's Guide**.

**Select top N products:** 4

The maximum of 4 best-selling products is displayed.

## 5.2.7 Wishlist

The **Wishlist** web part displays the wishlist of the currently logged-on user.

E-commerce starter site

Electronics for you

Shopping cart | My account | **My wishlist**

Your shopping cart is empty

Home News Products How to buy Company Silverlight Global Administrator Sign out

Search

Products

- » Cameras
- » Cell phones
- » MP3 Players
- » Notebooks
- » PDAs
- » PCs

Poll

How did you find this site?

- Referred by a friend
- Print advertising
- Link from another web site
- Search engine

Vote

Quick links

- » [How to buy](#)
- » [Company](#)
- » [News](#)
- » [Home](#)

Wishlist

**iPod Nano**

An anodized aluminum top and polished stainless steel back. Six eye-catching colors. A larger, brighter display with the most pixels per inch of any Apple display, ever. iPod nano stirs up visual effects from the outside in. And it'll wow you for hours. Play up to 5 hours of video or up to 24 hours of audio on a single charge.1 All that staying power and a wafer-thin, 6.5-mm profile makes iPod nano one small big attraction.

our price: \$199.00

[Remove from wishlist](#) [Add to cart](#)

**Acer Aspire 3105WLMi**

Great choice for mobile professionals who like to play as hard as they work--and who have an eye on the budget. Acer's acclaimed Folio design features chic lines, smooth curves, and a cool metallic sheen, while its slim-and-light form factor facilitates excellent mobility, while front-access ports and controls make connecting to a network convenient.

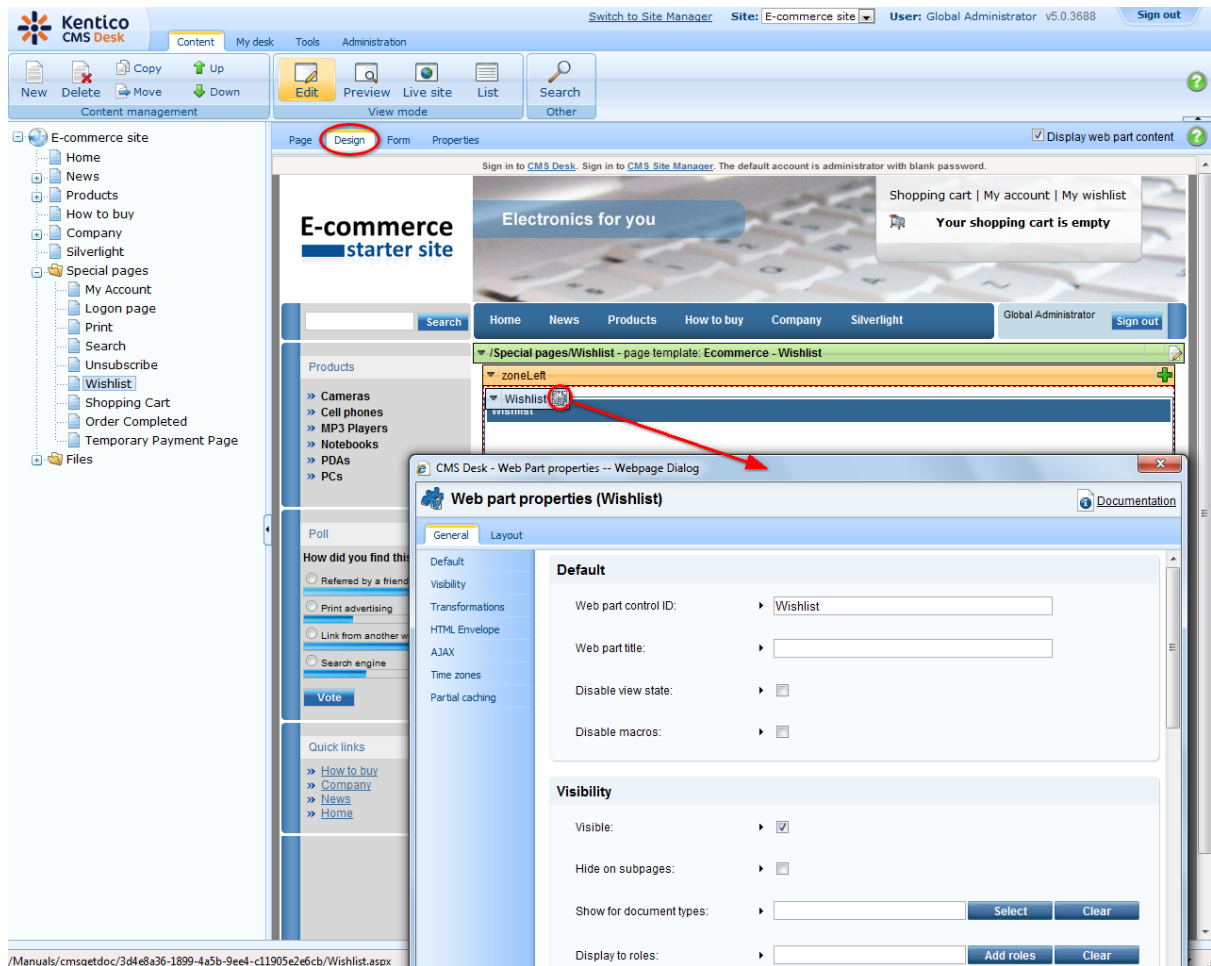
our price: \$490.00

[Remove from wishlist](#) [Add to cart](#)

[Continue shopping](#)

POWERED BY Kentico

You can modify the functionality of **My wishlist** by setting following properties in the **Web Part properties** dialog:



## Main properties

Transformation name	Transformation used in the list view mode.
---------------------	--

## 5.3 Displaying and resizing images

### 5.3.1 Overview

If you want to use a picture multiple times on your website, each time in different sizes, it is sufficient to upload it just once. KenticoCMS is able to resize it for you. Please consider that the size of a picture can only be decreased, not increased, though. For this reason, it is important that you upload your pictures in the maximum size you want to use on your website.

### 5.3.2 Displaying images

#### 1. How to display image in transformation

To get an image in the given size, you need to insert one of the following methods into the

transformation:

**a) Getting image by its attachment GUID**

**GetImage**(object attachmentGuidColumn, object maxSideSize, object width, object height, object alt)

**GetImage**(object attachmentGuidColumn)

**GetImage**(object attachmentGuidColumn, int maxSideSize)

**GetImage**(object attachmentGuidColumn, int width, int height)

**b) Getting image by its URL**

**GetImageByUrl**(object imageUrl, object maxSideSize, object width, object height, object alt)

**GetImageByUrl**(object imageUrl)

**GetImageByUrl**(object imageUrl, int maxSideSize)

**GetImageByUrl**(object imageUrl, int width, int height)

These methods use following parameters:

**attachmentGuidColumn** - attachment GUID

**imageUrl** - image url

**maxSideSize** - required image max side size

**width** - required image width

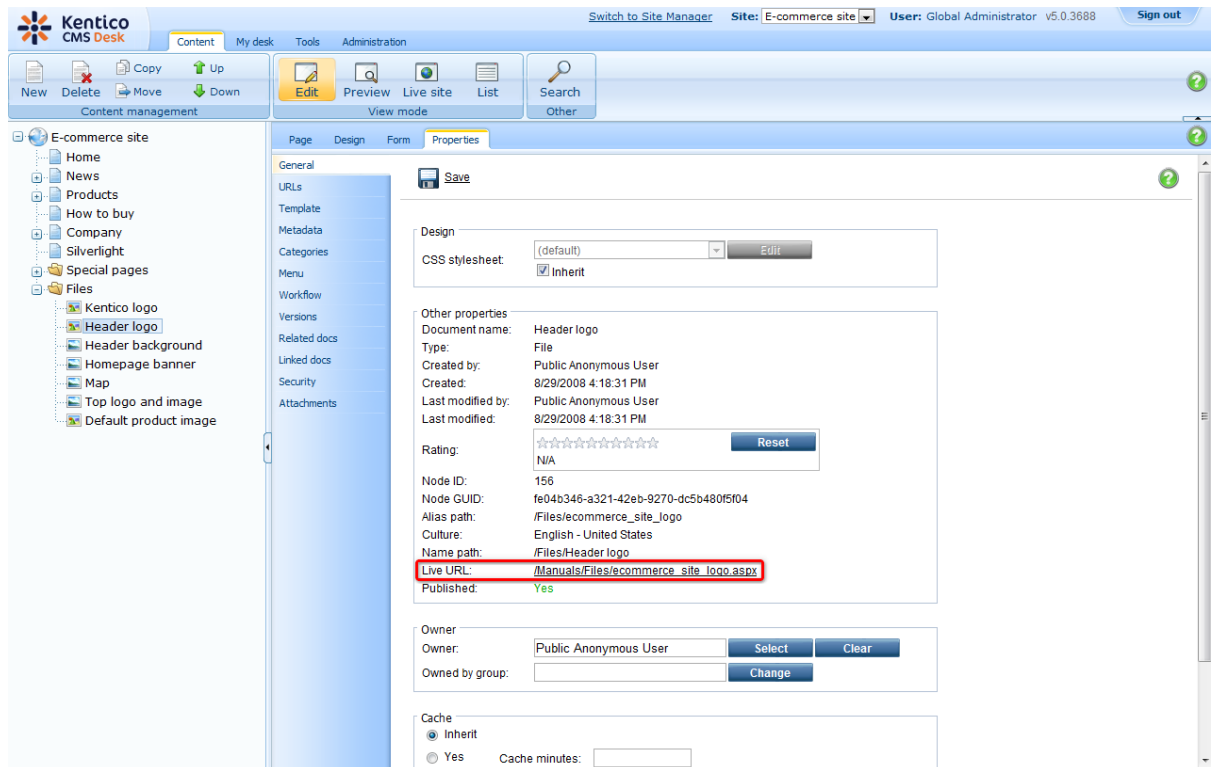
**height** - required image height

**alt** - image alternate text

All the methods generate HTML code for inserting an image according to given parameters.

**Examples of displaying image in transformation by its URL**

**a) Using Live URL**



```
<%#GetImageByUrl(~/Files/e-commerce_site_logo.aspx )%>
```

b) Using the field value of the given document type that represents the image URL (e.g. SKUImagePath):

### **Examples of displaying image in transformation by its attachment GUID**

a) Using the field value of the given document type that represents the attachment GUID


In the following example, we have defined the new document type **Employee**. This type has two attributes: **Employee name** specified in the **EmployeeName** column and **Employee photo** specified in the **EmployeePhoto** column.


Page Design **Form** Properties


Save Spell check

EmployeeID: 1

Employee name: David Simons

Employee photo:  david\_simons.gif  
Upload:  Browse...

Publish from:   Now

Publish to:   Now

To display the employee photo, you need to insert the **GetImage** method with the following syntax into the transformation.

```
<%#GetImage( Eval ( "EmployeePhoto" ) ) %>
```

## 2. How to display product image in transformation

To get a product image in the given size, you have to insert the **GetProductImage** method with the following syntax into the transformation:

**GetProductImage**(object imageUrl, object alt)

**GetProductImage**(object imageUrl, object width, object height, object alt)

**GetProductImage**(object imageUrl, object maxSideSize, object alt)

An example of displaying an image without resizing:

```
<%#EcommerceFunctions.GetProductImage( Eval ( "SKUImagePath" ), Eval ( "SKUName" ) ) %>
```

The **GetProductImage** method generates HTML code that will insert an image into your page. If the product image is not set, the method displays the default product image (see [Sitemanager settings](#) for more details) in the given size.



Please see the examples in the [Resizing images](#) chapter for more details.



### Storing images

For the best performance while loading images, please ensure images are enabled to be stored in the file system. For more details, please refer to the [Storing images](#) chapter.

### 5.3.3 Storing images

To improve the performance while loading images, please check **Store Files in File System** at **CMS Site Manager -> Files**.

Checking this option enables images to be stored in the file systems. Thus, anytime an image of the given size is needed, it can be loaded directly from the file system and doesn't have to be resized again from the original picture.

To be able to store images in the file system, the modify permissions have to be set for the whole web project (See the **Disk permissions problems** chapter in **Kentico CMS Developer's Guide** for more details).

All metafiles and attachments are stored in the folder specified at **CMS Site Manager -> Settings -> Files -> Files folder**. If the **Files folder** is not set, the following happens:

- The object metafiles that are not assigned to the specific website are stored in **<web project folder>/CMSFiles**.
- The object metafiles that are assigned to the specific website are stored in **<web project folder>/<site name>/metafiles**.
- Attachments (they are always assigned to the specific website) are stored in **<web project folder>/<site name>/files**.

### 5.3.4 Resizing images

This chapter explains how to resize product images. However, all images in general can be resized in the following manner with usage of the methods for displaying images (**GetImage()** and **GetImageByUrl()**) with appropriate parameters.

You can set a size of the original image (width: 300px, height: 150px) in the following ways:





## 1. Setting Maxsize value

This sets the longer side to the value specified as **Maxsize**. The other side is calculated automatically so that the aspect ratio remains the same with the original image. In the following image, **Maxsize** is set to 250px.

The **GetProductImage** method for resizing an image to the Maxsize value of 250px:

```
EcommerceFunctions.GetProductImage(Eval("SKUImagePath"), 250, Eval("SKUName"))
```



## 2. Setting Height value

This sets the height of an image. The width is calculated automatically so that the aspect ratio remains the same with the original image. In the following image, **Height** is set to 100px.

```
EcommerceFunctions.GetProductImage(Eval("SKUImagePath"), 0, 100, Eval("SKUName"))
```



### 3. Setting Width value

This sets the width of an image. The height is calculated automatically so that the aspect ratio remains the same with the original image. In the following image, **Width** is set to 150px.

```
EcommerceFunctions.GetProductImage(Eval("SKUImagePath"), 150, 0, Eval("SKUName"))
```



### 4. Setting Width and Height values

The given values are set on condition they are not greater than the original size of an image. Please note that the aspect ration may not be maintained. In the following image, **Width** is set to 100px and **Height** is set to 150px.

```
EcommerceFunctions.GetProductImage(Eval("SKUImagePath"), 100, 150, Eval("SKUName"))
```



In the following image, **Width** is set to 300px and **Height** is set to 50px.

```
EcommerceFunctions.GetProductImage(Eval("SKUImagePath"), 300, 50, Eval("SKUName"))
```



If at least one value is greater than the original size of an image, the original size is set. In the following image, **Width** is set to 600px and **Height** is set to 300px.

```
EcommerceFunctions.GetProductImage(Eval("SKUImagePath"), 600, 300, Eval("SKUName"))
```



## 5.4 Tools

### 5.4.1 Overview

Kentico CMS includes a wide range of modules which can be used to build your e-commerce solution.

The included modules are:

- Bizforms
- Blogs
- Booking system
- Content staging
- Event calendar
- File import
- Forums
- GeoMapping
- Image gallery
- Messaging
- Newsletters
- Polls

- Reporting
- RSS feeds
- User contributions
- Web analytics
- Web farm synchronization

Please note that you are not limited to the pre-defined modules. You can easily create your own modules with the desired functionality. Please see the **Custom modules** chapter in **Kentico CMS Developer's Guide** for more details.

Please note that not all versions of Kentico CMS support all modules. For more information about modules which are included in the specific version of Kentico CMS please see the feature matrix at Kentico website (<http://kentico.com/cms-asp-net-features/Feature-Matrix.aspx>).

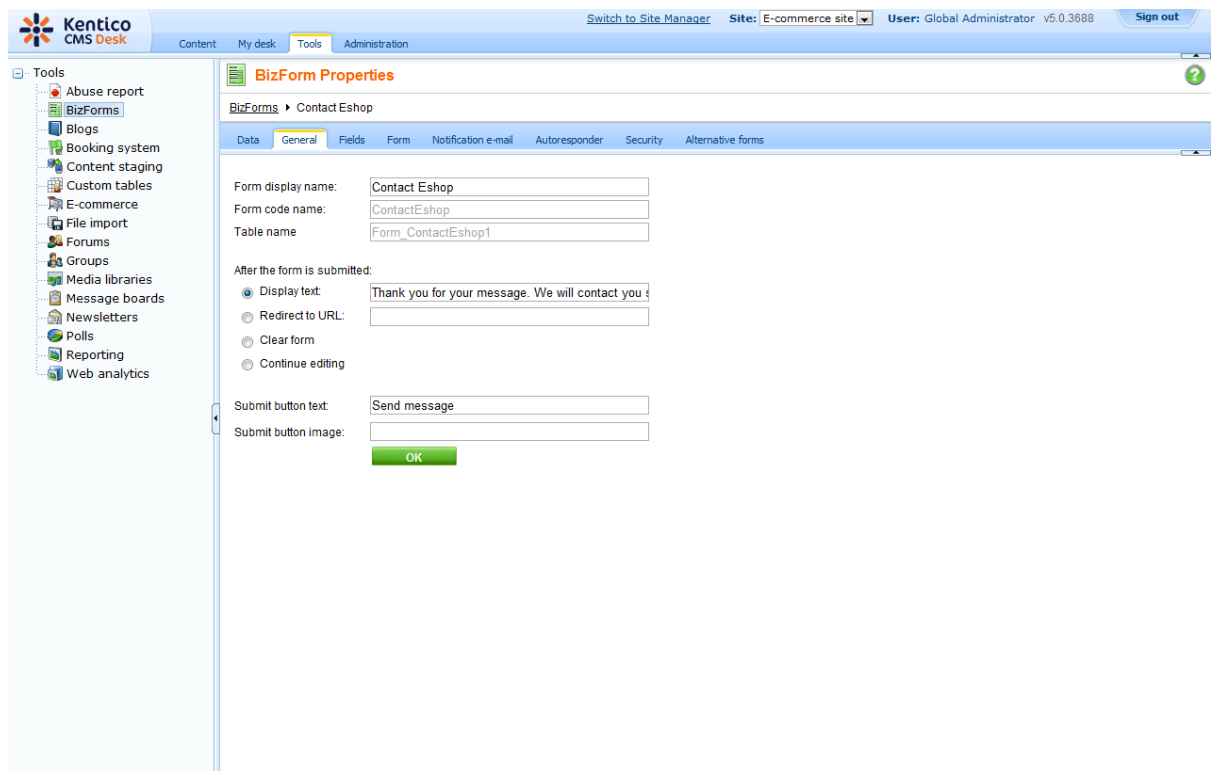
The screenshot shows the Kentico CMS Desk interface. The top navigation bar includes 'Switch to Site Manager', 'Site: E-commerce site', 'User: Global Administrator v5.0.3688', and 'Sign out'. The main content area is titled 'Tools' and contains the following sections:

- Abuse report**: In this section, you can view a list of abuse reports. Each of the reports is provided with a link to the page from that the abuse was reported. Reports can be marked as solved or rejected. They can also be edited and deleted here.
- BizForms**: This section displays a list of BizForms defined for the given site. You can either create new form, or edit, delete or export objects from the existing ones. For each BizForm, you can define its general information, fields, layout, whether the e-mail notification should be sent and others.
- Blogs**: This section displays a list of blogs owned by the logged-on user and the list of comments waiting for her approval. You can choose to edit blog or create a new post. In the list of comments, you can either edit, delete or approve comment. Approved comment will be displayed on the live site.
- Booking system**: This section displays a list of booked events defined for the given site. You can either view detail of each event or be redirected to it. You can manage attendees for a given event and send emails to all of them. For each event are displayed the following properties: its name, creation time, maximum capacity, number of attendees and open from/to dates.
- Content staging**: This section displays a tree view of documents or objects of the given site and a list of synchronization tasks. You can choose either to select and synchronize or delete a group of tasks or view, restore or delete an individual task. You can specify the content staging servers as well.
- Custom tables**: This section displays a list of custom tables. These are users' custom database tables that can be used in cases when storing data in the tree structure is inefficient.
- E-commerce**: This section contains information about data and settings of the e-commerce module. You can manage your orders, products, product options, manufacturers, suppliers, discount coupons and discount levels. You can choose to display reports or specify e-commerce settings as well.
- File import**: This section displays the file import interface. You can choose to start import or specify file import settings, e.g. target alias path or culture. The path to a directory, from which are files imported is displayed as well.

## 5.4.2 BizForms

The BizForms module enables you to create and publish simple on-line forms without writing a single line of code. For instance, you can manage communication with your customers more effectively using the **Contact us** form.

You can edit BizForms properties at **CMS Desk -> Tools -> BizForms -> <edit BizForm>**.

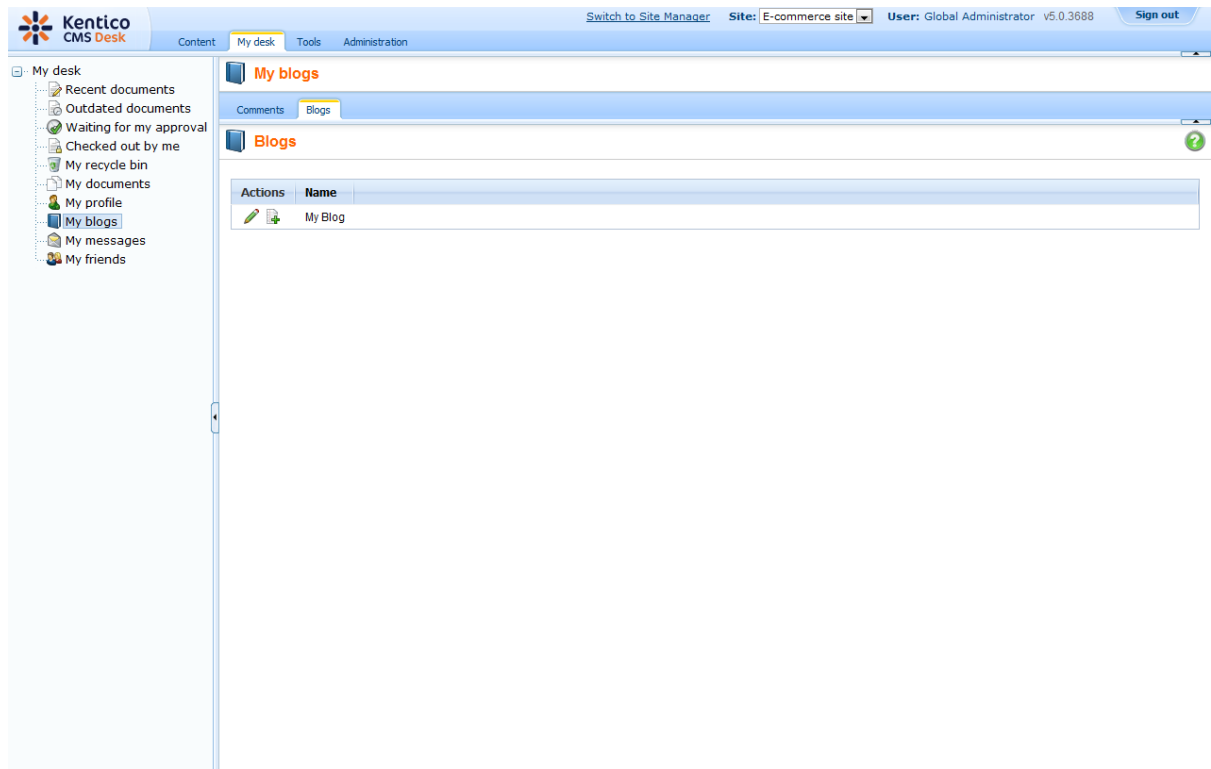


For more details and information about BizForms please refer to the **Module BizForms** chapter in **Kentico CMS Developer's guide**.

### 5.4.3 Blogs

The Blogs module allows you to publish a personal or company blog. You can publish multiple blogs on the same site and there can be multiple editors for each blog.

You can edit blogs at **CMS Desk -> My Desk -> My Blogs**.

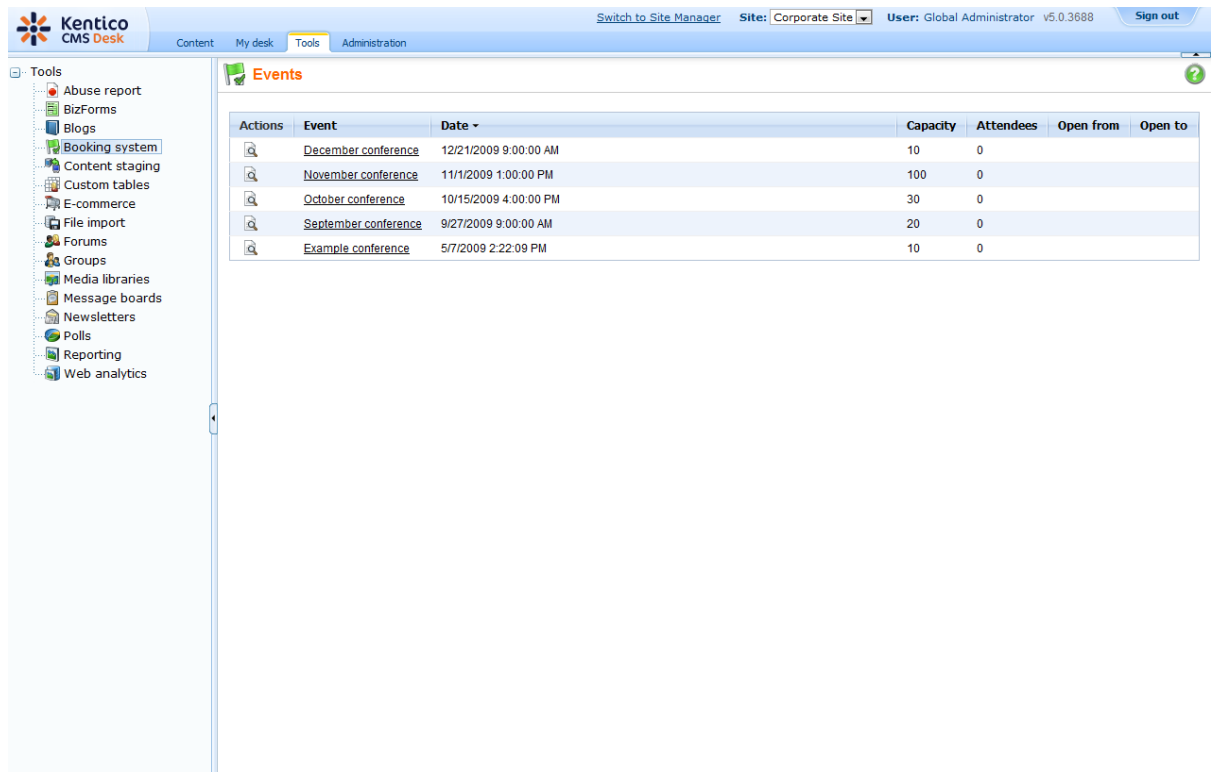


For more details and information about blogs please refer to the **Module Blogs** chapter in **Kentico CMS Developer's guide**.

#### 5.4.4 Booking system

The booking system enables you to create an event and manage its attendees. After attendees are added, you can send e-mails to them with the important information about the event. The booking system can be used both for on-line and off-line meetings.

You can edit events and add attendees at **CMS Desk -> Tools -> Booking system -> <view event>**.



The screenshot shows the Kentico CMS Desk interface. The top navigation bar includes the Kentico CMS Desk logo, a 'Switch to Site Manager' link, a 'Site: Corporate Site' dropdown, a 'User: Global Administrator v5.0.3688' status bar, and a 'Sign out' link. The main navigation tabs are 'Content', 'My desk', 'Tools', and 'Administration'. The 'Tools' tab is active, and the 'Events' module is selected, displaying a table of events.

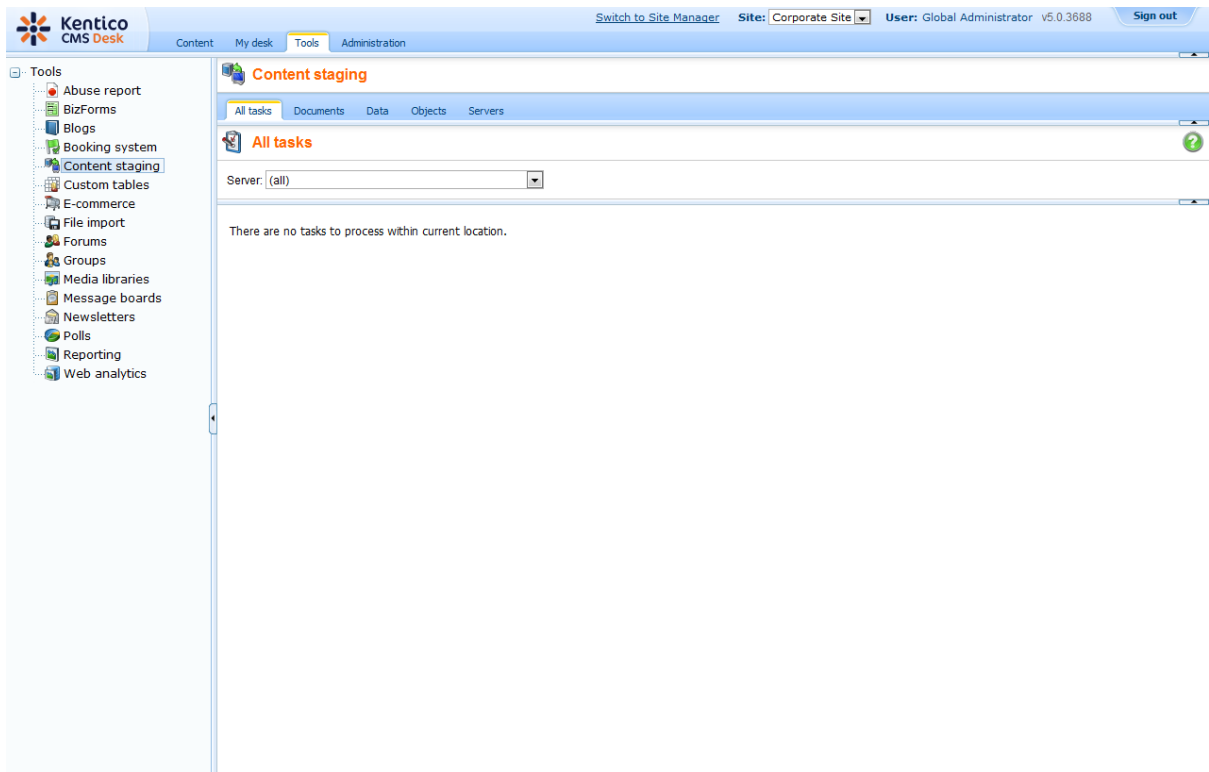
Actions	Event	Date	Capacity	Attendees	Open from	Open to
	<a href="#">December conference</a>	12/21/2009 9:00:00 AM	10	0		
	<a href="#">November conference</a>	11/1/2009 1:00:00 PM	100	0		
	<a href="#">October conference</a>	10/15/2009 4:00:00 PM	30	0		
	<a href="#">September conference</a>	9/27/2009 9:00:00 AM	20	0		
	<a href="#">Example conference</a>	5/7/2009 2:22:09 PM	10	0		

For more details and information about Booking system please refer to the **Module Booking system** chapter in **Kentico CMS Developer's guide**.

## 5.4.5 Content staging

The **Content staging** module allows you to store separately the content in development, staging (editing) and production (live) environment. It allows you to easily transfer document and object changes to another server or make a complete synchronization of documents and objects from one server to another.

You can manage content staging at **CMS Desk -> Tools -> Content staging**.



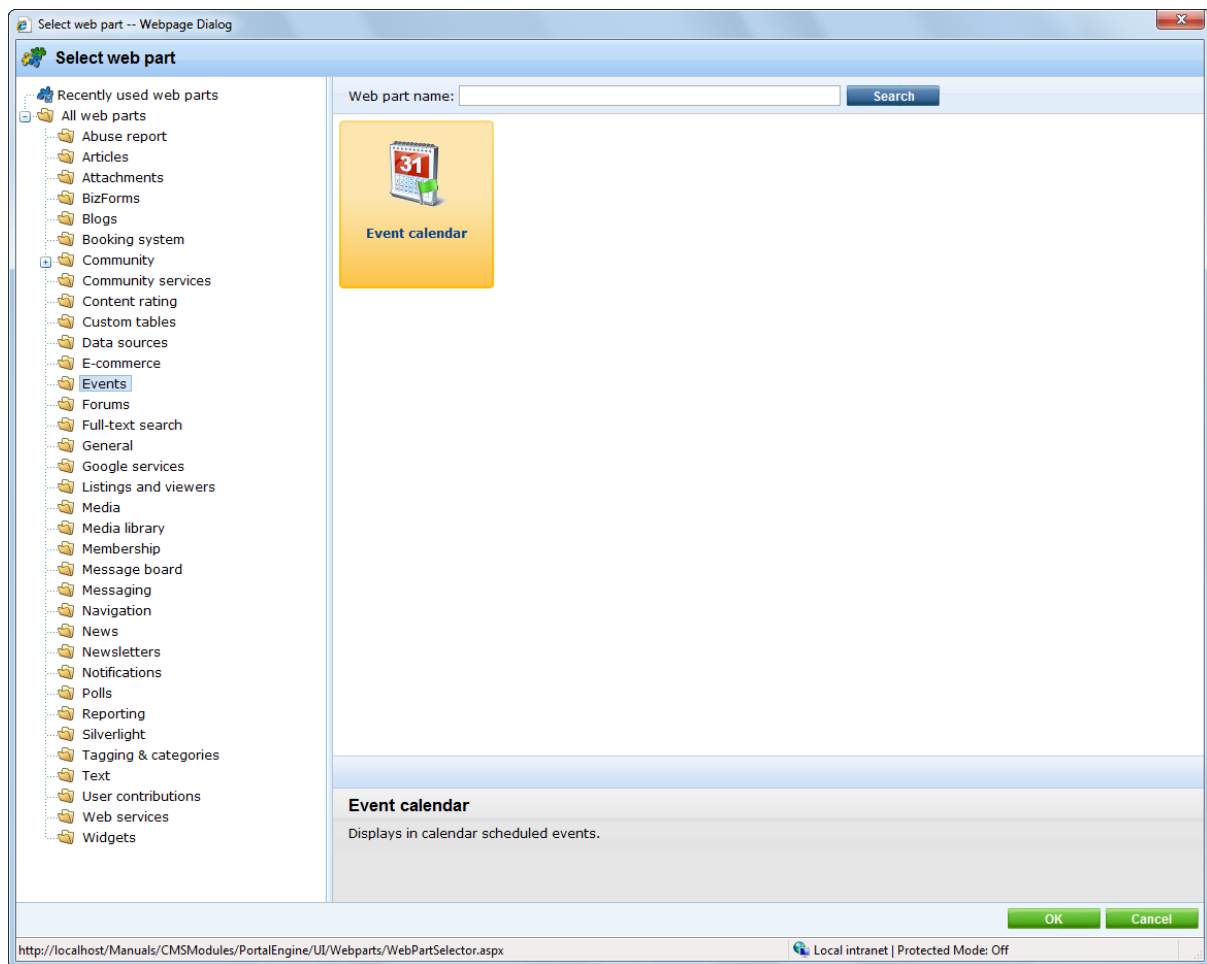
For more details and information about Content staging please refer to the **Module Content Staging** chapter in **Kentico CMS Developer's guide**.

### 5.4.6 Event calendar

You can use the **Event calendar** module for displaying documents ordered by date in a calendar. Various document types can be displayed in the calendar, depending on the settings.

For the **Event calendar** module implementation, use the **Events -> Event calendar** web part.



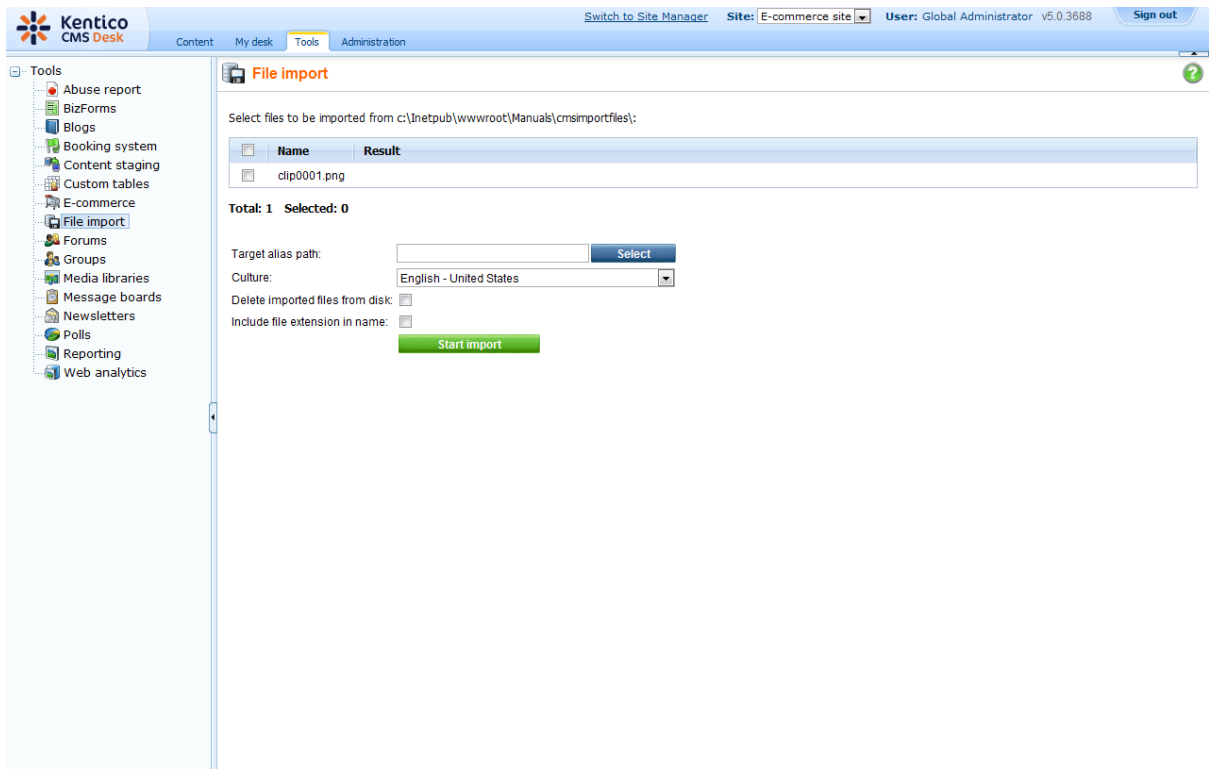


For more details and information about **Event calendar** please refer to the **Module Event calendar** chapter in **Kentico CMS Developer's guide**.

