Create a shopping cart with MX Kart

version 3.0

Table of Contents

Overview

This document aims to guide you through the process of creating a dynamic e-commerce site with the **MX Kart** Extensions.

- The site will be used to list and sell various goods online
- Customers will be able to view the existing products
- Add/subtract products to/from the shopping kart
- Update the cart
- Based on their authenticated user level customers will get discounts
- They can get other kind of discounts as well
- They will be guided through a three step checkout wizard where they will enter information allowing the application to set taxes and shipping rates

Such a dynamic e-commerce web site can now be created **intuitively in just a few minutes**. **MX Kart** supplies innovative features and tight integration with our **tNG** transaction engine and **Macromedia Dreamweaver MX** and **Dreamweaver MX 2004**.

The **MX Kart** tool is distributed in two versions: for PHP_MySQL and PHP_ADODB server models.

Prerequisites

Requirements

This tutorial requires basic knowledge of **Macromedia Dreamweaver MX** development practices.

To use **MX Kart** you will have to install the following software programs:

Macromedia Dreamweaver MX	http://www.macromedia.com/
PHAkt2 or **PHP_MySQL**	http://www.interaktonline.com/products/PHAkt/
MX Kart	http://www.interaktonline.com/products/MXKart/
Web server with PHP support	http://www.php.net
A MySQL database	http://www.mysql.com/

Note: Please follow the install notes found in each installation kit to configure your workspace. We presume you have a correctly configured platform for PHP development under Dreamweaver MX (the configured Windows or Linux server, share or FTP access, a Dreamweaver MX site).

Typographic Conventions

The notations and text formats used in this tutorial are found below:

- database name will be displayed using bold font "**kart_database**"
- database table: using an italic font "*products_prod*"
- database field will be displayed using a bold, italic font "***id_prod***"
- site name: underlined font "*new site*"
- site page: monospaced italic "*index.php*"
- recordset name: underlined italic "*recordset*"
- application button, menu or panel: bold font "**Button**"
- source code : monospaced font "`<?php echo "MX Kart tutorial"?>`"
- links in the generated code: ***Add to Cart***

Files

The **MX Kart** tutorial also includes the SQL script needed to create the database and the zip archive of the "mx_kart" site that is created below. You can use these files to overcome some technical difficulties you might have or to compare the results.

The database generation script: **mx_shop.sql**

mx_shop.mdb

The "mx_kart" website: **MXKart_tutorial_ADODB.zip**, **MXKart_tutorial_MySQL.zip** and **MXKart_tutorial_CF.zip**

In case you want to use your own database - **WE STRONGLY ADVISE YOU USE THE DEFAULT MX Kart DATABASE –** you should run the database generation scripts that will include the product properties tables – **mx_shop_properties.sql** and the orders tables – **mx_shop_orders.sql**.

Configuring Your Site

Managing your database

First you must create a new MySQL database you want to connect to and dump the **mx_shop.sql** file in your newly created database. In order to manage your database, use either phpMyAdmin or a Unix console.

In **phpMyAdmin** write your database name in the ***Create New Database*** text-field and click **Create**. Next, click on the ***SQL*** tab and browse (using the **Browse** button) to the **mx_shop.sql** file. Click **Go**.

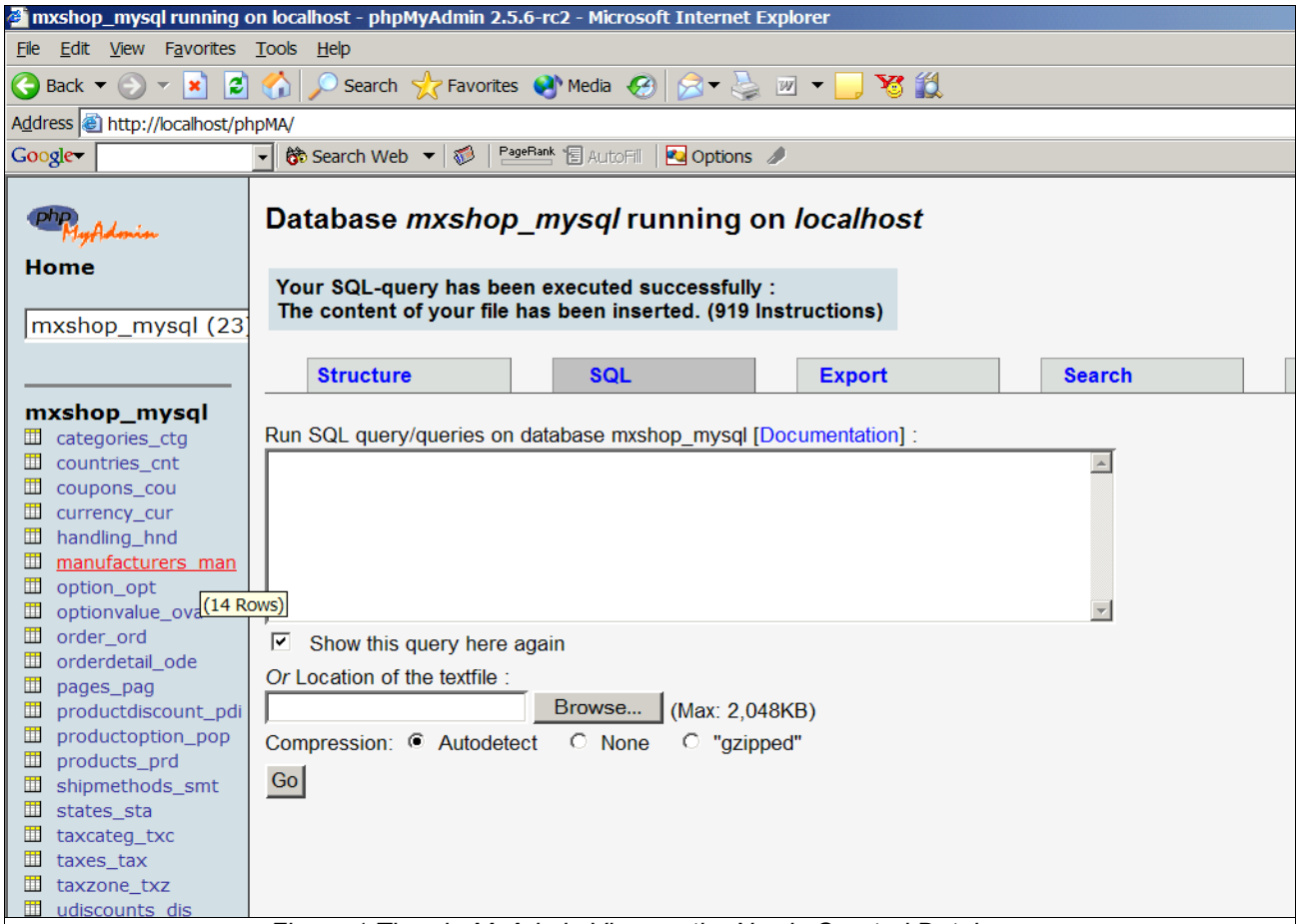


Figure 1 The phpMyAdmin View on the Newly Created Database

Database Design

For this tutorial, we will consider the **mx_shop** database having 24 tables. It is not a simple database structure, but it is complex because it will help you create complex e-commerce sites.

Below is a graphical representation of the database tables made using the **InterAKT** visual query editor **QuB** (http://www.interaktonline.com/products/QuB/).

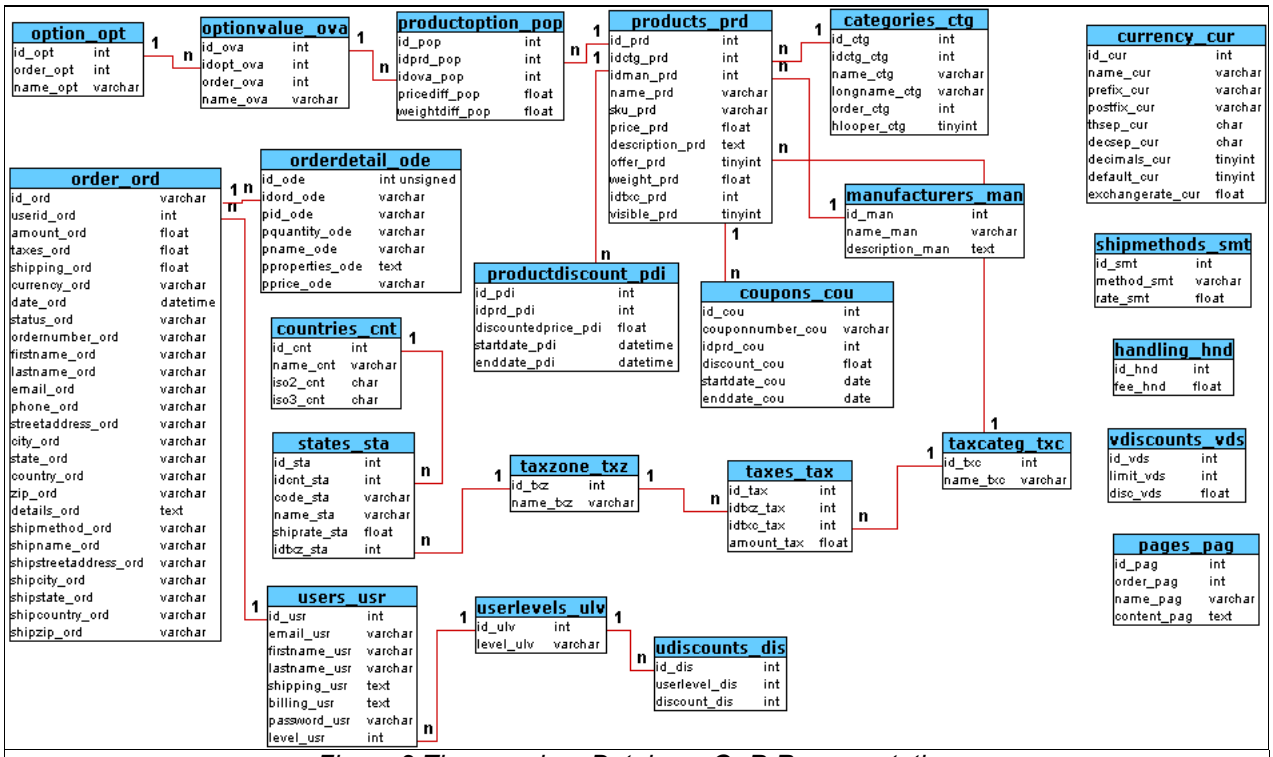


*Figure 2 The mx_shop Database **QuB** Representation*

Product related tables

These tables will store information (with their structure and properties) about the products sold with your **MX Kart** site.

- *products_prd* – this table stores all products to be sold on the website. The table fields are: ***id_prd*** (the primary key), ***idctg_prd*** (the foreign key to the *categories_ctg* table), ***idman_prd*** (the foreign key to the *manufacturers_man* table), ***name_prd*** (the product name), ***sku_prd*** (the product code), ***price_prd*** (the product price), ***description_prd*** (the product long description), ***offer_prd*** (if this field value is set to "1", the product is a special offer), ***weight_prd*** (the weight of the current product – will be used to compute the shipping costs), ***idtxc_prd*** (a foreign key to the tax categories table to select the tax category of the current product) and ***visible_prd*** (if this field value is set to "1", the product is visible on the site and can be bought by the customers
- *productoption_pop* – this is a many-to-many table which stores the properties values for each product along with the price and weight difference corresponding to each property value. The table fields are: ***id_pop*** (the primary key), ***idprd_pop*** (the foreign key to the the *products_prd* table), ***idova_pop*** (foreign key to the *optionvalue_ova* table, representing the selected option), ***pricediff_opp*** (the price difference corresponding to a selected property) and ***weightdiff_opp*** (the weight difference corresponding to a selected property).
- *option_opt* – this table stores all the possible product properties (eg. color, size etc.). The table fields are: ***id_opt*** (the primary key), ***name_opt*** (the option name) and ***order_opt*** (the properties display order when the Add to Kart by Form Server Behavior is applied).
- *optionvalue_ova* – this table stores the product properties values (eg. red, white; small, large etc.). The table fields are: ***id_ova*** (the primary key), ***name_ova*** (the property value name), ***order_ova*** (the property display order) and ***idopt_ova*** (the foreign key to the *option_opt* table to define the options category this property is belonging to).

- *manufacturers_man* – this table stores the product manufacturers. The table fields are: ***id_man*** (the primary key), ***name_man*** (the manufacturer name) and ***description_man*** (the manufacturer description).
- *categories_ctg* – this table stores all product categories. The table fields are: ***id_ctg*** (the primary key), ***idctg_ctg*** (parent category ID), ***name_ctg*** (the category name), ***longname_ctg*** (the category long name) ***order_ctg*** (the category order in the current subcategory) and ***hlooper_ctg*** (when 1, this can instruct the product list for a category to be rendered in a horizontal looper approach).