### 5.4.7 File import

The **File import** module allows you to import many files including their folder structure from the disk to Kentico CMS content repository, so that you do not have to upload the one-by-one using the user interface.

You can configure the path to your import directory at **CMS Desk -> Tools -> File import**.

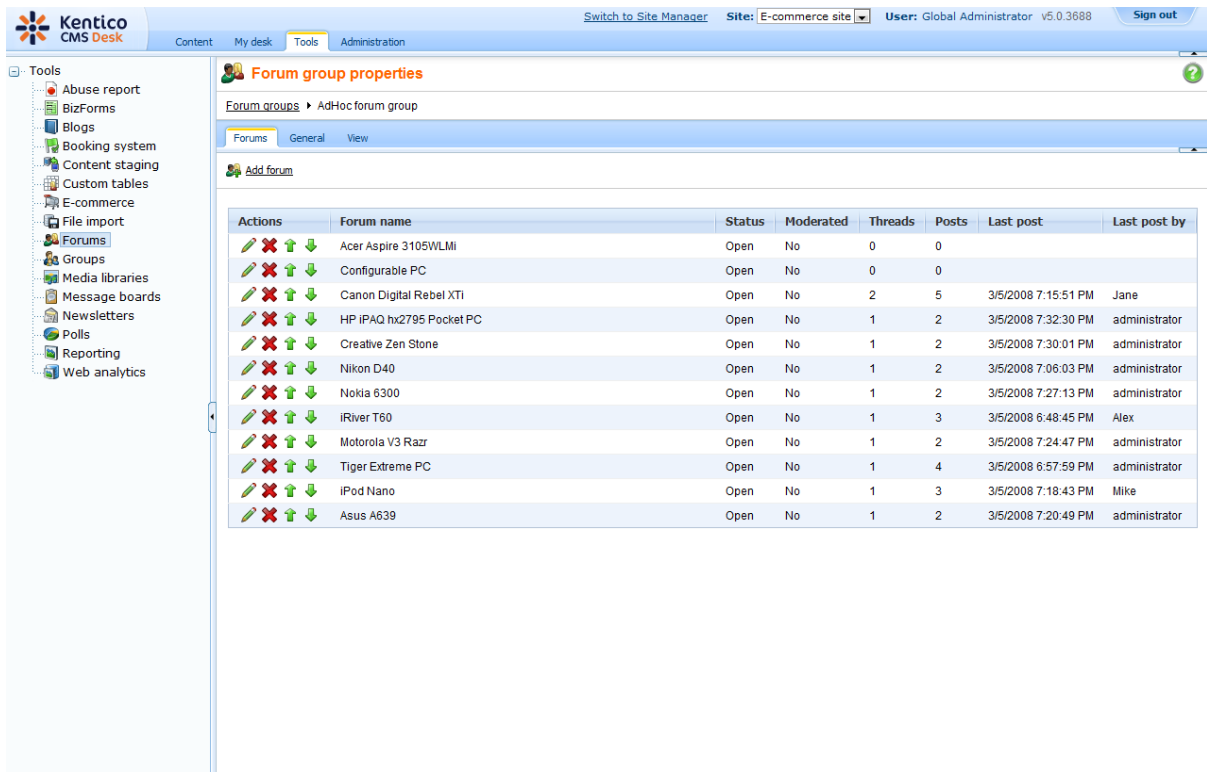


For more details and information about **File import** please refer to the **Module File import** chapter in **Kentico CMS Developer's guide**.

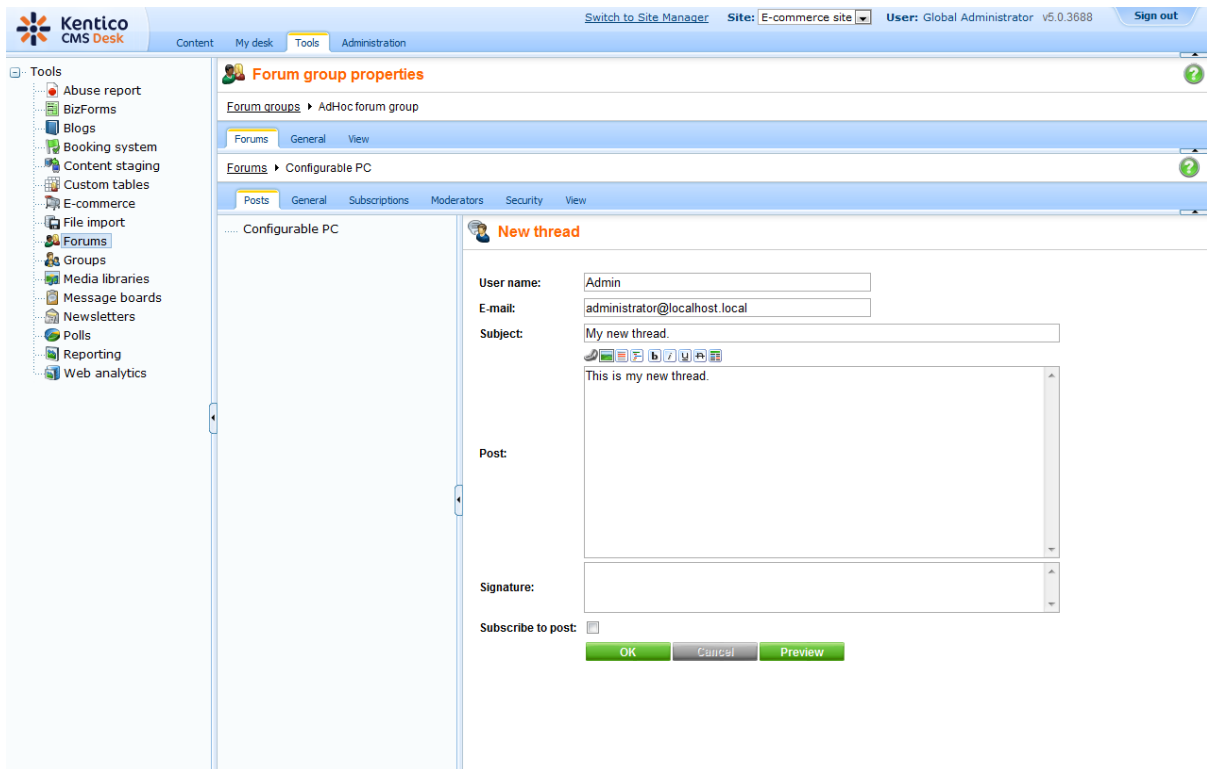
## 5.4.8 Forums

The Forums module allows your customers to add comments and feedback to each product.

You can edit forums on your website at **CMSDesk -> Tools -> Forums -> <edit forum group>**.



After choosing a forum group, you can **Edit** () , **Delete** () , **Move up** () and **Move down** () individual forums.



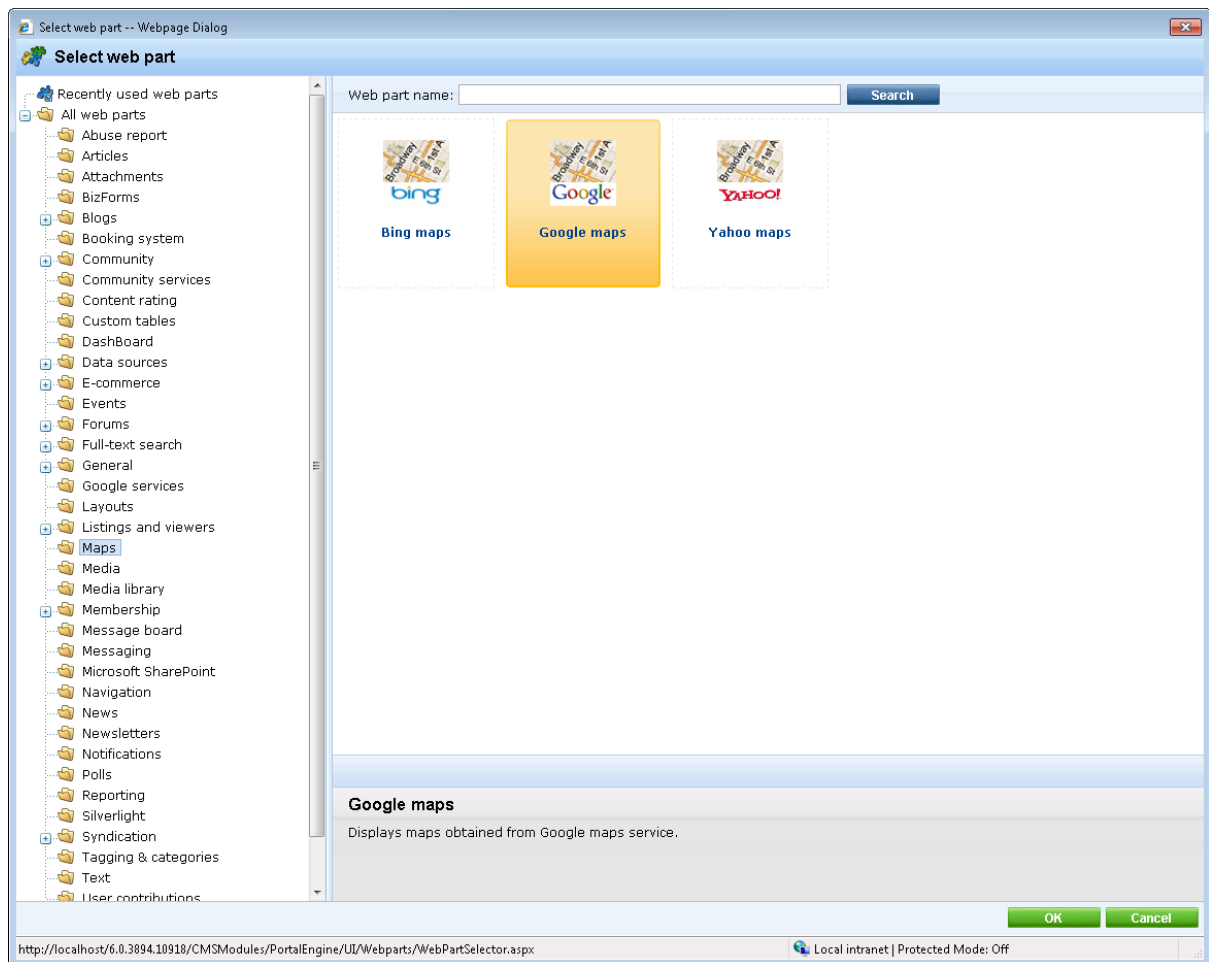
After you choose a forum to edit you can add, edit and delete threads and posts at of the given forum.

For more information please refer to the **Module Forums** chapter in **Kentico CMS Developer's guide**.

### 5.4.9 Geo mapping

The Geo mapping module uses Google Maps to display content on the map. You can use this module to display your offices, your store, your partners or real estates you offer on a map.

For the Geo mapping module implementation, use the **Maps -> Google Maps** web part.

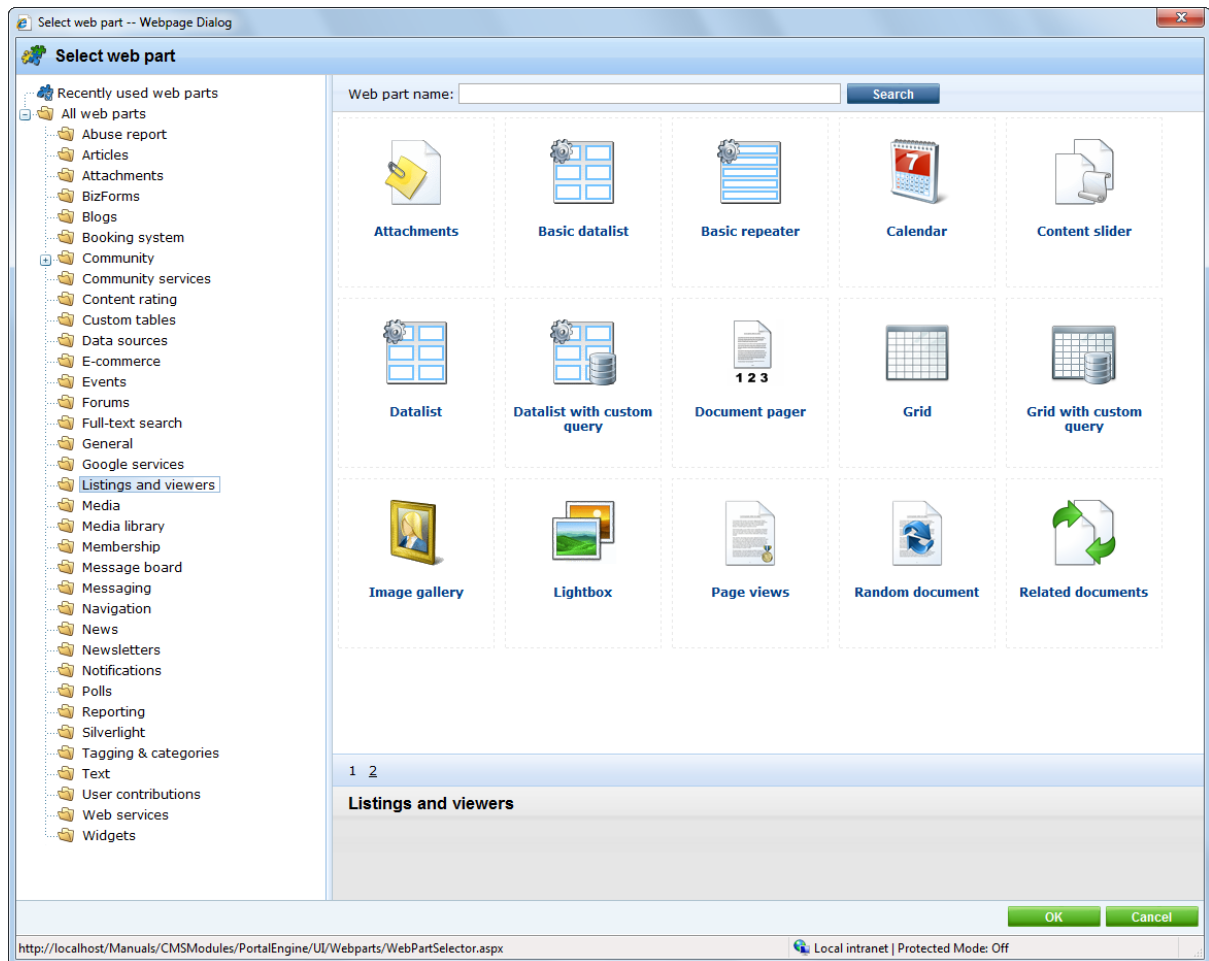


For more details and information about the Geo mapping module, please refer to the Geo mapping chapter in Kentico CMS Developer's Guide.

### 5.4.10 Image Gallery

Using the **Image Gallery** module, you can create easily image gallery pages. The module encompasses three page templates and three web parts suitable for creating image galleries.

For the implementations of **Image Gallery**, use the **Content slider**, **Image gallery** and **Lightbox** web parts.

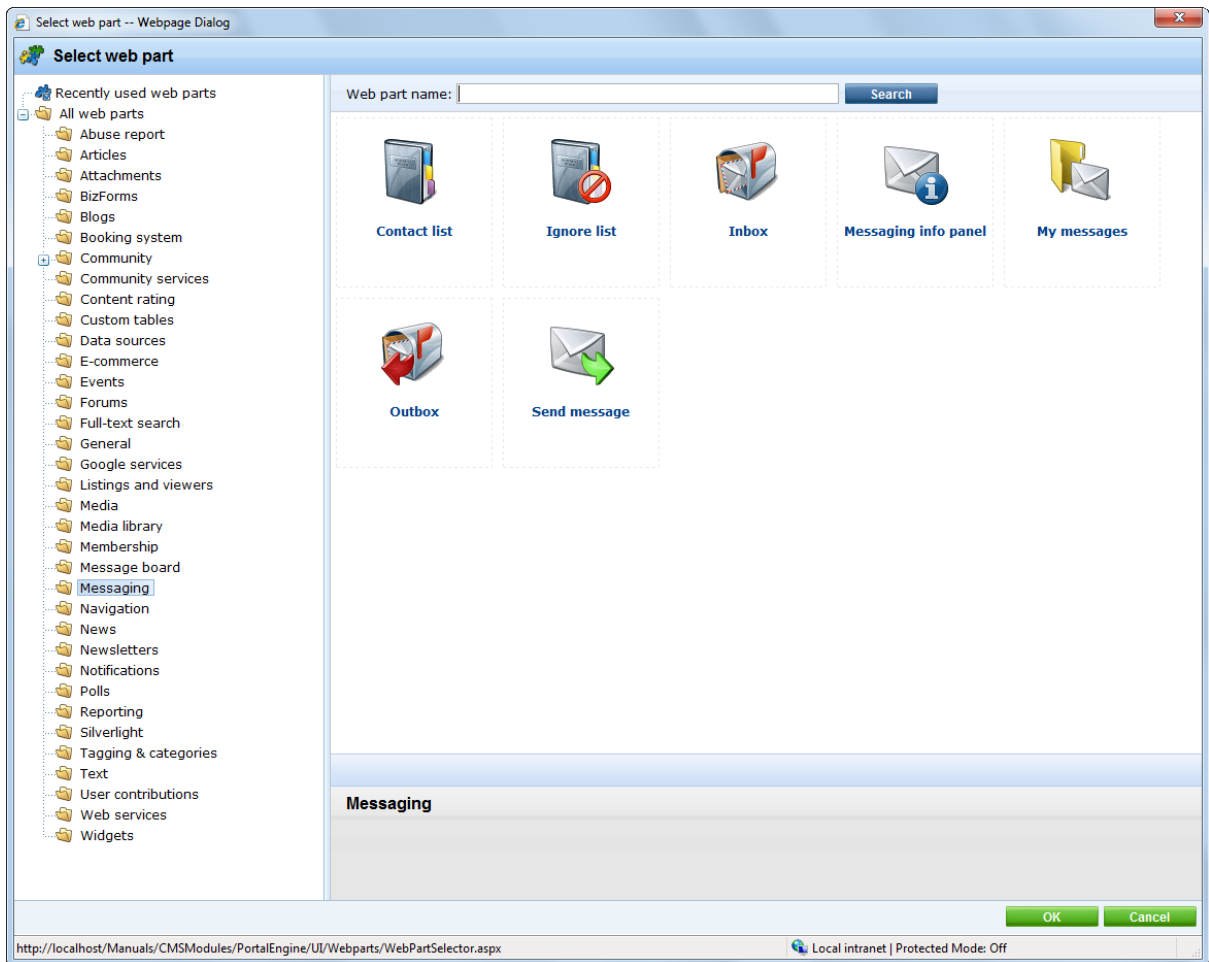


For more details and information about **Image Gallery** please refer to the **Module Image Gallery** chapter in **Kentico CMS Developer's guide**.

### 5.4.11 Messaging

The **Messaging** module allows you to send, receive and manage messages. Its purpose is to provide an internal way of communication with other users of the website.

For the **Messaging** module implementation, use the web parts from the **Messaging** folder in the **Web part selection** dialog.

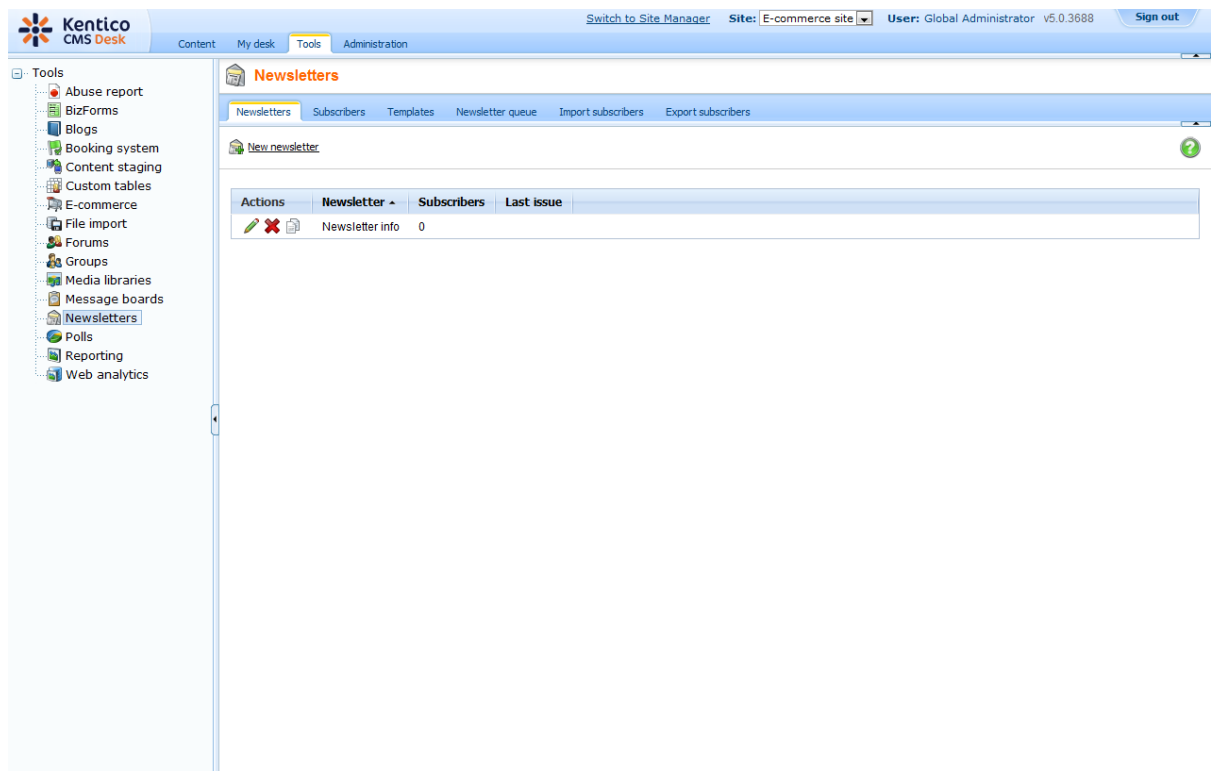


For more details and information about **Messaging** please refer to the **Module Messaging** chapter in **Kentico CMS Developer's guide**.

### 5.4.12 Newsletters

The Newsletter module allows you to author and mail out e-mail newsletters. Through newsletters, you can inform your customers about new products, current offers and discounts and much more. The newsletter can be static or dynamic.

You can edit newsletters at **CMS Desk -> Tools -> Newsletters**.

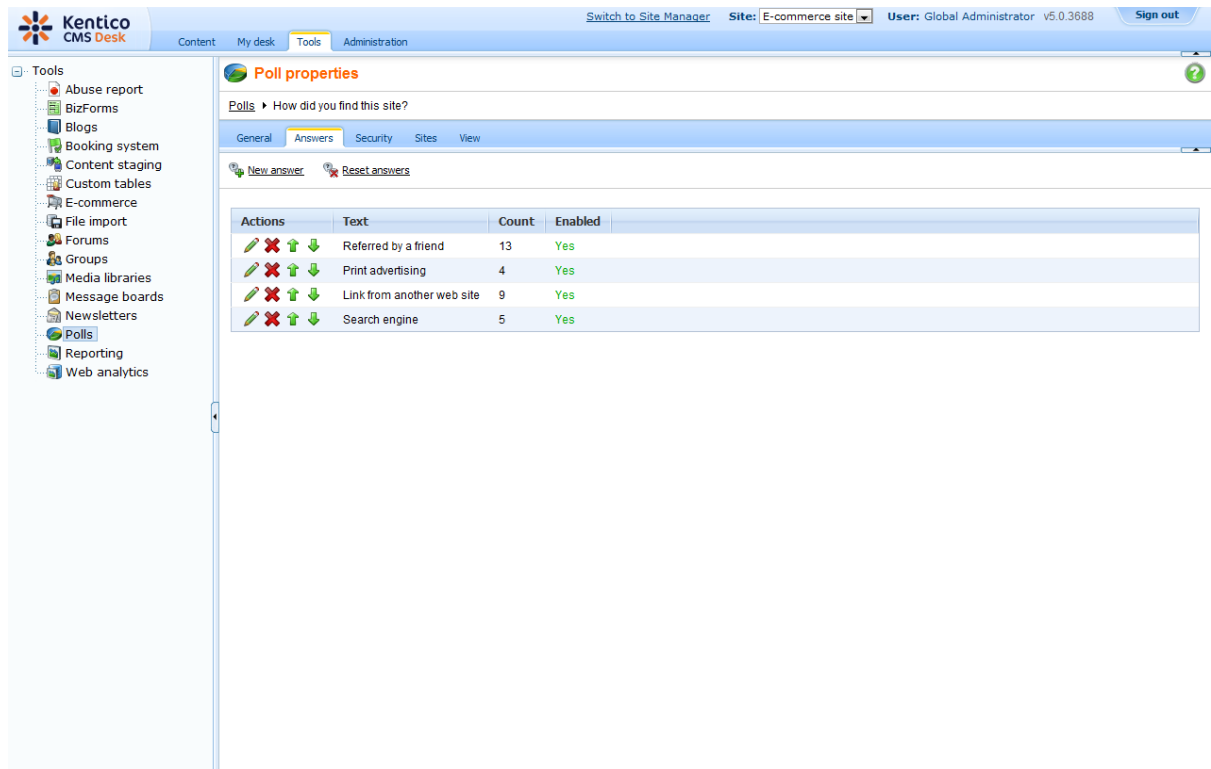


For more details and information about newsletters please refer to the **Module Newsletters** chapter in **Kentico CMS Developer's guide**.

### 5.4.13 Polls

The Polls module enables you to create and publish polls with single or multiple answer so that you are able to get vital information about how are your customers and what they want.

You can edit polls at **CMS Desk -> Polls -> <edit poll>**.



After you choose a poll to edit, you can set the poll question at the **General** tab and the poll answers at the **Answers** tab.

For more details and information about polls please refer to the **Module Polls** chapter in **Kentico CMS Developer's guide**.

#### 5.4.14 Reporting

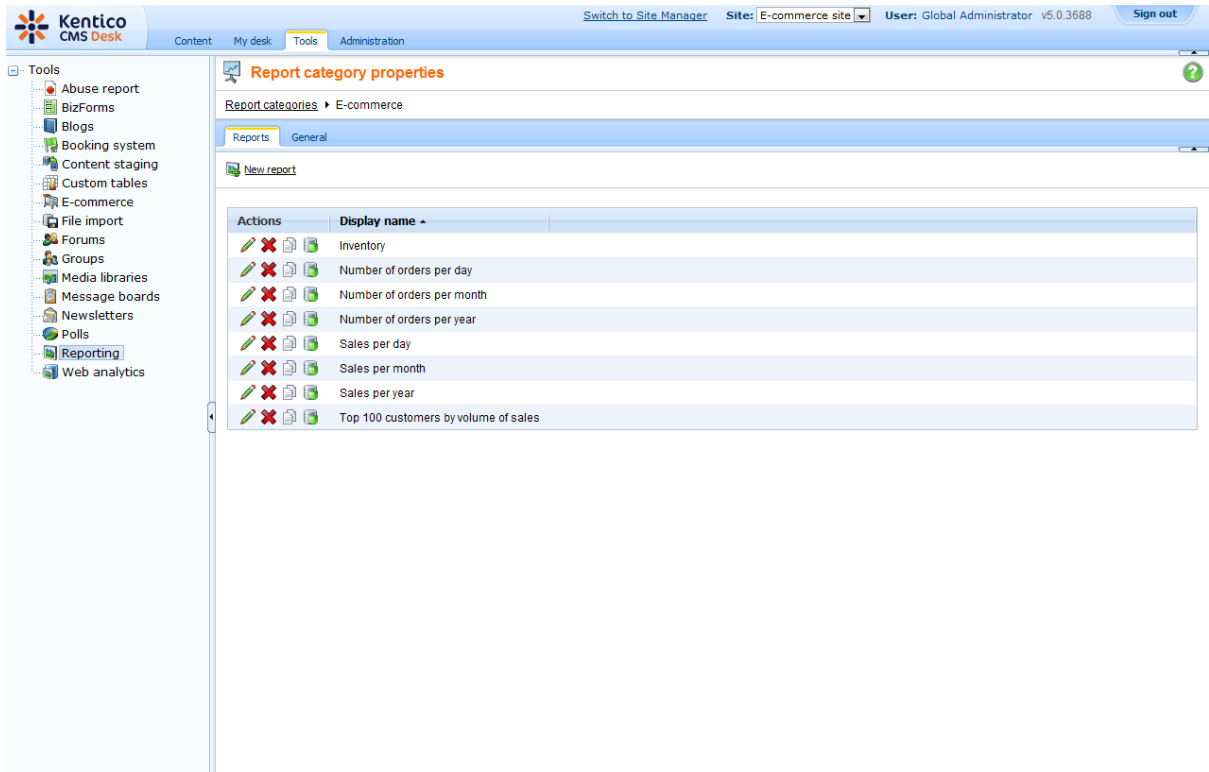
The Reporting module enables you to create internal reports to watch activity in the Kentico CMS system and on the website. Kentico CMS comes with several pre-defined e-commerce reports that show statistics related to your online business.

The following pre-defined e-commerce reports are available:

- Inventory
- Number of orders per day
- Number of orders per month
- Number of orders per year
- Sales per day
- Sales per month
- Sales per year
- Top 100 customers by volume sales

You can edit reports at **CMS Desk -> Tools -> Reporting -> <edit Report category>**.





After choosing a report category, you can **Edit** (✎) or **Delete** (✖) individual reports.

For more details and information about how to set report properties please refer to the **Module Reporting** chapter in **Kentico CMS Developer's guide**.

### 5.4.15 RSS Feeds

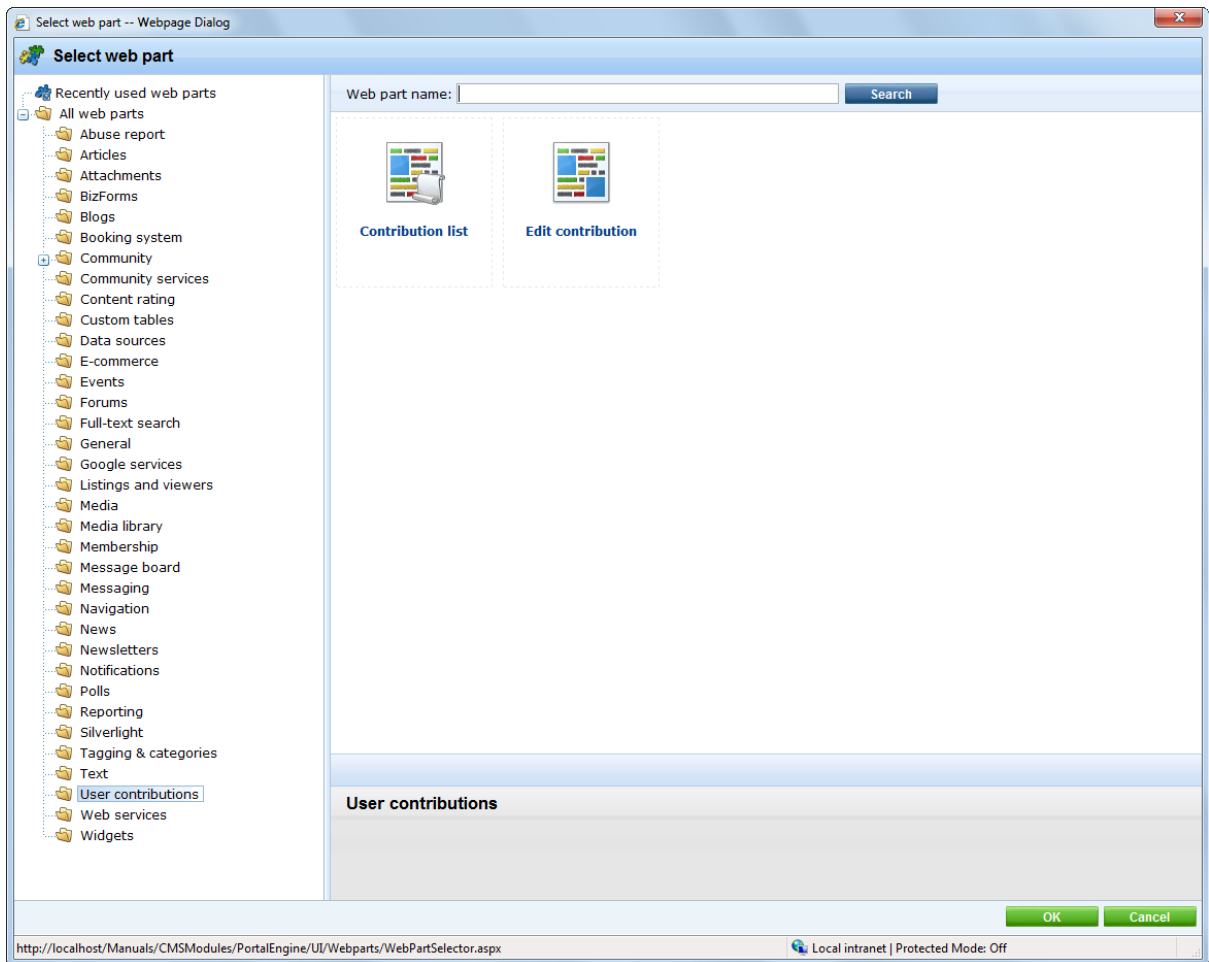
Kentico CMS allows you to publish content using RSS, Atom or XML feeds. This functionality is provided by a set of web parts belonging to the **Syndication** module.

For more details and information about feeds and how to create them for a different types of documents and objects, please refer to the [Modules -> Syndication \(RSS, Atom, XML\)](#) chapter of **Kentico CMS Developer's Guide**.

### 5.4.16 User Contributions

The **User Contributions** module allows you to create content editing interface for site members. Using this interface, the website visitors are able to create, edit and delete content, even if they are not editors and cannot access Kentico CMS Desk.

For the **User Contributions** module implementations, use the **Contribution list** and **Edit contribution** web part.

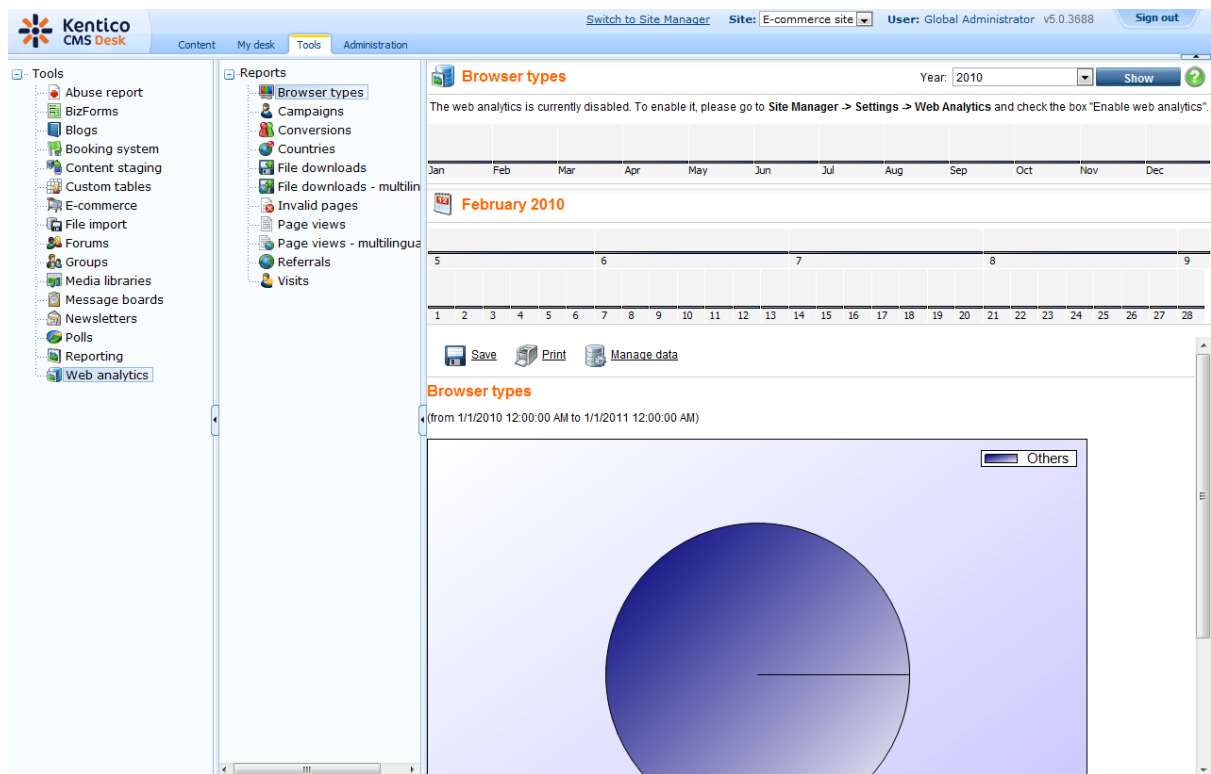


For more details and information about **User Contributions** please refer to the **Module User Contributions (Wiki)** chapter in **Kentico CMS Developer's guide**.

### 5.4.17 Web Analytics

The Web Analytics module enables you to track and analyze website visits, page views, file downloads and other metrics of the website.

You can modify view the Web Analytics reports at **CMS Desk -> Tools -> Web Analytics**.