Order related tables

These tables will store all the information on the orders placed on the **MX Kart** site.

- *order_ord* – this table stores the basic information related to customers' orders. This table together with the order details table will keep the order information unnormalized for archiving purposes (making sure the old purchase price is displayed even if the product price has changed).

  The table fields are: ***id_ord*** (the primary key), ***userid_ord*** (a foreign key to the users table that can be null), ***amount_ord*** (the order amount), ***taxes_ord*** (the taxes amount for the current order), ***shipping_ord*** (the shipping rates amount for the current order), ***currency_ord*** (the order currency), ***date_ord*** (the order date), ***status_ord*** (the order status – can be Initialized, Confirmed, Shipped or any other value as set from the payment gateway return message), ***ordernumber_ord*** (the order number), ***firstname_ord*** (the customer first name), ***lastname_ord*** (the customer last name), ***email_ord*** (the customer email), ***phone_ord*** (the customer phone), ***streetaddress_ord*** (the customer address), ***city_ord*** (the customer city), ***state_ord*** (the customer state), ***country_ord*** (the customer country), ***zip_ord*** (the customer zip code) and ***details_ord*** (the order status details), ***shipmethod_ord*** (the shipping method), ***shipname_ord*** (the name of the customer where the products will be shipped), ***shipstreetaddress_ord*** (the street address where the products will be shipped), ***shipcity_ord*** (the city where the products will be shipped), ***shipstate_ord*** (the state where the products will be shipped), ***shipcountry_ord*** (the country where the products will be shipped) and ***shipzip_ord*** (the zip code where the products will be shipped).
- *orderdetail_ode* – this table stores the detailed information of the products that were ordered. The table fields are: ***id_ode*** (the primary key), ***idord_ode*** (the foreign key to the *order_ord* table), ***pid_ode*** (the product ID), ***pquantity_ode*** (the ordered quantity), ***pname_ode*** (the product name), ***pproperties_ode*** (the product properties – summarized from the Add to Kart by Form command) and ***pprice_ode*** (the product price at the order date).

Discounts tables

These tables will store information on the possible discount functions that can be applied to an **MX Kart** site. Changing the information in these tables will allow the site administrator several option in changing the discount policies.

- *coupons_cou* – this table stores all the coupon numbers that can be used by customers to benefit from discounts. The table fields are: ***id_cou*** (the primary key), ***couponnumber_cou*** (the coupon number), ***idprd_cou*** (the product id that will be discounted using coupons), ***discount_cou*** (the discount amount – can be a percent or exact value), ***startdate_cou*** (the start date of the coupon validity period) and ***enddate_cou*** (the end date of the coupon validity period).
- *productdiscount_pdi* – this table stores some special offers for products. The table fields are: ***id_pdi*** (the primary key), ***idprd_pdi*** (a foreign key to the products table), ***discountedprice_pdi*** (the product discounted price), ***startdate_pdi*** (the start date of the special offer), ***enddate_pdi*** (the end date for the special offer).
- *vdiscounts_vds* – this table stores the volume discounts applied to an orders total price. The table fields are: ***id_vds*** (the primary key), ***limit_vds*** (the total price cart limit) and ***disc_vds*** (the discount value).
- *udiscounts_dis* – this table stores all user level discounts applied to products' prices. The table fields are: ***id_dis*** (the primary key), ***userlevel_dis*** (the user level corresponding to the current discount) and ***discount_dis*** (the discount for each user level).

Geographic related tables

These tables will store information on countries and states.

- *countries_cnt* – this table stores all the countries. The table fields are: ***id_cnt*** (the primary key), ***name_cnt*** (the country name), ***iso2_cnt*** (the country two letters acronym) and ***iso3_cnt*** (the country three letters acronym).

- *states_sta* – this table stores all the states in the countries. The default database provided is designed to have at least one state per country if nothing else is available the country's name is used. The table fields are: ***id_sta*** (the primary key)**,** ***idcnt_sta*** (the foreign key to the countries table), ***code_sta*** (the code of the state)**,** ***name_sta*** (the state name), ***shiprate_sta*** (the price per weight unit to be paid when shipping a product to this state) and ***idtxz_sta*** (a foreign key to the tax zones table – this will be used to determine the taxzone of the customer's billing address).

Tax related tables

These tables will store information on the configuration tables for the Tax functions of an **MX Kart** site.

- *taxcateg_txc* – this table stores all the tax categories. We created this table because some countries or states apply different taxes on different product categories (eg. food, non-food). The table fields are: ***id_txc*** (the primary key), ***name_txc*** (the tax category name).
- *taxes_tax* – this table stores all taxes. The table fields are: ***id_tax*** (the primary key), ***idtxz_tax*** (the foreign key to the tax zones table), i***dtxc_tax*** (the foreign key to the tax categories table), ***amount_tax*** (the tax amount – can be a percent).
- *taxzone_txz* – this table stores all the tax zones. The table fields are: ***id_txz*** (the primary key), ***name_txz*** (the tax zone name).

Shipping related tables

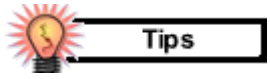
These tables will store the shipping functions configuration information.

- *handling_hnd* – this table stores the handling fees. The table fields are: ***id_hnd*** (the primary key), ***fee_hnd*** (the fee amount).
- *shipmethods_smt* – this table stores all the shipping methods. The table fields are: ***id_smt*** (the primary key), ***method_smt*** (the shipping method name), ***rate_smt*** (the price per weight unit to be paid when shipping using a certain method).

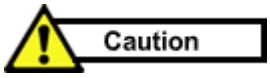
Currencies

This table will store the available currencies for your **MX Kart** site together with their formatting capabilities.

- *currency_cur* – this table stores all the available currencies. The table fields are: ***name_cur*** (the currency name), ***prefix_cur*** (the currency symbol that will be placed as prefix – for example $), ***postfix_cur*** (the currency symbol that will be placed as postfix – for example AUD), ***thsep_cur*** (the thousand separator – might be "." or ","), ***decsep_cur*** (the decimals separator), ***decimals_cur*** (the number of decimals), ***default_cur*** (this is a flag that sets the selected currency as default if its value is "1"), ***exchangerate_cur*** (the exchange rate to the default currency).



On your e-commerce website you should set the default currency by entering the "1" value in the ***default_cur*** field for the desired currency and "0" for the rest of the currencies. When changing the currency on your public site, **MX Kart** will automatically recalculate the prices according to the exchange rate that you entered into the database.



After entering the products into the corresponding table with prices displayed in the default currency we recommend you **NOT** change the default currency. If you change the default currency you will have to manually modify the prices in the products table. However on your website the prices can be displayed in any currency (see the **Choosing the Currency** section in **MX Kart User Manual**) even if in the database they are stored in the default currency.

Users tables

This table contains all the information regarding the users that can authenticate to an **MX Kart** site

- *users_usr* – this table stores all users registered at the site. The table fields are: ***id_usr*** (the primary key), ***email_usr*** (the user's email), ***firstname_usr*** (the user's first name), ***lastname_usr*** (the user's last name), ***shipping_usr*** (the user's shipping address), ***billing_usr*** (the user's billing address), ***password_usr*** (the user's password) and ***level_usr*** (the user's level).

The complete scripts used to generate the database tables are included with the **MX Kart** distribution package in a zip archive.

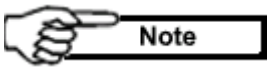
The Dreamweaver MX Site Creation

Our basic **MX Kart** sample site is a fully featured e-commerce site presenting product categories, a list of products in each category and the ability for the customer to easily manage the shopping process (add to cart, delete/update cart). The site includes support for many discount types. The checkout process proceeds through a 4 step wizard that will ask for the shipping and billing address so that the shipping and billing fees for the current cart contents may be computed.

Our site will contain 9 pages:

- `index.php` – a page that will display a list of products along with the add to cart link, a login nugget and a cart nugget.
- `viewKart.php` – so customers can update the cart (only the quantity of already chosen products), delete products from the cart or checkout to a payment gateway
- `prodDetail.php` – allows the customers to view all the product properties, select them, choose the quantity and then to add the product to the cart
- `logout.php` – allows customers to log out
- `addCoupons.php` – allows customers to enter a coupon code to receive discounts on certain products
- `step0.php` – this is the first step of the checkout wizard but it will not be visible to the customer. On this page, a trigger is executed to insert a record that contains all the kart fields into the *order_ord* table. If the user is logged in to the site these fields will be automatically filled-in with previous order data.
- `step1.php` – will ask the the customer's personal information and will update it in the *order_ord* table. It is the first page of the checkout wizard that will be visible to the customer.
- `step2.php` – update the *order_ord* table with the customer shipping information. This is the second page of the checkout wizard and it will be visible to the customer
- `step3.php` – displays the order confirmation page

Open **Macromedia Dreamweaver MX** and in the **New Site** option of the **Site** menu create a new site named "mx_kart". Configure your site following the next three steps.



All the screenshots in this tutorial are made when creating the mx_kart site using the **PHP_ADODB** server model. The screenshots for the **PHP_MySQL** are slightly different.

Step 1: Configuring the Local Information section

The information requested in the **Local Info** section refers to the local settings that you will use during the development process: Local Root Folder (for our purposes we used the "C:\My Documents\Sites\mx_kart" folder). The http:// address is the URL of the actual site (in this case, the HTTP address will be http://work.iakt.ro/test/mba/mx_kart (this is a local URL so don't try to load it in your browser).

Step 2: Configuring the Remote Information section

In the **Remote Info** section of this menu you will indicate the connection type used to upload the files to the production server and the actual path on the remote computer. You can also choose to automatically upload the files to the server after saving them (this option is very useful in saving you a lot of "Ctrl+Shift+U" key presses--this is the shortcut for uploading a file to the remote server). If you are working with a team you might also wish to activate the Check In/Check Out support that will forbid two users from editing the same file at the same time.

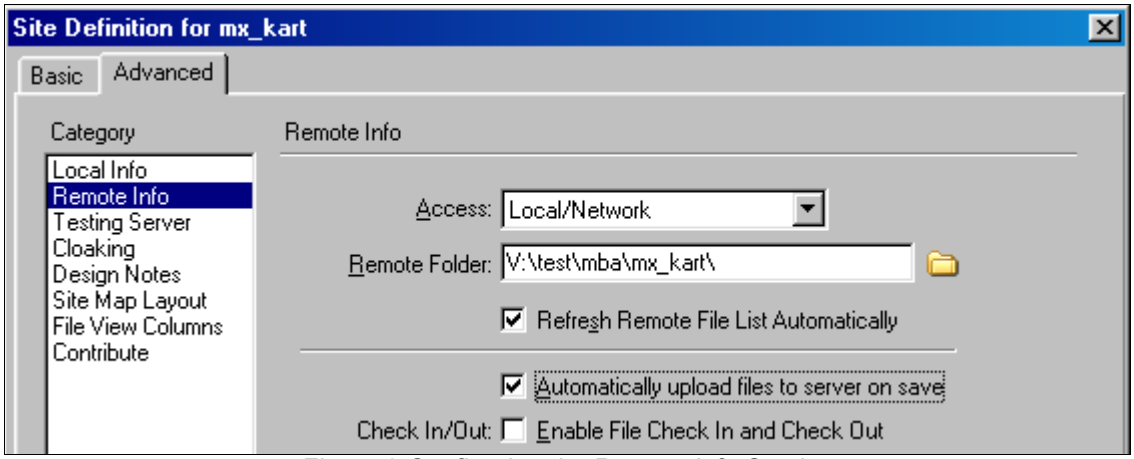


Figure 3 Configuring the Remote Info Section

Step 3: Configuring the Testing Server section

The **Testing Server** section refers to the type of connection and protocol used to connect to a test server. Right now **MX Kart** works with PHP_ADODB and PHP_MySQL only. You must select one of those server models. The HTTP address of the site and the path of the remote site root folder are also required:

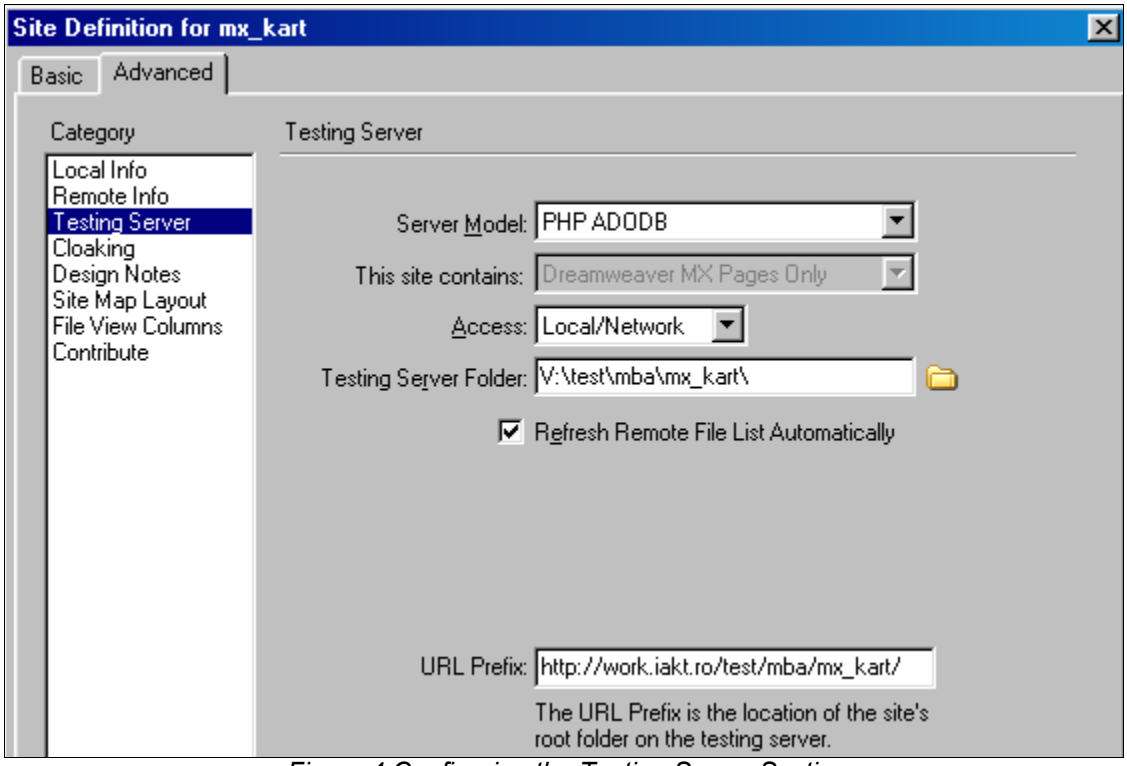


Figure 4 Configuring the Testing Server Section

Setting up a Database Connection

Create the 9 pages in your site root folder by right clicking on the site name in the **Files** panel (*index.php, viewKart.php, prodDetail.php, logout.php, addCoupons.php, step0.php, step1.php, step2.php, step3.php).*

Open the *index.php* page. This is the one that will display the *products_prd* table list. Because this is the first page of a new site no database connection will be available so you will have to create one from the **Databases** tab of the **Application** panel as explained below.

First we will create the database connection which will be called **kart_conn** for our tutorial site.

When using the **PHAkt2** (PHP_ADODB) server model you will create an **ADODB Connection** by using the + button from the **Databases** tab in the **Application** panel.

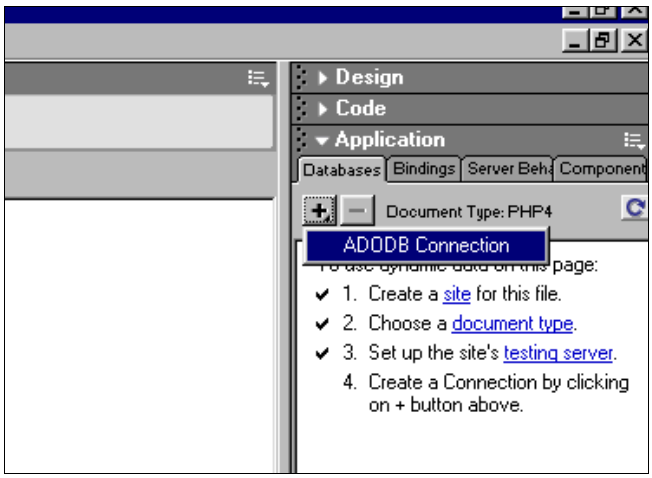


Figure 5 Creating an ADOdb Connection

A similar step is to be made for the **PHP_MySQL** server model, creating a **MySQL Connection**.

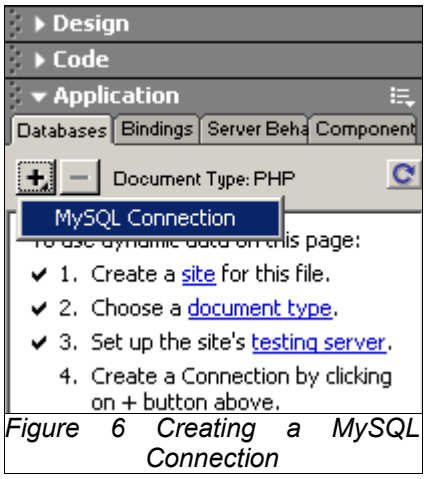


Figure 6 Creating a MySQL Connection

Follow the example below to properly configure your ADOdb server connection.

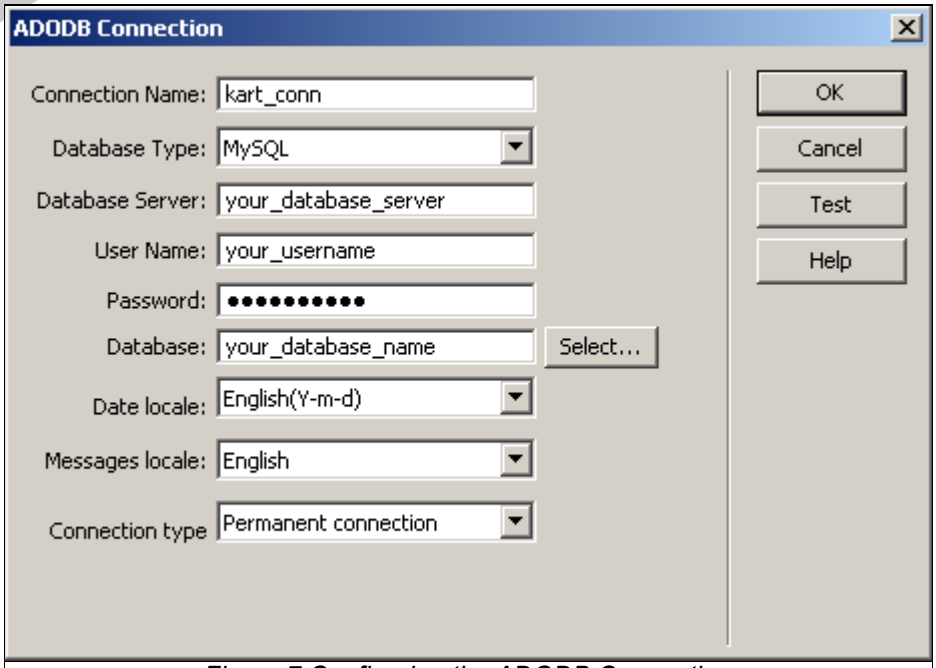


Figure 7 Configuring the ADODB Connection

Follow the example below to properly configure your MySQL server connection.

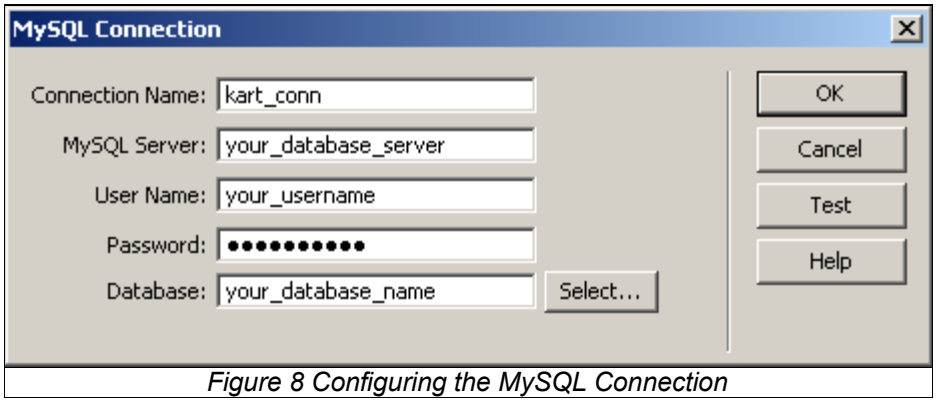


Figure 8 Configuring the MySQL Connection

Please replace the "***Database Server***", "***User Name***" and "***Password***" field values to reflect **your** actual configuration settings. By running the SQL script you can create the necessary tables and then use the **Select** button to choose the newly created database (`mx_shop.sql` in this case).

The MySQL user (the one entered in the ***User Name*** field) must have certain rights as: SELECT, INSERT, UPDATE and DELETE for the data and ALTER, DROP AND CREATE TEMPORARY TABLES for the structure. For more information about setting the users rights for MySQL databases, please visit: http://www.mysql.com/doc/en/User_Account_Management.html

After running the SQL script in your database to create all the necessary tables and creating the site with the new database connection, the next step is to initialize the **MX Kart**.

While the `index.php` page is opened apply the **Update Kart Version** server behavior from the Application panel –> Server Behaviors tab -> MX Kommerce -> Advanced. In the user interface that opens, you must select the MX Kart database conenction defined earlier:

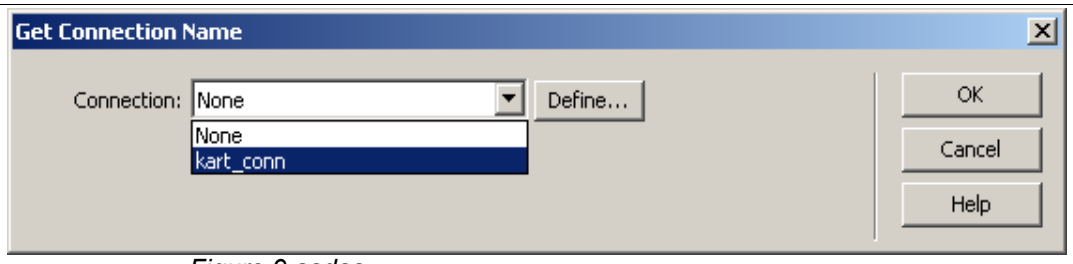


Figure 9 asdas

If you are asked if you are sure you want to put the entire site select Yes. Several required files will be added to the site, in the *MX Kart* and the *includes* folders.. Take special note of these files located in the MXKart directory as they will be needed later in the tutorial and they are the heart of the whole **MX Kart** e-commerce site: addToKart.php, checkOut.php, myDiscounts.inc.php, myShippingRates.inc.php, return.php, myTaxes.inc.php. You can find detailed information about these pages in the MX Kart – User Manual.

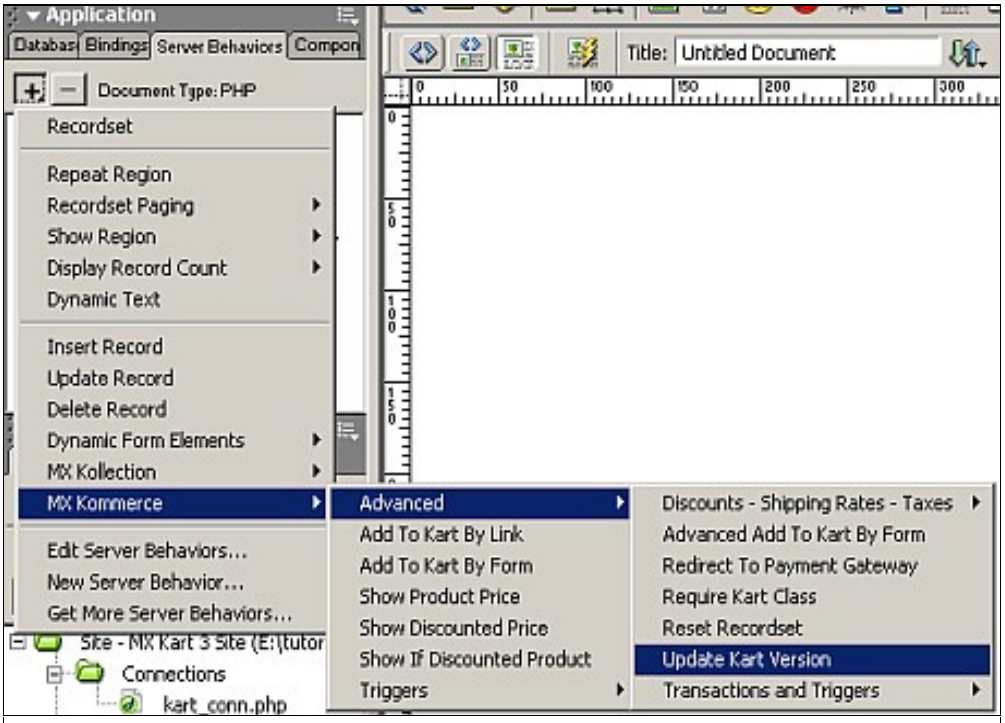


Figure 10 Initializing MX Kart

Caution

After creating the connection to the database **Macromedia Dreamweaver MX** generates a folder named *_mmServerScripts/*. This folder contains scripts that read information about your database and supply it to the Dreamweaver interface. **When publishing your site you should delete this folder from the production server because it allows an attacker to gain unauthorized access to the database.**

Using the MX Kart Dreamweaver MX Extension

Configuring MX Kart

MX Kart has a unified configuration center for the e-commerce website. You will be able to use it to define all the shop properties:

- Define the Orders properties tables and their fields
- Select the dynamic properties tables for the products (for color, weight, size, etc)
- The Kart default and additive fields
- Activate tax and the shipping functions
- Define which discounts will be used at the site (user level, special offers, coupons or volume discounts)
- Select the payment processor and its properties.

The **Kart Properties** command is accessible from the **Insert** Panel **-> MX Kommerce -> Kart Properties** command. A double click on each of the following nodes allows you to configure different cart properties. Once configured, these settings will be applied to the entire site, not just a particular page.

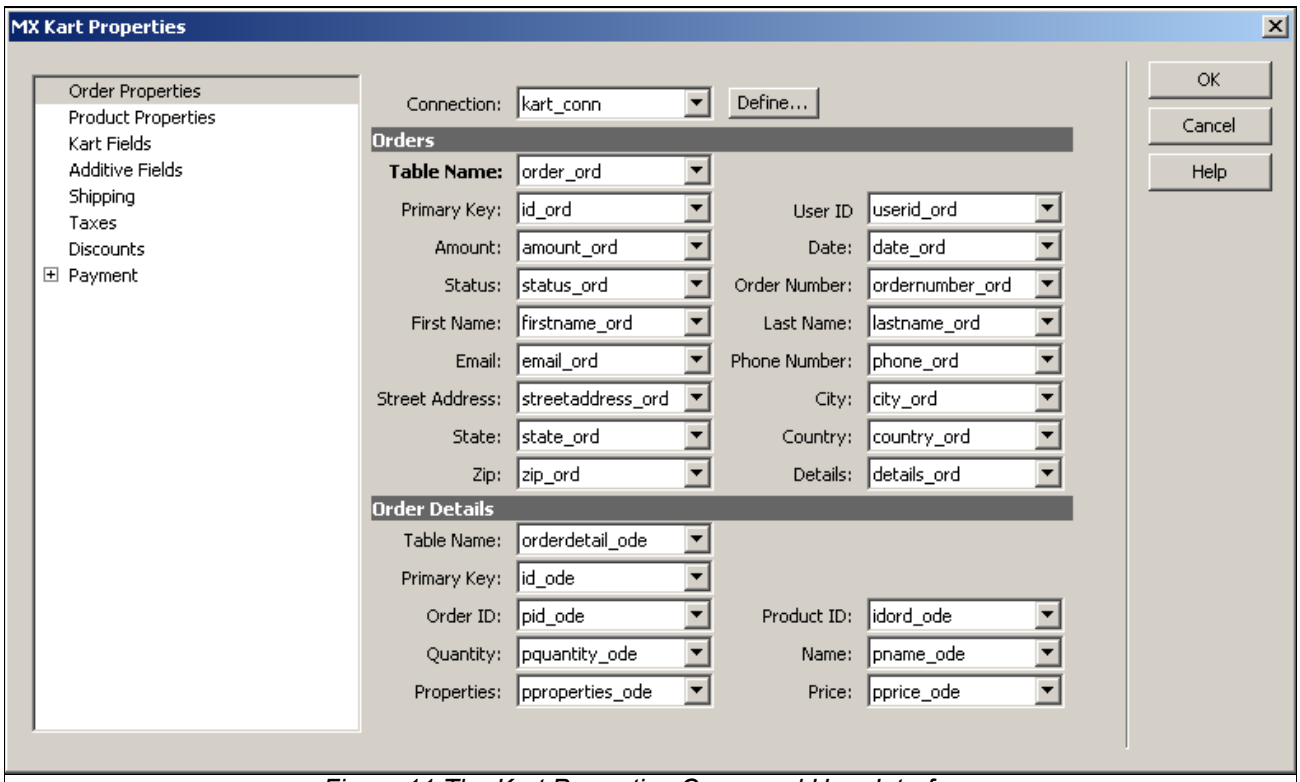


Figure 11 The Kart Properties Command User Interface

By default, the **MX Kart** generated files are already configured correctly for the default database. This allows you to start from a predefined base when creating your site.

We will leave all the configurations as they are now – and start creating the e-commerce site with the default settings.

Creating the Products Page

Creating the Products List

The *index.php* page will display the products list, some links for kart viewing and adding discounting coupons and the login nugget. You should now insert a table (**Insert** panel -> **Tables** tab) having 1 row and 3 columns so that the product list will be in the first column, the additional links in the second and the login nugget is in the third one.

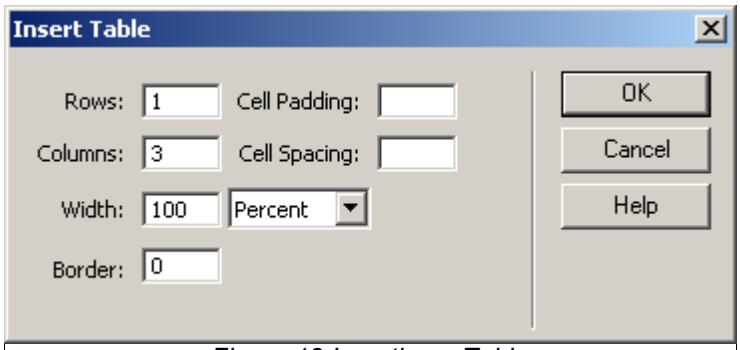


Figure 12 Inserting a Table

To set the table height to 100%, use the tables Properties inspector:

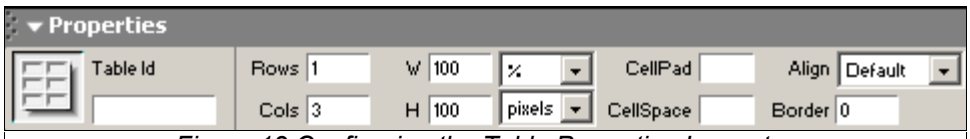


Figure 13 Configuring the Table Properties Inspector

Next, let's create a recordset named *rsProducts*. This recordset will contain all the product names found in the *products_prd* table along with their prices and the categories they belong to (from the *categories_ctg* table).

To create a new recordset go to the **Bindings** tab in the **Application** panel, press the "+" button and select the **Recordset (Query)** option. Select the **kart_conn** database connection. In order to display data from the two tables you must create an advanced recordset. Click on the **Advanced..** button and write the following SQL query:

```
SELECT *
FROM products_prd LEFT JOIN categories_ctg ON products_prd.idctg_prd =
categories_ctg.id_ctg ORDER BY id_ctg ASC
```

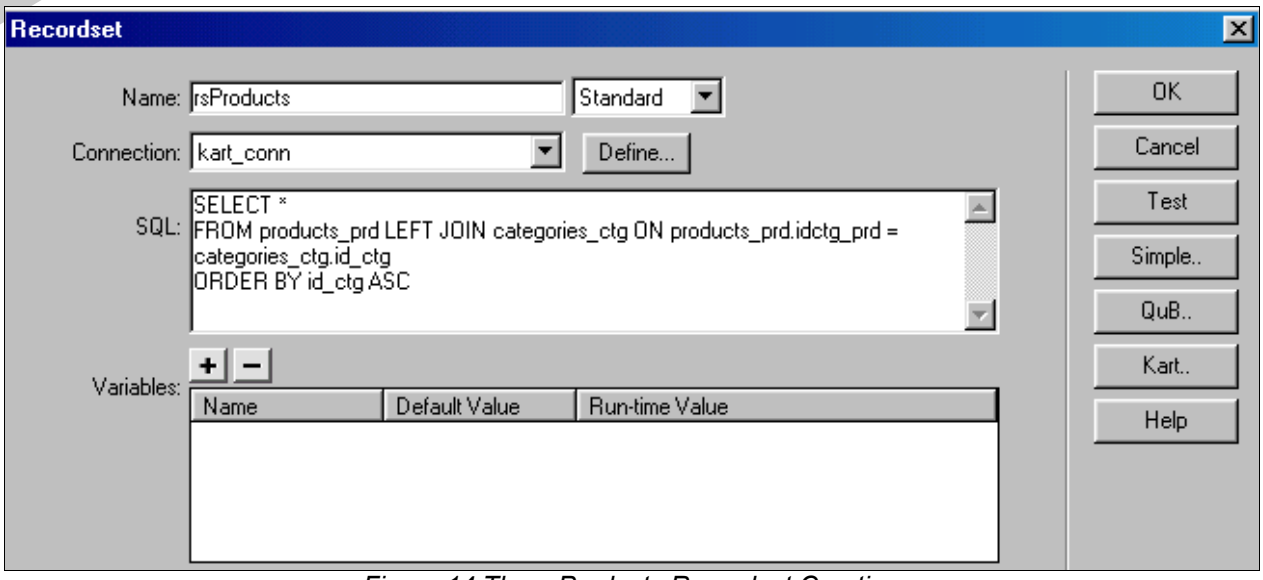


Figure 14 The rsProducts Recordset Creation

Select **OK**.

The next step is to display the product list. In the first column of the main table insert another table (**Insert** panel -> **Tables** tab) with 2 rows and 3 columns. Use the table Properties inspector to set the border to "1" and the alignment to "***Center***".

On the first row write : "Category", "Name" and "Price" (one label per cell). Then go to **Bindings** and drag-and-drop the ***name_ctg*** (into the cell under the Category label), ***name_prd*** (into the cell under the Name label) and ***price_prd*** (into the cell under the Price label) dynamic fields from the *rsProducts* recordset (they have little lightening bolt icons next to them).

For a more friendly display you could configure the first row as header (by checking the ***Header*** box from the cells inspector).

Select the entire second row (select the first cell of the row then shift + click on the last cell of the row) and apply a ***Repeat Region*** server behavior from the **Application** panel->**Server Behaviors** tab. When the dialog box opens, Select ***All records*** in the ***Repeat Region.*** Click **OK**. This way, all the products will be displayed and not just the first one.

If you want to display the products in a more organized and visible manner (grouping them by categories) you can select the ***name_ctg*** field apply a **Show If Field Has Changed** server behavior. Access this behavior from the **Application** panel > **Server Behaviors** tab > Plus (**+**) >**MX Kollection > Conditional Regions**. Configure the user interface to use the *rsProducts* recordset created earlier, and apply it on the ***name_ctg*** field.

This way the category name will be displayed only once for the products belonging to the same category. Be sure to select the correct Field before you click **OK**.

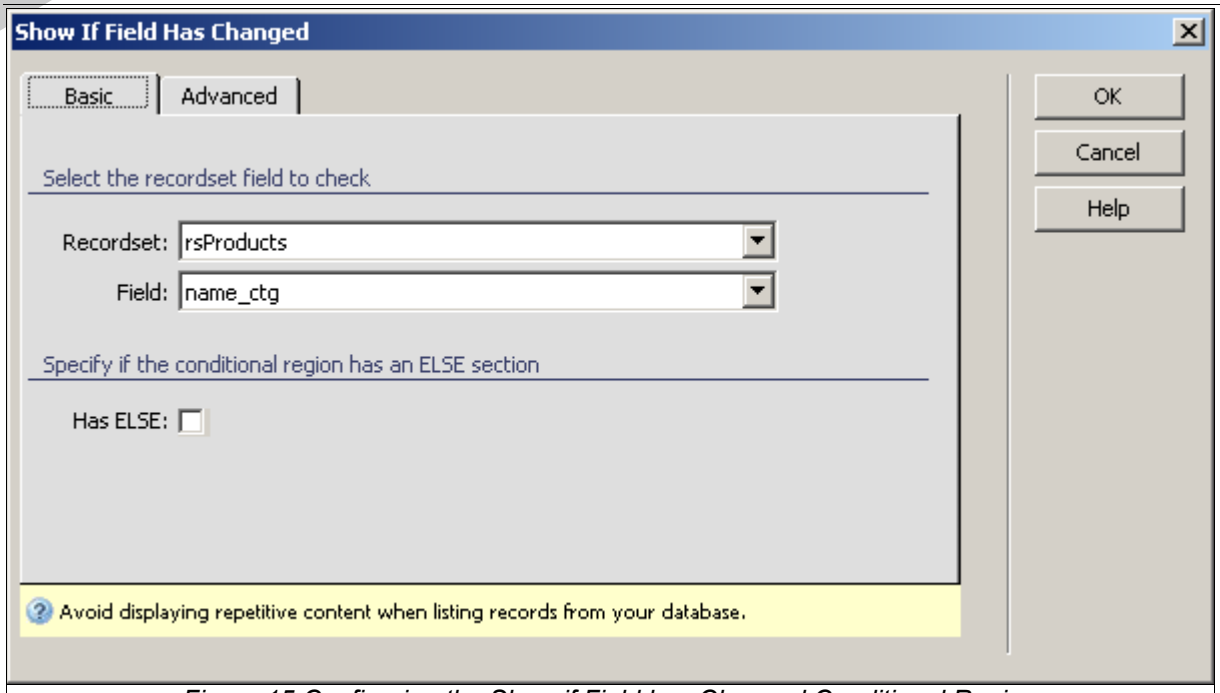
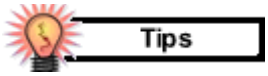


Figure 15 Configuring the Show if Field has Changed Conditional Region

We have designed this tutorial so that we can teach you to excite shoppers about a discounted price by showing the new lower price next to the original higher price with a strikethrough. To do this we will have to duplicate the product price.