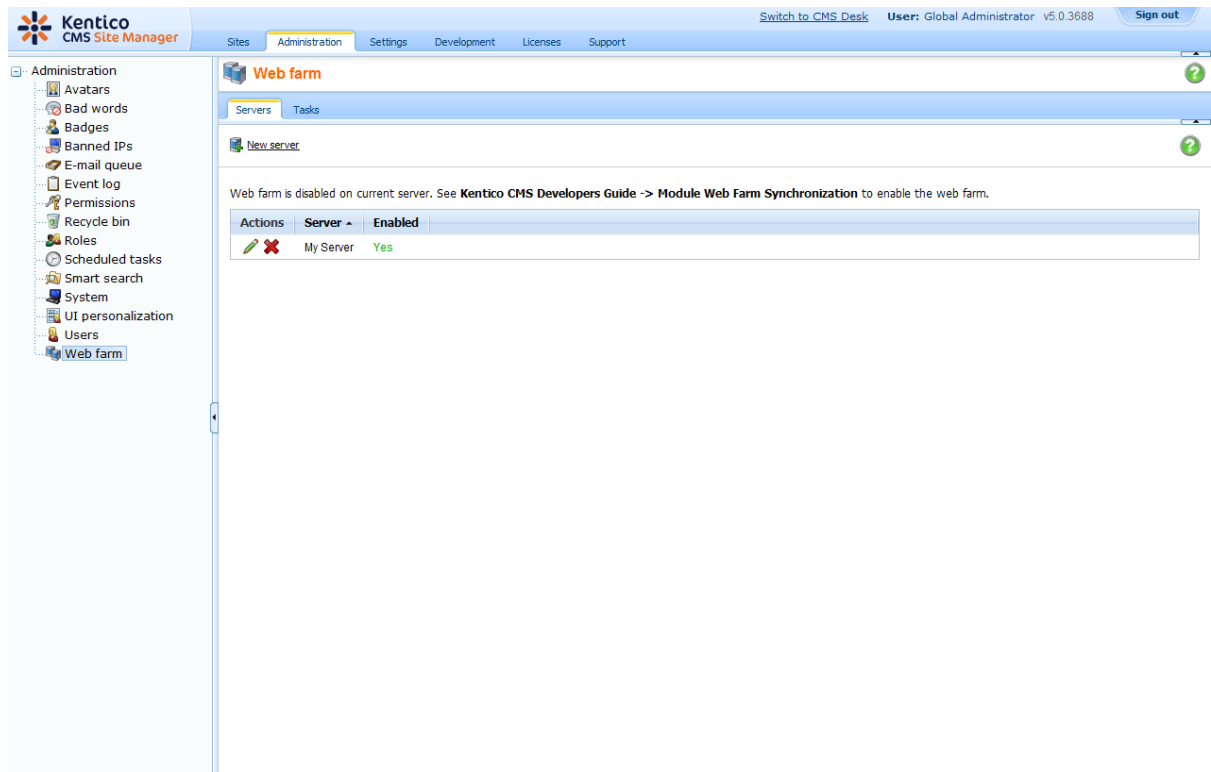


For more details and information about **Web Analytics** please refer to the **Module Web Analytics** chapter in **Kentico CMS Developer's guide**.

### 5.4.18 Web Farm synchronization

Kentico CMS offers native web farm support. The web farm support provides the synchronization of the changes made to the site settings on one of the servers to all the other servers and provides synchronization of the files uploaded to the site to all other servers.

You can manage web farm servers at **CMS Site Manager -> Administration -> Web farm**.



For more details and information about **Web Farm synchronization** please refer to the **Module Web Farm synchronization** chapter in **Kentico CMS Developer's guide**.

## 5.5 Adding items to the shopping cart

The items can be added to the shopping cart by calling the page with the **Shopping cart** web part with following URL parameters:

- **productId** - the ID of the product stored in the NodeSKUID database field (in the CMS\_Tree table) or SKUID (in the COM\_SKU table)
- **quantity** - the number of product units to be added to the shopping cart
- **options** - the string of the selected product option IDs separated by commas

It means your link will typically look like this:

```
<a href="~/shoppingcart.aspx?productId=10&quantity=1&options=12,24">Add to cart</a>
```

This link has been generated by the following **ShoppingCartItemSelector** control. All you have to do is adding the control to transformation and setting its properties accordingly.

### 1. Product detail

```
<%@ Register Src="~/CMSModules/Ecommerce/Controls/ProductOptions/ShoppingCartItemSelector.ascx" TagName="CartItemSelector" TagPrefix="uc1" %>
```

```
<uc1:CartItemSelector id="cartItemSelector" runat="server" SKUID='<%#
ValidationHelper.GetInteger(Eval("SKUID"), 0) %>' SKUEnabled='<%#
ValidationHelper.GetBoolean(Eval("SKUEnabled"), false) %>' AddToCartImageButton
="AddToCart.gif" ShowUnitsTextBox="true" ShowProductOptions="true"
AddToWishlistImageButton="AddToWishlist.gif" />
```

The control has the **ShowUnitsTextBox** and **ShowProductOptions** properties set to **true**. Therefore, the product options and units text box are visible on the website. Please note that **AddToWishlistImageButton** properties is set as well. Thus, the **Add to wishlist** button is displayed as well.

Please note: the `<% Register ... >` part needs to be placed at the beginning of the whole transformation code; otherwise, the transformation will not be functional

The screenshot shows a web page for an Acer configurator. The main content area is titled "Acer configurator" and features a laptop image with the text "Configure your notebook". To the right of the image is a "Parameters:" section listing specifications: Processor type, Display type (15.4 WXGA), Resolution (1280 x 800), Graphics card (ATI with shared memory), Memory type, Memory size, Optical drive (DVD-RW DL), Hard Drive, Wireless LAN (yes), Bluetooth (yes), Infraport (no), Battery type (Li-Ion), Weight (g), and Operating system (Windows XP Home). Below the parameters, the price is shown as "Our price: \$899.00" and "Without tax: \$899.00". A red box highlights the "Product options" section, which includes three dropdown menus: "Notebooks - Intel CPU" (Intel Core 1600Mhz (+ \$200.00)), "Notebooks - HDD" (Samsung 160GB SATA (+ \$99.00)), and "Notebooks - DDR2 modules" (512MB DDR2 (+ \$35.00)). Below these options, there is a paragraph of text: "Our world model was built specifically to investigate five major trends of global concern – accelerating industrialization, rapid population growth, widespread malnutrition, depletion of nonrenewable resources, and a deteriorating environment." At the bottom of the red box, the "Total price (without tax): \$1233.00" is displayed. Below the red box, there is an "Add to wishlist" button, a "Units text box" containing the number "1", and an "Add to cart" button. The page also features a left sidebar with navigation links (Cameras, Cell phones, MP3 Players, Notebooks, PDAs, PCs), a poll, and quick links. The right sidebar contains "Latest news" and a "Newsletter subscription" form.

## 2. Product list

```
<% Register Src="~/CMSModules/Ecommerce/Controls/ProductOptions/
ShoppingCartItemSelector.ascx" TagName="CartItemSelector" TagPrefix="uc1" %>
```

```
<uc1:CartItemSelector id="cartItemSelector" runat="server" SKUID='<%#
ValidationHelper.GetInteger(Eval("SKUID"), 0) %>' SKUEnabled='<%#
ValidationHelper.GetBoolean(Eval("SKUEnabled"), false) %>' AddToCartImageButton
="AddToCart.gif" ShowUnitsTextBox="false" ShowProductOptions="false" />
```

This control has **ShowUnitsTextBox** and **ShowProductOption** properties set to **false**. Therefore, the product options and units text box are not visible on the website.

The screenshot shows a web page for 'Products / Notebooks'. It features a navigation menu with links for Home, News, Products, How to buy, Company, and Silverlight. A search bar is located at the top left. The main content area displays a grid of laptop products. The first product is the Acer Aspire 3105WLMi, priced at \$490.00, with an 'Add to cart' button highlighted in red. Other products include Acer configurators, Acer Extensa 7620G, Asus F3U AP059C, and HP configurators. The right sidebar contains 'Similar products' (Canon PowerShot A720 IS, Nikon Coolpix S51, Nikon D40, Olympus Stylus FE-230) and 'Latest news RSS' with two news items dated 2/13/2008.

Complete list of shopping cart item selector properties:

<b>SKUID</b>	Product ID
<b>SKUEnabled</b>	Indicates whether the current product is enabled. If set to <b>true</b> , the button/link for adding product to the shopping cart is rendered, otherwise it is not rendered.
<b>AddToCartImageButton</b>	File name of the image which is used as a source for image button to add product to the shopping cart, default image folder is '~/App_Themes/<stylesheet name>/Images/ShoppingCart/'.
<b>AddToCartLinkText</b>	String (simple text or localizable string) of the link to add product to the shopping cart.
<b>ShowUnitsTextBox</b>	Indicates if text box for entering number of units to add to the shopping cart should be displayed. If it is hidden, number of units is equal to DefaultQuantity.
<b>ShowProductOptions</b>	Indicates if product options of the current product should be displayed.

	If <b>ShowProductOption</b> is set to <b>false</b> and a customer clicks <b>Add to shopping cart</b> for a products with specified product options, they are redirected to product detail where they are required to choose product option before the product can be added to the shopping cart.
<b>DefaultQuantity</b>	Default quantity when adding product to the shopping cart, it is set to 1 by default.
<b>ShowTotalPrice</b>	If true, total price will be shown in the bottom section of the product options list.
<b>ShowPriceIncludingTax</b>	If true, the total price enabled by the <b>ShowTotalPrice</b> property and product option prices will be displayed with tax added. If false, tax will not be included in the total price.

### Displaying total product price

If you enable the **ShowTotalPrice** property of the shopping cart item selector, a total price of the product including prices of product options will be displayed at the bottom section of the product options list, as you can see in the screenshot below. The price can either be displayed with or without tax, based on the value of the **ShowPriceIncludingTax** property of the control. Displayed prices of the product options are affected by the value of this property too.

Our price: \$899.00  
Without tax: \$899.00

**Notebooks - AMD CPU**  
AMD X2 1800 Mhz (+ \$150.00) ▼

**Notebooks - HDD**  
Samsung 80GB SATA (+ \$80.00) ▼

**Notebooks - DDR2 modules**  
512MB DDR2 (+ \$35.00) ▼

Our world model was built specifically to investigate five major trends of global concern – accelerating industrialization, rapid population growth, widespread malnutrition, depletion of nonrenewable resources, and a deteriorating environment.

**Total price (without tax): \$1164.00**

#### Shopping cart URL

If you use **ShoppingCartItemSelector**, the shopping cart URL is taken from the **Site Manager -> Settings -> E-commerce -> Shopping cart URL** value.

## 5.6 Adding items to the wish list

The items can be added to the wish list by calling the page with the **Wishlist** web part with following URL parameter:

- **productId** - the ID of the product stored in the NodeSKUID database field (in the CMS\_Tree table) or SKUID (in the COM\_SKU table)

It means your link will typically look like this:

```
<a href="~/wishlist.aspx?productId=10">Add to wish list</a>
```

In the transformations, you can use the following method to display the link to the wish list:

```
<%# EcommerceFunctions.GetAddToWishListLink(Eval("NodeSKUID")) %>
```

You can use the **ShoppingCartItemSelector** control to display the **Add to wishlist** button as well. All you have to do is to set one of the following properties.

<b>AddToWishlistImageButton</b>	File name of the image which is used as a source for image button to add product to the wish list, default image folder is '~/App_Themes/<stylesheet name>/Images/ShoppingCart/'.
<b>AddToWishlistLinkText</b>	String (simplet text or localizable string) of the link to add product to the wish list.



#### Wish list URL

If you use **ShoppingCartItemSelector** or functions like `GetAddToWishListLink`, the wish list URL is taken from the **Site Manager -> Settings -> E-commerce -> Wish list URL** value.

## 5.7 Displaying product price

### Displaying product price on the website

The price is stored in the COM\_SKU.SKUPrice field. The following method allows you to display the price **in your transformation** in the currently selected currency, with appropriate discount level and in the appropriate format specified at **CMS Desk -> Tools -> E-commerce -> Configuration -> Currencies -> <edit currency> -> Currency formatting string**:

```
<%# EcommerceFunctions.GetFormattedPrice(Eval("SKUPrice"), Eval("SKUDepartmentID")) %>
```

If you want to display prices with applied discount level and taxes as well, use the following function instead.



```
<%# EcommerceFunctions.GetFormattedPrice(Eval("SKUPrice"), Eval  
("SKUdepartmentID"),  
  
Eval("SKUID")) %>
```

**Please note:** computation of all product taxes is time-consuming. Therefore, the result is stored in the cache as specified at **CMS Site Manager -> Settings -> website -> Cache content (minutes)**.

If you don't want to display prices with the applied discount level and taxes, use the following function instead.

```
<%# EcommerceFunctions.GetFormattedPrice(Eval("SKUPrice")) %>
```

All product prices are displayed in currency chosen according to the following priorities:

1. **the currency of the shopping cart**
2. **the preferred currency of a customer**
3. **the default currency**

Therefore, if the currency of the shopping cart is not set yet (the shopping cart hasn't been created so far), all product prices are displayed in the preferred currency of the given customer. If the customer doesn't specify the preferred currency, all product prices are displayed in the default currency.

## 5.8 Getting product URL

In some cases, you may need to get the URL of the document to which some product is assigned. Knowing product URL can be useful when you work with specific web parts such as the **Product datalist** which is using the e-commerce product (SKU) properties to display products instead of the standard CMS documents.

For this, you need to insert the **GetProductUrl** method with the following syntax into your transformation:

```
<%# EcommerceFunctions.GetProductUrl(Eval("SKUID")) %>
```

As a result, you will get the product URL in the **/CMSPages/GetProduct.aspx?productId=125** format.

Alternatively, you can provide the method with values of the **SKUGUID** and **SKUName** columns:

```
<%# EcommerceFunctions.GetProductUrl(Eval("SKUGUID"), Eval("SKUName")) %>
```

This syntax returns the URL in the **~/getproduct/<skuguid>/<safe\_skuname>.<extension>** format.

## 5.9 Searching

Kentico CMS comes with the built-in **Smart search** module that enables you to search for the given expression in the content of your website.

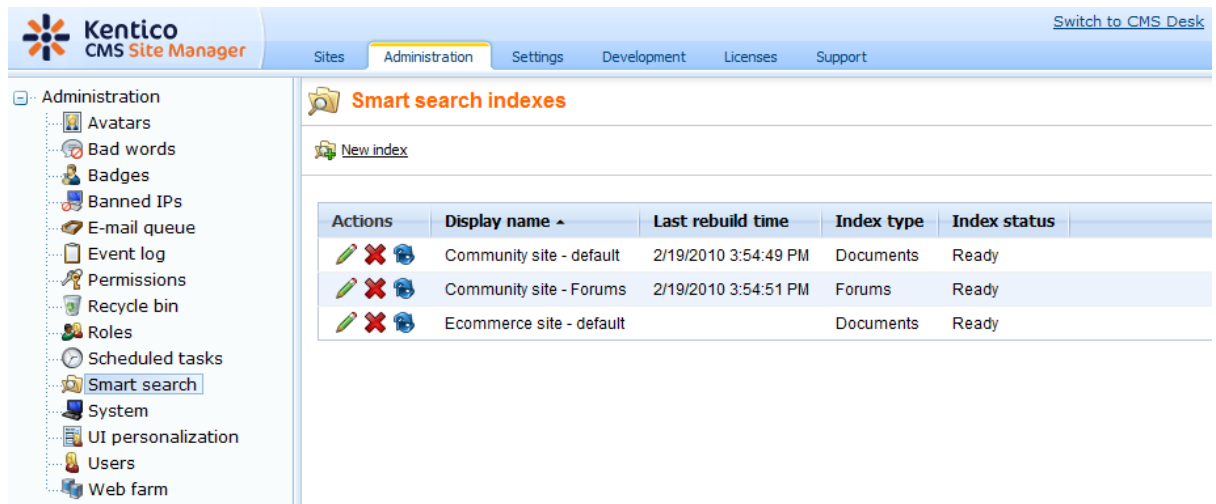
The search functionality is ensured by the **Special pages -> Search page**, where the **Smart search dialog with results** web part (1. in the screenshot) is placed. This web part allows searching and displays the search results.

The **Smart search box** web part (2. in the screenshot) on the master page is connected to the **Smart search dialog with results** web part by means of the **Search results page URL** property. When a user searches using the **Smart search box**, they are redirected to the **Special pages -> Search page** where the **Smart search dialog with results** web part displays the results.










The screenshot displays the Kentico CMS interface. On the left, there is a navigation tree for an 'E-commerce site' with various pages like Home, News, Products, and Special pages. The main content area shows a search box (2) and a search results page (1) with results for 'Sony Ericsson W800' and 'HP IPAQ 114'. The search results page includes a search mode dropdown set to 'Any word', a search button, and a list of search results with product images and descriptions.

The Smart search module provides index-based search. You need to specify which index should be searched using the Indexes property of the **Smart search dialog with results** web part.

Indexes can be created and managed in **Site Manager -> Administration -> Smart search**. For detailed information about Kentico CMS Smart search module, please refer to [Kentico CMS Developer's Guide -> Modules -> Smart search](#).



The screenshot displays the Kentico CMS Site Manager Administration interface. The top navigation bar includes 'Sites', 'Administration' (selected), 'Settings', 'Development', 'Licenses', and 'Support'. A 'Switch to CMS Desk' link is visible in the top right. The left sidebar lists various administration categories, with 'Smart search' highlighted. The main content area is titled 'Smart search indexes' and features a 'New index' button. Below this is a table listing existing search indexes.

Actions	Display name ^	Last rebuild time	Index type	Index status
  	Community site - default	2/19/2010 3:54:49 PM	Documents	Ready
  	Community site - Forums	2/19/2010 3:54:51 PM	Forums	Ready
  	Ecommerce site - default		Documents	Ready

**Part**

---

**VI**

**Configuration settings**

---

## 6 Configuration settings

### 6.1 Configuration overview

When you start using the **E-commerce module** it's recommended that you configure it in the following order:

1. Enable the **E-commerce module** at **Site Manager -> Settings** if it's not enabled yet.  
More details: [Enabling the e-commerce module](#)
2. Configure **access rights** to the **E-commerce module** interface.  
More details: [Security configuration](#)
3. Set up **countries and states**.  
More details: [Countries and states](#)
4. Configure **store settings**.  
More details: [Store settings](#)
5. Set up **departments** and assign **store administrators** to their **departments**.  
More details: [Departments](#)
6. Configure **currencies**. If you plan to use multiple currencies, specify the **exchange rates**.  
More details: [Currencies](#), [Exchange rates](#)
7. Set up **shipping options**.  
More details: [Shipping options](#)
8. Set up **payment methods**.  
More details: [Payment methods](#)
9. Configure **tax classes**.  
More details: [Tax classes](#)
10. Set up order processing **workflow**.  
More details: [Order status](#)
11. Configure **public and internal product status**.  
More details: [Public status](#), [Internal status](#)
12. Modify the **invoice/receipt** design.  
More details: [Invoice](#)
13. Create documents with product details at **CMS Desk -> Content** and mark the documents as **products**. Enter the product details.  
More details: [Products](#)
14. Make a testing order for each **payment option**.

#### Possible configurations

The **E-commerce module** allows you to manage products in two basic modes:

- **standard mode** - you manage the products in **Kentico CMS**. Each product is represented by a combination of document that contains product information displayed to the visitor and a product record that contains standard product information, such as product price, sizes, taxes, etc.

- **custom product provider** - you manage the products in an external system and publish them inside Kentico CMS-based website. You can learn more on custom providers in chapter [Developing custom providers](#).

## 6.2 Orders

The Orders can be managed at **Kentico CMS Desk -> Tools -> E-commerce -> Orders** of the e-commerce module.

### General properties

- **Order ID** - unique identifier of the order
- **Date** - date and time when the order was placed
- **Invoice number** - the assigned invoice number (it can be changed on **Invoice** tab)
- **Status** - order processing status
- **Customer** - customer who made this order
- **Company address** - address of the company which made the order
- **Order note** - notes added by customer or by store managers

### Shipping

- **Shipping option** - chosen shipping option
- **Shipping address** - chosen shipping address
- **Tracking number** - tracking number of the order

### Billing

- **Payment method** - chosen payment method
- **Currency** - currency used in the invoice
- **Billing address** - chosen billing address
- **Payment result** - payment results as described in the [Payment results](#) chapter

### Items

Here you can modify the order during its processing. Please note that to be able to additionally modify order item name and order item price **CMUseCurrentSKUData** and **CMSEnableOrderItemEditing** keys in the web.config files have to be set accordingly. Please refer to the [Web.config settings](#) chapter for more information.

### Invoice

Here you can re-generate or print the invoice and change the invoice number manually.

### History

Here you can see the history of order processing.

## 6.3 Customers

You can manage customers at **Kentico CMS Desk -> Tools -> E-commerce -> Customers**. The customer profile can be created either manually in this administration interface or it can be created automatically when the customer registers during the purchase process.

You can create new customers here. While creating a new customer, you can choose to provide them with **User name** and **Password** straight away and create the user account for them.

### General customer properties

- **First name** - customer's first name
- **Last name** - customer's last name
- **Company** - company name (optional)
- **Organization ID** - registration ID of the customer's company
- **Tax registration ID** - tax registration ID of the customer's company
- **E-mail** - customer's e-mail address
- **Phone** - customer's phone number
- **Fax** - customer's fax number
- **Preferred currency** - the currency in which the prices are displayed when the customer signs in
- **Preferred payment method** - customer's preferred payment option (automatically set based on their last purchase)
- **Preferred shipping option** - customer's preferred shipping option (automatically set based on their last purchase)
- **Country/state** - customer's main country and state
- **Enabled** - if you uncheck the box, the customer profile will not be listed in the customer list, while still being stored in the database for your records and to keep customer's purchase history.
- **Discount level** - discount level applied to the customer's orders

### Addresses

Each customer can have several addresses for billing and shipping stored in their profile. Every address has the following properties:

- **Personal or company name** - name line in the address
- **Address lines** - address line 1 and address line 2
- **City** - city
- **ZIP code** - zip/postal code
- **Country** - country
- **Phone number** - phone number
- **Enabled** - indicates if the address should be offered to the client. If you set it to disabled, it will no longer be displayed, but it will be kept in the database for your records and to keep customer's purchase history.
- **Use as shipping address** - indicates if the address should be offered to customer as a shipping address
- **Use as billing address** - indicates if the address should be offered to customer as a billing address
- **Use as company address** - indicates if the address should be offered to customer as a company address

### Orders

- the list of orders made by this customer

## Credit

The customer credit history.

The customer credit is stored in the default currency. If a payment is made in a currency different from default, the payment amount is converted to the default currency and the customer credit is decreased by this amount - a new credit event is created.

For information about enabling payments with the customer credit please refer to the [Customer credit](#) chapter.

The displayed total value of the customer credit is sum of all credit events.

## Newsletters

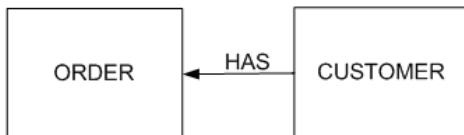
Here you can select newsletters to which a customer is subscribed.

For more information about newsletters please refer to the **Newsletter module overview** chapter in **Kentico CMS Developer's Guide**.

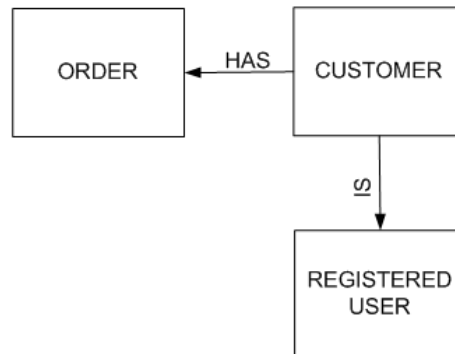
## Login

Here you can manage user accounts that users use to sign in to your e-commerce site (if they are registered users, not only anonymous site visitors)

A. Customer-order relation if an order is made by an anonymous customer.



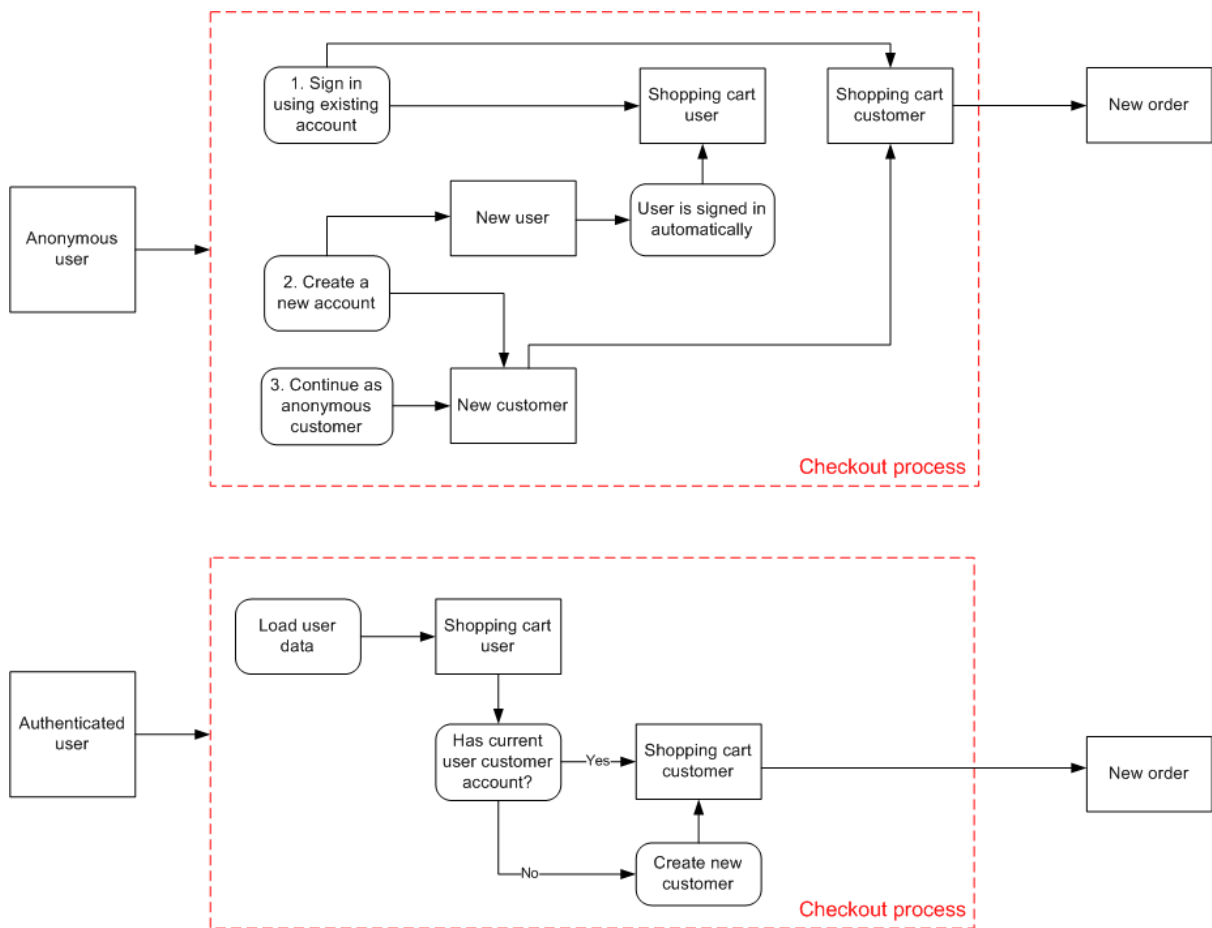
B. Customer-order-user relation if an order is made by a registered user.



You can create and edit users at **CMS Desk -> Administration -> Users**.

The following diagram shows the checkout process for both an anonymous and authenticated user in greater detail.





## 6.4 Products

Kentico CMS allows you to integrate the e-commerce data with documents. You can mark any document as a product and define its product properties.

There are two ways how you can enter the products:

1. Go to **CMS Desk -> Content -> Product**, check the **Mark document as product** box and choose **Create a new product**.
2. Go to **CMS Desk -> Tools -> E-commerce -> Products** and enter all your product details. Then go to **CMS Desk -> Content -> Product**, check the **Mark document as product** and select an existing product.

### General tab

Every product has the following properties:

- **Product name** - product name displayed on the website and in the shopping cart
- **Product number** - product number (serial number or SKU number) for your records
- **Description** - product description used for special product listings
- **Price** - product price in the main currency
- **Enabled** - indicates if product can be added to the shopping cart
- **Department** - the department that is responsible for this product
- **Manufacturer** - the manufacturer of the product

- **Supplier** - the supplier of the product
- **Public status** - status of the product displayed to site visitors - e.g. Available immediately
- **Internal status** - status of the product displayed only to the site owners - e.g. Products going off-sale. Invisible to the site visitors.
- **Availability (days)** - number of days required for processing the order (e.g. the order may require 1 day so that you get the product from the distributor)
- **Product image URL** - URL of the product image
- **Package weight** - weight in your chosen unit
- **Package height** - height in your chosen unit
- **Package width** - width in your chosen unit
- **Package depth** - depth in your chosen unit
- **Available items** - number of items available in stock
- **Sell only if items are available** - indicates if customers can purchase only quantity that is in stock (checked box)

### Custom fields tab

Here you can see list of the custom fields available for all products. You can specify the custom fields at **CMS Site Manager -> Development -> System tables -> Ecommerce - SKU**.

Please note that the **Custom fields tab** is visible only if there are some custom fields defined.

### Tax classes tab

You can specify **Tax classes** for every product - you can choose any number of taxes that apply to the given product.

### Volume discounts

On the **Volume discounts** tab in the **Product properties** dialog, you can define discount that applies to the ordered product when the customer purchases specified amount of the product.

For every volume discount line, you need to enter:

- **minimum amount** - number of ordered items from which the discount applies
- **discount value** - relative or absolute discount for the given minimum amount

When the customer updates the number of ordered products, the discount is automatically calculated and the customer is displayed with the final price. For more details please refer to the [Discounts overview](#) chapter.

### Options tab

Here you can specify option categories (e.g. size of clothes) - you can choose any number of categories that assign to the given product. Customers are asked to choose their product configuration before the product is added to the shopping cart.

For further information please refer to the [Product options](#) chapter.

#### Assigning product to document

In Kentico CMS, products are defined separately from the documents. When you

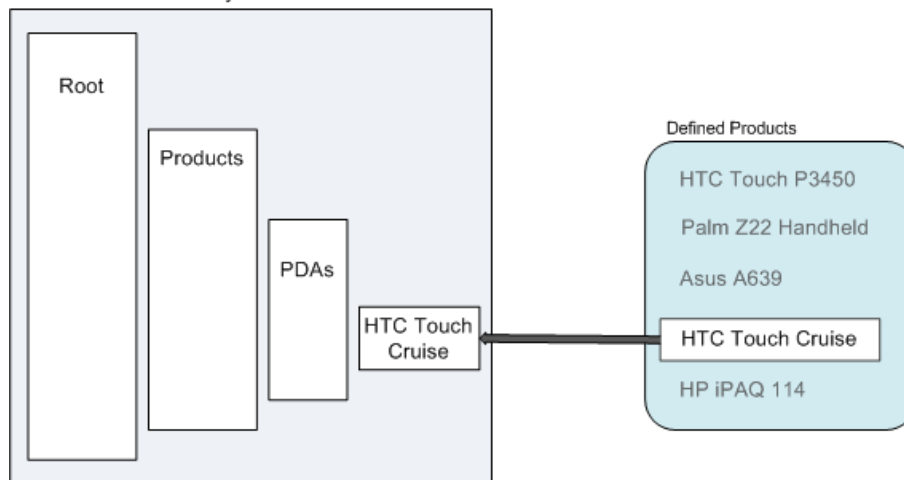


want to display a product on your website, you firstly choose a document that will act as a container for the product and then you assign the given product to the chosen document.

While creating document, you can also choose to create document and product at the same time and product will be assign to the given document automatically.

The fact that products are defined separately enables them to be independent of the document structure of a specific website. This allows you to use the same product on multiple sites.

Document structure of your web site.



**Please note:** Should you want your products to be created automatically or to get its product field value directly from the document field value, please refer to the [Mapping document fields to product properties](#) chapter.

## 6.5 Product options

You can manage product options at **Kentico CMS Desk -> Tools -> E-commerce -> Product options**.

Here you can either create a new option category by clicking the **New category** link or **Edit** and **Delete** the existing one.

Every option category has following properties:

### General

**Category name** - category name displayed to the customer

**Selection type** - type of selection, a customer use to choose the product option

**Display price** - if enabled, product option prices will be displayed next to product options






**Default option(s)** product options which are selected by default before a customer chooses their own one(s)

**Description** - the description of the category as it is visible to customers

**Default record text** - the caption of the default option, which is added to the product options collection

**Enabled** - indicates if the category is enabled

## Options

Here you can either add a new product option to the current option category by clicking the  **New product option** link or you can **Edit** () or **Delete** () existing options or change their succession () () within the current option category.

**Product options** have the same properties as **Products** with the exception of the volume discounts which cannot be assigned (as well as discount coupon) to a product option.

On contrary with the product price, the product option price can be negative. The product option price is the amount that is either added to or subtracted from the given product price. The product option is sold as an addition to the given product, not as its replacement. If certain discount level applies to the given customer, this discount level applies for the chosen product option as well.

## 6.6 Manufacturers

The **Manufacturers** dialog at **Kentico CMS Desk -> Tools -> E-commerce -> Manufactures** allows you to manage the list of manufacturers who make the products you sell. This information is optional, so you do not need to supply it. You can use it for your internal records or you can display it on your website (e.g. you can display a link to the manufacturer's website where clients can find more details).

Every manufacturer has the following properties:

- **Manufacturer display name** - manufacturer name
- **Homepage** - URL of the internet home page of the manufacturer
- **Enabled** - indicates if the given manufacturer can be used for newly created products

## 6.7 Suppliers

You can define your suppliers at **Kentico CMS Desk -> Tools -> E-commerce -> Suppliers**. This information is typically used only for your records. This information is optional and you do not need to enter it.

Every supplier has the following properties:

- **Supplier display name** - name of the supplier
- **Supplier e-mail** - supplier e-mail contact
- **Supplier phone** - supplier phone contact
- **Supplier fax** - supplier fax contact
- **Enabled** - indicates if the supplier is active and can be used for newly added products

## 6.8 Discount coupons

You can define discount coupons at **Kentico CMS Desk -> Tools -> E-commerce -> Discount coupons**. The discount coupon is represented by a special code that is used by the customer on the Shopping Cart page.

Every discount coupon has the following properties:

- **Discount coupon name** - friendly name of the discount displayed to customers and store managers
- **Discount coupon code** - the coupon code that will be used by the customer during the purchase
- **Discount value** - you can choose between absolute and relative discount and you can enter the discount value in the box below

- **Valid from, Valid to** - time period when the discount can be applied

On the **Products** tab, you can choose which products the discount should be applied to or not applied to. Please note that you can assign a discount coupon only to the given product, not to its product options. **Discount coupons** cannot be defined for **Product options**.

For more details please refer to the [Discounts overview](#) chapter.

## 6.9 Discount levels

You can define discount levels that apply to all products at **Kentico CMS Desk -> Tools -> E-commerce -> Discount levels**. Every customer can be assigned with a discount level. For example, you can assign all Gold Partners with 30% discount on all products they purchase on your website.

The discount level can be defined on the **Discount levels** tab. A discount level has the following properties:

- **Display name** - the discount level name displayed to the users
- **Code name** - the discount level name used in the code
- **Value** - percentage value of the discount
- **Valid from/to** - time period when the discount level is applied. If you leave both values empty, the discount is time-unlimited.
- **Enabled** - indicates if the discount level should be used

On the **Departments** tab, you can choose the departments on which the discount level can be applied.

Once you define the discount levels, you need to assign them to the customers in the **Customer properties** dialog, in the **Discount level** drop-down list.

For more details please refer to the [Discounts overview](#) chapter.

## 6.10 Reports

At **Kentico CMS Desk -> Tools -> E-commerce -> Reports** you can display, save and print following e-commerce statistics:

- **Numbers of orders per year** - number of orders made on the website in one year
- **Numbers of orders per month** - number of orders made on the website in one month
- **Number of orders per day** - number of orders made on the website in one day
- **Sales per year** - sales in one year in your default currency
- **Sales per month** - sales in one month in your default currency
- **Sales per day** - sales in one day in your default currency
- **Inventory** - the inventory list of all products defined in the **Products** tab
- **Top 100 customers by volume of sales**

For more information, please refer to the **Module Reporting** chapter in **Kentico CMS Developer's guide**.