Insert the ***price_prd*** dynamic field one more time next to the original price in the third column (it's under the **Bindings** tab).



A more elegant solution is to insert a new table containing 1 row, 2 columns and having border 0. Clear the old ***price_prd*** dynamic field and insert it into each new table's columns.

Select the first price field, right click and choose ***Style/Strikethrough***.

This is how the product list table should look like at this point:

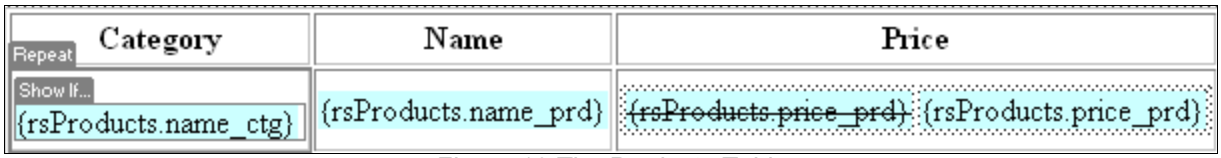


Figure 16 The Products Table

Applying the Show If Discounted Product Server Behavior

The old strikethrough price will be displayed only if the product is discounted. For that you must select this old price field (the one WITH the strikethrough) and apply the **Show If Discounted Product** Server Behavior from the **Application** panel-> **Server Behaviors** tab ->Plus (**+**)**->MX Kommerce**. The ***Product Price*** field will be automatically filled in with the ***Product Primary Key Value.*** now select the product ID field from the *rsProducts* recordset and click **OK**:

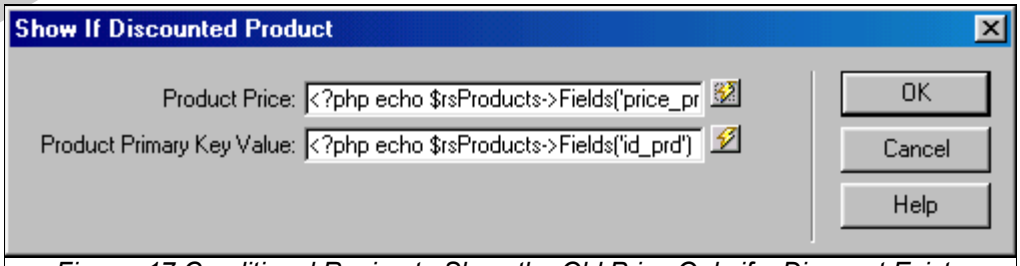
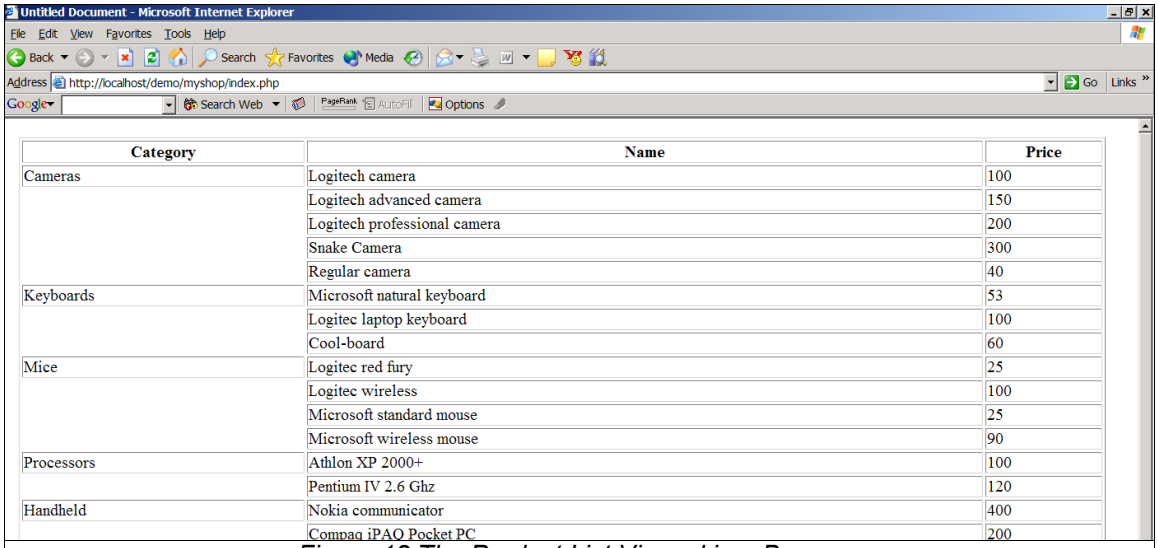


Figure 17 Conditional Region to Show the Old Price Only if a Discount Exists

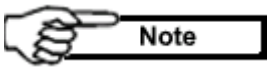
Save the *index.php* file, upload it to the server and test it into the browser by using the F12 key.

Depending on your actual database information, the result should be similar to the one below:



Category	Name	Price
Cameras	Logitech camera	100
	Logitech advanced camera	150
	Logitech professional camera	200
	Snake Camera	300
	Regular camera	40
Keyboards	Microsoft natural keyboard	53
	Logitec laptop keyboard	100
	Cool-board	60
Mice	Logitec red fury	25
	Logitec wireless	100
	Microsoft standard mouse	25
	Microsoft wireless mouse	90
Processors	Athlon XP 2000+	100
	Pentium IV 2.6 Ghz	120
Handheld	Nokia communicator	400
	Compaq iPAQ Pocket PC	200

Figure 18 The Product List Viewed in a Browser



Price column will be displayed without symbols (i.e. "$") now. We will format the symbols later in the tutorial.

Applying the Show Product Price Server Behavior

As you can see the current list presents the product prices in an unformatted way.

In order to format the old price field to display the currency and other formatting rules select it and apply the **Show Product Price** Server Behavior from the **Application** panel-> **Server Behaviors** tab->Plus (**+**)->**MX Kommerce->Show Product Price**. Because you selected the price field it will be automatically inserted into the ***Price*** field of the **Show Product Price** server behavior interface:

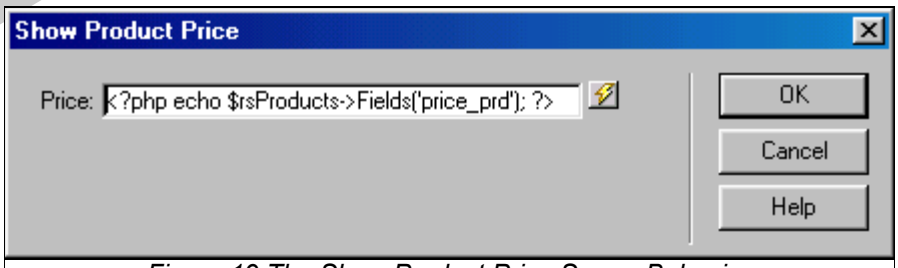


Figure 19 The Show Product Price Server Behavior

Save the `index.php` file, upload it to the server and test it with the browser by using the F12 key.

For now you will not be able to see the regular dollar formatting rules applied on all your visible product price fields because no discount has been yet applied. USD is the default currency in the database provided.

Applying the Add to Kart By Link Server Behavior

Now let's add a new column in the product list table to accommodate the new ***Add to cart*** column. Right click in the price cell and select ***Table***->***Insert Rows or Columns*** – Add one column After the current one. Write "Add to cart" in the new column on the second row ----- and select ***No wrap*** in the property inspector.

Select the "Add to cart" text and apply the **Add to Kart By Link** Server Behavior from the **Application** panel -> **Server Behaviors tab -> Plus (+) -> MX Kommerce -> Add to Kart By Link.**
The next step will be to configure the dynamic bindings (by pressing the "lightning bolt" button and selecting from the recordset the corresponding fields—See image below) for the attributes that will be passed to the Kart recordset.

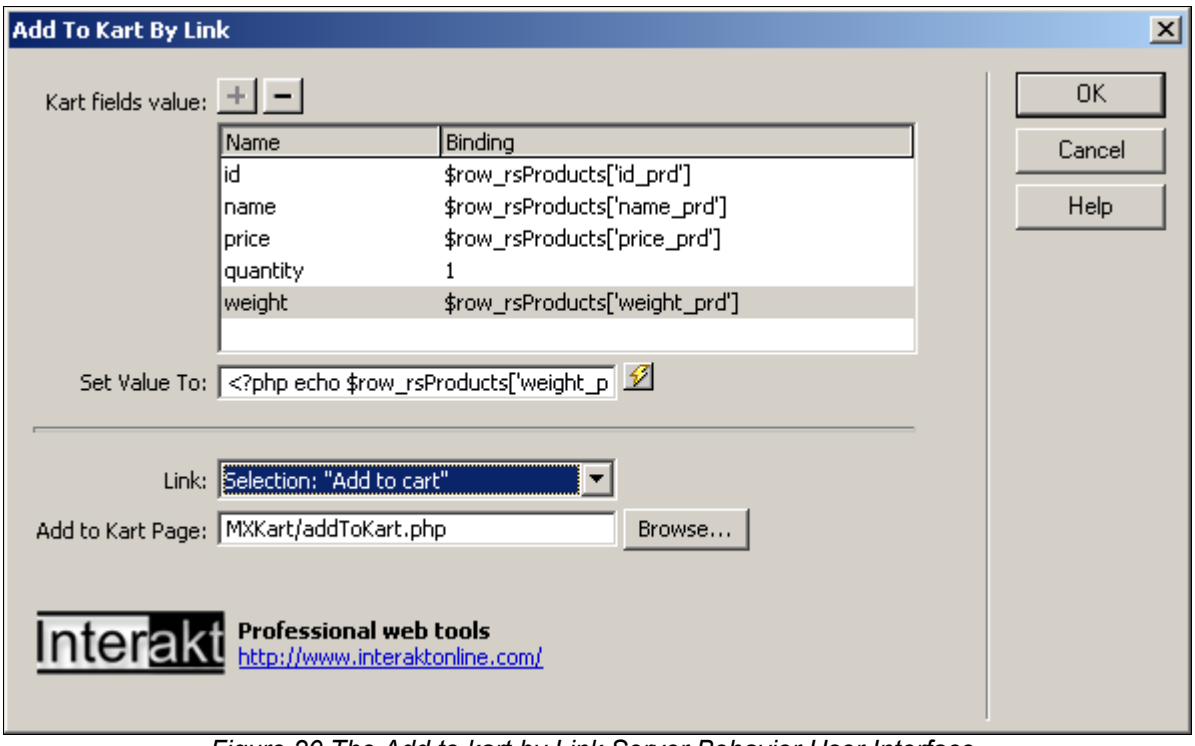


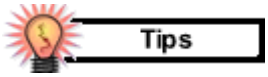
Figure 20 The Add to kart by Link Server Behavior User Interface

The value to be entered for the "***quantity***" parameter should be numerical. This is the quantity of the selected product to be added to the shopping cart when the ***Add to cart*** link is clicked.

In the ***Add to Kart Page*** text-field you should enter the name of the page that actually executes the add to cart transaction. In this case you should not change the page that is already entered – *MXKart/addToKart.php*, since this page behaves correctly and is included with the **MX Kart** kit.

Creating the Login Nugget

We started creating the *index.php* page by inserting a table with three columns. In the first column you should display the products list. The next step is to create a login nugget into the third column.



A nugget is a customizable block of content that can contain content from various sources used to perform specific tasks. It usually appears on the left or right side of the page.

One can use a nugget in order to allow customers to log into the site:

Figure 21 Login Nugget Example

or to display the shopping cart content in a minimal approach:

Figure 22 Shopping Cart Nugget Example

For an easier to read layout you may want to select a background color for the third column. For example a gray shade can be applied by placing the cursor inside the main table third column and by entering the "#CCCCCC" value into the ***Bg*** field from the ***Cell*** properties inspector. Configure the cell properties as shown below:

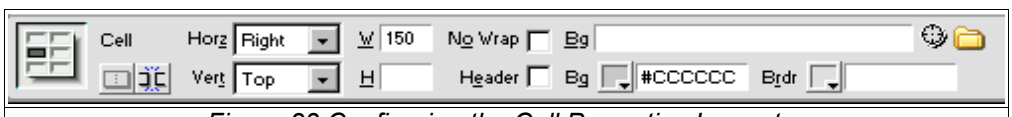


Figure 23 Configuring the Cell Properties Inspector

Now we will insert the login form in the third column of the outside table.