## 6.11 Configuration

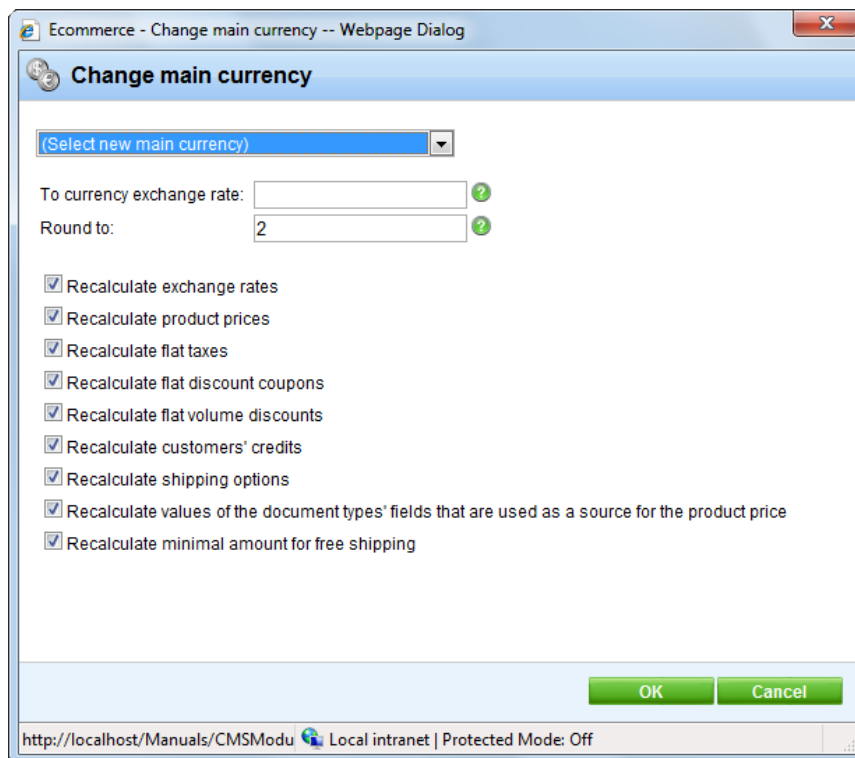
### 6.11.1 Store settings

Go to **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Store Settings** and set the following values:

#### General tab

- **Default country** - <choose your country or country where you sell most>
- **Minimum amount for free shipping** - the minimum order value when the shipping is not charged
- **Allow anonymous customers** - indicates if customers need to register in your site so that they can make the purchase
- **Use an extra company address** - indicates if the option of providing company address is available in the check out process
- **Require company account information** - indicates if it is compulsory to provide company account information in the check out process
- **Show Organization ID field** - indicates if organization ID field should be displayed
- **Show Tax registration ID field** - indicates if organization ID field (e.g. VAT registration ID) should be displayed
- **Main currency** - default currency use as base for calculation of exchange rates

After clicking the **Change** button next to main currency, a new dialog appears. In the dialog, you can select items which should be recalculate according to exchange rate to the new main currency.








#### Emails tab

- **Send e-commerce e-mails from** - an e-mail address the e-commerce notification e-mails are sent

from

- **Send e-commerce e-mails to** - an e-mail address (e.g. merchant's e-mail address) the e-commerce notification e-mails are sent to
- **Send order notification** - indicates if e-mail notifications are sent after an order is finished and saved. The e-mail template **E-commerce order notification to customer** is used when sending notification to customer. The e-mail template **E-commerce order notification to administrator** is used when sending notification to administrator.
- **Send payment notification** - indicates if e-mail notifications are sent after payment is completed. The e-mail template **E-commerce - Order payment notification to customer** is used when sending notification to customer. The e-mail template **E-commerce - Order payment notification to administrator** is used when sending notification to administrator.

## Checkout process tab

Here you can either create a new step in the checkout process by clicking the  **New step** link or you can **Edit** () or **Delete** () existing steps or change their succession () , () .

For more information about purchase process please refer to the [Customizing the purchase process](#) chapter.

## 6.11.2 Departments

Go to **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Departments**. Department is used for organization of your product management efforts - you can specify users who can manage products in the given department. It means you can have a different product manager for e.g. books and for electronics departments.

Please note that the **departments are not the same as product categories** displayed to site visitors.

You can create either only single department and place all products in it or you can create separate departments if you have several product managers.

Every department has these properties:

- **Department display name** - friendly name displayed to users
- **Department code name** - code name used by developers in the code
- **Default tax class for new products** - the tax class that is automatically assigned to the products created in the given department

In the **Users** tab, you can specify the users who can manage products in these departments.

## 6.11.3 Shipping options

You can manage shipping options at **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Shipping options**. Each shipping option has the following properties on the **General** tab:

- **Shipping option display name** - friendly name displayed to customers and store managers
- **Shipping option name** - code name used by developers
- **Shipping option charge** - amount charged for this shipping option (in the main currency)
- **Enabled** - indicates if shipping option is offered to customers

On the **Payment methods** tab of the Shipping option properties dialog, you can choose which payment methods will be available for the given shipping option.

### 6.11.4 Payment methods

You can manage payment methods at **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Payment methods**. Every payment method has the following properties:

- **Payment method display name** – Friendly name (localizable string or simple text) displayed to the users.
- **Payment method code name** – Unique identifier which must be unique within all of the site payment methods.
- **Enabled** – Indicates if payment method could be used for payment.
- **Payment gateway URL** - External payment gateway url or relative path to your document controlled by the Kentico CMS which represents your custom payment processor. It can be parameterized with data macros to evaluate the data fields of the order and related objects.

The following example shows how you can insert finished order ID and customer billing address state code to the payment url, however, you can get any value from the related objects. See **Kentico CMS Database Reference** for detailed column listing of the objects:

```
http://www.SomePaymentGateway.com?orderid={%Order.OrderId%}&state={%BillingAddress.State.StateCode%}
```

<code>{%Order.OrderID%}</code>	Displays the value of specified order data column ( <b>COM_Order</b> )
<code>{%ShoppingCart.ShoppingCartID%}</code>	Displays the value of specified shopping cart data column ( <b>COM_ShoppingCart</b> )
<code>{%OrderStatus.StatusID%}</code>	Displays the value of specified order status data column ( <b>COM_OrderStatus</b> )
<code>{%BillingAddress.AddressID%}</code>	Displays the value of specified billing address data column ( <b>COM_Address</b> )
<code>{%BillingAddress.Country.CountryID%}</code>	Displays the value of specified billing address country data column ( <b>CMS_Country</b> )
<code>{%BillingAddress.State.StateID%}</code>	Displays the value of specified billing address state data column ( <b>CMS_State</b> )
<code>{%ShippingAddress.AddressID%}</code>	Displays the value of specified shipping address data column ( <b>COM_Address</b> )
<code>{%ShippingAddress.Country.CountryID%}</code>	Displays the value of specified shipping address country data column ( <b>CMS_Country</b> )
<code>{%ShippingAddress.State.StateID%}</code>	Displays the value of specified shipping address state data column ( <b>CMS_State</b> )
<code>{%CompanyAddress.AddressID%}</code>	Displays the value of specified company address data column ( <b>COM_Address</b> )
<code>{%CompanyAddress.Country.CountryID%}</code>	Displays the value of specified company address country data column ( <b>CMS_Country</b> )
<code>{%CompanyAddress.State.StateID%}</code>	Displays the value of specified company address state data column ( <b>CMS_State</b> )



<b>StateID%</b>	<b>CMS_State)</b>
<b>{%ShippingOption.ShippingOptionID%</b>	Displays the value of specified shipping option data column ( <b>COM_ShippingOption</b> )
<b>{%PaymentOption.PaymentOptionID%</b>	Displays the value of specified payment option data column ( <b>COM_PaymentOption</b> )
<b>{%Currency.CurrencyID%</b>	Displays the value of specified currency data column ( <b>COM_Currency</b> )
<b>{%Customer.CustomerID%</b>	Displays the value of specified customer data column ( <b>COM_Customer</b> )
<b>{%DiscountCoupon.DiscountCouponID%</b>	Displays the value of specified discount coupon data column ( <b>COM_DiscountCoupon</b> )

- **Payment gateway assembly name** – Name of a library which includes your payment gateway class inside.
- **Payment gateway class name** – Name of a class which represents your custom payment gateway processor.
- **Order status when payment succeeds** – Order status which is set to the existing order when its payment succeeds.
- **Order status when payment fails** – Order status which is set to the existing order when its payment fails.

**Please note:** the offered payment methods depend on the chosen [shipping option](#). You need to specify this relationship when editing required shipping option. If there is no enabled shipping option all payment methods are offered.

### 6.11.5 Tax classes

You can manage tax classes at **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Tax classes**.

**Tax classes** represent a customizable system that allows you to configure taxes paid for goods. You can specify different tax classes with different rates for each country. Then you apply these taxes to particular products.

Every tax class has the following properties:

- **Tax class display name** - friendly name displayed to users
- **Tax class code name** - code name used by developers in the code
- **Zero tax if Tax ID is supplied** - indicates if the tax should be calculated as zero (0) if the customer enters the Tax ID. The Tax ID is only displayed during the purchase process if the **Configuration -> Store Settings -> Show Tax registration ID field** box is checked.

On the **Countries** and **States** tab, you can set the value for particular countries (the country is recognized based on the billing address). The tax value can be either percentage (by default) or it can be a flat fee. If both country and state tax are specified, only the state tax is used.

### 6.11.6 Currencies

The store may offer goods for prices in several currencies. The available currencies can be managed at **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Currencies**.

## How to configure currencies

1. Enter all currencies you will use and delete or disable (uncheck the box **Enabled**) the currencies that you do not want to use. You will need to enter the following properties:

- **Currency display name** - the friendly name displayed to site visitors
- **Currency code name** - the code name used by developers
- **Currency code** - the official code of the currency used in exchange rates
- **Currency formatting string** - the format used to display amounts in the given currency - use {0} expression to insert the value into the formatting text
- **Significant digits** - the number of digits in the price that will be used in the total amount. The value will be rounded if the actual number of decimal digits is higher
- **Enabled** - indicates if the currency should be displayed in the currency drop-down list

2. If applicable, set the new currency as main currency at **Configuration -> General**.

3. If you're using multiple currencies, please specify exchange rates at **Configuration -> Exchange rates**.

### 6.11.7 Exchange rates

You can configure the exchange rates between the main currency and the alternative currencies at **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Exchange rates**. The exchange rates are organized into **Exchange tables** that specify the complete exchange rate table for a given time period. You can define a new exchange table e.g. for every day.

Every exchange table has the following properties:

- **Table display name** - friendly name of the exchange table - e.g. "January 15, 2007 - January 20, 2007"
- **Valid from, Valid to** - time period the exchange table applies to
- **Exchange rates** - here you can enter "how much you have to pay for a single unit of the given currency"

### 6.11.8 Order status

You can configure various order states at **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Order status**. This can be e.g. New -> In progress -> Closed or Canceled. You can also specify their flow using the up/down arrows.

Every status has the following properties:

- **Order status display name** - friendly name displayed to store managers
- **Order status code name** - code name used by developers
- **Order status color** - the color used to highlight the orders in given status in the Orders dialog.
- **Enabled** - indicates whether this status should be used for new orders
- **Send notification** - indicates whether the e-mail notifications are sent to the customer and administrator when status of the existing order is set to the given status. The e-mail template "E-commerce order status notification to customer" is used when sending notification to customer. The e-mail template "E-commerce order status notification to administrator" is used when sending notification to administrator.

**Please note:** This setting is independent of the **Send order notification** setting at **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Store settings -> Emails**. Furthermore, the **Send e-commerce e-mails from** and **Send e-commerce e-mails to** text boxes at **CMSDesk -> Tools -> E-commerce -> Configuration -> Store settings -> Emails** need to have specified values

so that you would be able to receive and send notifications.

### 6.11.9 Public status

You can configure various states of the product displayed on the public site at **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Public status**. This can be e.g. "Available", "Accepting pre-orders" or "Not available".

Every status has the following states:

- **Public status display name** - friendly name displayed to store managers
- **Public status code name** - code name used by developers
- **Enabled** - indicates if this status should be used for new orders

This can be e.g. "Available", "Accepting pre-orders" or "Not available".

### 6.11.10 Internal status

You can configure various states of the product used by store managers to differentiate products at **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Internal status**. This can be e.g. "Featured product", "New!" or "Sale". You can then use this information on your website to display the products e.g. in the "New products" box on your home page.

Every status has the following states:

- **Internal status display name** - friendly name displayed to store managers
- **Internal status code name** - code name used by developers
- **Enabled** - indicates if this status should be used for new orders

This can be e.g. "Featured product", "New!" or "Sale". You can then use this information on your website to display the products e.g. in the **Featured products** list on your home page.

### 6.11.11 Invoice

You can customize the invoice (or receipt) design at **Kentico CMS Desk -> Tools -> E-commerce -> Configuration -> Invoice** dialog. You can edit the HTML code representing your invoice here and use special expressions to insert dynamic parts of the invoice.

For example, you can use the **##BILLINGADDRESS##** expression to insert the customer's billing address into the invoice.

You can find a complete list of available dynamic expressions at the bottom of the dialog.

**Please note:** In Kentico CMS offers you the capability of splitting an invoice into several pages for printing. Should you experience difficulties with the printed design of an invoice in Internet Explorer, try to print it using the Firefox web browser instead.

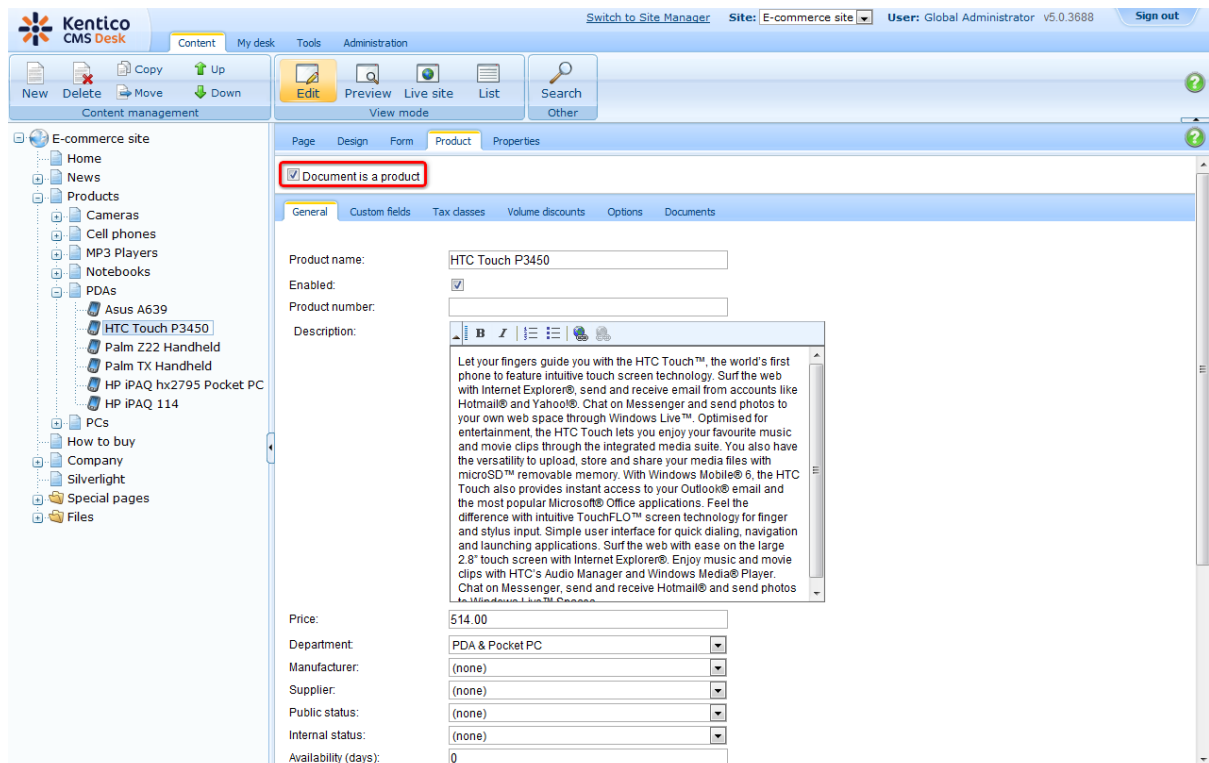
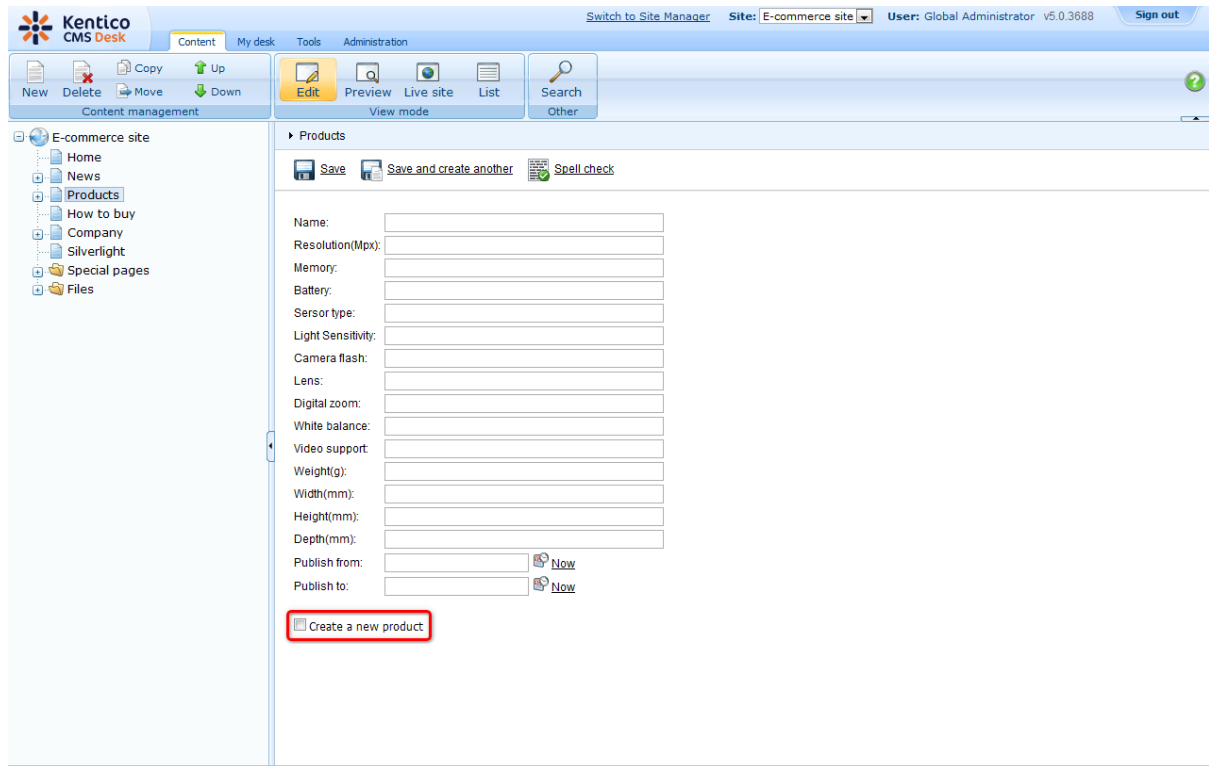
Invoice		<i>Company logo</i>		
Invoice number:	2	Order date:	4/6/2008	
Supplier	Customer			
Company address	Development Soft 1020 Blueberry Ln. Tucson 85754 USA, Arizona			
Payment option				
Cash on delivery				
Product name	Units	Price/unit	Tax	Subtotal
Configurable PC	1	0.00	0.00	0.00
- Windows Vista Home Premium	1	216.00	0.00	216.00
- CD RW Samsung Black SATA	1	42.00	0.00	42.00
- INTEL Core 2 Quad Q6600 2.40GHz	1	220.00	0.00	220.00
- nVIDIA GEFORCE 6600GT 512MB	1	440.00	0.00	440.00
- SAMSUNG 500GB SATA	1	139.00	0.00	139.00
- DDR2 1024MB 667MHz	1	30.00	0.00	30.00
- ACER 22"	1	460.00	0.00	460.00
- AUDIGY SE bulk	1	30.00	0.00	30.00
- Repr0 5.1	1	60.00	0.00	60.00
- Printer	1	90.00	0.00	90.00
- Scanner	1	69.00	0.00	69.00
Configurable PC	1	0.00	0.00	0.00
- Windows XP Home	1	191.00	0.00	191.00
- CD RW Samsung Black SATA	1	42.00	0.00	42.00
- INTEL Core 2 Quad Q6600 2.40GHz	1	220.00	0.00	220.00
- nVIDIA GEFORCE 6600GT 512MB	1	440.00	0.00	440.00
- SAMSUNG 500GB SATA	1	139.00	0.00	139.00
- DDR2 1024MB 667MHz	1	30.00	0.00	30.00
- ACER 22"	1	460.00	0.00	460.00
- AUDIGY SE bulk	1	30.00	0.00	30.00
- Repr0 5.1	1	60.00	0.00	60.00
- Printer	1	90.00	0.00	90.00
- Scanner	1	69.00	0.00	69.00
Configurable PC	2	0.00	0.00	0.00
- Windows Vista Business	2	270.00	0.00	540.00
- CD RW Samsung Black SATA	2	42.00	0.00	84.00
- INTEL Core 2 Quad Q6600 2.40GHz	2	220.00	0.00	440.00
- nVIDIA GEFORCE 6600GT 512MB	2	440.00	0.00	880.00
- SAMSUNG 500GB SATA	2	139.00	0.00	278.00
- DDR2 1024MB 667MHz	2	30.00	0.00	60.00
- ACER 22"	2	460.00	0.00	920.00
- AUDIGY SE bulk	2	30.00	0.00	60.00
- Repr0 5.1	2	60.00	0.00	120.00

Product name	Units	Price/unit	Tax	Subtotal
- Printer	2	50.00	0.00	100.00
- Scanner	2	69.00	0.00	138.00
Configurable PC	1	0.00	0.00	0.00
- Red Hat Linux	1	55.00	0.00	55.00
- CD RW Samsung Black SATA	1	42.00	0.00	42.00
- INTEL Core 2 Quad Q6600 2.40GHz	1	220.00	0.00	220.00
- nVIDIA GEFORCE 6600GT 512MB	1	440.00	0.00	440.00
- SAMSUNG 500GB SATA	1	139.00	0.00	139.00
- DDR2 1024MB 667MHz	1	30.00	0.00	30.00
- ACER 22"	1	460.00	0.00	460.00
- AUDIGY SE bulk	1	30.00	0.00	30.00
- Repr0 5.1	1	80.00	0.00	80.00
- Printer	1	50.00	0.00	50.00
- Scanner	1	69.00	0.00	69.00
Total shipping:				\$ 5.00
Total price:				\$ 9197.00
Tax summary:				

## 6.12 Enabling the e-commerce module

You may need to enable the e-commerce product before you start using it. Go to **Kentico CMS Site Manager** -> **Settings** -> **E-commerce**, select your website in the drop-down list and check the box "Product" tab enabled. Click **Save**.

It will ensure that documents can be marked as products in the **Kentico CMS Desk -> Content** section.



## 6.13 Security configuration

You can configure the following permissions for the E-commerce module in the **Kentico CMS Site Manager** -> **Administration** -> **Permissions** -> <select the "Module: E-commerce"> permission matrix. The E-commerce module has the following permissions:

- **Modify configuration** - allows members of the role to edit configuration of the E-commerce module on the Configuration tab
- **Modify data** - allows members of the role to modify configuration and data on all other tabs than the Configuration tab + in the CMS Desk -> Content -> Product dialog
- **Read configuration** - allows members of the role to view configuration of the E-commerce module on the Configuration tab
- **Read data** - allows members of the role to read configuration and data on all other tabs than the Configuration tab + in the CMS Desk -> Content -> Product dialog

Moreover, you can assign store managers to particular [departments](#), so that they cannot accidentally modify products they are not responsible for.

You can also use document-level permissions if you need to restrict access to modifications of product documents in the content tree.

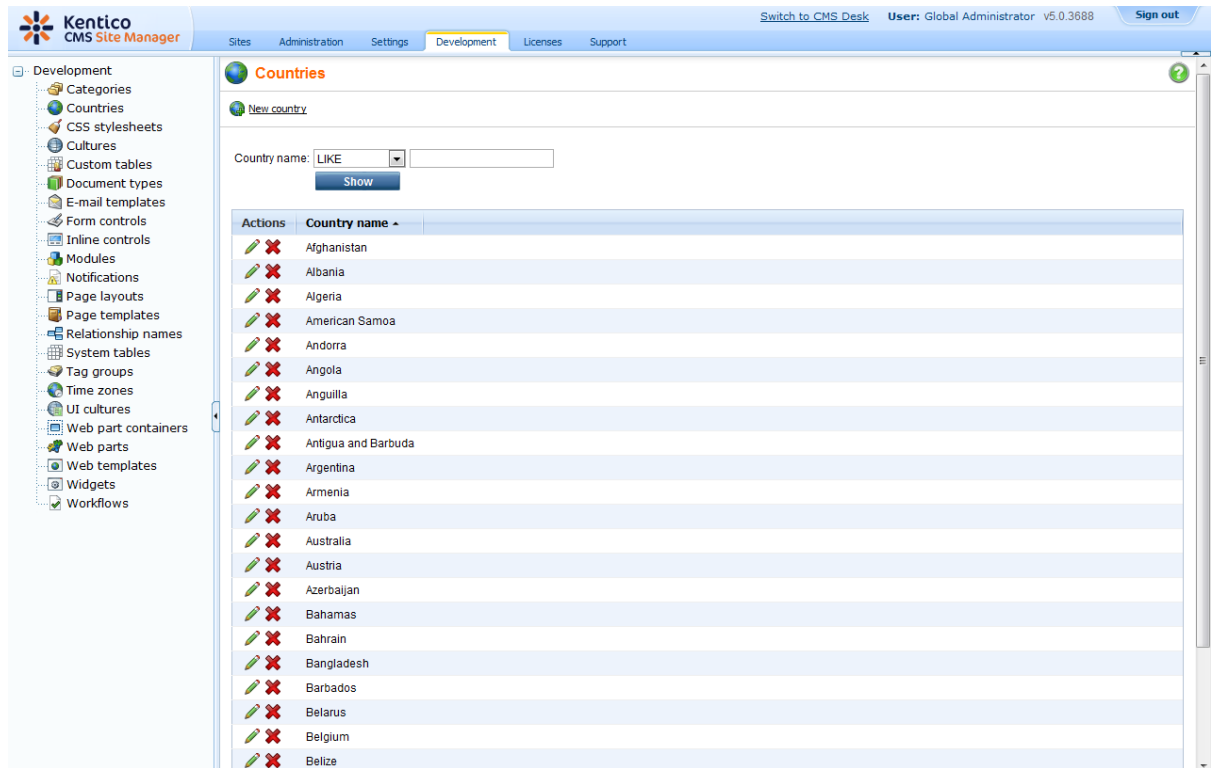
The screenshot shows the Kentico CMS Site Manager Administration interface. The main content area displays the 'Permissions' configuration for the 'E-commerce' module. The configuration includes dropdown menus for 'Site' (E-commerce site), 'Permission type' (Modules), and 'Permission matrix' (E-commerce). Below these is a table showing permissions for various user roles.

	Modify configuration	Modify data	Read configuration	Read data
Authenticated users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CMS Basic users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CMS Community administrators	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CMS Designers	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CMS Desk Administrators	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CMS Editors	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CMS Readers	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Everyone	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Live ID users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Not authenticated users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

## 6.14 Countries and states

Various dialogs through-out the e-commerce module and the check-out process display a list of countries and states. You can manage the list of available countries and states at **Site Manager** ->

**Development -> Countries.** These settings need to be pre-defined by some of the global administrators.



## 6.15 E-commerce and multi-site configuration

When you run multiple sites from a single Kentico CMS interface, it's important to consider that some e-commerce settings are shared by all websites. Thus, it's recommended that you use the multi-site support only for a single company that manages multiple e-commerce stores with shared settings.

The following settings are **shared** among websites:

- orders
- customers
- products
- product options
- manufacturers
- suppliers
- discounts
- departments
- tax classes
- currencies
- exchange rates
- order status
- public status
- internal status

The following settings are **site-specific**:

- store settings
- shipping options
- payment methods
- invoice template

## 6.16 Mapping document fields to product properties

You can map document fields into the product fields. Then, when the document is modified, the values are automatically updated in its product properties.

Please note that you cannot map product fields to document fields. The mapping doesn't work the other way round.

You can map the fields in the **Site Manager -> Development -> Document Types -> <select document type> -> E-commerce** dialog.

You can also choose to automatically mark the document as a product when a new document of the given type is created. In this case, you need to check the box **Automatically create product when a new document is created** and choose the **default department** to which the product will be assigned. In this case, you need to map at least the **Product name** and **Product price** source fields in the drop-down lists above.

The screenshot shows the Kentico CMS Site Manager interface. The left sidebar contains a tree view with categories like Categories, Countries, CSS stylesheets, Cultures, Custom tables, Document types (selected), E-mail templates, Form controls, Inline controls, Modules, Notifications, Page layouts, Page templates, Relationship names, System tables, Tag groups, Time zones, UI cultures, Web part containers, Web parts, Web templates, Widgets, and Workflows. The main content area displays the 'Document type properties' dialog for 'Product - Camera'. The dialog has tabs for General, Fields, Form, Transformations, Queries, Child types, Sites, E-commerce (selected), Alternative forms, Search fields, and Documents. Under the 'E-commerce' tab, there is a section titled 'Select document fields that will be used as a source for the specified product properties:' with the following fields:

- Product name source: CameraName
- Product image source: (none)
- Package weight source: (none)
- Package height source: (none)
- Package width source: (none)
- Package depth source: (none)
- Product price source: (none)
- Product description source: (none)

Below these fields, there is a checkbox labeled 'Automatically create product when a new document is created' which is currently unchecked. Underneath, there is a dropdown menu for 'Default department for new product:' set to 'Cameras'. At the bottom of the dialog is a green 'OK' button.

## 6.17 Sitemanager settings

### E-commerce



You can set following e-commerce module properties at **Site Manager -> Settings -> E-commerce**:

- **"Product" tab enabled** - Please see the [Enabling the e-commerce module](#) chapter for more details.
- **Default product image URL** - Default product image URL (virtual path). This image is used when product image is not specified.
- **My account URL** - My account page URL (virtual path)
- **Shopping cart URL** - Shopping cart page URL (virtual path)
- **Wishlist URL** - Wishlist page URL (virtual path)

### Payment Gateway - Authorize.NET

At **Site Manager -> Settings -> Payment Gateway - Authorize.NET** you can set following properties:

- **API Login**
- **Test mode**
- **Transaction key**

For more information please see the [Authorize.NET configuration](#) chapter.

### Payment Gateway - PayPal

At **Site Manager -> Settings -> Payment Gateway - PayPal** you can set following properties:

- **Business**
- **Cancel Return Url**
- **Notify Url**
- **Return Url**

For more information please see the [PayPal configuration](#) chapter.

## 6.18 Web.config settings

In the **web.config** file, you can set appropriate keys in the following location to modify functionality of the e-commerce module:

```
<configuration>
  <appSettings>
    <add key="CMSUseCustomEcommerceProviders" value="false" />
    .
    .
    .
  </appSettings>
</configuration>
```

The following keys can be set:

<b>CMSUseCurrentSKUData</b>	<p>If set to <b>false</b> (default value), the name and price of the existing order item are used while editing order items.</p> <p>If set to <b>true</b>, the current product name and product price of a product</p>
-----------------------------	--

<p><b>CMSEnableOrderItemEditing</b></p>	<p>are used while editing order items.</p> <p>If set to <b>true</b>, it is possible to edit order item price or order item name after the order is made. Please note that you can make these changes only if the <b>CMSUseCurrentSKUData</b> key is set to <b>false</b> at the same time.</p>  <p>It is set to <b>false</b> by default.</p>
<p><b>CMSUseMetaFileForProductImage</b></p>	<p>If set to <b>true</b> (default value), user is asked to choose a product image from the file system on their computer. The product image is uploaded to server and saved as a metafile of the given product (product image document of type <b>cms.file</b> is not created). The path to the given metafile is saved as a product image path (SKUImagePath).</p> <p>If set to <b>false</b>, user is asked to choose a document of type <b>cms.file</b> to become a product image. The path to the selected document is saved as a product image path (SKUImagePath).</p>
<p><b>CMSUseCustomEcommerceProviders</b></p>	<p>For more information, please see the <a href="#">Using custom providers</a> chapter,</p>
<p><b>CMSShoppingCartExpirationPeriod</b></p>	<p>Specify the number of days after which a shopping cart is considered to be old and is erased with the <b>Deleting old shopping carts</b> task.</p>

## Custom web.config settings

You can add your own settings (key and its value) into the **web.config** file. To work with these settings, use methods of the **CMS.SettingsProvider.SettingsKeyProvider** class.

The following example shows how to get the value (type of double) of the key named **MyCustomKey**.

**[C#]**

```
double value = CMS.SettingsProvider.SettingsKeyProvider.GetDoubleValue(
    "MyCustomKey" );
```

## 6.19 Localization settings

**KenticoCMS** supports localization of its features through localization expressions.

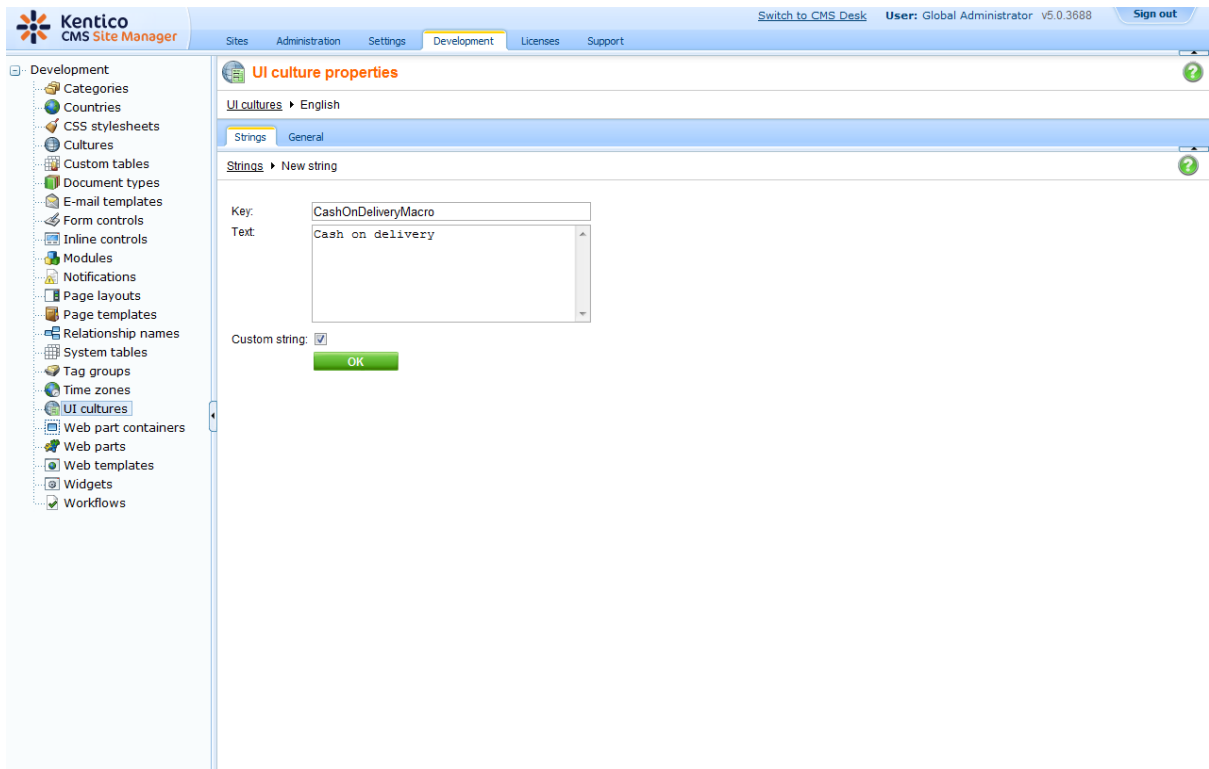
In the e-commerce module, the following features can be localized meaning that you can set their value either as plain text or as the macro expression.

- **Product display name**
- **Product option display name**
- **Product option category display name**
- **Payment method display name**
- **Shipping option display name**
- **Currency display name**
- **Tax class display name**
- **Discount coupon display name**
- **Discount level display name**
- **Order status display name**
- **Public status display name**
- **Country display name**
- **State display name**

Following example should suggest you how to supply a localized value into the document type transformation code or into your own aspx page.

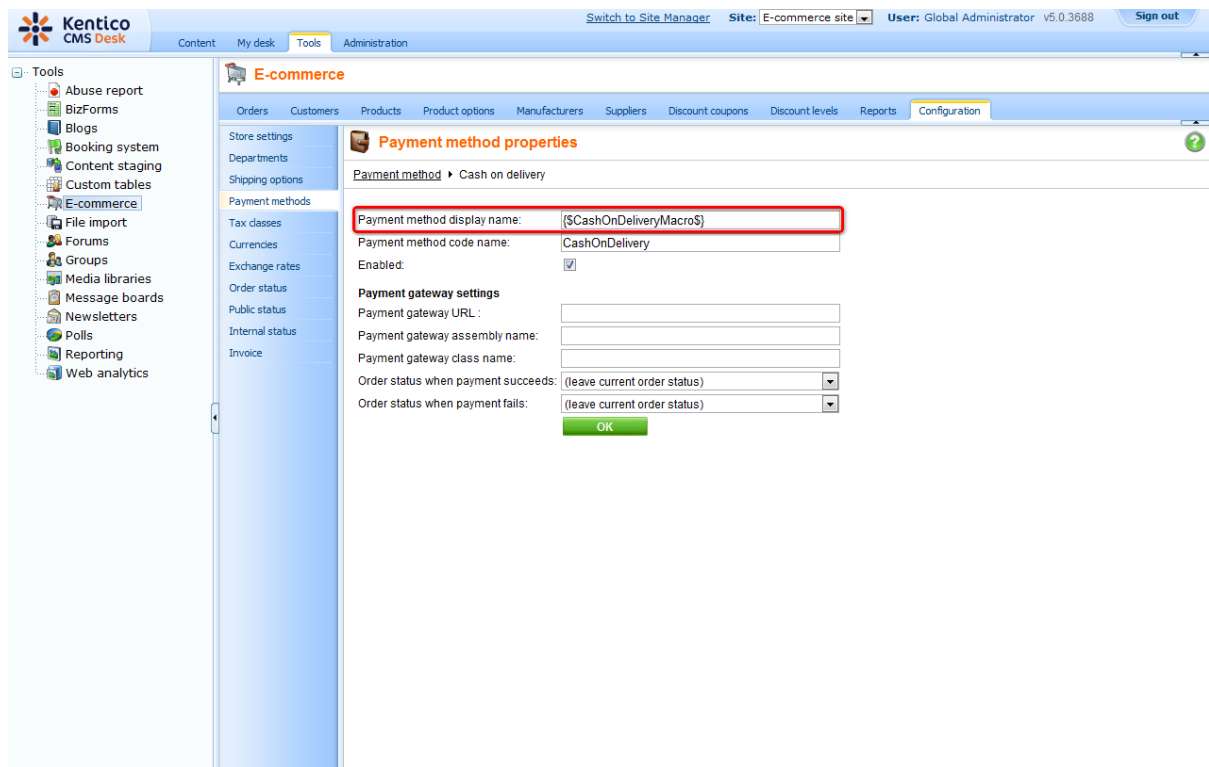
### 1. Creating localized expressions

At **CMS Site Manager -> Development -> UI cultures** you can create a new string for the given cultures.