Tab your cursor so that it is in the third column. Set the ***"Vert"*** alignment of the cell to ***"Top"*** in the properties inspector. Since the **MX Kart** extension comes with **ImpAKT** we will use a simple way of creating a login form. Select the **MX Kollection** tab on the Insert panel near the top of your screen. Now apply the **Login Form Wizard** from the **Insert** panel -> **MX Kollection** tab.

The user interface is divided into several steps, each allowing you to set some options.

The first step of the wizard serves mostly to inform about the login setting. These are global for the entire site, and can be configured in the InterAKT Control Panel > Login Settings. As shown in the image below, the dialog box displays the current settings, as well as a button that allows opening the Login Settings dialog box.

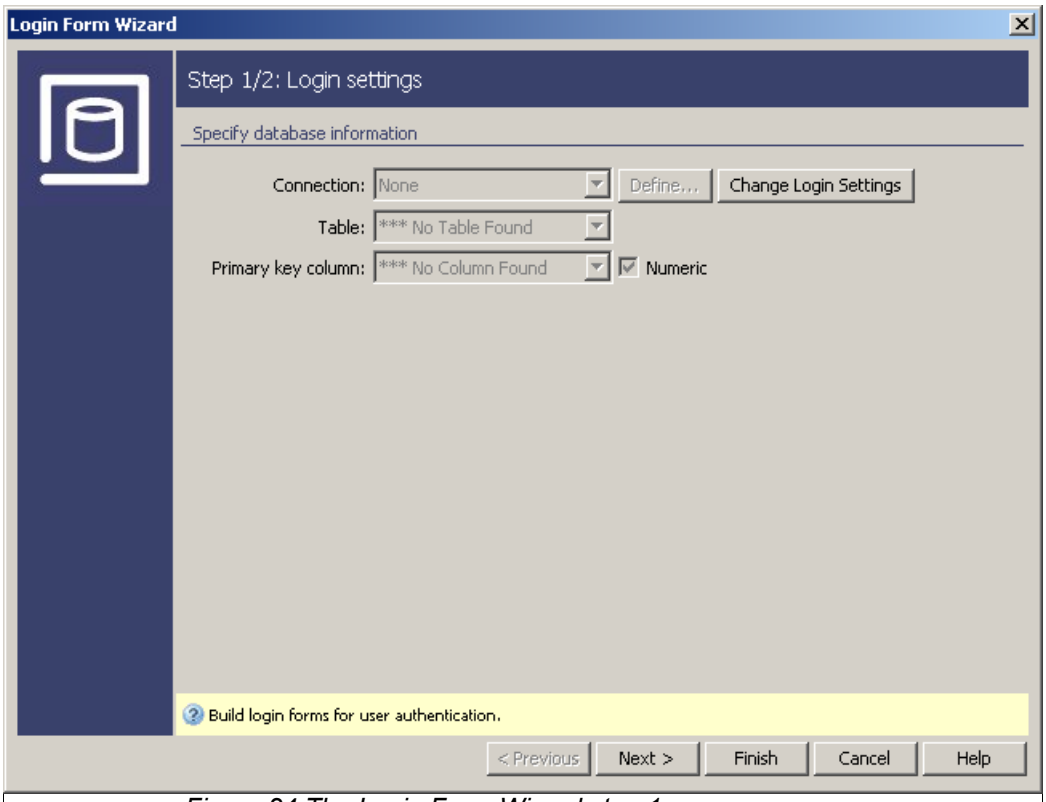


Figure 24 The Login Form Wizard step 1

No settings have been configured yet, so you must click the **Change Login Settings** button. This will open the Login Settings user interface, which is divided into several tabs:

1. The options tab: this is where you will set the main login options:
   - decide if you need password encryption
   - choose what to check at login : username, password and / or user level
   - For how long the auto-login will function.

Configure this tab as shown in the image below, using the username, password and access level for authentication, the standard 30 days auto-login validity and no password encryption.

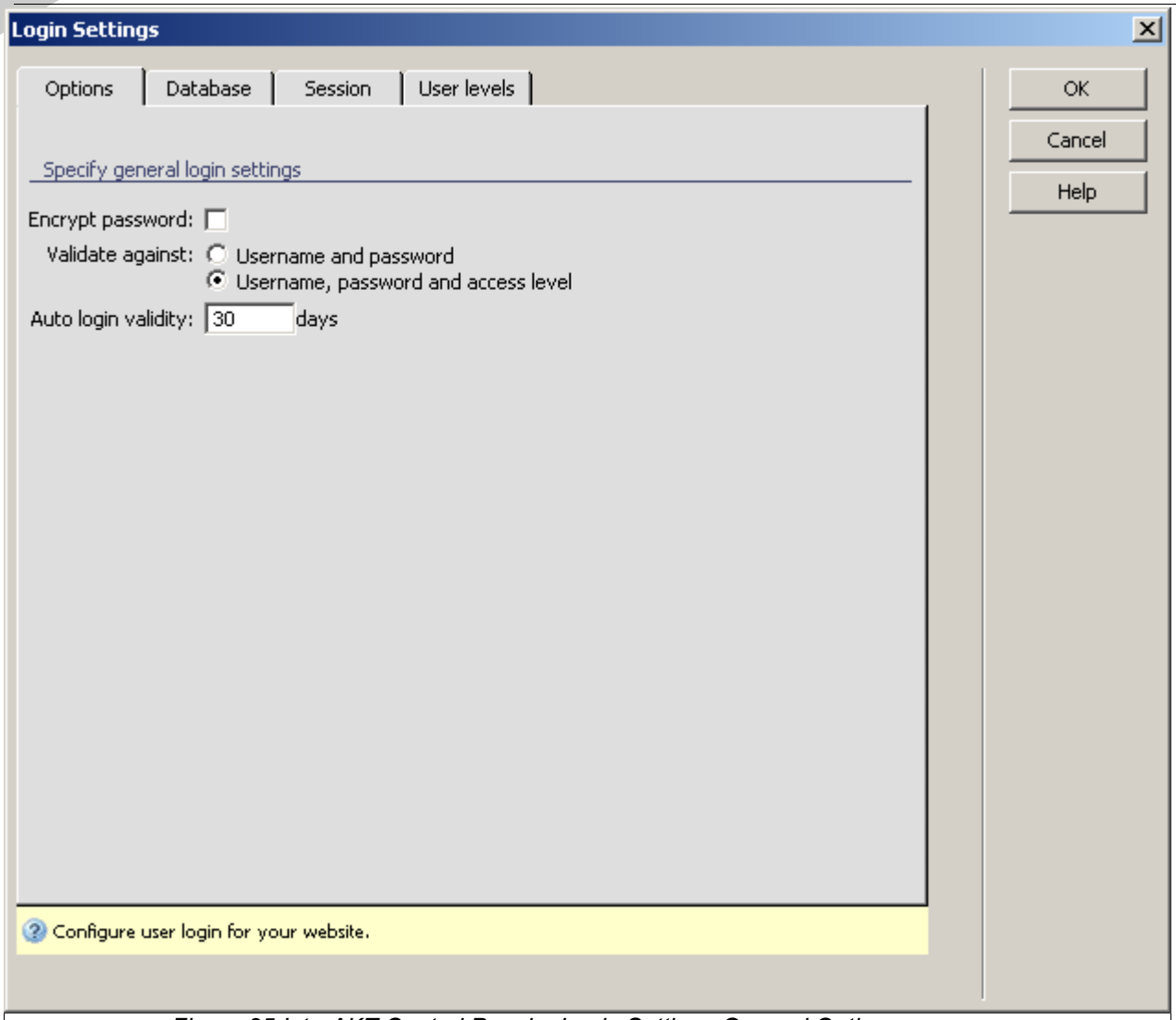


Figure 25 InterAKT Control Panel > Login Settings General Options

2. The database tab: on this tab you set the correlation between the table fields and their role in the login process. Configure the fields to use with the login process based on the user table retrieved through the kart_conn connection. As you can see you can use the ***email_usr*** field for the ***User Name Column*** or if your "users" table has a "username" field, you can use it.

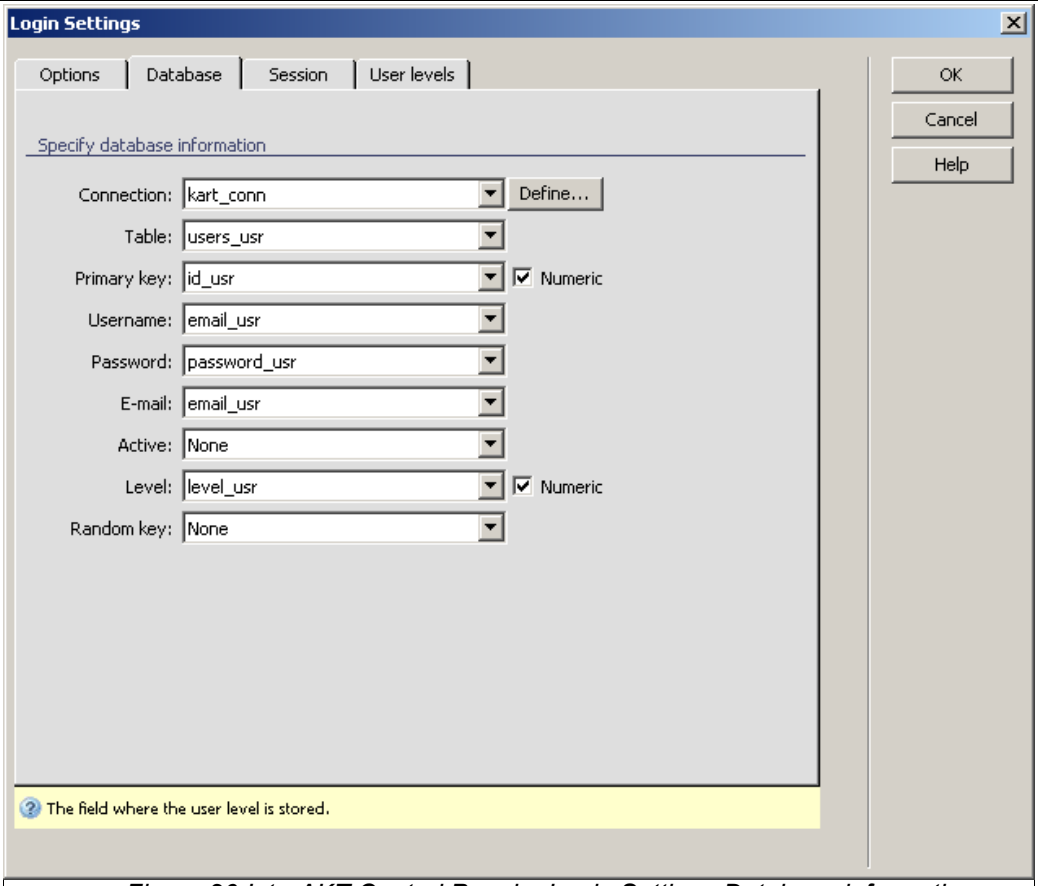


Figure 26 InterAKT Control Panel > Login Settings Database information

3. The Session tab : on this tab you can define additional SESSION variables to be created on login, which can be used later on. By default, the user id and user name (in this case e-mail address) are stored as SESSION variables. Since no additional variables are needed at this point, you can safely skip this step.
4. The User Levels tab: this is an important section of the user interface, as it allows you to define redirect pagess well as user levels. There are two types of redirect pages you can define on this tab:
   - Global redirect pages – these do not depend on a particular user level
   - User level specific redirects – depending on the level, each user can be redirected differently.

Since the login form is placed on the site index, this page will be used as login, redirect on success and redirect on fail page for all levels, and the global redirects. To define a user level you simply have to click the plus (+) button on the grid and define it. Because of the possibility of entering a loop if the page to open for success and failure) especially when using Restrict access to page), the user interface will not allow for the two fields to have the same content (an alert will appear). This is why you must add a bogus URL parameter to one of them (e.g. ?1=1). It has no real meaning, but it is used as workaround.

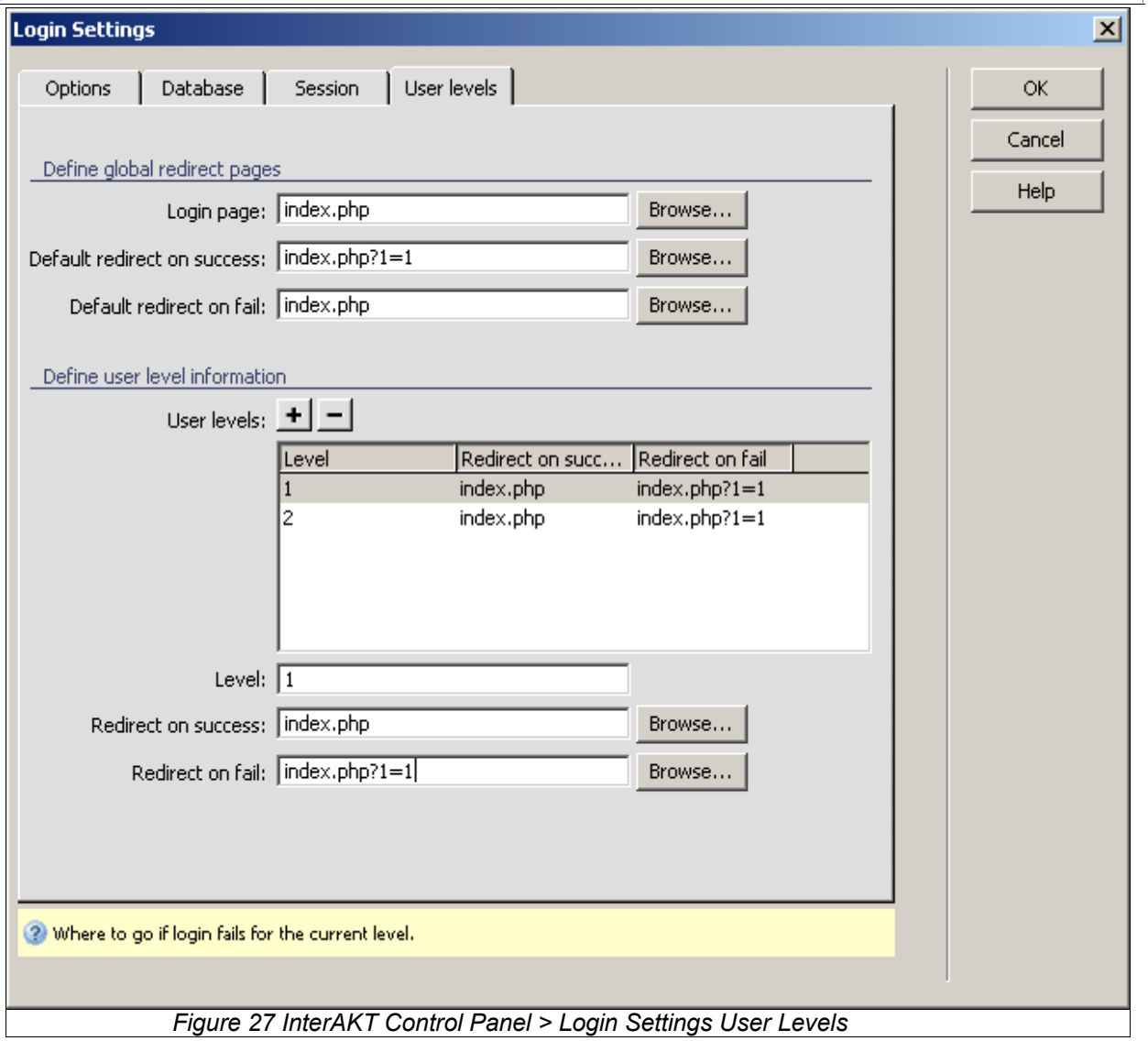


Figure 27 InterAKT Control Panel > Login Settings User Levels

With this last tab, the Login settings are completely defined, and you can click the OK button to close the dialog box and return to the Login Form Wizard. If you are asked if you want to put the entire site, click Yes. This happens because the new Login Settings are being uploaded to the server.

Back in the Login Wizard, the first step's fields are now completed. If the settings are OK, move on to the next and final step. In this step, you can define the following login options:

- If you want a remember me checkbox in the form.
- If you want to generate a forgot password page.

Check both options, and click on Finish to apply the wizard. The wizard will insert both HTML code and server behaviors into the page that perform the entire login operation.

At this point the *index.php* page should look as follows:

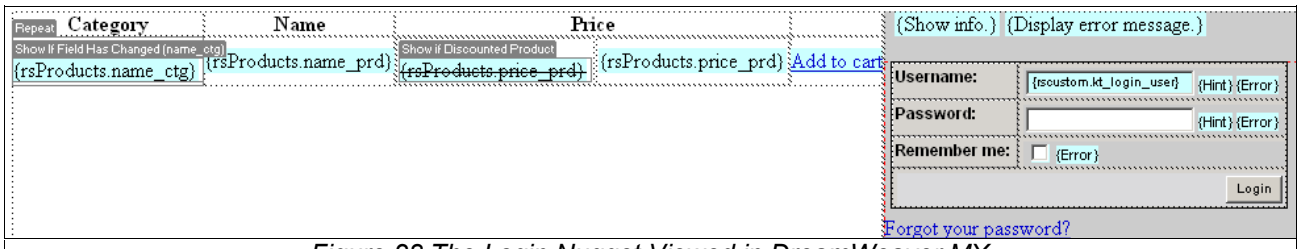


Figure 28 The Login Nugget Viewed in DreamWeaver MX

As you can see creating a login page by using the **Login Form** Wizard is quick and easy.

For test purposes the username and password to access the database provided by **InterAKT** are:

- for a user having the access level set to 1 :
  - username: ***reseller1***
  - password: ***reseller1***
- for a user having the access level set to 2 :
  - username: ***reseller2***
  - password: ***reseller2***

Save the `index.php` page, upload it to the server and test it into the browser by using the F12 key.

You may wish to set some initial values to be displayed in the ***User Name*** and ***Password*** field. For that you must select the corresponding textfields and type the values (from the database) into the ***Init Val*** field in the **Properties** inspector as shown below:

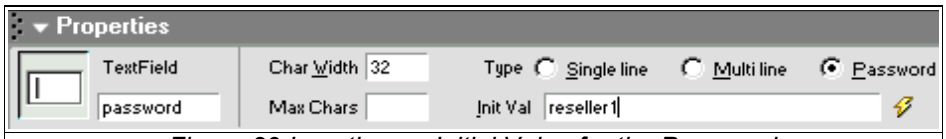


Figure 29 Inserting an Initial Value for the Password

This login form should be shown when the user is not logged in. Otherwise the username and a ***Logout*** link will be displayed.

Therefore you should select the login form (together with all other components — error display triggers, forgot password link - Shift click to select them all on the page) and apply the **Show If User Is Logged In** server behavior from the **Application** panel-> **Server Behaviors** tab->+->**MX Kollection->Conditional Regions->Show If User Is Logged** . Configure it as shown below and **be sure to select the** Has else option.

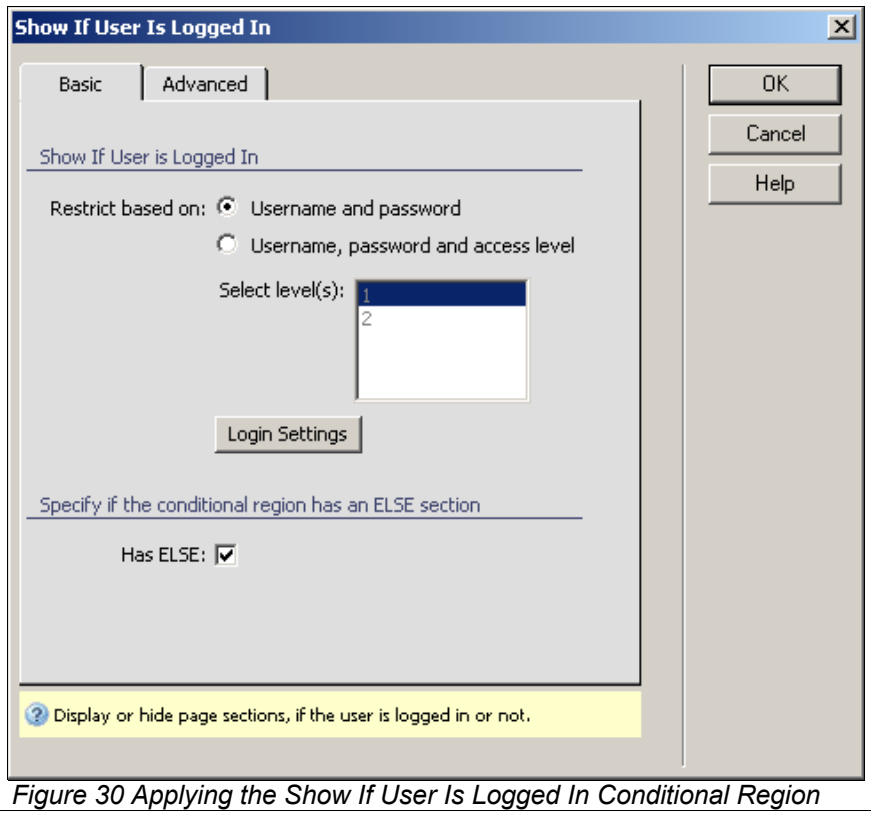


Figure 30 Applying the Show If User Is Logged In Conditional Region

At this point, the server behavior will display the login form only when the user is logged in. But this is exactly the opposite of what you want. With the entire login section selected (the form and error message display triggers – just as before applying the Show If User Is Logged In SB) select Edit -> Cut. This will remove the elements from the If section of the SB. Next select the Else text: Replace this and paste the contents of the clipboard. Now the login form will only appear when the user is not logged in.

.To display the user name and the logout password when the user is logged in, click in the IF section of the SB (if you cannot see it in Design view, switch to code view. It is right before the beginning of the Else section.

Type "**User:**" and drag-and-drop the ***Session*** variable ***kt_login_user*** from the **Application** panel-> **Bindings** tab -> **Session+** next to it. You may wish to bold this text and put a space between it and the dragged and dropped session. Now type the word "**Logout**", select it, right click on it and create a link to the `logout.php` page using **Make link**..

Alternatively, you may choose to use a default Dreamweaver button. Under **Insert-> Forms-> *"Button".*** If you are asked if you want to add a form tag select NO. Now select the new button and under properties label it ***Logout***. Set the action to none. In the **Window** menu, select the **Behaviors** option and the **Tag <body>** panel will appear in the right side. Select "+" in the **Behaviors** tab and then choose ***Go to URL*** . Leave ***Open in*** at Main Window and use the **Browse** button to point to the `logout.php` file.

If you test this page and receive a "`Notice: Undefined index:...`" error don't worry. Continue with the next section and then test again. The error should be resolved.

Creating the Kart Nugget

You can also add a cart nugget on the `index.php` page that will keep the customer posted with minimal information about his shopping cart contents. You may wish to display only the total price with a ***Checkout*** link. For that place the cursor under the ELSE translator (below the Forgot password link) and apply the **Create Shopping Kart View** Command from the **Insert** Panel (near the top of the Dreamweaver interface)->**MX Kommerce** tab. In the displayed interface select the connection you are using then simply remove all the fields (by using the "-" button) except for the ***MXK_SUBTOTAL*** and to uncheck all the boxes except the ***Checkout Link***. Click **OK**. See image below.

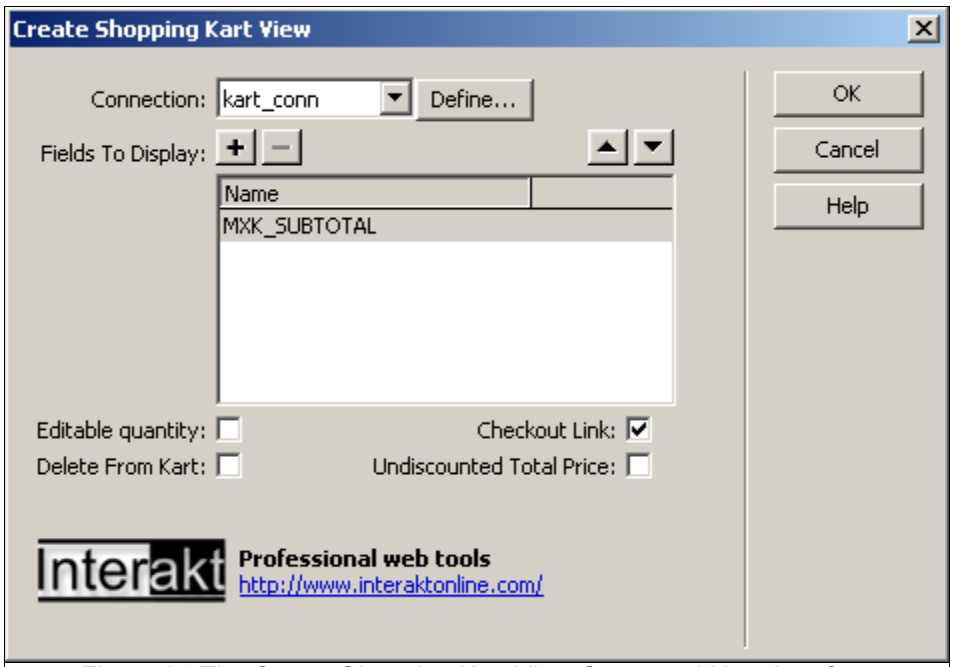


Figure 31 The Create Shopping Kart View Command User Interface

You will notice the **Create Shopping Kart View** Command already includes a conditional region – a **Show if Recordset Is Empty** server behavior that displays the message: "The Cart is empty" if there are no products into the kart session (***KartFV_RS***).