## 2. Displaying localized expressions

For instance, if you have defined new string with the **Key** attribute *CashOnDeliveryMacro* and **Text** attribute *Cash on delivery* for the English culture and *Comptant à la livraison* for the french culture, using localization method **CMS.GlobalHelper.ResHelper.LocalizeString("{key}")** will result in displaying **Cash on delivery** to the English-speaking person and **Comptant à la livraison** to the French-speaking person.



For further information on the localization macros please refer to the **Localization Expressions** and **Appendix A - Macro expressions** chapters in **Kentico CMS Developer's Guide**.

## 6.20 Customizing invoice and e-mail templates

For your invoice and e-mail templates, you can use a whole range of pre-defined and data macros as you can find them at **CMSDesk -> Tools -> Ecommerce -> Configuration -> Invoice**.

### Pre-defined macros

Should you want to replace the current pre-defined macros, you need to enable the usage of the custom e-commerce provider (see [Using custom providers](#) for more details) and replace the appropriate method from [CustomOrderInfoProvider](#).

Macro	Description	Method
##BILLINGADDRESS##	Displays customer address	<code>string GetAddress(int addressId)</code>
##COMPANYADDRESS##	Displays company address	<code>string GetAddress(int addressId)</code>
##SHIPPINGADDRESS##	Displays shipping address	<code>string GetAddress(int addressId)</code>
##PAYMENTOPTION##	Displays payment method	<code>string GetPaymentOption(int paymentId)</code>
##SHIPPINGOPTION##	Displays shipping option	<code>string GetShippingOption(int shippingId)</code>
##INVOICENUMBER##	Displays invoice number	<code>string GetInvoiceNumber(object order)</code>

##ORDERDATE##	Displays order date	<code>string GetOrderDate(object order)</code>
##ORDERNOTE##	Displays order note	<code>string GetOrderNote(object order)</code>
##PRODUCTLIST##	Displays list of ordered products	<code>string GetProductList(DataTable dt, object currency, bool renderDiscount)</code>
##TOTALSHIPPING##	Displays total shipping for order	<code>string GetTotalShipping(double value, object currency)</code>
##TOTALPRICE##	Displays total price for order	<code>string GetTotalPrice(double value, object currency)</code>
##TAXRECAPITULATION##	Displays list of all taxes used in order	<code>string GetTaxRecapitulation(DataTable dt, object currency)</code>
##TAXREGISTRATIONID##	Displays customer tax registration ID	<code>string GetTaxRegistrationID(object customerObj)</code>
##ORGANIZATIONID##	Displays customer organization ID	<code>string GetOrganizationID(object customerObj)</code>

## Data macros

You can use data macros to evaluate the data fields of the order and related objects. See 'Kentico CMS Database Reference' for detailed column listing of the objects.

Following example displays code of the customer billing address state, however, you can get any value from the related objects: `{%BillingAddress.State.StateCode%}`

Macro	Description
<code>{%Order.OrderID%}</code>	Displays the value of specified order data column ( <b>COM_Order</b> )
<code>{%ShoppingCart.ShoppingCartID%}</code>	Displays the value of specified shopping cart data column ( <b>COM_ShoppingCart</b> )
<code>{%OrderStatus.StatusID%}</code>	Displays the value of specified order status data column ( <b>COM_OrderStatus</b> )
<code>{%BillingAddress.AddressID%}</code>	Displays the value of specified billing address data column ( <b>COM_Address</b> )
<code>{%BillingAddress.Country.CountryID%}</code>	Displays the value of specified billing address country data column ( <b>CMS_Country</b> )
<code>{%BillingAddress.State.StateID%}</code>	Displays the value of specified billing address state data column ( <b>CMS_State</b> )
<code>{%ShippingAddress.AddressID%}</code>	Displays the value of specified shipping address data column ( <b>COM_Address</b> )
<code>{%ShippingAddress.Country.CountryID%}</code>	Displays the value of specified shipping address country data column ( <b>CMS_Country</b> )
<code>{%ShippingAddress.State.StateID%}</code>	Displays the value of specified shipping address state data column ( <b>CMS_State</b> )
<code>{%CompanyAddress.AddressID%}</code>	Displays the value of specified company address data column ( <b>COM_Address</b> )
<code>{%CompanyAddress.Country.CountryID%}</code>	Displays the value of specified company address country data column ( <b>CMS_Country</b> )
<code>{%CompanyAddress.State.StateID%}</code>	Displays the value of specified company address state data column ( <b>CMS_State</b> )

<b>{%ShippingOption.ShippingOptionID%}</b>	Displays the value of specified shipping option data column ( <b>COM_ShippingOption</b> )
<b>{%PaymentOption.PaymentOptionID%}</b>	Displays the value of specified payment option data column ( <b>COM_PaymentOption</b> )
<b>{%Currency.CurrencyID%}</b>	Displays the value of specified currency data column ( <b>COM_Currency</b> )
<b>{%Customer.CustomerID%}</b>	Displays the value of specified customer data column ( <b>COM_Customer</b> )
<b>{%DiscountCoupon.DiscountCouponID%}</b>	Displays the value of specified discount coupon data column ( <b>COM_DiscountCoupon</b> )

## Custom macros

Following example shows how you can display customer total credit and current time in invoice or e-mails, however you can define your own custom macros.

Please refer to the **Appendix A - Macro expressions** chapter in **Kentico CMS Developer's Guide** for more details.

**[C#]**

```
using System;
using CMS.GlobalHelper;

public static string ResolveCustomMacro(MacroResolver sender, string expression,
out bool match)
{
    match = false;
    string result = expression;

    //
    // Custom macro examples
    //
    switch (expression.ToLower())
    {
        // Display customer total credit, macro in template: {#totalcredit#}
        case "totalcredit":

            match = true;

            // Get shopping cart object from resolver
            CMS.Ecommerce.ShoppingCartInfo cartObj = sender.SourceObject as
CMS.Ecommerce.ShoppingCartInfo;
            if ((cartObj != null) && (cartObj.ShoppingCartCustomerID > 0))
            {
                // Get customer total credit in default currency
                double totalCredit =
CMS.Ecommerce.CreditEventInfoProvider.GetCustomerTotalCredit(cartObj.
ShoppingCartCustomerID);

                CMS.Ecommerce.CurrencyInfo currentCurrency = null;
                double currentRate = 1.0;

                // Use shopping cart currency and its exchange rate
                if (cartObj.CurrencyInfoObj != null)
                {
                    currentCurrency = cartObj.CurrencyInfoObj;
                    currentRate = cartObj.ExchangeRate;

                    // Apply exchange rate
                    totalCredit =
CMS.Ecommerce.ExchangeTableInfoProvider.ApplyExchangeRate(totalCredit,
currentRate);

                }
                // Use default currency
                else
                {
                    currentCurrency = CMS.Ecommerce.CurrencyInfoProvider.
GetMainCurrency();
                }

                // Round value
                totalCredit =
CMS.Ecommerce.CurrencyInfoProvider.RoundTo(totalCredit, currentCurrency);

                // Return formatted value
                result =
```



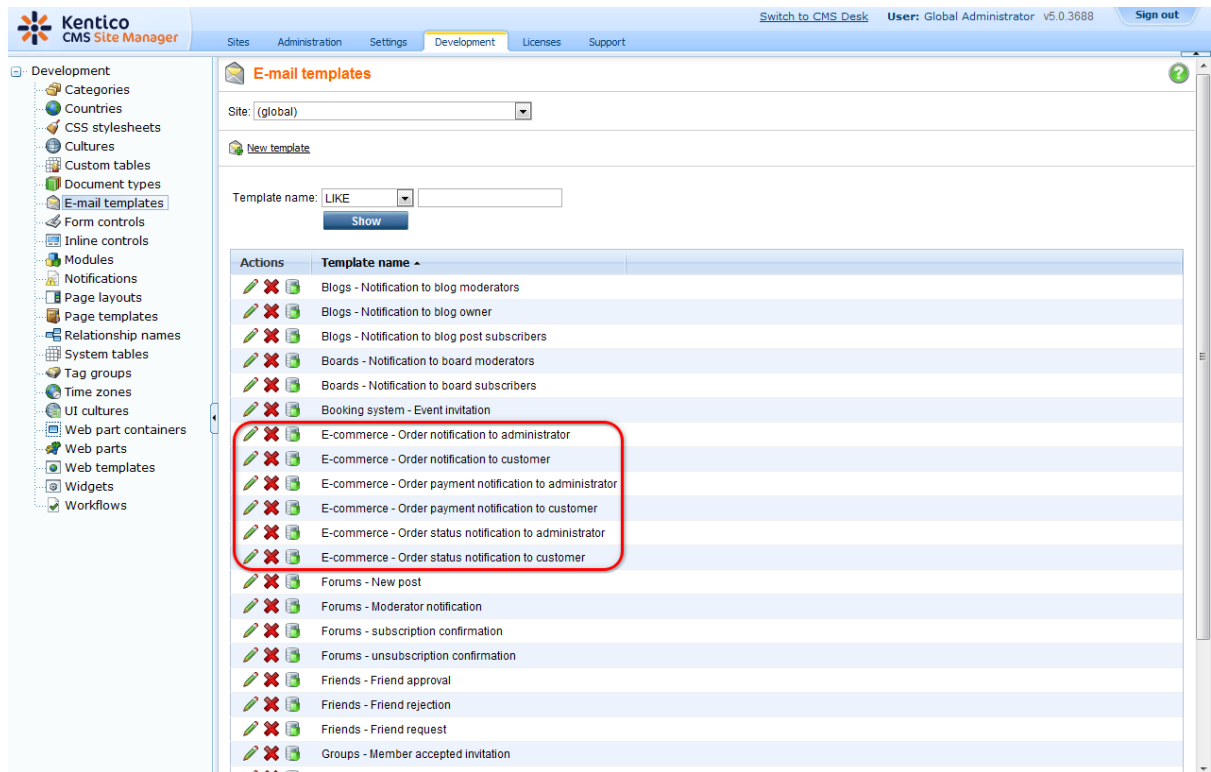
```
CMS.Ecommerce.CurrencyInfoProvider.GetFormattedPrice(totalCredit, currentCurrency);

    }
    break;

    // Display current date and time, macro in template: {#currenttime#}
    case "currenttime":
        result = DateTime.Now.ToString();
        break;
    }
    return result;
}
```

## E-mail templates

You can manage e-mail templates at **CMS Site Manager -> Development -> E-mail templates**.



## Customizing e-mail subject

Beside e-mail templates, you can use the very same macros in subject of an e-mail. All you have to do is change the appropriate localization strings as specified in the table below. You can change the given localization strings at **<your web project folder>/CMSResources/CMS.<given UI culture code >.resx** (for default culture it's **CMS.resx**)

E-mail template	E-mail subject localization string
order notification to administrator	ordernotification.administratorsubject
order notification to customer	ordernotification.customersubject
order payment notification to administrator	orderpaymentnotification.administratorsubject
order payment notification to customer	orderpaymentnotification.customersubject
order status notification to administrator	orderstatusnotification.administratorsubject
order status notification to customer	orderstatusnotification.customersubject

Following example shows how to customize subject of the order notifications which are sent to administrator:

1. Go to **<your web project folder>/CMSResources/** and open the **CMS.resx** file.
2. Find the **ordernotification.administratorsubject** key and change its value to **Order with ID {%**

`Order.OrderID%` - `{%OrderStatus.StatusDisplayName%}` so that the given part of code look in the following way.

```
<data name="ordernotification.administratorssubject" xml:space="preserve">
  <value>
    Order with ID {%Order.OrderID%} - {%OrderStatus.StatusDisplayName%}
  </value>
  <comment>IsNotCustom</comment>
</data>
```

This will include information about order ID and order status in the subject of an e-mail send to the administrator.

## Default settings of the e-mail subjects

When new order is made or items of the existing one are changed:

E-mail subject string	Default value
ordernotification.administratorssubject	Order: {%Order.OrderInvoiceNumber%}
ordernotification.customerssubject	Your order: {%Order.OrderInvoiceNumber%}

When payment is received:

E-mail subject string	Default value
orderpaymentnotification.administratorssubject	Order {%Order.OrderInvoiceNumber%}: Payment received
orderpaymentnotification.customerssubject	Your order {%Order.OrderInvoiceNumber%}: Payment received

When status of the existing order is changed:

E-mail subject string	Default value
orderstatusnotification.administratorssubject	Order {%Order.OrderInvoiceNumber%}: {%OrderStatus.StatusDisplayName%}
orderstatusnotification.customerssubject	Your order {%Order.OrderInvoiceNumber%}: {%OrderStatus.StatusDisplayName%}

## 6.21 Using product inventory

Kentico CMS allows you to manage your product inventory at **CMS Desk -> Tools -> E-commerce -> Products**. Here, you can quickly check if a product is available and how many items do you have in stock.

The screenshot shows the Kentico CMS Desk interface. The top navigation bar includes 'Content', 'My desk', 'Tools', and 'Administration'. The 'Tools' menu is expanded, showing 'E-commerce' as the selected tool. The 'E-commerce' sub-menu is also expanded, showing 'Orders', 'Customers', 'Products', 'Product options', 'Manufacturers', 'Suppliers', 'Discount coupons', 'Discount levels', 'Reports', and 'Configuration'. The 'Products' sub-menu is selected, and the 'New product' form is visible. The form includes fields for 'Product name', 'Product number', 'Department', 'Public status', 'Internal status', and 'Sites'. Below the form is a table of products with columns for 'Actions', 'Product name', 'Product number', 'Product price', 'Available items', 'Public status', 'Internal status', and 'Enabled'. The 'Available items' column is highlighted with a red box.

Actions	Product name	Product number	Product price	Available items	Public status	Internal status	Enabled
	Acer Aspire 3105WLMi		490	-10			Yes
	Acer configurator		899	2			Yes
	Acer configurator		899	-1			Yes
	Acer Extensa 7620G		99	0			Yes
	Apple iPhone		499	0	Featured products		Yes
	Apple MacBook Air		2094	2			Yes
	Asus A639		470	1			Yes
	Asus F3U AP059C		999	1			Yes
	Canon Digital Rebel XTi		597	0	Featured products		Yes
	Canon PowerShot A720 IS		195	95			Yes
	Compaq Presario		659	-1			Yes
	Configurable PC		0	0			Yes
	Creative ZEN		150	-7	Featured products		Yes
	Creative Zen Stone		50	0			Yes
	Emgeton M6 Cult		145	0			Yes
	Emgeton M7 Cult		215	4	Featured products		Yes

There are two ways you can handle products in your inventory:

- 1.) Sell a product only if they are some available items in stock at **CMS Desk -> Tools -> E-commerce -> Products**.
- 2.) Always sell a product no matter if there are any available items in stock.

You can distinguish between these two by checking the **Sell only if items are available** checkbox either at **CMS Desk -> Tools -> E-commerce -> Products -> <edit given product> -> General** or on the **Product** tab of the product chosen in the content tree.

The screenshot shows the Kentico CMS Desk interface for configuring an e-commerce product. The left sidebar lists various tools, and the main area displays the configuration for 'Apple MacBook Air'. The 'Inventory' section is highlighted with a red box, showing the following settings:

- Price: 2094.00
- Department: Hardware
- Manufacturer: Apple
- Supplier: (none)
- Public status: (none)
- Internal status: (none)
- Availability (days): 0
- Product image URL: Upload: [Browse...]
- Package weight: 1.36
- Package height: [ ]
- Package width: [ ]
- Package depth: [ ]
- Inventory Available items: 2
- Sell only if items are available:

## 1. Sell product only if there are available items in stock

For enabling this option, you have to enter the number of available items into the **Available items** text box and check the **Sell only if items are available** checkbox. An order is accepted only if the number of ordered items is no greater than the number of available items.

**Inventory**

Available items:


Sell only if items are available:

The sufficiency of the available items is checked every time a user does one of the following:

- Add product to the **shopping cart**
- Click the **Update** button in the shopping cart
- Click the **Checkout** button in the shopping cart
- Click the **Order now** button in the Order Preview

If you want to buy a product with product options, the chosen product and product options have to be enabled and there has to be sufficient number of available items of the given product with the given product option. Otherwise the error message appears.

Step 1 of 6 - Add some products to the shopping cart



### Shopping cart

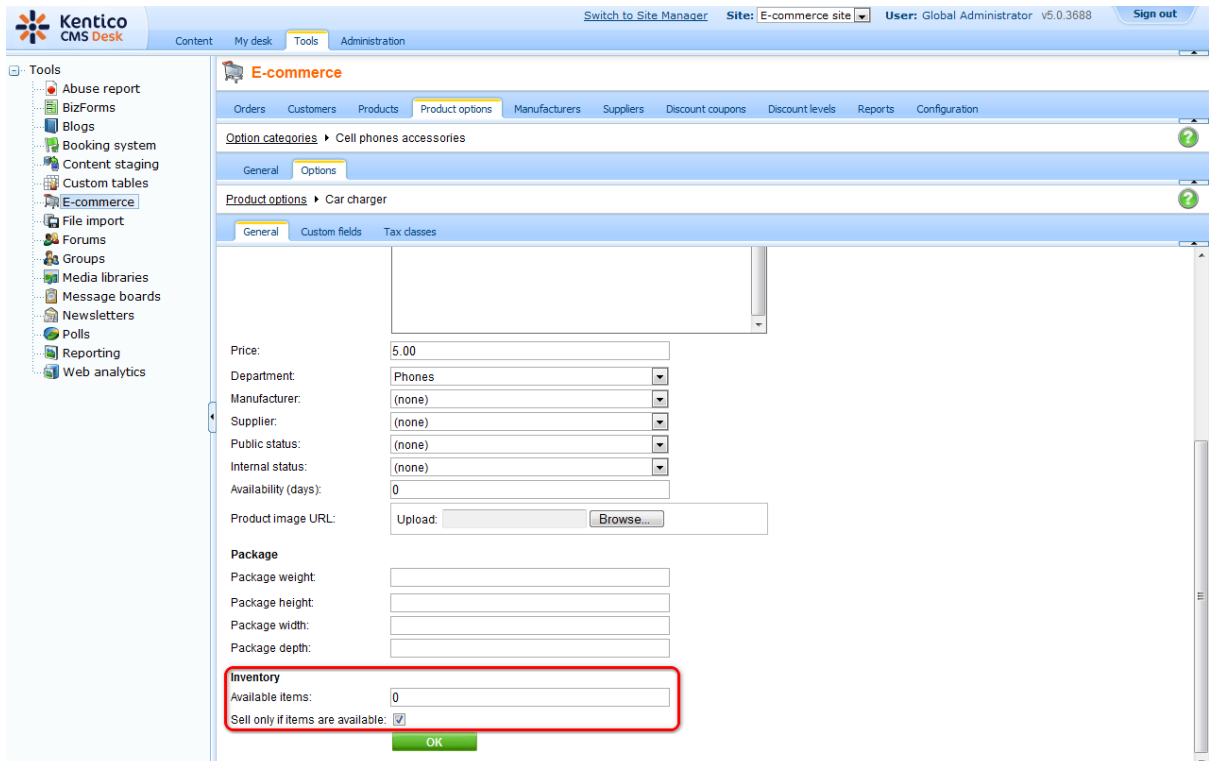
Currency: U.S. Dollar

Remove	Product name	Units	Unit price	Tax	Subtotal
<input type="checkbox"/>	<a href="#">Asus A639</a> The product is currently not available in required quantity. The quantity was set to maximum available quantity, which is 1.	<input type="text" value="1"/>	470.00	0.00	<input type="text" value="470.00"/>

If you have a coupon code, please enter it here:

Total shipping: \$0.00  
Total price: **\$470.00**

Please note that you can check the **Sell only if items are available** checkbox for product options as well. You can set check the **Sell only if items are available** checkbox and enter the number of available items for product options at **CMS Desk -> Tools -> E-commerce -> Products Options -> <edit option category> -> Options -> <edit product options>**.



Kentico CMS Desk

Switch to Site Manager Site: E-commerce site User: Global Administrator v5 0.3688 Sign out

Tools Administration

E-commerce

Orders Customers Products **Product options** Manufacturers Suppliers Discount coupons Discount levels Reports Configuration

Option categories Cell phones accessories

General Options

Product options Car charger

General Custom fields Tax classes

Price:

Department:

Manufacturer:

Supplier:

Public status:

Internal status:

Availability (days):

Product image URL: Upload:

Package

Package weight:

Package height:

Package width:

Package depth:


Inventory

Available items:

Sell only if items are available:

If the checkbox is checked, there has to be the sufficient quantity of the given product option in stock so that an order is accepted. Otherwise the error message appears.

Step 1 of 6 - Add some products to the shopping cart



**Shopping cart**

Currency: U.S. Dollar

Remove	Product name	Units	Unit price	Tax	Subtotal
<input type="checkbox"/>	<a href="#">Motorola V3 Razr</a>	<input type="text" value="1"/>	130.00	0.00	<input type="text" value="130.00"/>
	- Car charger The product is currently not available in required quantity. The maximum available quantity is 0.	<input type="text" value="1"/>	5.00	0.00	<input type="text" value="5.00"/>

If you have a coupon code, please enter it here:

Total shipping: \$0.00  
Total price: **\$135.00**

Please note that the very same check is made anytime a user edit the already existing order. When an item is removed from the order, it is added back to stocked items.

## 2. Always sell product

When the **Sell only if items available** checkbox is unchecked, an order is processed disregarding the number of available items of the given product or product option. Therefore, a customer can order more items that it is readily available in stock. Should that happen, the number of available items will be negative. Therefore, you'll know exactly how many items do you need to order to meet demands.

**Please note:** You can check the number of available items in stock at **CMSDesk -> Tools -> Ecommerce -> Reports -> Inventory** as well.

The screenshot displays the Kentico CMS Desk interface for the E-commerce module. The left sidebar shows a navigation tree with 'Tools' expanded, and 'E-commerce' selected. The main content area is titled 'E-commerce' and has a sub-tab 'Reports' selected. Under 'Reports', 'Inventory' is chosen. The 'Inventory' report shows a table of products with columns for SKU name, Product number, Department name, and Available items. A search filter 'Available items less than: 10' is visible with an 'Update' button. The table lists various hardware products like DDR2 memory, CPUs, and laptops.

SKU name	Product number	Department name	Available items
1024MB DDR2		Hardware	0
2048MB DDR2		Hardware	0
512MB DDR2		Hardware	-12
Abit AW9D - Intel 975X		Hardware	2
ACER 22"		Hardware	-1
Acer Aspire 3105WLMi		Hardware	-10
Acer configurator		Hardware	-1
Acer configurator		Hardware	2
Acer Extensa 7620G		Hardware	0
ActionTech Gladiator CK-18B blue		Hardware	4
AMD Sempron 1800Mhz		Hardware	0
AMD X2 1800 Mhz		Hardware	0
AMD X2 2000 Mhz		Hardware	0
AOC 19"		Hardware	0
AOC 20"		Hardware	0
Apple iPhone		Phones	0
Asus A639		PDA & Pocket PC	1
Asus F3U AP059C		Hardware	1
Asus M2N-E - nForce 570 Ultra		Hardware	4

## 6.22 Discounts overview


In the Kentico CMS E-commerce module, you can define three different types of discounts for your customers:

- **Discount coupons** - discount from the price of the specified products. For more details see the [Discount coupons](#) chapter.
- **Discount levels** - discount that is assigned to the customer for all their purchases. For more details see the [Discount levels](#) chapter.
- **Volume discount** - discount assigned to the product that applies when the customer purchases specified amount of the given product. For more details see the [Products](#) chapter.

In the following examples, we purchase five units of **Apple MacBook Air** with a total price of \$10470 without any discount. Each example demonstrates where you can specify the given type of discount and how it affects the total price of the purchase.



Step 1 of 6 - Add some products to the shopping cart



**Shopping cart**

Currency: U.S. Dollar

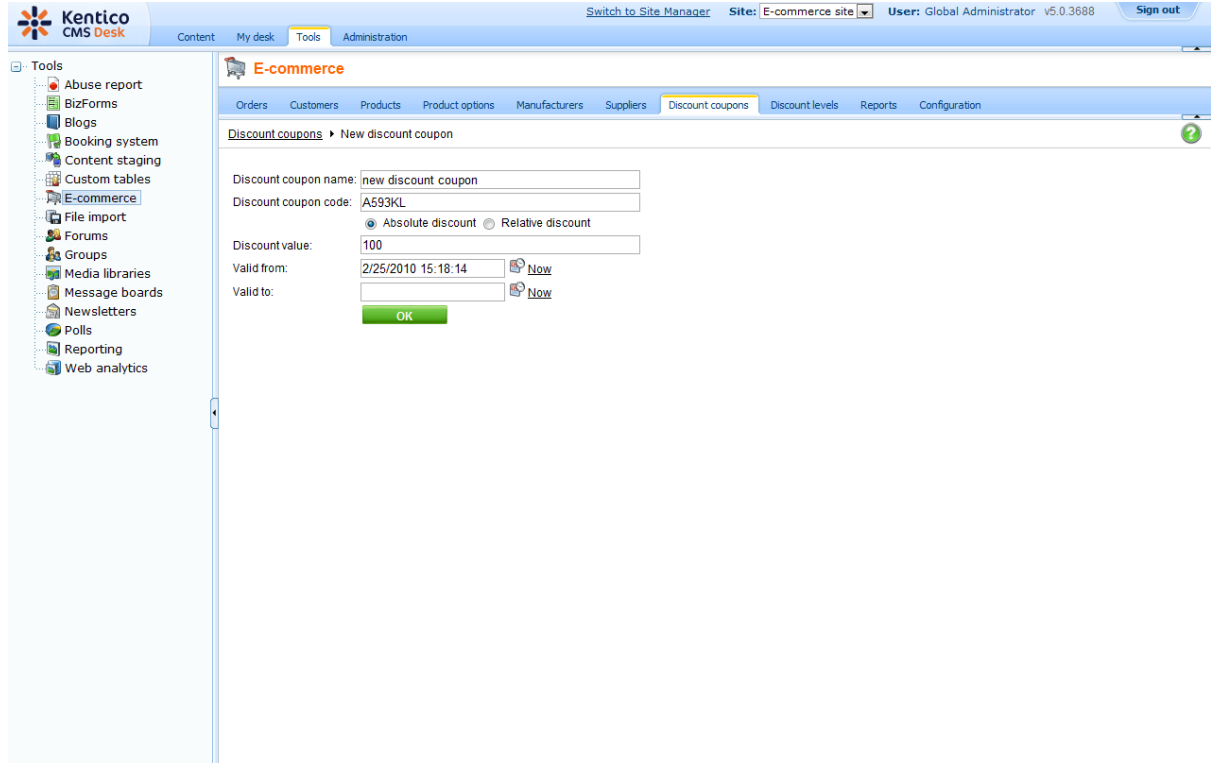
Remove	Product name	Units	Unit price	Tax	Subtotal
<input type="checkbox"/>	<a href="#">Apple MacBook Air</a>	5	2094.00	0.00	10470.00

If you have a coupon code, please enter it here:

Total shipping: \$0.00  
Total price: **\$10470.00**

## Discount coupons

You can create a new discount coupon or edit existing ones at **CMS Desk -> Tools -> E-commerce -> Discount coupons**.



Kentico CMS Desk

Switch to Site Manager Site: E-commerce site User: Global Administrator v5.0.3688 Sign out

Content My desk Tools Administration

**E-commerce**

Orders Customers Products Product options Manufacturers Suppliers **Discount coupons** Discount levels Reports Configuration

Discount coupons ▶ New discount coupon

Discount coupon name:

Discount coupon code:

Absolute discount  Relative discount

Discount value:

Valid from:


Valid to:

Tools

- Abuse report
- BizForms
- Blogs
- Booking system
- Content staging
- Custom tables
- E-commerce**
- File import
- Forums
- Groups
- Media libraries
- Message boards
- Newsletters
- Polls
- Reporting
- Web analytics

While making purchase, a customer is required to enter a coupon code into the given text box to make use of it. The given discount is subtracted from the product price.

Step 1 of 6 - Add some products to the shopping cart



**Shopping cart**

Currency: U.S. Dollar

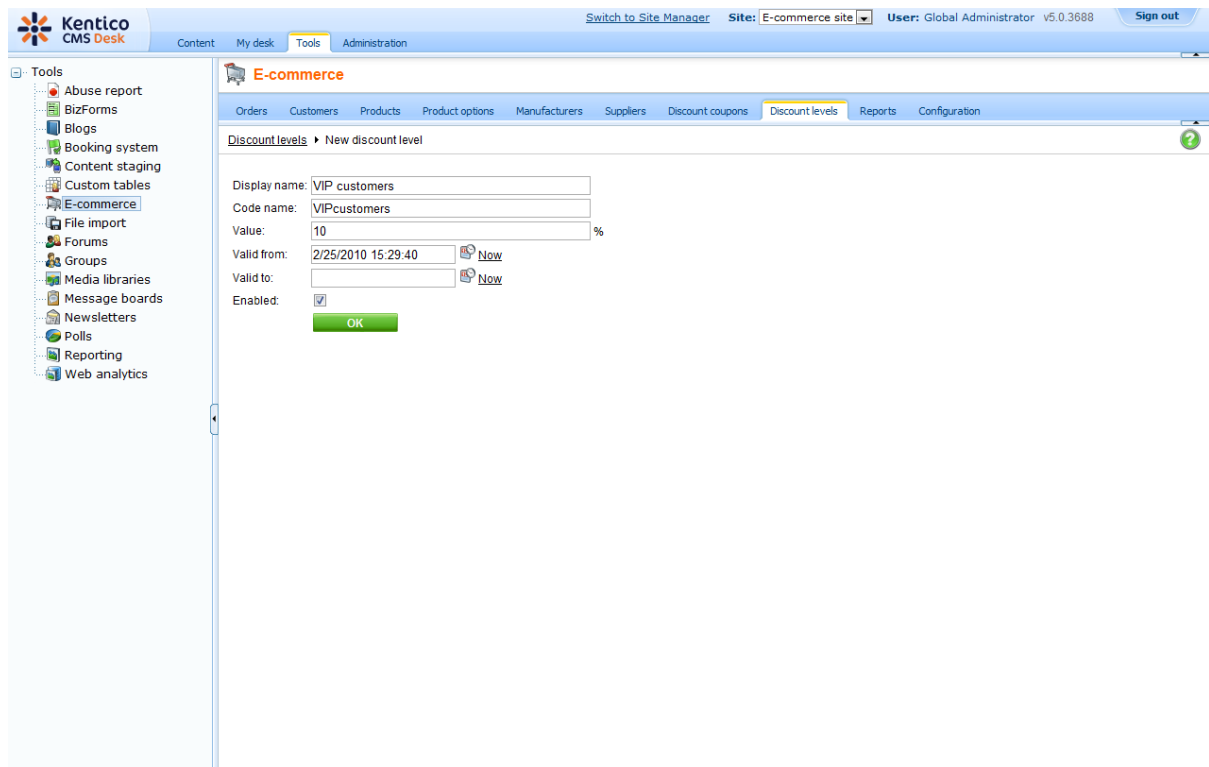
Remove	Product name	Units	Unit price	Unit discount	Tax	Subtotal
<input type="checkbox"/>	<a href="#">Apple MacBook Air</a>	<input type="text" value="5"/>	2094.00	100.00	0.00	9970.00

If you have a coupon code, please enter it here:

Total shipping: \$0.00  
**Total price: \$9970.00**

## Discount levels

You can set up a new discount level or edit existing ones at **CMS Desk -> Tools -> E-commerce -> Discount levels**.



Kentico CMS Desk

Switch to Site Manager Site: E-commerce site User: Global Administrator v5.0.3688 Sign out

Content My desk Tools Administration

**E-commerce**

Orders Customers Products Product options Manufacturers Suppliers Discount coupons **Discount levels** Reports Configuration

Discount levels ▶ New discount level

Display name:

Code name:

Value:  %

Valid from:

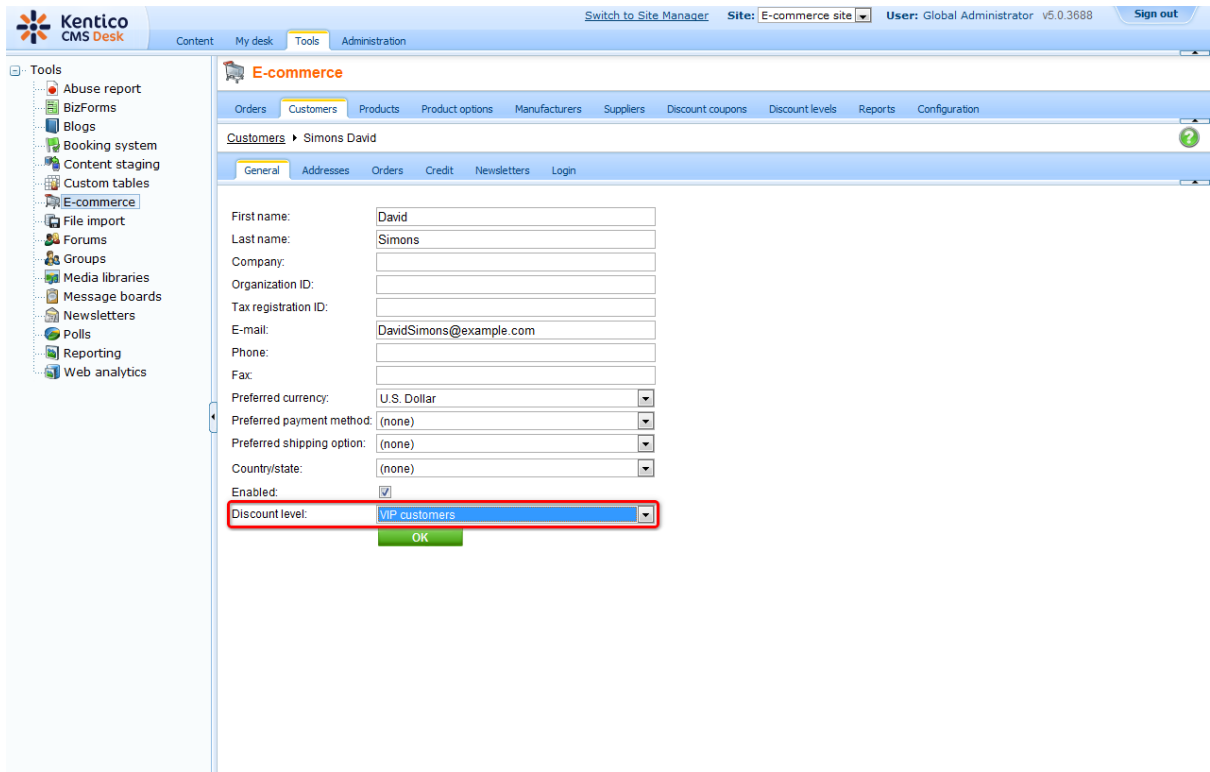
Valid to:

Enabled:

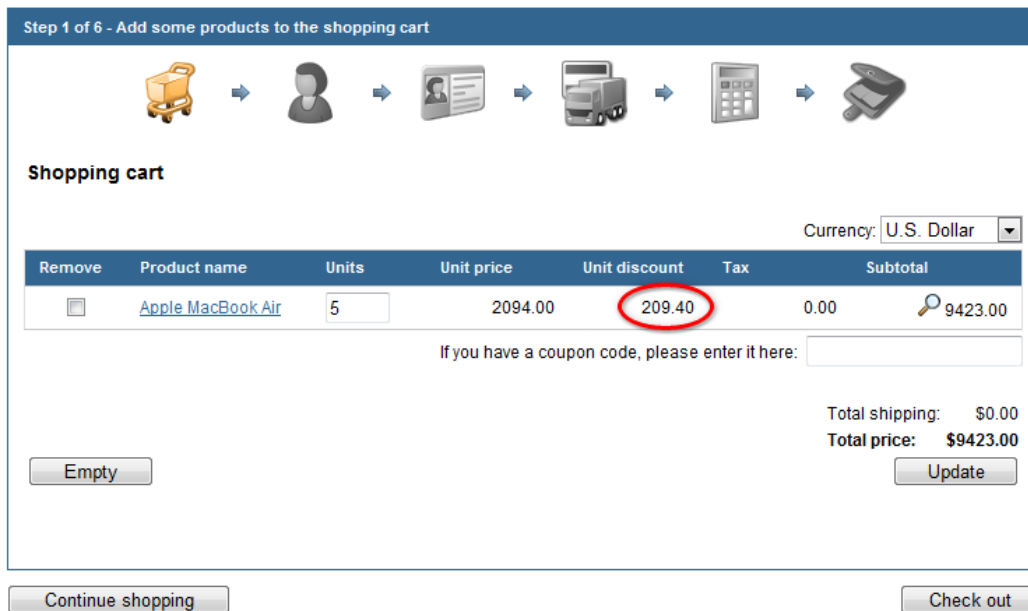
Tools

- Abuse report
- BizForms
- Blogs
- Booking system
- Content staging
- Custom tables
- E-commerce**
- File import
- Forums
- Groups
- Media libraries
- Message boards
- Newsletters
- Polls
- Reporting
- Web analytics

After creating a new discount level, you have to assign it to those users who you want to take advantage of it.

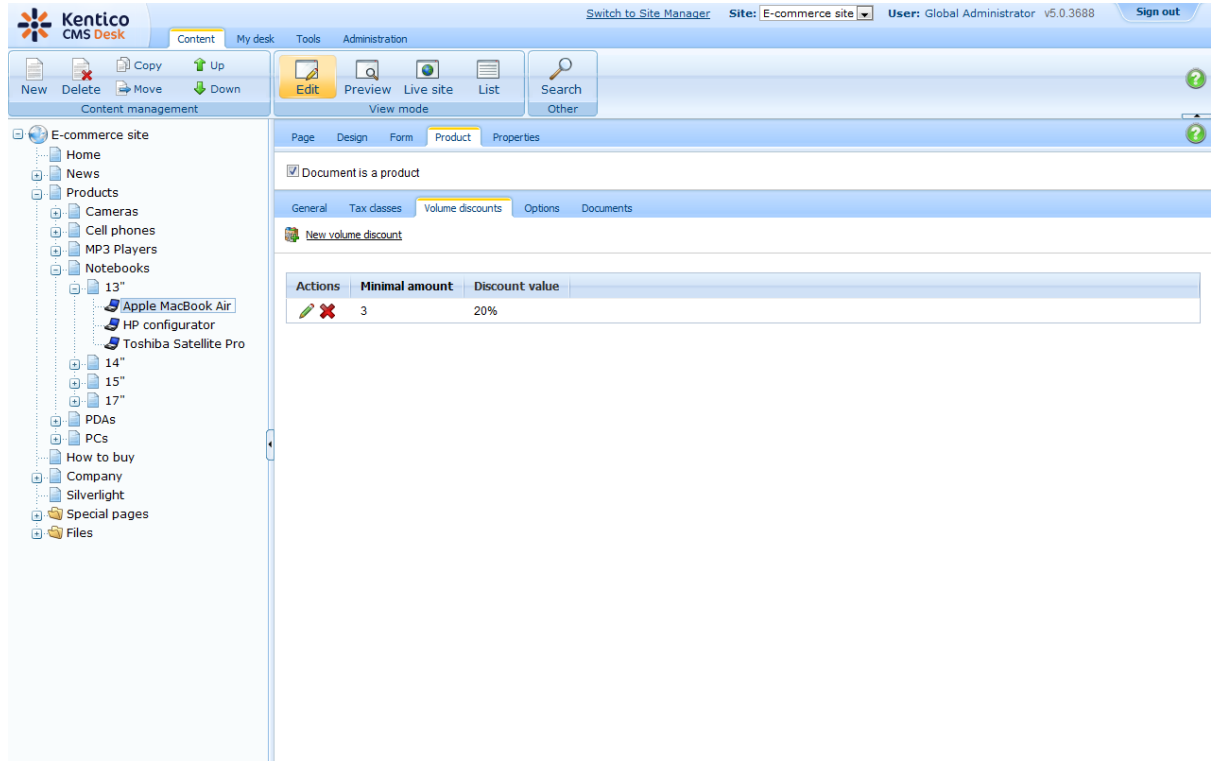


Before making a purchase, a user has to sign in to their account so that the assigned discount level is automatically take into account while calculating the unit and total price.





## Volume discounts

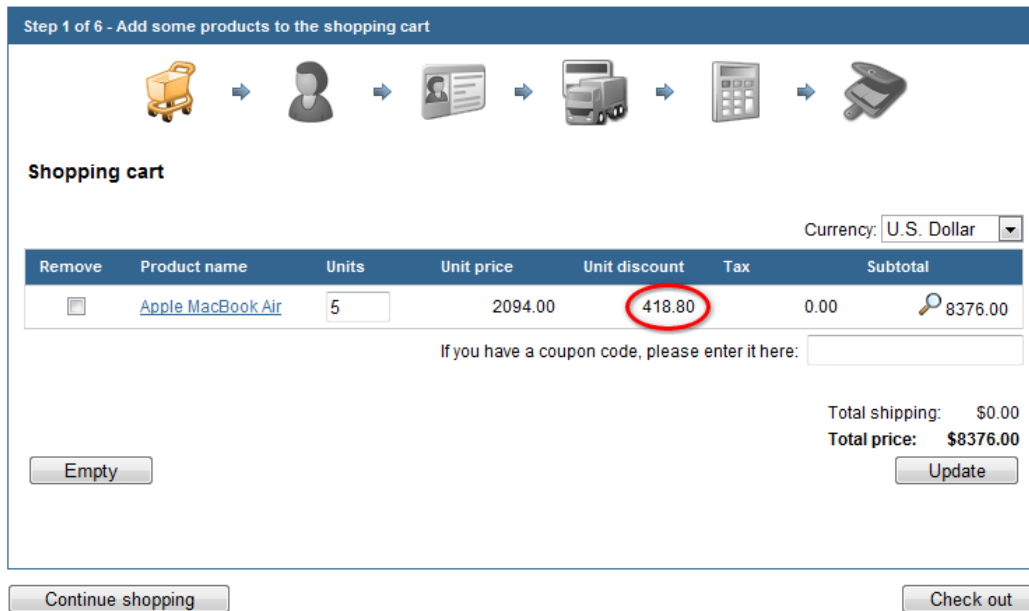
You can define volume discounts at **CMS Desk -> Content -> <choose product> -> Product -> Volume discounts**.



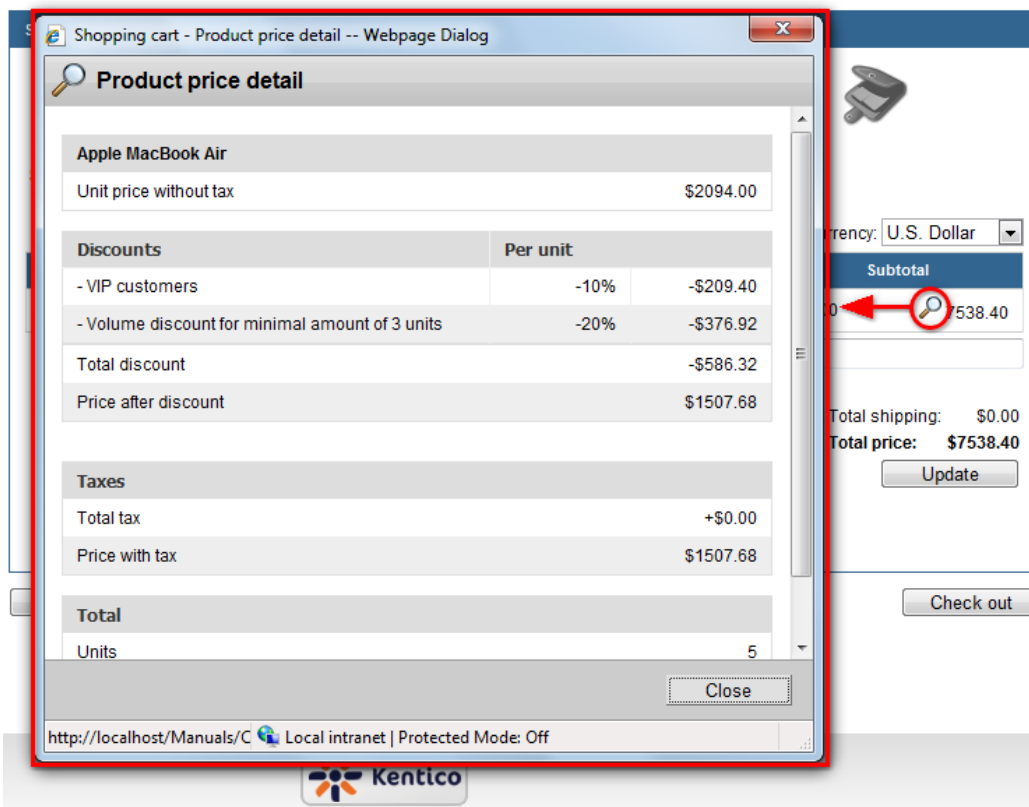
The screenshot displays the Kentico CMS Desk interface. The left sidebar shows a tree view of the site structure, with 'Products' expanded to show various categories like 'Cameras', 'Cell phones', and 'Notebooks'. The main content area is titled 'Volume discounts' and includes a 'New volume discount' button. Below this, a table lists the configured volume discounts:

Actions	Minimal amount	Discount value
 	3	20%

On condition a user adds to the **shopping cart** sufficient amount of units of the product with the assigned volume discount, the given discount level is automatically taken into account and its subtracted from the unit price.



You can combine all the discount types. While combining the discount coupon and the discount level, though, only higher discount applies.



For more details about how are discounts calculated see [Figure B: Discount calculation](#).

## 6.23 Deleting old shopping carts

The deletion of old shopping carts can be done automatically with the pre-defined task.

By default, all the shopping carts older than 30 days are considered old and therefore are erased when running the **Deleting old shopping carts** task. You can change the period of time after which the shopping cart expires by modifying the **CMSShoppingCartExpirationPeriod** key in the **web.config** file. See the [Web.config settings](#) chapter for more details.

This task erases not only old shopping carts, but anonymous customers with no order as well. This anonymous customer with no order appears when an anonymous user adds an item into the shopping cart. They are then required to enter their name and e-mail and they become a customer. If this anonymous user, for whatever reason, doesn't get through all steps of the checkout process, they remain in the system as anonymous customers with no order.

You can schedule the deletion of old shopping carts task by setting the **Scheduler** at **CMS Site Manager -> Administration -> Scheduled tasks**.

The screenshot displays the 'Task properties' configuration window for the 'Delete old shopping carts' task. The interface includes a navigation menu on the left with categories like Administration, Avatars, Bad words, Badges, Banned IPs, E-mail queue, Event log, Permissions, Recycle bin, Roles, Scheduled tasks, Smart search, System, UI personalization, Users, and Web farm. The main content area shows the following configuration details:

- Task display name:** Delete old shopping carts
- Task name:** Ecommerce.DeleteOldShoppingCarts
- Task assembly name:** CMS.Ecommerce
- Task class name:** CMS.Ecommerce.ShoppingCartCleaner
- Period:** Day
- Start time:** 1/30/2007 11:50:00 (with a 'Now' button)
- Every:** 1 Day
- Task interval:** Days:  Monday,  Tuesday,  Wednesday,  Thursday,  Friday,  Saturday,  Sunday
- Task data:** (Empty text area)
- Task enabled:**
- Delete task after last run:**
- Server name:** (Empty text field)

An 'OK' button is located at the bottom of the configuration area.

For more information please refer to the **Scheduler** chapter in **Kentico CMS Developer's guide**.

**Part**

---

**VII**

**Purchase process and payment gateways**

## 7 Purchase process and payment gateways

### 7.1 Purchase process overview

The standard purchase (checkout) process looks like this:

1. The user browses the website and adds the products to the **shopping cart**. They can modify the number of items in the cart. They can also insert a discount coupon.
2. The user clicks the **Checkout** button. They can choose from three options:
  - **Sign in using an existing account** - if they already have a user account
  - **Create a new account**
  - **Continue as anonymous customer** - this option doesn't require the user to create a user account, but they need to enter their details with every purchase. You can enable/disable this option at **CMS Desk -> Tools -> E-commerce -> Configuration -> Store Settings -> Allow anonymous customers**.
3. The user enters/updates customer data, including first name, last name, e-mail and company name.
4. The customer enters billing/shipping address or chooses from the list of previously used addresses. The user can also choose to enter an additional company headquarters address that can be enabled at **CMS Desk -> Tools -> E-commerce -> Configuration -> Store Settings -> Use an extra company address**.
5. The user chooses the payment and shipping method.
6. The user previews the order.
7. The user completes the order or continues with on-line payment. The order status is set to **New**. The order notification e-mail is sent to store owner and to the customer if it's configured in **CMS Desk -> Tools -> E-commerce -> Configuration -> Store Settings** dialog.
8. The payment is processed by the payment gateway. If the payment succeeds/fails, the order status is set to values specified at **CMS Desk -> Tools -> E-commerce -> Configuration -> Payment methods -> <select some payment method> -> Order status when payment succeeds/fails**. A payment notification e-mail is sent to store owner and to the customer if it's configured in **CMS Desk -> Tools -> E-commerce -> Configuration -> Store Settings** dialog and if the payment is successfully completed.
9. The user is redirected to the page configured in the properties of the **Shopping cart** web part, in the **Default URL after purchase** value.

#### Shopping cart content

The shopping cart content is stored in the database and it's bound to the current user name. If the user is anonymous, the cart ID is stored in a browser cookie. It ensures that the cart content is preserved even if the session is lost during application restart. You can find more details on how the shopping cart is retrieved in [Figure D - Shopping cart retrieval](#).

When the checkout process is completed (regardless if the payment succeeds or fails), a new order is created and the shopping cart content is moved to the order. The cart is empty then.



The cart content is accessible in your code using the **ShoppingCartInfoObj** class in the checkout process steps.

## 7.2 Customizing the purchase process

The purchase process can be enhanced with your custom steps or you can remove some steps. This can be done at **CMS Desk -> Tools -> E-commerce -> Configuration -> Store settings -> Checkout process**.

For each step, you can specify the following values:

<b>Caption</b>	The step caption in the checkout process.
<b>Code name</b>	The name of the site used in the code.
<b>Image file name</b>	Name of the file used in the checkout process header. The image must be stored in folder ~\App_Themes\<<stylesheet name>\Images\ShoppingCart.
<b>Control path</b>	Virtual path to the ASCX control representing the dialog in the given step. Example:  ~/CMSEcommerce/ShoppingCart/ShoppingCartOrderAddresses.ascx
<b>Show on the live site</b>	Indicates if this step should be used in the checkout process on the live site. (see below)
<b>Show in Customer section</b>	Indicates if this step should be used in the checkout process when creating a new order for the given customer. (see below)
<b>Show in Order section</b>	Indicates if this step should be used in the checkout process for a new order created from the administration interface. (see below)
<b>Show in Order items section</b>	Indicates if this step should be used in the checkout process when editing items of an existing order. (see below)

See chapter [Developing custom dialogs for the checkout process](#) for more details on development of custom steps.

### Types of the checkout process

There are four different types of checkout process:

- Checkout process 1 - New order on the live site
- Checkout process 2 - New order from section **CMSDesk -> Tools -> E-commerce -> Customers/ (some customer) -> Orders**.
- Checkout process 3 - New order from section **CMSDesk -> Tools -> E-commerce -> Orders**.
- Checkout process 4 – Editing existing order from section **CMSDesk -> Tools -> E-commerce -> Orders -> (some order) -> Items**.

Below, you can find the screenshots of these types:

#### Checkout process 1 – New order on the live site

**E-commerce starter site**

Electronics for you

Shopping cart | My account | My wishlist  
**Total price:** **\$537.30**

Home News Products How to buy Company Silverlight Global Administrator [Sign out](#)

Step 1 of 6 - Add some products to the shopping cart

Shopping cart

Currency: U.S. Dollar

Remove	Product name	Units	Unit price	Unit discount	Tax	Subtotal
	<a href="#">Canon Digital Rebel XT</a>	<input type="text" value="1"/>	597.00	59.70	0.00	537.30

If you have a coupon code, please enter it here:

Total shipping: \$0.00  
**Total price: \$537.30**

Quick links  
 >> [How to buy](#)  
 >> [Company](#)  
 >> [News](#)  
 >> [Home](#)

Powered by Kentico

**Checkout process 2 - New order from section CMSDesk -> Tools -> E-commerce -> Customers -> (some customer) -> Orders**

**E-commerce**

Orders Customers Products Product options Manufacturers Suppliers Discount coupons Discount levels Reports Configuration

Customers Administrator Global

General Addresses Orders Credit Newsletters Login

Orders New order

**Step 1 of 5 - Select billing and shipping address**

**Billing address**

Billing address: (new) [v]

Name (company or personal): Global Administrator

Address lines: [ ] [ ] [ ]

City: [ ]

ZIP: [ ]

Country: USA [v]

(none) [v]

Phone number: [ ]

**Shipping address**

My shipping address is different from the billing address.

Back Next

### Checkout process 3 - New order from section CMSDesk -> Orders

**E-commerce**

Orders Customers Products Product options Manufacturers Suppliers Discount coupons Discount levels Reports Configuration

Orders New order

**Step 1 of 6 - Select customer**

**Customer**

Customer detail or its part: [ ] Search

[ ]

Next

### Checkout process 4 - Editing existing order from section CMSDesk -> Tools -> E-commerce -> Orders -> (some order) -> Items.

**E-commerce**

Orders Customers Products Product options Manufacturers Suppliers Discount coupons Discount levels Reports Configuration

Orders ▾ 3

General Shipping Billing **Items** Invoice History

Step 1 of 2 - Add some products to the shopping cart

Shopping cart

Currency: U.S. Dollar ▾

Remove	Product name	Units	Unit price	Unit discount	Tax	Subtotal
<input type="checkbox"/>	Creative ZEN	<input type="text" value="1"/>	150.00	15.00	20.25	155.25
<input type="checkbox"/>	Samsung SGH i620	<input type="text" value="1"/>	650.00	65.00	0.00	585.00
	- Red	1	0.00	0.00	0.00	0.00
<input type="checkbox"/>	Nikon Coolpix S51	<input type="text" value="1"/>	1950.00	598.65	0.00	1351.35

If you have a coupon code, please enter it here:

Total shipping: \$0.00  
Total price: **\$2091.60**

Send order changes by e-mail

## 7.3 Developing custom dialogs for the checkout process

### Creating a custom checkout process step

1. Create new web user control (\*.ascx) and place your own form controls into it.
2. Go to its code behind and set control class to inherit from **CMS.EcommerceProvider.ShoppingCartStep**.
3. There are several methods you can override to reach the required functionality:
  - **IsValid()** – Validates current step custom data and returns validation result. True – all step data is correct and can be processed, False – some step data is not correct or missing and cannot be processed. In this case appropriate error message should be displayed. By default, it returns True.
  - **ProcessStep()** – Processes current step information (updates shopping cart data) and returns result of this action. True – shopping cart data was updated successfully and customer can be moved to the next checkout process step, False – shopping cart update failed and customer cannot be moved to the next step. In this case appropriate error message should be displayed. By default, it returns True.
  - **ButtonBackClickAction()** – Defines action which is run after the Back button is clicked. By default, the parent shopping cart control method ButtonBackClickAction() is called which moves customer one step backward in the checkout process.
  - **ButtonNextClickAction()** – Defines action which is run after the “Next button” is clicked. By default, the parent shopping cart control method ButtonNextClickAction() is called which moves customer one step forward in the checkout process when the current step data is valid and processed successfully.
4. There are several properties you should use to get or set required information:
  - **ShoppingCartControl** – parent shopping cart control the step belongs to
  - **ShoppingCartInfoObj** – shopping cart object which stores all data during the checkout process

- **CheckoutProcessStep** – checkout process step information

5. The step control is created and can be registered as your custom checkout process step.

If the control represents checkout process steps in different [checkout process types](#) and these steps differ from each other only a little, you can create one control and specify a different behavior depending on the checkout process type as follows:

**[C#]**

```
using System;
using CMS.EcommerceProvider;
using CMS.GlobalHelper;

switch (this.ShoppingCartControl.CheckoutProcessType)
{
    case CheckoutProcessEnum.LiveSite:
        // Here comes the code which will be run only
        // when it is a checkout process on the live site
        break;

    case CheckoutProcessEnum.CMSDeskOrder:
        // Here comes the code which will be run only
        // when it is a checkout process in the section CMSDesk/Tools/E-commerce/
        Orders
        break;

    default:
        // Here comes the code which will be run in all other cases
        break;
}
```

### Example 1 - My step

The following example shows a simple checkout process step definition. It displays the total price and an editable field to insert a customer comment. After the **Next** button is clicked, the editable field is checked for emptiness. If it is not empty the customer comment is saved and shopping cart data is updated. Otherwise, an appropriate error message is displayed. If the customer is a member of role "VipCustomers", an extra step with an additional form for VIP customers is loaded (MyVipStep.ascx). The **Back** button action is not overridden.

**[MyStep.ascx]**

Please note: If you installed the Kentico CMS project as a web application, you need to rename the *CodeFile* property on the first line to *CodeBehind* for the code example to be functional.

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="MyStep.ascx.cs"
Inherits="MyStep" %>
<asp:Label ID="lblError" runat="server" EnableViewState="false" Visible="false"></
asp:Label>
<table>
<tr>
<td><asp:Label ID="lblTotalPrice" runat="server" /></td>
<td><asp:Label ID="lblTotalPriceValue" runat="server" /></td>
```

```
</tr>
<tr>
    <td><asp:Label ID="lblComment" runat="server" /></td>
    <td><asp:TextBox ID="txtComment" runat="server" TextMode="MultiLine" Rows
="3" /></td>
</tr>
</table>
```

**[MyStep.ascx.cs]****[C#]**

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using CMS.GlobalHelper;
using CMS.Ecommerce;
using CMS.EcommerceProvider;

public partial class MyStep : ShoppingCartStep
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Initialize labels
        lblTotalPrice.Text = ResHelper.GetString("MyStep.TotalPrice");
        lblComment.Text = ResHelper.GetString("MyStep.Comment");

        // Display rounded and formatted total price
        lblTotalPriceValue.Text =
CurrencyInfoProvider.GetFormattedPrice(ShoppingCartInfoObj.RoundedTotalPrice,
ShoppingCartInfoObj.CurrencyInfoObj);

        if (!this.ShoppingCartControl.IsCurrentStepPostBack)
        {
            // Load customer comment
            txtComment.Text = this.ShoppingCartInfoObj.ShoppingCartCustomData[
"CustomerComment"];
        }
    }

    /// <summary>
    /// Validates current step data.
    /// </summary>
    /// <returns></returns>
```

```
public override bool IsValid()
{
    // Check customer comment for emptiness
    if (txtComment.Text.Trim() == "")
    {
        // Display error message
        lblError.Text = ResHelper.GetString("MyStep.Error.CommentMissing");
        lblError.Visible = true;

        // Data are not correct - customer comment missing
        return false;
    }
    else
    {
        // Data are correct
        return true;
    }
}

/// <summary>
/// Process current step data
/// </summary>
/// <returns></returns>
public override bool ProcessStep()
{
    // Update shopping cart with customer comment
    this.ShoppingCartInfoObj.ShoppingCartCustomData["CustomerComment"] =
    txtComment.Text.Trim();

    try
    {
        // Update shopping cart in database
        ShoppingCartInfoProvider.SetShoppingCartInfo(this.
ShoppingCartInfoObj);

        // Current step data were processed and saved succesfully
        return true;
    }
    catch
    {
        // Display error message
        lblError.Text = ResHelper.GetString("MyStep.Error.ShoppingCartUpdate"
);
        lblError.Visible = true;

        // Current step data update failed
        return false;
    }
}

/// <summary>
/// Action after the "Next button" is clicked
/// </summary>
public override void ButtonNextClickAction()
{

```

```

        // If customer is registered and is a member of role "VipCustomers"
        if ((this.ShoppingCartInfoObj.UserInfoObj != null)
            && (this.ShoppingCartInfoObj.UserInfoObj.IsInRole("VipCustomers",
this.ShoppingCartInfoObj.SiteName)))
        {
            if (IsValid() && ProcessStep())
            {
                try
                {
                    // Load extra step for VIP customers which is not included
                    // in standard checkout process definition
                    ShoppingCartStep ctrl =
                    (ShoppingCartStep)this.Page.LoadControl("~/CMSecommerce/ShoppingCart/MyVipStep.
                    ascx");

                    this.ShoppingCartControl.LoadStep(ctrl);

                    // Note: Current step index is not increased

                }
                catch
                {
                    // Error while loading extra step control -> Do standard
                    action
                    base.ButtonNextClickAction();
                }
            }
            else
            {
                // Do standard action (validate step data, process step data, load
                next step)
                base.ButtonNextClickAction();
            }
        }
    }
}

```

### My VIP step

This is an external checkout process step which is not included in standard checkout process definition. It is loaded only when the current customer is a member of role "VipCustomers", however, you will need to use your own condition for loading your external steps. There is no data validation (Vip customer comment can be empty). Neither "Back button action" nor "Next button action" is overridden so it means the standard methods are called after the **Back** button and **Next** button are clicked.

### **[MyVipStep.ascx]**

Please note: If you installed the Kentico CMS project as a web application, you need to rename the *CodeFile* property on the first line to *Codebehind* for the code example to be functional.

```

<%@ Control Language="C#" AutoEventWireup="true" CodeFile="MyVipStep.ascx.cs"
Inherits="MyVipStep" %>
<asp:Label ID="lblError" runat="server" EnableViewState="false" Visible="false"></

```



```
asp:Label>
<asp:Label ID="lblComment" runat="server" />
<asp:TextBox ID="txtComment" runat="server" TextMode="MultiLine" Rows="3" />
```

**[MyVipStep.ascx.cs]****[C#]**

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using CMS.GlobalHelper;
using CMS.Ecommerce;
using CMS.EcommerceProvider;

public partial class CMSEcommerce_TestVipCustomers : ShoppingCartStep
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Initialize label
        lblComment.Text = ResHelper.GetString("MyVipStep.Comment");

        if (!this.ShoppingCartControl.IsCurrentStepPostBack)
        {
            // Load VIP customer extra comment
            txtComment.Text =
Convert.ToString(this.ShoppingCartInfoObj.ShoppingCartCustomData["VipComment"]);
        }

        /// <summary>
        /// Process current step data
        /// </summary>
        /// <returns></returns>
        public override bool ProcessStep()
        {
            // Update shopping cart with VIP customer extra comment
            this.ShoppingCartInfoObj.ShoppingCartCustomData["VipComment"] =
txtComment.Text.Trim();

            try
            {
                // Update shopping cart in database
                ShoppingCartInfoProvider.SetShoppingCartInfo(this.
ShoppingCartInfoObj);
            }
        }
    }
}
```

```
        // Current step data were processed and saved succesfully
        return true;
    }
    catch
    {
        // Display error message
        lblError.Text = ResHelper.GetString("MyStep.Error.ShoppingCartUpdate"
);
        lblError.Visible = true;

        // Current step data update failed
        return false;
    }
}
```



#### Important notes on step order

Notice that data of all steps (including Kentico CMS standard shopping cart steps) are always processed by themselves, not by the parent shopping cart control. It means if you reorder standard checkout process steps you can experience a strange behavior because of missing information or omitting some important action.

For example if you move step "Order preview" in the standard checkout process definition for the live site before steps "Select billing and shipping address" and "Select payment and shipping methods" some order preview data will be missing (billing and shipping addresses, payment method and shipping option) because they were not entered yet. But even worse is that entered data of the following two steps ("Select billing and shipping address", "Select payment and shipping methods") will not be saved.

This happens because the order is saved after the "Next button" of the step "Order preview" is clicked and these two steps include completely different actions after their "Next buttons" are clicked.

## 7.4 Payment gateways

Once the customer enters appropriate information, you can redirect them to an on-line payment gateway. You can choose from the following options:

- **No on-line payment** - the customer is only displayed with some "Thank you" page.
- **Customer credit** - the customer can pay using their credit. The credit must be entered in the customer details by the store owner. This option is useful for customer loyalty programs. For further information please refer to the [Customer credit](#) chapter.
- **Authorize.NET** - the customer pays using their card. See [Authorize.NET configuration](#) for more details.
- **PayPal** - the customer pays using their card or PayPal account. See [PayPal configuration](#) for more details.

- **Custom payment gateway** - you can integrate some other payment gateway using your code. See [Developing custom payment gateways](#) for more details.

## 7.5 Authorize.NET configuration

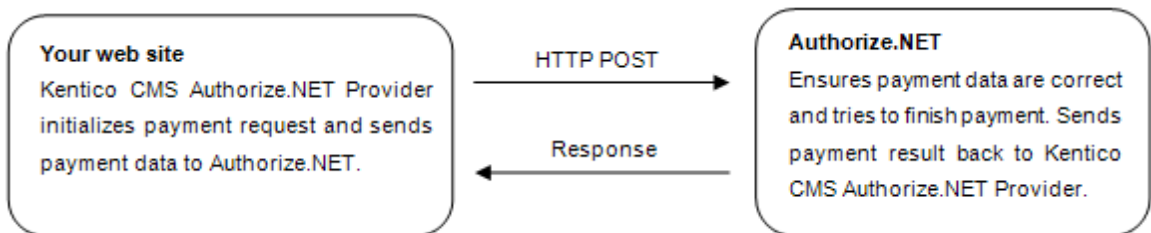
Authorize.NET is one of the most popular gateway providers. It uses plain HTTPS POST operations against its gateway server.

### What do I need?

1. **Kentico CMS 3.0 or higher** with built in Authorize.NET support.
2. **Internet Merchant Account** - A type of bank account that allows a business to accept internet credit card payments (the card is not physically presented to the merchant).
3. **Payment Gateway Account** - A secure internet bridge between your website and the credit card processing networks.

See **Authorize.NET Getting Started Guide** at <http://www.authorize.net/files/ecommerceguide.pdf> for more details.

### How does it work?



### Authorize.NET settings in Kentico CMS 3.1 or higher

Before you can offer customers to use Authorize.NET payment gateway you will need to do some necessary settings:

1. Go to the section *CMSSiteManager -> Settings -> Payment Gateways – Authorize.NET*
2. Enter **API Login** (API Login ID for the payment gateway account) and **Transaction key** (Transaction key obtained from the Merchant Interface) to identify your payments.
3. Turn off **Test Mode** of the Authorize.NET payment gateway. Gateway behavior depends on both test mode settings: in CMSSiteManager and in Authorize.NET Merchant Interface. See table below for more details about test mode settings:

Kentico CMS	Merchant Interface	Transaction processed as
ON	ON	test transaction
OFF	ON	test transaction
ON	OFF	test transaction
OFF	OFF	live transaction

4. Ensure that Authorize.NET payment method is registered and enabled.
5. Check Authorize.NET payment method payment gateway settings:
  - Payment gateway url: *https://secure.authorize.net/gateway/transact.dll*
  - Payment gateway assembly name: *CMS.EcommerceProvider*
  - Payment gateway class name: *CMS.EcommerceProvider.CMSAuthorizeNetProvider*

**Please note:** If you want your transaction to be processed as a test transaction turn on the Test Mode and use *https://test.authorize.net/gateway/transact.dll* as your payment gateway url.

While using the Authorize.NET payment method, a customer is required to fill their credit card information in the last step of the checkout process to finish payment. After the payment is finished the order payment result is updated.

The screenshot displays the 'Step 6 of 6 - Payment' stage of an e-commerce checkout process. At the top, there's a navigation bar with 'Home', 'News', 'Products', 'How to buy', 'Company', and 'Silverlight'. A shopping cart icon shows a total price of \$537.30. The main content area features a progress bar with icons for cart, user, address, payment, and shipping. Below this, a 'Payment summary' section lists: Order ID: 4, Payment method: Credit card, and Total price: \$537.30. The 'Your credit card details' section includes input fields for 'Credit card number' (1222365948790015), 'Credit card CCV' (528), and 'Credit card expiration' (01/2019). At the bottom, there are 'Skip payment' and 'Finish payment' buttons. The footer contains the 'POWERED BY Kentico' logo.

## 7.6 PayPal configuration

### What do I need?

1. **Kentico CMS 3.0 or higher** with built in PayPal support.
2. **PayPal account**, see <https://www.paypal.com/us/cgi-bin/webscr?cmd=registration-run> for more details.

### PayPal settings in Kentico CMS 3.1 or higher

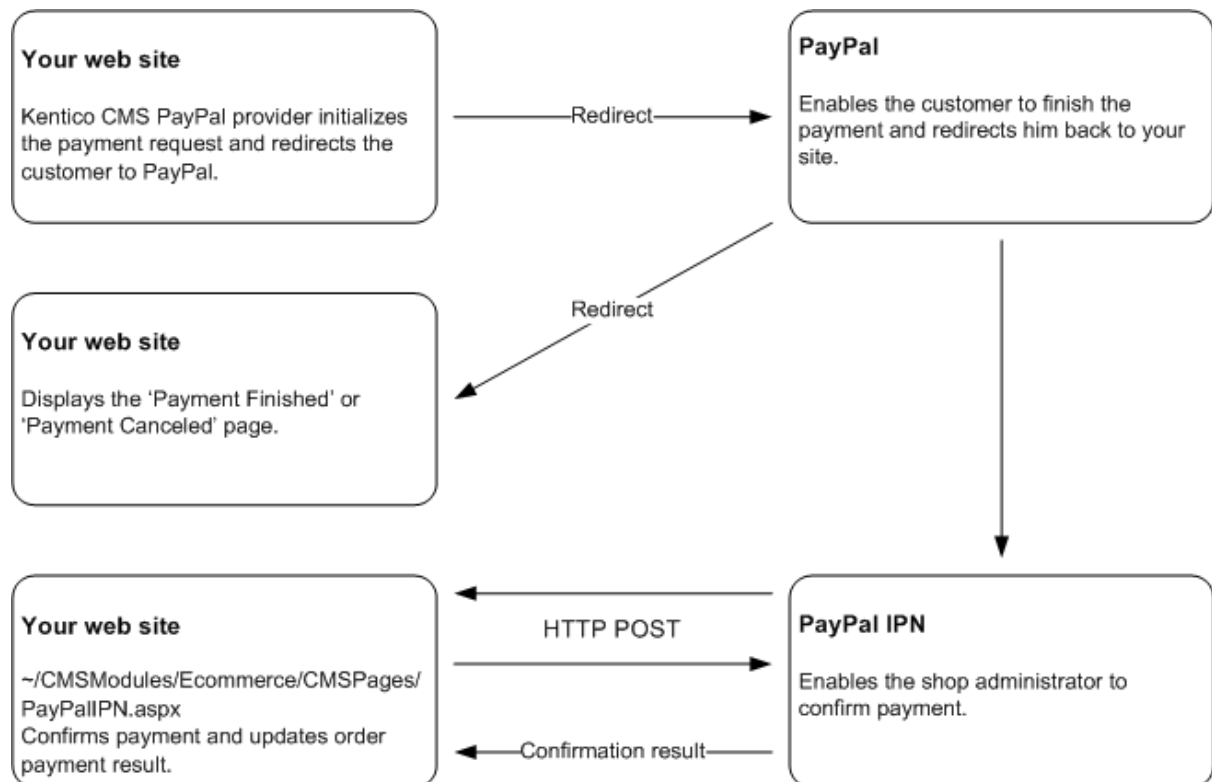
Before you can offer customers to use PayPal payment gateway you will need to do some necessary settings:

1. Go to the section *CMSSiteManager -> Settings -> Payment Gateways – PayPal*
2. Enter **Business** (E-mail address for merchant's PayPal account) to identify your payments. Other settings are optional:
  - **Notify Url:** The URL to which PayPal posts information about the transaction, see *PayPal IPN* for more details. If set overrides settings in PayPal merchant's interface.
  - **Cancel Return Url:** The URL to which the buyer's browser is redirected if payment is cancelled; for example, a URL on your website that displays a "Payment Canceled" page. Browser is redirected to a PayPal web page by default.
  - **Return Url:** The URL to which the buyer's browser is redirected after completing the payment; for example, a URL on your site that displays a "Thank you for your payment" page. Buyer is redirected to a PayPal web page by default.
3. Ensure that PayPal payment method is registered and enabled.
4. Check PayPal payment method payment gateway settings:
  - Payment gateway url: *https://www.paypal.com/cgi-bin/webscr*
  - Payment gateway assembly name: *CMS.EcommerceProvider*
  - Payment gateway class name: *CMS.EcommerceProvider.CMSPayPalProvider*

**Please note:**

- If you want your transaction to be processed as test transaction use *https://www.sandbox.paypal.com/cgi-bin/webscr* as your payment gateway url and sign up for PayPal SandBox testing environment, see [https://www.paypal.com/IntegrationCenter/ic\\_sandbox.html](https://www.paypal.com/IntegrationCenter/ic_sandbox.html) for more details.
- The price is always rounded to two decimal places, otherwise the PayPal payment gateway wouldn't allow the payment.

**How does it work?**



## IPN - Instant Payment Notification

It's easy to spoof the return URL you sent to PayPal since it's visible on the query string. Therefore a user could simply type the Confirmation Url in directly and you should not confirm the order at this point. You can manually check for orders on the PayPal site or wait for PayPal's confirmation e-mails etc. all of which let you know for sure that the order was processed in a 'manual' way.

To automate this process, PayPal can optionally ping you back at another URL with order completion information. It uses a mechanism called Instant Payment Notification (IPN) which is essentially a web based callback mechanism that calls a pre-configured url on your site. IPN must be enabled on the PayPal site and when enabled IPN sends a confirmation to you at this url after the order was processed. PayPal then expects a return from you within a certain timeframe (a few minutes) and return a response to you to confirm you that the customer has paid. To do this you have to POST the data back to PayPal by echoing back all the form data that PayPal sends to you. IPN is optional, but it's a requirement if you need to confirm your orders immediately with your customers.

While using the PayPal payment method, a customer is not required to fill any additional information in the last step of the checkout process. After the "Finish payment" button is clicked the user is redirected to the PayPal payment gateway to finish payment:

The screenshot displays an e-commerce checkout page titled "E-commerce starter site" with a navigation menu including Home, News, Products, How to buy, Company, and Silverlight. The main content area is titled "Step 6 of 6 - Payment" and shows a progress bar with icons for cart, user, address, shipping, payment, and confirmation. Below the progress bar, it states "Your order was saved." and provides a "Payment summary" with the following details:

- Order ID: 4
- Payment method: PayPal
- Total price: \$537.30

The "PayPal" section includes the instruction: "Click 'Finish payment' button to redirect to PayPal payment gateway." At the bottom of the payment section, there are two buttons: "Skip payment" and "Finish payment".

On the left side of the page, there is a sidebar with a search bar, a "Products" list (Cameras, Cell phones, MP3 Players, Notebooks, PDAs, PCs), a "Poll" titled "How did you find this site?" with radio button options (Referred by a friend, Print advertising, Link from another web site, Search engine), and "Quick links" (How to buy, Company, News, Home). The top right corner shows a shopping cart icon and links for "My account" and "My wishlist", with a "Total price: \$537.30" displayed.

The footer of the page features the "POWERED BY Kentico" logo.

## 7.7 Developing custom payment gateways

### Creating a custom payment gateway

Here's a general overview of the process of integrating a custom payment gateway:

1. Create a payment gateway form with your custom controls to enable customers to enter their payment data such as credit card number, see section [Creating custom payment gateway form](#).
2. Create your custom payment gateway class and override required methods for payment processing, see section [Creating custom payment gateway class](#).
3. Go to the **CMSDesk -> Tools -> E-commerce -> Configuration -> Payment methods**.
4. Create a new payment method and register your custom payment gateway as described in the [Payment methods](#) chapter.

### Creating a custom payment gateway form

1. Create a new web user control (\*.ascx) and place it into your site folder which is located in the root of your web project.

Since the control is located in the site folder it is included in export package of your site.

2. Set control class to inherit from abstract class *CMS.EcommerceProvider*.  
*CMSPaymentGatewayForm*
3. There are several methods you need to override to reach your required functionality:
  - **LoadData()** – Initializes form controls with customer payment data.
  - **ValidateData()** – Validates customer payment data.
  - **ProcessData()** – Process customer payment data and saves them to the *ShoppingCartInfo* object.
4. There are several properties to get or set required information:
  - **ShoppingCartControl** – Parent shopping cart control the current shopping cart step belongs to.
  - **ShoppingCartInfoObj** – Shopping cart object which stores all data during the checkout process.

**Please note:** Payment data, such as credit card numbers, credit card codes and others are not saved into the database because of security reasons.

### Creating custom payment gateway class

1. Create a new library (assembly) as a part of your solution and a new class inside this library.
2. Add a reference to *System.Web* assembly into the project with the payment gateway (Right-click the References folder, choose Add reference, select .NET -> System.Web).
3. Set your class to inherit from abstract class *CMS.EcommerceProvider*.  
*CMSPaymentGatewayProvider*.
4. There are several methods you can override to reach your required functionality:
  - **AddCustomData()** – Adds payment gateway custom controls to the current shopping cart step. By default *CMSPaymentGatewayForm* control is added to the payment data container and its data are loaded.
  - **RemoveCustomData()** – Removes payment gateway custom controls from the current shopping cart step. By default all controls from payment data container are removed.
  - **ValidateCustomData()** - Validates payment gateway custom data of the current shopping cart step. By default *CMSPaymentGatewayForm* control data are validated.
  - **ProcessCustomData()** - Process payment gateway custom data of the current shopping cart step. By default *CMSPaymentGatewayForm* data are processed.
  - **ProcessPayment()** - Process payment - you need to override this method to process payment by your payment processor.
  - **GetPaymentDataForm()** – Loads payment gateway form with custom controls – you need to override this method to get your own payment gateway form.

5. There are several properties to get or set required information:
  - **ShoppingCartControl** – Parent shopping cart control the current shopping cart step belongs to.
  - **ShoppingCartInfoObj** – Shopping cart object which stores all data during the checkout process .If *OrderId* is set it is created from existing order, otherwise it is returned from current shopping cart control.
  - **OrderId** – Order ID. Set this value when you want to process payment for the existing order outside the checkout process.
  - **PaymentDataContainer** – Payment gateway custom controls container of the current shopping cart step control.
  - **PaymentResult** – Payment result (see caption *Payment Result*)
  - **IsPaymentCompleted** – Indicates whether order payment was already completed. It is determined by order payment result.
  - **InfoMessage** - Payment result message displayed to user when payment succeeds.
  - **ErrorMessage** - Payment result message displayed to user when payment fails.
6. Compile the library.
7. Ensure the library file (\*.dll) is included in *<your web project folder>/Bin* directory.

## Example

The following example shows a custom payment processor implementation. It allows customers to pay for their orders using some external payment gateway similar to PayPal, let's call it Custom Gateway. Customer is asked for their credit card number in the last step of checkout process. Credit card number is validated for emptiness and processed after the "Finish payment" button is clicked. If it succeeds payment process is performed – required payment data are attached to the payment url and customer is redirected to Custom Gateway. If payment process fails (payment gateway url is not defined) order payment result is updated and appropriate error message is displayed. Notice that order is saved before the customer is asked to pay for it, it happens immediately after the "Order now" button is clicked.

### Please note:

- It is not secure to send credit card information as a part of payment gateway url. Customers usually asked for their credit card number after they are redirected to the payment gateway itself; otherwise another way of sending credit card information should be used instead.
- For more details about how payment gateway could inform merchant about the result of the payment which was finished outside their website, see caption PayPal (IPN – Instant Payment Notification).



The screenshot shows the 'New payment method' configuration form in the Kentico CMS. The form is titled 'New payment method' and is located under the 'Configuration' menu. The left sidebar shows a navigation tree with 'Payment methods' selected. The main form area contains the following fields and options:

- Payment method display name:** Custom gateway
- Payment method code name:** CustomGateway
- Enabled:**
- Payment gateway settings:**
  - Payment gateway URL:** http://www.somePaymentGateway.com
  - Payment gateway assembly name:** CMS.CustomProvider
  - Payment gateway class name:** CMS.CustomProvider.CustomGateway
  - Order status when payment succeeds:** Payment received
  - Order status when payment fails:** Payment failed

At the bottom of the form is a green 'OK' button.

Example of custom payment gateway definition.

## Custom payment gateway form

It is a simple form with one input field to enter customer credit card number, see image bellow.

The screenshot shows the 'Step 6 of 6 - Payment' form. The form is titled 'Step 6 of 6 - Payment' and is located under the 'Payment' menu. The form contains a progress bar with icons representing the checkout process: shopping cart, user profile, credit card, payment gateway, calculator, and receipt. Below the progress bar, the text 'Your order was saved.' is displayed. The form is divided into two main sections:

- Payment summary:**
  - Order ID: 5
  - Payment method: Custom gateway
  - Total price: \$537.30
- Your credit card details:**
  - Credit card number:

At the bottom of the form are two buttons: 'Skip payment' and 'Finish payment'.

## CustomGatewayForm.ascx

Please note: If you installed the Kentico CMS project as a web application, you need to rename the *CodeFile* property on the first line to *Codebehind* for the code example to be functional.

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="CustomGatewayForm.ascx.
cs" Inherits="CMSEcommerce_Example_CustomGatewayForm" %>
<asp:Label ID="lblTitle" runat="server" EnableViewState="false" CssClass
="BlockTitle" />
```

```
<asp:Label ID="lblError" runat="server" EnableViewState="false" CssClass
="ErrorLabel" Visible="false" />
<asp:Label ID="lblCardNumber" EnableViewState="false" runat="server" />
<asp:TextBox ID="txtCardNumber" runat="server" />
```

## CustomGatewayForm.ascx.cs

[C#]

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using CMS.EcommerceProvider;
using CMS.GlobalHelper;

public partial class CMSEcommerce_Example_CustomGatewayForm :
    CMSPaymentGatewayForm
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Initialize label
        lblTitle.Text = "Your credit card details";
        lblCardNumber.Text = "Credit card number:";
    }

    /// <summary>
    /// Initializes form controls with customer payment data.
    /// </summary>
    public override void LoadData()
    {
        // Display customer credit card number
        txtCardNumber.Text = ValidationHelper.GetString(this.ShoppingCartInfoObj.
            PaymentGatewayCustomData["CustomGatewayCardNumber"], "");
    }

    /// <summary>
    /// Validates customer payment data.
    /// </summary>
    /// <returns></returns>
    public override string ValidateData()
    {
        if (txtCardNumber.Text.Trim() == "")
        {
            lblError.Visible = true;
        }
    }
}
```

```
        lblError.Text = "Please enter your credit card number";
        return lblError.Text;
    }
    return "";
}

/// <summary>
/// Process customer payment data.
/// </summary>
/// <returns></returns>
public override string ProcessData()
{
    // Save credit card number
    this.ShoppingCartInfoObj.PaymentGatewayCustomData[
"CustomGatewayCardNumber"] = txtCardNumber.Text.Trim();
    return "";
}
}
```

## Custom payment gateway class

The following example uses assembly name *CMS.CustomProvider* and class *CMS.CustomProvider.CustomGateway*, however, you will need to use your own names.

### CustomGateway.cs

[C#]

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Web;
using System.Collections;

using CMS.EcommerceProvider;
using CMS.GlobalHelper;
using CMS.UIControls;
using CMS.ExtendedControls;

namespace CMS.CustomProvider
{
    /// <summary>
    /// Class representing Custom Gateway processor.
    /// </summary>
    public class CustomGateway : CMSPaymentGatewayProvider
    {
        /// <summary>
        /// Returns payment gateway form with custom controls.
        /// </summary>
        /// <returns></returns>
        public override CMSPaymentGatewayForm GetPaymentDataForm()
        {
            try
            {
```

```
        return (CMSPaymentGatewayForm)this.ShoppingCartControl.LoadControl
        ("~/CMSEcommerce/Example/CustomGatewayForm.ascx");
    }
    catch
    {
        return null;
    }
}

/// <summary>
/// Process payment.
/// </summary>
public override void ProcessPayment()
{
    // Get payment gateway url
    string url = this.GetPaymentGatewayUrl();

    if (url != "")
    {
        // Initialize payment parameters
        Hashtable parameters = InitializePaymentParameters();

        // Add required payment data to the url
        url = GetFullPaymentGatewayUrl(url, parameters);

        // Redirect to payment gateway to finish payment
        this.ShoppingCartControl.Page.Response.Redirect(url);
    }
    else
    {
        // Show error message - payment gateway url not found
        this.ErrorMessage = "Unable to finish payment: Payment gateway url
not found.";

        // Update payment result
        this.PaymentResult.PaymentDescription = this.ErrorMessage;
        this.PaymentResult.PaymentIsCompleted = false;

        // Update order payment result in database
        this.UpdateOrderPaymentResult();
    }
}

/// <summary>
/// Returns table with initialized payment parameters.
/// </summary>
/// <returns></returns>
private Hashtable InitializePaymentParameters()
{
    Hashtable parameters = new Hashtable();

    parameters["orderid"] = this.ShoppingCartInfoObj.OrderId;
    parameters["price"] = this.ShoppingCartInfoObj.TotalPrice;
    parameters["currency"] = this.ShoppingCartInfoObj.CurrencyInfoObj.
CurrencyCode;
}
```

```
        parameters["cardnumber"] = Convert.ToString(this.ShoppingCartInfoObj.
PaymentGatewayCustomData["CustomGatewayCardNumber"]);

        return parameters;
    }

    /// <summary>
    /// Returns payment gateway url with payment data in query string.
    /// </summary>
    /// <param name="url">Payment gateway url.</param>
    /// <param name="parameters">Initialized payment paremeters.</param>
    /// <returns></returns>
    private string GetFullPaymentGatewayUrl(string url, Hashtable parameters)
    {
        foreach (DictionaryEntry parameter in parameters)
        {
            // Add payment data to the url
            url = UrlHelper.AddParameterToUrl(url, Convert.ToString(parameter.
Key), HttpUtility.UrlEncode(Convert.ToString(parameter.Value)));
        }
        return url;
    }
}
```

## 7.8 Payment results

### Payment Results

Payment result is stored in xml format which is represented by the object *CMS.Ecommerce.PaymentResultInfo*. Each payment result xml node is equal to the single payment result item which is represented by the object *CMS.Ecommerce.PaymentResultItemInfo*.

Base payment result items are:

- **Payment date** – Date and time when the payment result was last updated.
- **Payment method** – Payment method which was used for payment.
- **Payment is completed** – Yes/No. Indicates whether payment was already completed.
- **Payment status** – Completed/Failed/(your custom status). Status of the payment.
- **Payment transaction ID** – Completed payment unique identifier generated by the payment gateway.
- **Payment description** – Describes payment result in more details.

Payment result item properties are:

- **Name**: Unique identifier of the item.
- **Header**: Friendly name of the item visible to user (simple text or localizable string).
- **Text**: Outer representation of the item value visible to user (simple text or localizable string).
- **Value**: Inner representation of the item value used by developers.

Following example shows xml definition of the order payment result extended by the item “*authorizationcode*” used by the Authorize.NET:

```

<result>
  <item name="date" header="{ $PaymentGateway.Result.Date$ }" value="1/27/2008
5:01:41 PM" />
  <item name="method" header="{ $PaymentGateway.Result.PaymentMethod$ }" text="
Credit card" value="230" />
  <item name="completed" header="{ $PaymentGateway.Result.IsCompleted$ }" value
="1" text="{ $PaymentGateway.Result.PaymentCompleted$ }" />
  <item name="status" header="{ $PaymentGateway.Result.Status$ }" text="
{ $PaymentGateway.Result.Status.Completed$ }" value="completed" />
  <item name="transactionid" header="{ $PaymentGateway.Result.TransactionID$ }"
value="0" />
  <item name="description" header="{ $PaymentGateway.Result.Description$ }" />
  <item name="authorizationcode" header="{ $AuthorizeNet.AuthorizationCode$ }"
value="000000" />
</result>

```

The following example shows order payment result which is visible to the user in CMSDesk:

*Date: 1/27/2008 5:01:41 PM*

*Method: Credit card*

*Is completed: YES*

*Status: Completed*

*Transaction ID: 0*

*Authorization code: 000000*

**Please note:**

- o The order payment result remains empty until it is updated by the payment gateway processor.
- o You don't need to specify both item value and item text if they are identical because payment result rendering method can manage this and renders payment result as follows: try to render item text, if not found, try to render item value.

## How to customize payment result

You can use *PaymentResultInfo* properties to get or set specified item text or value:

- PaymentDate
- PaymentMethodID
- PaymentMethodName
- PaymentIsCompleted
- PaymentStatusName
- PaymentStatusCode
- PaymentTransactionID

You will need to use public methods ***GetPaymentResultItemInfo(string itemName)*** and ***SetPaymentResultItemInfo(PaymentResultItemInfo itemObj)*** to get and set your custom payment result items.

The following example shows how to get and set custom payment result item while payment processing by your custom payment gateway provider:

**[C#]**

```
using CMS.Ecommerce;

// Set authorization code
PaymentResultItemInfo item = new PaymentResultItemInfo();
item.Header = "{$AuthorizeNet.AuthorizationCode$}";
item.Name = "authorizationcode";
item.Value = "00000";
this.PaymentResult.SetPaymentResultItemInfo(item);
```

[C#]

```
using CMS.Ecommerce;

// Get authorization code
PaymentResultItemInfo item = this.PaymentResult.GetPaymentResultItemInfo(
"authorizationcode");
```

## 7.9 Customer credit

Customers may receive a credit on their account. They can then purchase products using this credit. This feature is useful for customer loyalty competitions where customers receive bonus points/credit for their previous purchases and can order some products once they achieve an appropriate amount of credit.

Please note: the purchase can be made only using one payment method, so it's not possible to combine credit payment with another form of payment and the whole order must be paid using the credit.

The customer receives credit when the store owner adds some credit event to the customer history. This can be done in **Customer properties** dialog, on the **Credit** tab.

Before you can offer registered customers to use the **Customer credit** you will need to do some necessary settings:

1. Ensure that the **Customer credit** payment method is registered and enabled.
2. Check the **Customer credit** payment method settings:
  - Payment gateway url: leave blank
  - Payment gateway assembly name: *CMS.EcommerceProvider*
  - Payment gateway class name: *CMS.EcommerceProvider.CMSCreditPaymentProvider*

While using the customer credit, a customer is not required to fill any additional information in the last step of the checkout process. After the "Finish payment" button is clicked, their credit is decreased by specified amount which is equal to the order total price in default currency and the order payment result is updated.

Step 6 of 6 - Payment



Your order was saved.

**Payment summary**

Order ID: 6  
Payment method: Customer credit  
Total price: \$155.25

**Use credit for payment**

Available credit: \$1000.00

[Skip payment](#) [Finish payment](#)



**Part**

---



**VIII**

**Developing custom providers**

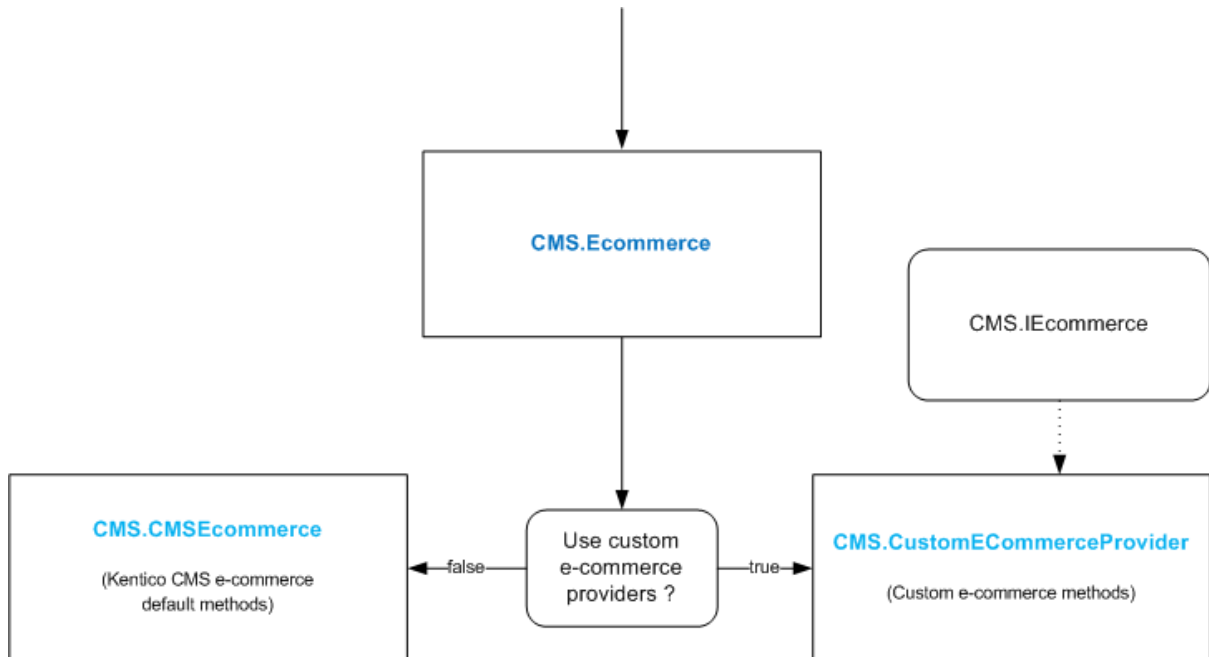
---

## 8 Developing custom providers

### 8.1 Overview

The E-commerce module allows you to override the default behavior and calculations by using custom providers that ensure various operations.

#### How does it work?



For more information please refer to the [Using custom providers](#) chapter.

The following example shows the way of customizing the calculation of the shipping cost by modifying the **CMS.CustomECommerceProvider.CustomShippingOptionInfoProvider.CalculateShipping** method. Similarly, you can modify any method of any class from the **CMS.**

**CustomECommerceProvider** library. All the methods that can be customized call by default the equivalent methods from the **CMS.CMSEcommerce** namespace. This ensures the standard behaviour in the case the given method is not modified and the using custom e-commerce provider is enabled at the same time.

1. The default code of the **CMS.CustomECommerceProvider.CustomShippingOptionInfoProvider.CalculateShipping** method.

[C#]

```

public double CalculateShipping(object cartObj, string siteName)
{
    return CMS.CMSEcommerce.ShippingOptionInfoProvider.CalculateShipping
    ((CMS.Ecommerce.ShoppingCartInfo)cartObj, siteName);
}
  
```

## 2. The customized code of the `CMS.CustomECommerceProvider.CustomShippingOptionInfoProvider.CalculateShipping` method

[C#]

```
using CMS.Ecommerce;
using CMS.SettingsProvider;
using CMS.SiteProvider;

public double CalculateShipping(object cartObj, string siteName)
{
    CMS.Ecommerce.ShoppingCartInfo cart = cartObj as CMS.Ecommerce.
    ShoppingCartInfo;
    if (cart != null)
    {
        if ((cart.UserInfoObj != null) && (cart.UserInfoObj.IsInRole(
"VIPCutomers", siteName)))
        {
            // Free shipping for VIP customers
            return 0;
        }
        else
        {
            // Get shipping address details
            CMS.Ecommerce.AddressInfo address = CMS.Ecommerce.
            AddressInfoProvider.GetAddressInfo(cart.ShoppingCartShippingAddressID);
            if (address != null)
            {
                // Get shipping address country
                CountryInfo country = CountryInfoProvider.GetCountryInfo
                (address.AddressCountryID);
                if ((country != null) && (country.CountryName.ToLower() !
= "usa"))
                {
                    // Add an extra charge to standard shipping price for
                    non-usa customers
                    double extraCharge = SettingsKeyProvider.
                    GetDoubleValue("ShippingExtraCharge");
                    return CMS.CMSEcommerce.ShippingOptionInfoProvider.
                    CalculateShipping(cart, siteName) + extraCharge;
                }
            }
        }

        // Get standard shipping price in other cases
        return CMS.CMSEcommerce.ShippingOptionInfoProvider.CalculateShipping
        (cart, siteName);
    }
}
```

## 8.2 Using custom providers

By default, Kentico CMS is delivered with sample custom providers with samples of method implementation. To include custom providers within your web project, please follow these steps:

1. Open your web project in Visual Studio.
2. Copy the C:\Program Files\Kentico CMS\<version>\CodeSamples\CustomECommerceProvider directory to your working directory and add the project **CustomECommerceProvider.csproj** to your solution.
3. Add the following references from your web project /bin directory:
  - CMS.CMSHelper.dll
  - CMS.DataEngine.dll
  - CMS.Ecommerce.dll
  - CMS.GlobalHelper.dll
  - CMS.IECommerce.dll
  - CMS.SettingsProvider.dll
  - CMS.SiteProvider.dll
  - CMS.Staging.dll
4. Delete the **CMS.CustomECommerceProvider.dll** file from the /bin directory of the CMS project.
5. Add the project reference to the CustomECommerceProvider project to your web project.
6. Compile the solution.

Now your project is ready to use the custom providers. To enable custom providers, add following settings key to you web.config file:

```
<appSettings>
  <add key="CMSUseCustomEcommerceProviders" value="true" />
</appSettings>
```

Since now, your web application will call your custom providers to handle the E-commerce requests. If you do not want to override the behaviour of certain methods, you should just call the CMS provider method to perform the original action. Original CMS providers are located in the **CMS.CMSEcommerce** namespace. See the next chapters to get the information about the providers and actions that they should perform in your methods to ensure the system consistency.

Every custom provider must implement a corresponding interface from the **CMS.IECommerce** namespace. For example: the **CustomSKUInfoProvider** implements the **ISKUInfoProvider** interface.

All providers use **System.Object** parameter types, so you need to type the method parameters to appropriate data types.

**Please note:** If you implement only some of the custom providers, you should ensure that your data are consistent with the other E-commerce database data. In case of inconsistency the system might not work well.

#### Additional information

If you need additional information on some class or database table structure, please see the **Database Reference** or **API Reference** documentation.

## 8.3 CustomAddressInfoProvider

This provider works with customer addresses and provides the address data manipulation in general. The methods work with **CMS.Ecommerce.AddressInfo** class which is replaced by object in the method headers. The provider implements the following methods:

- **object GetAddressInfo(int addressId)** – returns the AddressInfo object by address ID.
- **object GetAddressInfo(string addressName)** – returns the AddressInfo object address name.
- **void SetAddressInfo(object address)** – inserts/updates the AddressInfo object to the database. If the AddressID property value is set, the database record is updated, if not, the record is inserted and new record ID must be set in the object.
- **void DeleteAddressInfo(int addressId)** – deletes the address record from the database by given address ID.
- **string GetAddressName(object addressInfoObj)** – generates the address name.
- **DataSet GetAddresses(string where, string orderBy)** - returns the DataSet of all the addresses filtered by WHERE condition and ordered by orderBy expression, see the database reference for the address column definition.
- **DataSet GetAddressesForCustomer(int customerId, bool showBilling, bool showShipping, bool showCompany, bool onlyEnabled)** - returns the DataSet of all the addresses for the specified user according to the input parameters, see the database reference for the currency column definition.
- **bool CheckDependencies(int addressId)** - returns true if some objects depend on the address specified by its ID.

## 8.4 CustomCurrencyInfoProvider

This class provides the methods to work with currencies. Methods work with **CMS.Ecommerce.CurrencyInfo** class.

- **object GetCurrencyInfo(int currencyId)** – returns the CurrencyInfo object by currency ID.
- **object GetCurrencyInfo(string currencyName)** – returns the CurrencyInfo object by code name.
- **void SetCurrencyInfo(object currency)** – inserts/updates the currency record to the database. If CurrencyID is set, it performs an update, else it performs a new currency insertion and initializes new record ID.

- `void DeleteCurrencyInfo(int currencyId)` – deletes the currency record specified by its ID.
- `DataSet GetAllCurrencies()` – returns the DataSet of all the currencies, see the database reference for the currency column definition.
- `DataSet GetCurrencies(string where, string orderBy)` – returns the DataSet of all the currencies filtered by where condition and ordered by orderBy expression, see the database reference for the currency column definition.
- `DataSet GetCurrenciesWithExchangeRate()` – returns the DataSet with currencies, that have a currently valid an exchange rate specified.
- `object GetMainCurrency()` – returns the main (default) currency. The main currency has the `CurrencyIsMain` value set to true, all the others must have it set to false.
- `void RemoveMainCurrency()` – removes the main currency flag from the currency records (it sets all `CurrencyIsMain` values to false)
- `double RoundTo(double price, int currencyId)` – performs the price rounding based on the currency defined by its ID.
- `double RoundTo(double price, object ciObj)` – performs the price rounding based on the given `CurrencyInfo` object.
- `string GetFormattedPrice(double price, object ciObj)` – returns the price string formatted by given `CurrencyInfo` object content (`CurrencyFormatString`).
- `string GetFormattedValue(double price, object ciObj)` – returns the string formatted by given `CurrencyInfo` object (to display the value without price, in grids etc.).
- `double GetLastValidExchangeRate(int currencyId)` – returns the last valid exchange rate for given currency ID.
- `double GetCurrentValidExchangeRate(int currencyId)` – returns the currently valid exchange rate for given currency ID.
- `double GetExchangeRate(int currencyId, DateTime time)` – returns exchange rate for given currency ID at given time.
- `bool CheckDependencies(int currencyId)` – returns true if some objects depend on the currency specified by its ID.
- `void RecalculateValues(double rate, bool updExchangeRates, bool updSKUs, bool updTaxes, bool updDiscountCoupons, bool updVolumeDiscounts, bool updCredit, bool updShippingOptions, bool updDocuments, bool updFreeShipping)` - recalculates specified values with given exchange rate.

## 8.5 CustomCustomerInfoProvider

This class works with customer information. It uses the **CMS.Ecommerce.CustomerInfo** class.

- `object GetCustomerInfo(int customerId)` – returns the CustomerInfo object by customer ID.
- `object GetCustomerInfoByUserId(int userId)` – returns the CustomerInfo object for the registered user with specified user ID. Please note that registered customers are connected to corresponding CMS\_User records. This couple ensures proper authentication and shopping cart privacy. Anonymous customers do not have their corresponding CMS\_User record and are strictly used for one-time orders.
- `object GetCustomerInfo(Guid customerGuid)` – Returns the CustomerInfo object selected by customer GUID.
- `void SetCustomerInfo(object customer)` – updates or inserts the customer database record using given CustomerInfo object data. If CustomerID is set, update is performed, else new record is inserted.
- `void DeleteCustomerInfo(int customerId)` – deletes the customer record specified by its customer ID.
- `bool CheckDependencies(int customerId)` – returns true if some objects depend on the record with given customer ID.
- `DataSet GetCustomers()` – returns the DataSet with all customers. Please see the database reference for columns and their types.
- `DataSet GetCustomers(string where, string orderBy)` – returns the DataSet of all customers selected with given WHERE condition and ORDER BY clause.
- `DataSet GetCustomersList()` – returns the DataSet with customers joined with their Country and State data, see View\_COM\_Customer\_Joined view for the required result columns.
- `DataSet GetCustomersList(string where, string orderBy)` – returns the customer list as a DataSet selected by given WHERE condition and ORDER BY specification.

## 8.6 CustomDepartmentInfoProvider

Department provider works with the department records and definition that can be used to separate the product groups and branches. It uses the **CMS.Ecommerce.DepartmentInfo** class.

- `object GetDepartmentInfo(int departmentId)` – returns the DepartmentInfo object by given department ID.

- `object GetDepartmentInfo(string departmentName)` – returns the `DepartmentInfo` object selected by given department code name.
- `void SetDepartmentInfo(object department)` – updates the department record with the data from given `DepartmentInfo` object. If `DepartmentID` value is set, it updates the existing record, else it inserts a new record.
- `void DeleteDepartmentInfo(int departmentId)` – deletes department record specified by its department ID.
- `void AddUserToDepartment(int departmentId, int userId)` – adds the user to the department. User-Department is N:N relationship. This assignment is defined for the administration purposes and access control.
- `void RemoveUserFromDepartment(int departmentId, int userId)` – removes user from the department.
- `bool IsUserInDepartment(int departmentId, int userId)` – returns true if the user is member of given department.
- `DataSet GetUserDepartments(int userId)` – returns the `DataSet` of the user departments, see the database reference for required column definition.
- `DataSet GetDepartments()` – returns `DataSet` of all departments defined in the E-commerce system.
- `DataSet GetDepartments(string where, string orderBy)` – returns `DataSet` of the departments selected with given condition and order by specification.
- `bool CheckDependencies(int departmentId)` – returns true if some objects depend on given department to prevent the deletion.

## 8.7 CustomDiscountCouponInfoProvider

This class works with discount coupon information. It uses the `CMS.Ecommerce.DiscountCouponInfo` class.

- `object GetDiscountCouponInfo(int discountCouponId)` – returns `DiscountCouponInfo` object selected by given discount coupon ID.
- `object GetDiscountCouponInfo(string discountCouponCode)` – returns the `DiscountCouponInfo` object selected by given coupon code.
- `void SetDiscountCouponInfo(object discountCoupon)` – updates the discount coupon record with the data from given `DiscountCouponInfo` object. If `DiscountCouponID` is set, it updates the record, else it inserts a new record and initializes its ID.



- `void DeleteDiscountCouponInfo(int discountCouponId)` – deletes specified coupon record by given ID.
- `bool CheckDependencies(int discountCouponId)` – returns true if there are some objects dependent on the given coupon.
- `DataSet GetCouponProducts(int couponId)` – returns the DataSet products assigned to the discount coupon. There is an N:N relationship between products (SKUs) and coupons.
- `DataSet GetCouponProducts(int couponId, string where, string orderBy)` – returns the DataSet of coupon products selected with given condition and order.
- `void AddSKUToDiscountCoupon(int skuId, int discountId)` – adds the relationship between product and coupon.
- `void RemoveSKUToDiscountCoupon(int skuId, int discountId)` – removes the relationship between product and coupon.
- `bool IsSKUAssignedToDiscountCoupon(int skuId, int discountId)` – returns true if product is assigned to the discount coupon.
- `bool IsSKURelatedToDiscountCoupon(int skuId, int discountId)` – returns true if there is a relation between specified discount coupon and SKU.
- `DataSet GetDiscountCoupons()` – returns the DataSet of all discount coupons. See database reference for required fields.
- `DataSet GetDiscountCoupons(string where, string orderBy)` – returns the coupons with specified condition and order applied.

## 8.8 CustomDiscountLevelInfoProvider

This class works with discount level information. It uses the `CMS.Ecommerce.DiscountLevelInfo` class.

- `void DeleteDiscountLevelInfo(int discountLevelId)` – deletes the given discount level.
- `object GetDiscountLevelInfo(int discountLevelId), object GetDiscountLevelInfo(discountLevelCodeName)` - returns the DiscountLevelInfo for the specified discount level.
- `void SetDiscountLevelInfo(object discountLevel)` - updates discount level.
- `object GetAllDiscountLevelInfo()` - get all discount levels
- `object GetAllDiscountLevelInfo(string where, string orderBy)` - get all discount levels filtered by where condition and ordered by orderBy expression.

- `void AddToDepartment(int discountLevelId, int departmentId)` - assigns specified discount level to the department. It means that discount is applied only to products from the specified departments then.
- `void RemoveFromDepartment(int discountLevelId, int departmentId)` - removes specified discount level from the department.
- `DataSet GetDepartments(int discountLevelId)` - returns DataSet of all the departments to which the specified discount level is applied.

## 8.9 CustomExchangeTableInfoProvider

This class works with exchange rates. It uses the `CMS.Ecommerce.ExchangeTableInfo` class.

- `object GetExchangeTableInfo(int exchangeTableId)` - returns the ExchangeTableInfo object selected by given table ID.
- `object GetExchangeTableInfo(string tableName)` - returns the ExchangeTableInfo object selected by given table name.
- `void SetExchangeTableInfo(object exchangeTable)` - updates the exchange table record based on given ExchangeTableInfo object data. If ExchangeTableID is set, updates the record, else inserts a new one.
- `void DeleteExchangeTableInfo(int exchangeTableId)` - deletes the exchange table.
- `DataSet GetRatesByTableID(int tableId)` - returns the DataSet of exchange rates for given table ID.
- `void DeleteRateByRateId(int currencyId, int tableId)` - deletes specified currency exchange rate from exchange table.
- `void SetRateByRateId(int rateId, int CurrencyId, double rateValue, int tableId)` - sets the specified rate value.
- `object GetLastValidExchangeTableInfo()` - returns ExchangeTableInfo object that was last valid.
- `double ApplyExchangeRate(double value, double exchangeRate)` - applies the exchange rate to the given price. By default  $resulting\_price = original\_price / exchange\_rate$ .
- `DataSet GetExchangeTables()` - returns the DataSet of all the exchange tables. See database reference for column details.
- `DataSet GetExchangeTables(string where, string orderBy)` - returns the DataSet of exchange tables selected with given condition and order.

## 8.10 CustomInternalStatusInfoProvider

This class manages the internal status lookup table. It uses the **InternalStatusInfo** class.

- `object GetInternalStatusInfo(int internalStatusId)` – returns the **InternalStatusInfo** object selected by its ID.
- `object GetInternalStatusInfo(string internalStatusName)` – returns the **InternalStatusInfo** object selected by its name.
- `void SetInternalStatusInfo(object internalStatus)` – updates status record with data from given **InternalStatusInfo** object. If **StatusID** is set, updates the record, if not inserts a new record.
- `void DeleteInternalStatusInfo(int internalStatusId)` – deletes the internal status record with given ID.
- `bool CheckDependencies(int internalStatusId)` – returns true if some records depend on the status record.
- `DataSet GetStatuses(object status)` – returns the **DataSet** of all the statuses. See Database Reference for column definition.
- `object GetInternalStatusInfos(string where, string orderBy)` - returns the **DataSet** of all the statuses filtered by where condition and ordered by orderBy expression.

## 8.11 CustomManufacturerInfoProvider

This class manages the manufacturer lookup table. It uses the **ManufacturerInfo** class.

- `object GetManufacturerInfo(int manufacturerId)` – returns the **ManufacturerInfo** object selected by given ID
- `object GetManufacturerInfo(string manufacturerName)` – returns the **ManufacturerInfo** object selected by given name.
- `void SetManufacturerInfo(object manufacturer)` – updates/inserts the manufacturer record with data from given **ManufacturerInfo** object. If **ManufacturerID** is set, updates the record, if not inserts the record.
- `void DeleteManufacturerInfo(int manufacturerId)` – deletes the specified manufacturer record.
- `bool CheckDependencies(int manufacturerId)` – returns true if some objects depend on given manufacturer record.

- `DataSet GetAllManufacturers(object status)` – returns the DataSet of all the manufacturers. See database reference for column details.
- `object GetManufacturerInfos(string where, string orderBy)` – returns the DataSet of all the manufacturers filtered by where condition and ordered by orderBy expression.

## 8.12 CustomOptionCategoryInfoProvider

This class works with product option category information. It uses the **CMS.Ecommerce.OptionCategoryInfo** class.

- `object GetOptionCategoryInfo(int optionCategoryId)` – returns the OptionCategoryInfo object selected by given ID.
- `void SetOptionCategoryInfo(object manufacturer)` – updates/inserts the product option category record with data from given **OptionCategoryInfo** object. If CategoryID is set, updates the record, if not inserts the record.
- `void DeleteOptionCategoryInfo(int optionCategoryId)` – deletes the specified option category record.
- `bool CheckDependencies(int categoryId)` – returns true if some objects depend on given product option category record.
- `DataSet GetOptionCategories()` – returns the DataSet of all the product option categories. See database reference for column details.
- `DataSet GetOptionCategories(string where, string orderBy)` – returns the DataSet of all the product option categories filtered by where condition and ordered by orderBy expression.
- `DataSet GetSKUOptionCategories(int skuId, string where)` – returns the DataSet of all the option categories which are assigned to the specified SKU filtered by specified where condition.
- `void AddOptionCategoryToSKU(int categoryId, int skuId)` – assigns option category to the specified SKU.
- `void RemoveOptionCategoryFromSKU(int categoryId, int skuId)` – removes option category from the specified SKU.
- `void SortProductOptionsAlphabetically(int categoryId)` – sorts product options of the specified option category alphabetically (A-Z).

## 8.13 CustomOrderInfoProvider

This class manages the orders. It uses the OrderInfo class.

- `object GetOrderInfo(int orderId)` – returns the OrderInfo object selected by order ID.
- `void SetOrderInfo(object order)` – updates/inserts the order record based on the given OrderInfo object. If OrderID is set, updates the record, else inserts a new one.
- `void DeleteOrderInfo(int orderId)` – deletes the specified order record.
- `DataSet GetOrders()` – returns the DataSet with the orders. See Database Reference for column details.
- `DataSet GetOrders(string where, string orderBy)` – returns the DataSet of column orders with given condition and order.
- `DataSet GetOrderList()` – returns the DataSet of the orders data joined with their customer, status and currency data.
- `DataSet GetOrderList(string where, string orderBy)` – returns the order list DataSet with given condition and order.
- `string GetInvoice(int orderId, int siteId)` – returns the HTML code of an order invoice.
- `string GetPaymentOption(int paymentId)` – returns the HTML code for the payment option.
- `string GetShippingOption(int shippingId)` – returns the HTML code for the shipping option.
- `string GetAddress(int addressId)` – returns the HTML code for the given address.
- `string GetInvoiceNumber(object order)` – returns HTML code for the invoice number.
- `string GetOrderDate(object order)` – returns the HTML code for the order date.
- `string GetOrderNote(object order)` – returns the HTML code for the order note
- `string GetTaxRegistrationID(object customerObj)` – returns HTML code for the customer tax registration ID
- `string GetOrganizationID(object customerObj)` – returns HTML code for the customer organization ID
- `string GetTotalShipping(double value, object currency)` – returns the HTML code for the total shipping
- `string GetTotalPrice(double value, object currency)` – returns the HTML code for the total price
- `string GetProductList(DataTable dt, object currency, bool`

`renderDiscount`) – returns the HTML code for the product list (by default simple table of order items).

- `string GetTaxRecapitulation(DataTable dt, object currency)` – returns the HTML code for the tax recapitulation (by default table of the tax summary).
- `void SendOrderNotificationToCustomer(object cartObj)` – sends the order e-mail notification to the customer.
- `void SendOrderNotificationToAdministrator(object cartObj)` – sends the order e-mail notification to the administrator.
- `void SendOrderPaymentNotificationToCustomer(object cartObj)` – sends the order payment e-mail notification to the customer.
- `void SendOrderPaymentNotificationToAdministrator(object cartObj)` – sends the order payment e-mail notification to the administrator.
- `public void SendOrderStatusNotificationToAdministrator(object cartObj)` – sends the e-mail notification to administrator about new order status of the specified order.
- `public void SendOrderStatusNotificationToCustomer(object cartObj)` – sends the e-mail notification to customer about new order status of the specified order.

## 8.14 CustomOrderItemInfoProvider

This class manages the order items. It uses the **OrderItemInfo** class.

- `object GetOrderItemInfo(int orderId)` – returns the OrderItemInfo object selected by given ID.
- `void SetOrderItemInfo(object orderItem)` – updates/inserts the order item with given OrderItemInfo object data. If ItemID is set, updates the existing data, else inserts a new record.
- `void DeleteAllOrderItems(int orderId)` – deletes all items for specified order.
- `void DeleteOrderItemInfo(int orderId)` – deletes the order item with given ID.
- `DataSet GetOrderItems(string where, string orderBy)` – returns the DataSet of all the order items filtered by where condition and ordered by orderBy expression, see the database reference for the order item column definition.

## 8.15 CustomOrderStatusInfoProvider

Provider contains methods for order status lookup table. It uses the **OrderStatusInfo** class.

- `object GetOrderStatusInfo(int orderStatusId)` – returns the **OrderStatusInfo** object selected by given ID.
- `object GetOrderStatusInfo(string orderStatusName)` – returns the **OrderStatusInfo** object selected by given name.
- `DataSet GetStatuses(object status)` – returns the **DataSet** of order statuses with specified status. See database reference for column definitions.
- `DataSet GetStatuses(string where, string orderBy)` – returns the **DataSet** of all the order statuses filtered by where condition and ordered by orderBy expression, see the database reference for the order status column definition.
- `void SetOrderStatusInfo(object orderStatus)` – updates/inserts the order status record with data from given **OrderStatusInfo** object. If StatusID is set, updates, else inserts a new record.
- `void DeleteOrderStatusInfo(int orderStatusId)` – deletes the specified order status.
- `void MoveStatusUp(int statusId)` – moves the status up (changes the status order).
- `void MoveStatusDown(int statusId)` – moves the status down (changes the status order).
- `int GetLastStatusOrder()` – returns the order ID of the last status.
- `object GetFirstEnabledStatus()` – returns the first enabled **OrderStatusInfo** object.
- `object GetNextEnabledStatus(int statusId)` – returns next enabled **OrderStatusInfo** object which is situated after the specified status in order status flow
- `object GetPreviousEnabledStatus(int statusId)` – returns previous enabled **OrderStatusInfo** object which is situated before the specified status in order status flow.
- `bool CheckDependencies(int orderStatusId)` – returns true if some objects depend on the status.

## 8.16 CustomOrderStatusUserInfoProvider

The provider contains methods for order auditing log. It uses the **OrderStatusUserInfo** class.

- `object GetOrderStatusUserInfo(int orderStatusUserId)` – returns the

OrderStatusUserInfo object selected by given ID.

- `void SetOrderStatusUserInfo(object orderStatusUser)` – updates/inserts the order status history record with data from given OrderStatusUserInfo object. Updates when OrderStatusUserID is set, else inserts new record.
- `void DeleteOrderStatusUserInfo(int orderStatusUserId)` – deletes specified order status history record.
- `DataSet GetOrderStatusList(int orderId)` – returns the DataSet with order status history list, see View\_CMS\_OrderStatusUser\_Joined for required columns.

## 8.17 CustomPaymentOptionInfoProvider

This class manages the payment option lookup table. It uses the **PaymentOptionInfo** class.

- `object GetPaymentOptionInfo(int paymentOptionId)` – returns the PaymentOptionInfo object with specified ID.
- `object GetPaymentOptionInfo(string paymentOptionName, string siteName)` – returns PaymentOptionInfo object with specified name on given site.
- `void SetPaymentOptionInfo(object paymentOption)` – updates/inserts the payment option record with data from given PaymentOptionInfo object. If PaymentOptionID is set, updates, else insert a new record.
- `void DeletePaymentOptionInfo(int paymentOptionId)` – deletes the payment option info record.
- `DataSet GetPaymentOptions()` – returns the DataSet of payment options. See database reference for detailed column information.
- `DataSet GetPaymentOptions(string where, string orderBy)` – returns the DataSet of payment options with condition and order.
- `DataSet GetAllPaymentsForSite(int siteId, object status)` – returns all payment options for given site with specified status.
- `bool CheckDependencies(int paymentOptionId)` – returns true if some objects depend on given payment option.
- `string GetPaymentURL(object cartObj)` – returns the payment gateway URL to use with given ShoppingCartInfo object.
- `void AddPaymentToShipping(paymentId, shippingId)` - adds payment method to the given shipping option.
- `DataSet GetShippingPaymentList(int shippingID)` - fills dataset with payment



methods for specific shippingID; contains all details.

- `DataSet GetShippingPayments(int shippingOptionId, bool enabled, bool disabled)` - fills dataset with payment methods for specific shippingID; contains only payment method details.
- `void RemovePaymentFromShipping(int paymentId, int shippingId)` - removes payment method from the given shipping option.

## 8.18 CustomPublicStatusInfoProvider

This class manages the product status lookup table. Methods work with **PublicStatusInfo** class objects, replaced by **object** in method headers. Provider implements following methods:

- `object GetPublicStatusInfo(int publicStatusId)` – returns the **PublicStatusInfo** object with given ID.
- `object GetPublicStatusInfo(string publicStatusName)` – returns the **PublicStatusInfo** object with given name.
- `void SetPublicStatusInfo(object publicStatus)` – updates/inserts the public status record with data from given **PublicStatusInfo** object. If **StatusID** is set, updates, else inserts a new record.
- `void DeletePublicStatusInfo(int publicStatusId)` – deletes specified public status record.
- `DataSet GetStatuses(object status)` – returns the **DataSet** of all the public statuses with given status.
- `object GetPublicStatusInfos(string where, string orderBy)` – returns the **DataSet** of all the public statuses filtered by where condition and ordered by orderBy expression.
- `bool CheckDependencies(int publicStatusId)` – returns true if some objects depend on given status.

See database reference for the public status column definition or **PublicStatusInfo** class definition for the **PublicStatusInfo** object details.

## 8.19 CustomShippingOptionInfoProvider

This class manages the shipping option lookup table. It uses the **ShippingOptionInfo** class.

- `object GetShippingOptionInfo(int shippingOptionId)` – returns ShippingOptionInfo object with given ID.
- `object GetShippingOptionInfo(string shippingOptionName, string siteName)` – returns ShippingOptionInfo object with given name on the given site.
- `void SetShippingOptionInfo(object shippingOption)` – updates/inserts the shipping option with data from given ShippingOptionInfo object.
- `void DeleteShippingOptionInfo(int shippingOptionId)` – deletes the specified shipping option.
- `bool CheckDependencies(int shippingOptionID)` – returns true if some objects depend on given shipping option.
- `DataSet GetShippingOptions()` – returns the DataSet of all shipping options. See database reference for detailed column information.
- `DataSet GetShippingOptions(string where, string orderBy)` – returns DataSet of the shipping options with specified condition and order.
- `DataSet GetAllShippingsForSite(int siteId, object status)` – returns the DataSet of shipping options on given site with specified status.
- `double CalculateShipping(object cartObj, string siteName)` – returns the shipping price for given ShoppingCartInfo object, in default currency.
- `DataSet GetShippingOptions(object cart, object status)` – returns the DataSet of shipping options with specified status and depending on the shopping cart data.
- `double ApplyShippingFreeLimit(double shipping, double totalPrice, double siteStoreFreeLimit)` – Returns shipping price or zero according to site store shipping free limit.

## 8.20 CustomShoppingCartInfoProvider

This class provides methods for the shopping cart calculations. The methods internally work with **ShoppingCartInfo** object that stores and works with complete order information.

- `void EvaluateShoppingCartContent(object shoppingCart)` – evaluates the content in the given ShoppingCartInfo object. This method is required to fill at least both

ShoppingCartContentTable and ShoppingCartTaxTable objects with proper data for shopping cart to work properly.

- `void EvaluateShoppingCartContent(object shoppingCart, bool evaluateForInvoice)` – evaluates the content in the given ShoppingCartInfo object. This method is required to fill at least both ShoppingCartContentTable and ShoppingCartTaxTable objects with proper data for generating an order invoice from them.
- `object GetShoppingCartInfo(int shoppingCartId)` – returns ShoppingCartInfo object with given ID.
- `object GetShoppingCartInfo(Guid shoppingCartGUID)` – returns ShoppingCartInfo object with given GUID.
- `object GetShoppingCartInfo(int userId, string siteName)` – returns ShoppingCartInfo object for the specified user and site.
- `object GetShoppingCartInfoFromOrder(int orderId)` – returns ShoppingCartInfo object created from the specified order.
- `void SetShoppingCartInfo(object shoppingCart)` – updates/inserts the shopping cart with data from given ShoppingCartInfo object.
- `void DeleteShoppingCartInfo(int shoppingCartId)` – deletes the shopping cart with given ID.
- `void DeleteShoppingCartInfo(Guid shoppingCartGuid)` – deletes the shopping cart with given GUID.
- `void DeleteShoppingCartInfo(int userId, string siteName)` – deletes the shopping cart of the specified user and site.
- `void DeleteShoppingCartItems(int cartId)` – deletes all the shopping cart items.
- `DataSet GetExpiredShoppingCarts()` – returns shopping carts that are older than specified time interval from now.
- `object CreateShoppingCartInfo(int siteId)` – returns new empty ShoppingCartInfo object.
- `object CreateShoppingCartInfo(DataRow dr)` – returns new ShoppingCartInfo object created from the given data row.
- `object CreateShoppingCartInfo(SerializationInfo info, StreamingContext context)` – returns new ShoppingCartInfo object created from deserialized data.
- `void ClearShoppingCartPrivateData(object shoppingCart)` – clears customer's private data from shopping cart.

- `void ActualizeSKUAvailableItems(object shoppingCart)` - actualizes products' available items according to the products' units of the specified shopping cart.
- `public static void SetOrder(ShoppingCartItemInfo shoppingCart)` - inserts or updates order according to the shopping cart data.

## 8.21 CustomShoppingCartItemInfoProvider

This class manages the shopping cart items. It uses the **ShoppingCartItemInfo** class.

- `object GetShoppingCartItemInfo(Guid cartItemGuid)` - returns ShoppingCartItemInfo object with given GUID.
- `object GetShoppingCartItemInfo(int cartItemId)` - returns ShoppingCartItemInfo object with given ID.
- `void DeleteShoppingCartItem(Guid cartItemGuid)` - deletes the shopping cart item with given GUID.
- `void DeleteShoppingCartItem(int cartItemId)` - deletes the shopping cart item with given ID.
- `void SetShoppingCartItemInfo(object cartItem)` - updates/inserts the shopping cart item with data from given ShoppingCartItemInfo object.
- `void SetShoppingCartItems(ArrayList cartItems)` - updates/inserts all the shopping cart items with data from given list items.
- `DataSet GetShoppingCartItems(int cartId)` - returns the DataSet of all shopping cart items including their SKU data of the specified shopping cart.
- `DataSet GetShoppingCartItemsFromOrder(int orderId)` - returns the DataSet of all shopping cart items including their SKU data of the specified order.
- `ArrayList GetShoppingCartItemList(int cartId)` - returns the list of the ShoppingCartItemInfo objects from the specified shopping cart.
- `ArrayList GetShoppingCartItemListFromOrder(int orderId)` - returns the list of the ShoppingCartItemInfo objects created from the order items of the specified order.

## 8.22 CustomSKUInfoProvider

This provider manages the products (SKUs). It uses the **SKUInfo** class.

- `object GetSKUInfo(int skuId)` – returns the SKUInfo object selected by given ID.
- `object GetSKUInfo(Guid skuGuid)` - returns the SKUInfo object selected by given GUID.
- `void SetSKUInfo(object sku)` – updates/inserts the SKU record from given SKUInfo object. If SKUID is set, updates, else inserts a new record.
- `void DeleteSKUInfo(int skuId)` – deletes specified SKU record.
- `bool CheckDependencies(int skuId)` – returns true if some objects depend on given SKU.
- `DataSet GetSKUs()` – returns the DataSet of all products. See database reference for detailed column information
- `DataSet GetSKUs(string where, string orderBy)` – returns DataSet of products selected with condition and order.
- `DataSet GetSKUs(bool products, bool productOptions, string where, string orderBy)` – returns DataSet of products and/or product options selected with condition and order.
- `DataSet GetSKUList(string where, string orderBy)` – returns the DataSet of product list. Result contains the SKU data and public and internal status display name.
- `DataSet GetTopNProductsBySales(int topN, int siteId, string where)` – returns the DataSet with top N products by sales for the specified site and according to the where condition.
- `void AddSKUtoWishlist(int userId, int skuId, int siteId)` – adds specified SKU to user's wishlist.
- `void RemoveSKUfromWishlist(int userId, int skuId, int siteId)` – removes specified SKU from the wishlist of the given user.
- `bool IsSKUinWishlist(int userId, int skuId, int siteId)` - returns true if the given SKU is present in the user's wish list.
- `DataSet GetWishlist(int userId, int siteId)` - returns the content of wish list for the given user as a DataSet.
- `DataSet GetOptionCategorySKUs(int categoryId, string where)` – returns the DataSet of all product options from the specified product option category filtered by specified where condition.
- `void MoveSKUOptionUp(int skuId)` – moves specified product option up within the parent product option category.
- `void MoveSKUOptionDown(int skuId)`– moves specified product option down within the parent product option category.

Read also: [Using the Product datalist web part](#)

## 8.23 CustomSupplierInfoProvider

This class manages the supplier lookup table. It uses the **SupplierInfo** class.

- `object GetSupplierInfo(int supplierId)` – returns the **SupplierInfo** object with given ID.
- `object GetSupplierInfo(string supplierName)` – returns the **SupplierInfo** object with given name.
- `void SetSupplierInfo(object supplier)` – updates/inserts the supplier record from the given **SupplierInfo** object. If **SupplierID** is set, updates, else inserts a new record.
- `void DeleteSupplierInfo(int supplierId)` – deletes the supplier record.
- `DataSet GetAllSuppliers(object status)` – returns the **DataSet** of all the suppliers. See database reference for detailed column information.
- `object GetSupplierInfos(string where, string orderBy)` – returns the **DataSet** of all the suppliers filtered by where condition and ordered by orderBy expression, see the database reference for the supplier column definition.
- `bool CheckDependencies(int supplierId)` – returns true if some objects depend on given supplier ID.

## 8.24 CustomTaxClassInfoProvider

This class manages the tax classes. It uses the **TaxClassInfo** and **TaxClassCountryInfo** classes.

- `object GetTaxClassInfo(int taxClassId)` – returns the **TaxClassInfo** object with given ID.
- `object GetTaxClassInfo(string taxClassName)` – returns the **TaxClassInfo** object with given name.
- `void SetTaxClassInfo(object taxClass)` – updates/inserts the tax class record with data from given **TaxClassInfo** object. If **TaxClassID** is set, updates, else inserts a new record.
- `void DeleteTaxClassInfo(int taxClassId)` – deletes specified tax class record.
- `DataSet GetTaxClasses()` – returns the **DataSet** of all the tax classes. See database reference for detailed column information.

- `DataSet GetTaxClasses(string where, string orderBy)` – returns the DataSet of tax classes with selection condition and order.
- `void SetTaxClassCountryInfo(object taxClassCountry)` – updates/inserts the tax class country record. If IDs are set, updates, else inserts a new record.
- `void DeleteTaxClassCountryInfo(int taxClassCountryId, int taxClassId)` – deletes specified tax class country record.
- `object GetTaxClassCountryInfo(int taxClassCountryId, int taxClassId)` – returns the TaxClassCountryInfo object for given tax class and country.
- `DataSet GetTaxClassCountries(int taxClassId)` – returns the DataSet of all the countries assigned to the given tax class.
- `DataSet GetTaxClassesForCart(object cartInfo)` – returns the DataSet of all the classes used within the given ShoppingCartInfo object.
- `DataSet GetTaxClassesForCart(object cartInfo, string where)` – returns the DataSet of all the country tax classes used within the given ShoppingCartInfo object.
- `void AddTaxClassToSKU(int taxClassId, int skuId)` – assigns the given tax class to the specified product.
- `void RemoveTaxClassFromSKU(int taxClassId, int skuId)` – removes the tax class binding from the specified product.
- `bool IsTaxClassAssignedToSKU(int taxClassId, int skuId)` – returns true if the product is assigned for the given tax class.
- `DataSet GetSKUTaxClasses(int skuId)` – returns the DataSet of tax classes assigned to given product.
- `void SetTaxClassStateInfo(object taxClassState)` - saves the information about assignment of the state to the tax class.
- `void DeleteTaxClassStateInfo(int stateId, int taxClassId)` - removes the state from the given tax class.
- `DataSet GetTaxClassStates(int taxClassId, int countryId)` - returns DataSet with states of the given country assigned to the given tax class.
- `DataSet GetStateTaxClassesForCart(object cartInfo)` - returns the DataSet of all the state tax classes used within the given ShoppingCartInfo object.
- `DataSet GetStateTaxClassesForCart(object cartInfo, string where)` - returns the DataSet of all the state tax classes used within the given ShoppingCartInfo object.
- `DataSet GetAllCountriesWithStates()` - returns the DataSet of all the countries which have some states.

## 8.25 CustomVolumeDiscountInfoProvider

This class manages the volume discounts for particular products.

- `object DeleteVolumeDiscountInfo(int volumeDiscountId)` – deletes the `TaxClassInfo` object with given ID.
- `void DeleteVolumeDiscounts(int SKUId)` - deletes all volume discounts for the given SKU (product).
- `object GetVolumeDiscountInfo(int volumeDiscountId)` - returns volume discount.
- `object GetVolumeDiscountInfo(int SKUId, int units)` - returns volume discount object for given SKU (product) and given quantity of the product.
- `DataSet GetVolumeDiscounts(int SKUId)` - returns `DataSet` with all volume discounts for the given SKU (product).
- `void SetVolumeDiscountInfo(object volumeDiscount)` - saves the volume discount to the database.
- `object GetVolumeDiscounts(string where, string orderBy)` - returns the `DataSet` of all the volume discounts filtered by where condition and ordered by orderBy expression, see the database reference for the volume discount column definition.

## 8.26 Using the Product datalist web part

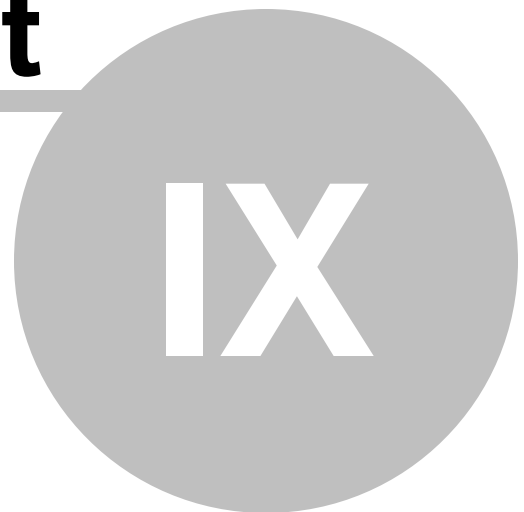
If you're using a custom provider for managing product information (using the **CustomSKUInfoProvider**), you will need to use the **Product datalist** web part or develop a similar web part that will display the products using the custom provider.

You can find more details on the **Product datalist** web part in **Kentico CMS Web Parts and Controls Reference/Web Parts/E-commerce/Product datalist**.



**Part**

---



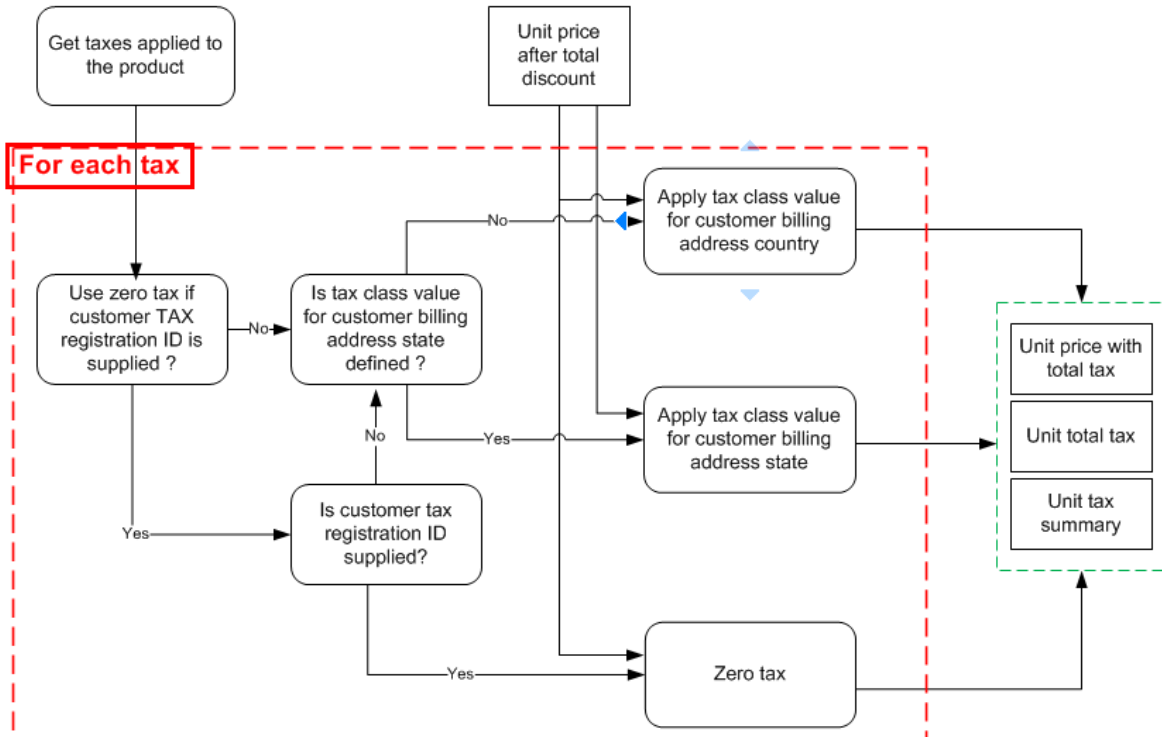
**Figure A: Tax calculation process**

---

## 9 Figure A: Tax calculation process

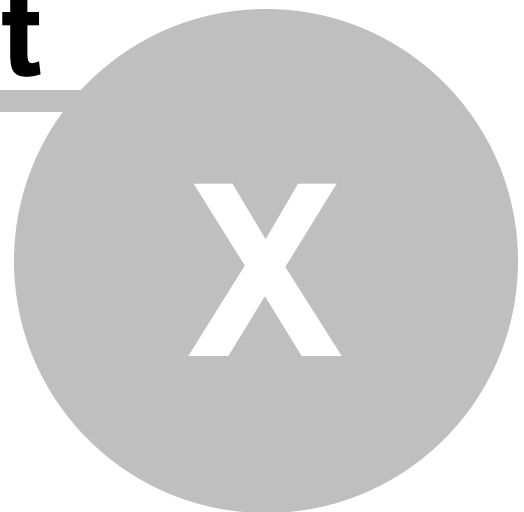
The following figure shows how the tax is calculated:

### Taxes



**Part**

---



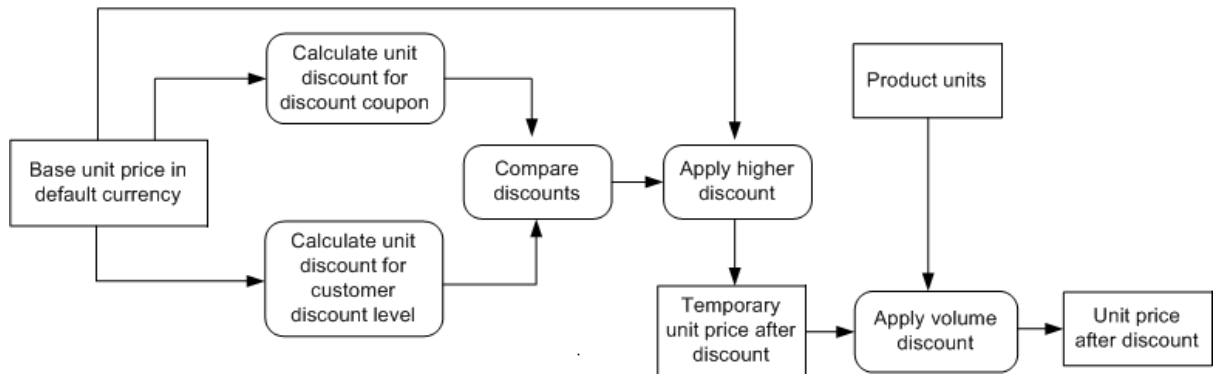
**Figure B: Discount calculation**

---

## 10 Figure B: Discount calculation

The following figure shows how discounts are calculated:

### Discounts



**Part**

---

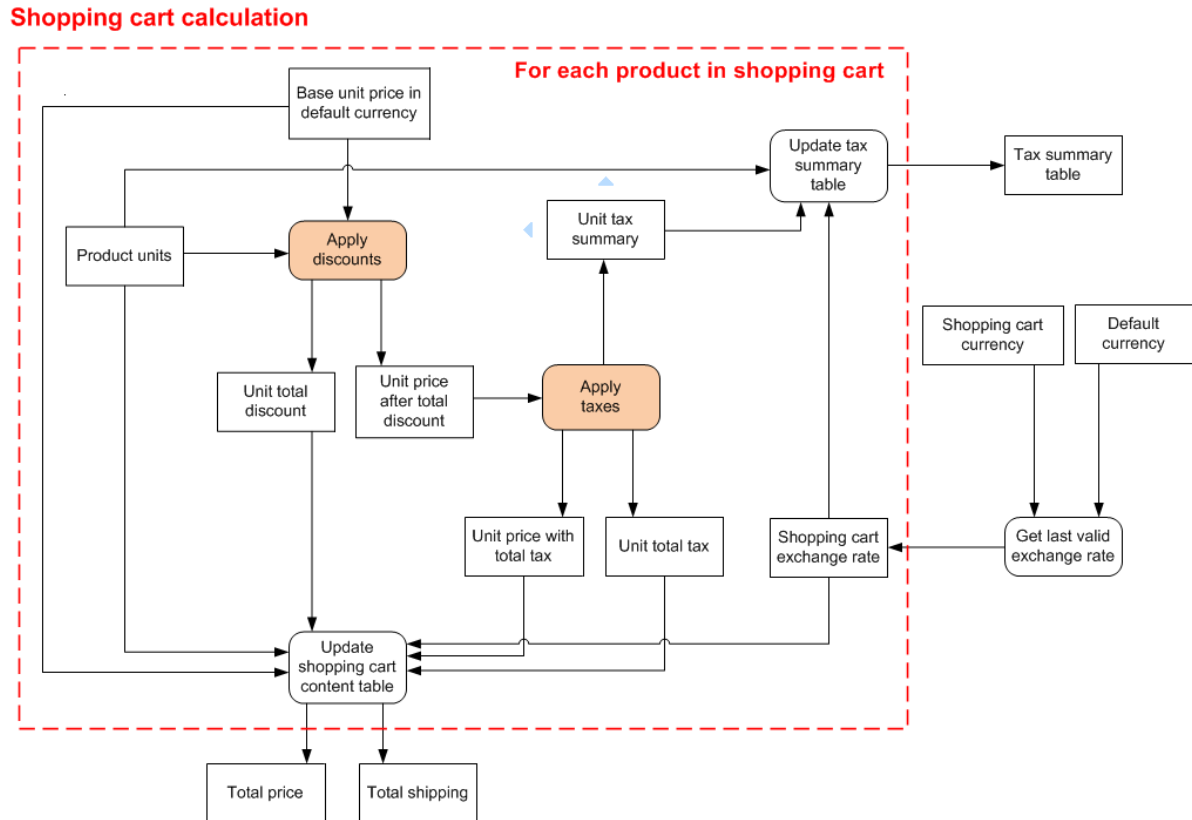
**XI**

**Figure C: Shopping cart price calculation**

---

## 11 Figure C: Shopping cart price calculation

The following figure shows how the total price is calculated:



**Part**

---

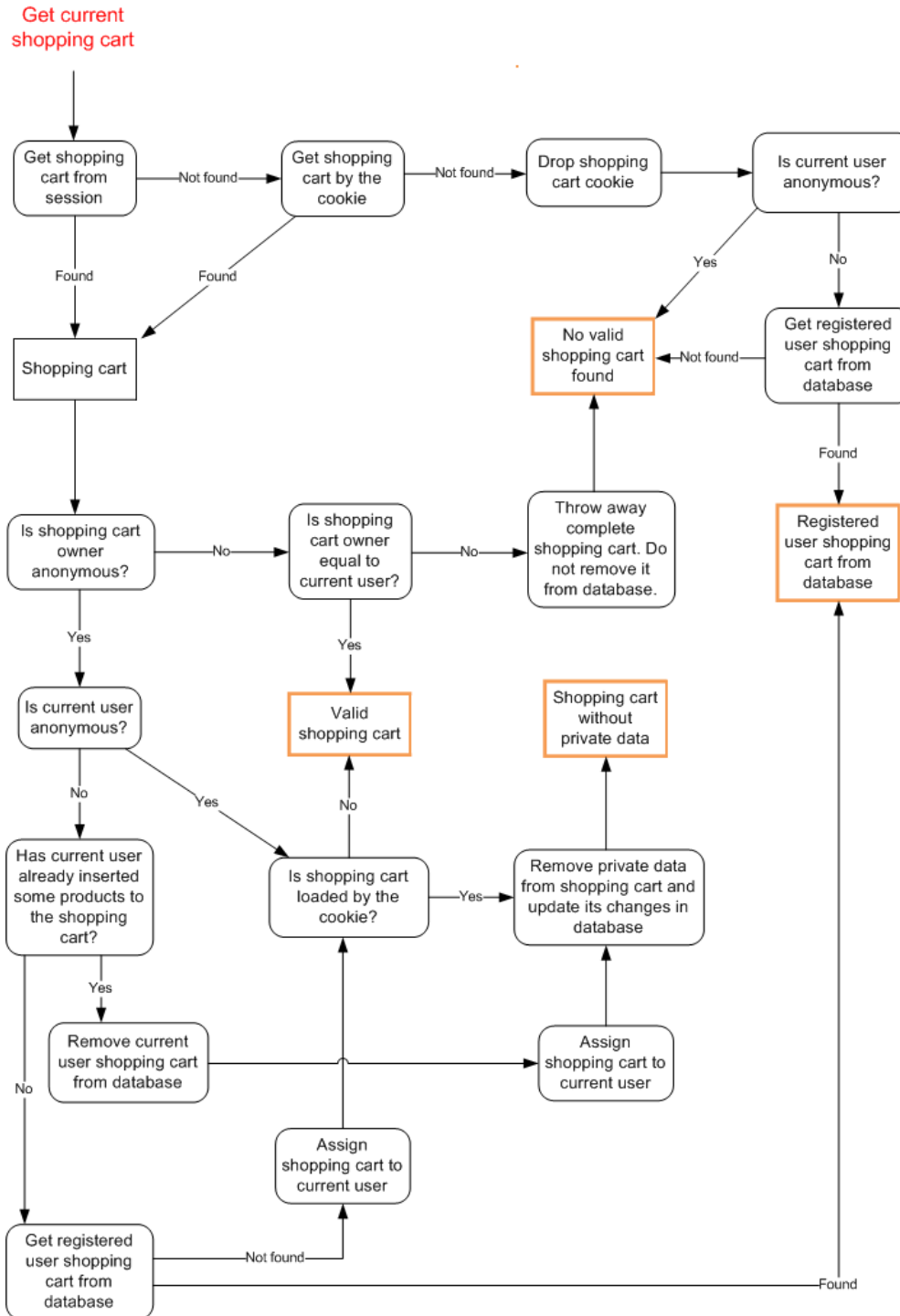
**XII**

**Figure D: Shopping cart retrieval**

---

## 12 Figure D: Shopping cart retrieval

The following figure shows how the shopping cart is retrieved for the current visitor:





# Index

## - A -

- Adding items to the shopping cart 116
- Adding items to the wish list 119

## - C -

- Company details
  - changing 37
- Configuration settings
  - countries and states 142
  - currencies 137
  - customers 127
  - customizing an e-mail template 149
  - customizing invoice template 149
  - deleting old shopping carts 166
  - departments 135
  - discount coupons 132
  - discount levels 133
  - discounts overview 160
  - enabling the e-commerce module 140
  - exchange rates 138
  - internal status 139
  - invoice 139
  - localization 147
  - manufacturers 132
  - mapping document fields 144
  - multi-site configuration 143
  - order status 138
  - orders 126
  - overview 125
  - payment methods 136
  - product options 131
  - products 129
  - public status 139
  - reports 133
  - security 142
  - shipping options 135
  - sitemanager settings 144
  - store 134
  - suppliers 132
  - tax classes 137
  - using product inventory 155

- web.config settings 145
- Custom providers
  - CustomAddressInfoProvider 197
  - CustomCurrencyInfoProvider 197
  - CustomCustomerInfoProvider 199
  - CustomDepartmentInfoProvider 199
  - CustomDiscountCouponInfoProvider 200
  - CustomDiscountLevelInfoProvider 201
  - CustomExchangeTableInfoProvider 202
  - CustomInternalStatusInfoProvider 203
  - CustomManufacturerInfoProvider 203
  - CustomOptionCategoryInfoProvider 204
  - CustomOrderInfoProvider 205
  - CustomOrderItemInfoProvider 206
  - CustomOrderStatusInfoProvider 207
  - CustomOrderStatusUserInfoProvider 207
  - CustomPaymentOptionInfoProvider 208
  - CustomPublicStatusInfoProvider 209
  - CustomShippingOptionInfoProvider 210
  - CustomShoppingCartInfoProvider 210
  - CustomShoppingCartItemInfoProvider 212
  - CustomSKUInfoProvider 212
  - CustomSupplierInfoProvider 214
  - CustomTaxClassInfoProvider 214
  - CustomVolumeDiscountInfoProvider 216
- overview 194
- Product datalist web part 216
- using 195

## - D -

- Discount calculation 220

## - F -

- First purchase 9

## - I -

- Images
  - displaying 92
  - overview 92
  - resizing 96
  - storing 96
- Integration with existing site
  - overview 75

**- O -**

Overview 7

**- P -**

## Product

- adding new 14
- adding product options 18
- adding to multiple categories 26
- customizing product categories 22
- displaying featured products 30
- sorting and paging 33

Product datalist web part 216

## Product fields

- adding custom fields 57
- adding images 54
- customizing product design 51
- defining new product type 44

## Product price

- displaying 120

## Product URL

- getting 121

## Purchasing and payment

- Authorize.NET configuration 179
- custom checkout dialogs 172
- custom payment gateways 183
- customer credit 191
- customizing 169
- overview 168
- payment gateways 178
- payment results 189
- PayPal configuration 180

**- S -**

Searching 122

Shopping cart price calculation 222

Shopping cart retrieval 224

**- T -**

Tax calculation process 218

## Tools

- BizForms 100
- Blogs 101

- Booking system 102
- Content staging 103
- Event calendar 104
- File import 105
- Forums 106
- GeoMapping 108
- Image gallery 108
- Messaging 109
- Newsletters 110
- overview 99
- Polls 111
- Reporting 112
- RSS feeds 113
- User contributions 113
- Web analytics 114
- Web farm synchronization 115

**- W -**

## Web design

- colors using CSS styles 71
- customizing master page 62

## Web parts

- my account 76
- random products 78
- shopping cart 82
- shopping cart preview 85
- similar products by sale 87
- top N products by sale 88
- wishlist 90