Go in the second table column type “Full cart view” and make a link to the `viewKart.php` page.

By default, the ***Checkout*** link redirects to the `checkOut.php` file from the **MX Kart** folder. But, as the customer should follow some steps before checking out, you should change the file this link is pointing at. Select the ***Checkout*** word and replace the “MXKart/checkOut.php” selection from the ***Link*** field (Properties inspector) with `step0.php`.

Save the `index.php` file, upload it to the server and test it into the browser by using the F12 key.

In the Internet browser, this page should look as follows:

Figure 32 The Product List Page Viewed in a Browser

Creating the Logout Page

Open the `logout.php` page and add a **Logout User** server behavior from the **Application** panel -> **Server Behaviors** -> + ->**MX Kollection** -> **User Login** > **Logout user**. In the displayed interface check the ***Page Loads*** radio button:

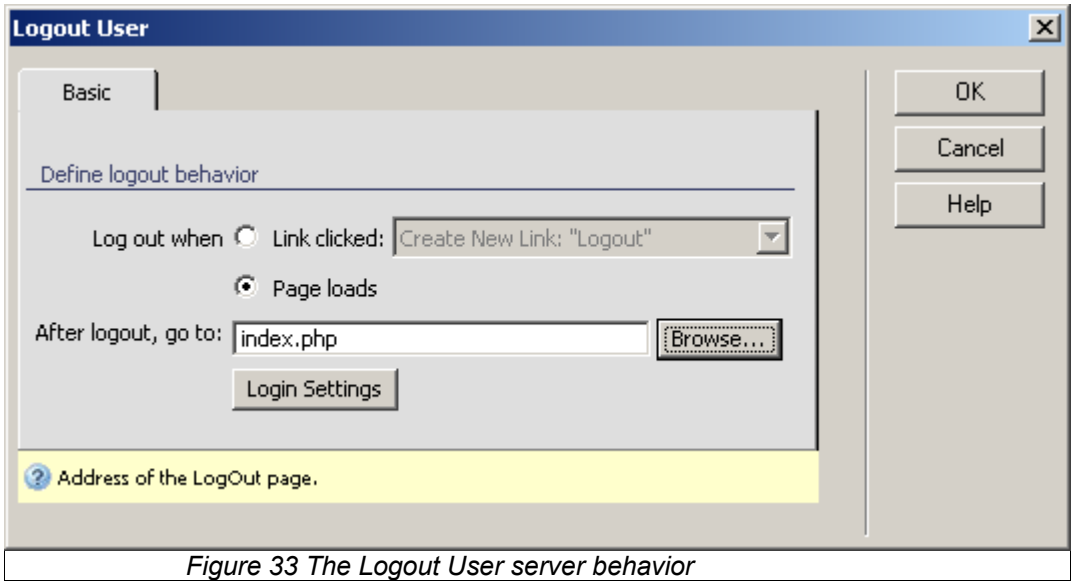


Figure 33 The Logout User server behavior

Now what will happen is after the customer logs out he will be redirected to the `index.php` page.

Creating the View Kart Page

As we have already mentioned the `viewKart.php` page unlike the kart nugget will display the entire contents of the cart not just the total price and ***Checkout*** link.

Applying the Create Shopping Kart View Command

Open the `viewKart.php` page and apply the **Create Shopping Kart View** Command from the **Insert** Panel (near the top of the Dreamweaver interface)->**MX Kommerce** tab.

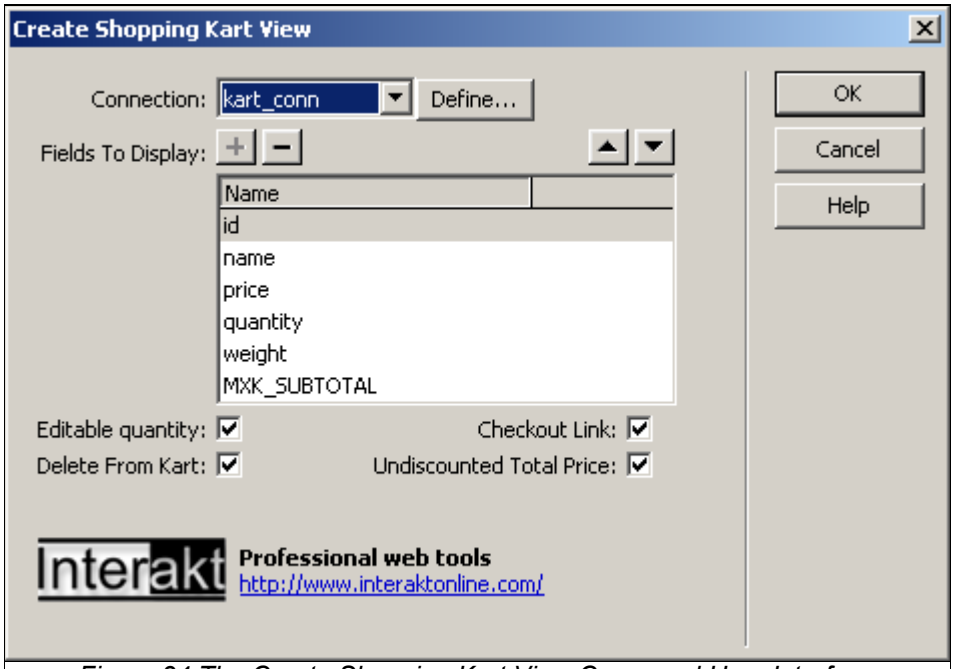


Figure 34 The Create Shopping Kart View Command User Interface

- ***Connection*** – in this drop-down menu select the name of the database connection the transaction is registered to. The **Define** button will open a configuration window allowing you to create another connection.
- a list with the default parameters that are to be stored in the session recordset. You can add or delete parameters by using the "+" or "-" parameters and you can also set the parameters' display order in the cart by using the up/down arrows.
- ***Editable quantity*** – when checked it will make the **Update** button visible, the quantity field editable and it will introduce an update transaction for it.
- ***Checkout Link*** – when checked it will insert a ***Checkout*** link that will start the checkout process.
- ***Delete From Kart***– when checked it will present a ***Delete*** link and a Delete transaction option which will perform the delete operation for the items in the cart .
- ***Undiscounted Total Price***– when checked it will display the regular ("undiscounted") price next to the discounted one.

The code generated by this command is a repeated region that adds dynamic values for the fields in the shopping cart recordset. Besides these values the **Create Shopping Kart View** Command also includes a ***Total Price*** dynamic value.

The Dreamweaver *viewKart.php* page should look similar to the one presented in the image below:

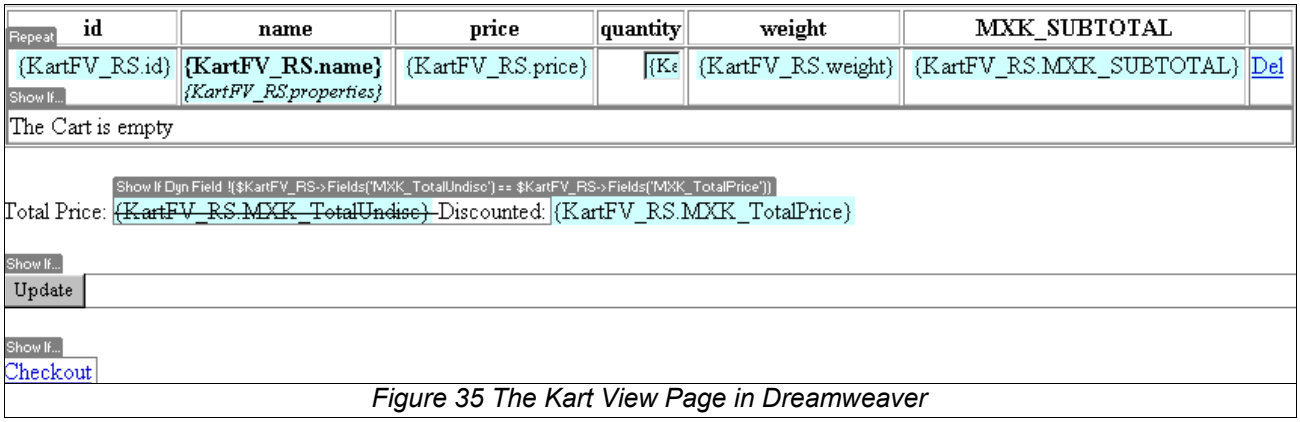


Figure 35 The Kart View Page in Dreamweaver

By default, the ***Checkout*** link redirects to the *MXKart/checkOut.php* file. However since your customer should follow some steps before checking out, you should change this link. Select the ***Checkout*** word and replace the *MXKart/checkOut.php* selection from the ***Link*** field (Properties inspector) with *step0.php*.

Next to the checkout text type ***Shop some more*** and make a link to the *index.php* page.

Save the *viewKart.php* file and upload it to the server. Test it with the browser by using the F12 key.

Creating the Product Detail Page

Configuring the Recordset

Open the *prodDetail.php* page which will allow the customer to select the desired product properties and quantities and then to add the product to cart.

First you will create a new recordset named *rsDetails*. This recordset will contain all the fields from the *products_prd* table for a certain product. If you want to display the corresponding category name (from the *categories_ctg* table) as well then you should create an advanced recordset.

To create a new recordset go to the **Bindings** tab in the **Application** panel and press the "+" button and select the **Recordset (Query)** option.

If the interface is different than the one shown below you are probably viewing the "simple" view. Click on the **Advanced** button. Now configure the recordset as in the picture below. Press the **OK** button: You may wish to copy and paste from the text under the image below.

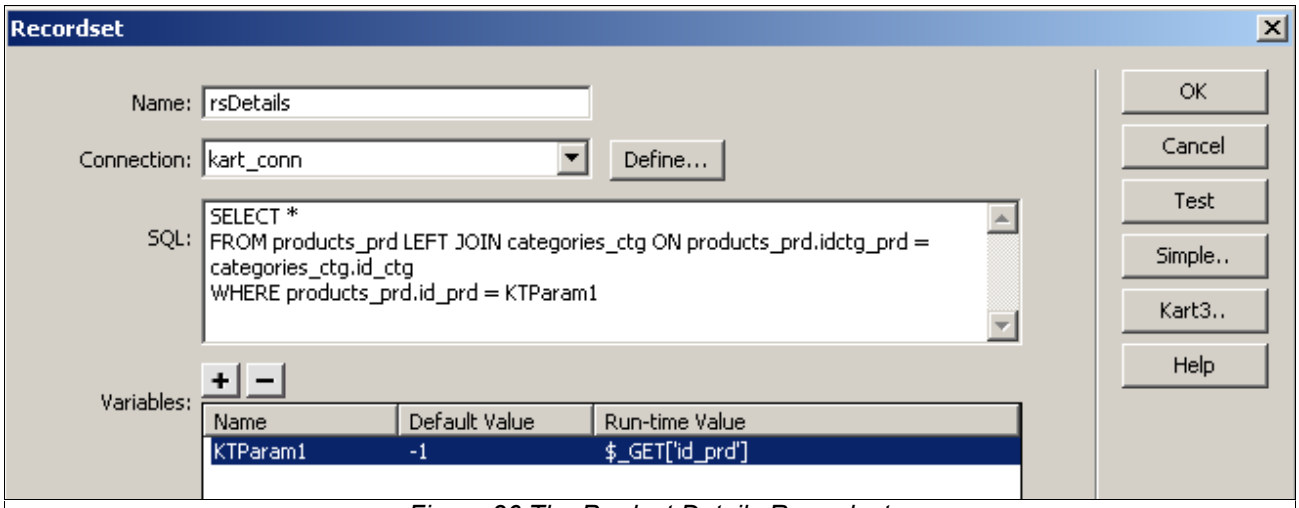


Figure 36 The Product Details Recordset

The SQL query should be the following:

```
SELECT *
FROM products_prd LEFT JOIN categories_ctg ON products_prd.idctg_prd = categories_ctg.id_ctg
WHERE products_prd.id_prd = KTParam1
```

As you can see we created a variable named ***KTParam1*** (you can choose another name) that will pass the ***id_prd*** value obtained by a GET procedure to the SQL. To create this variable press the + in the variables section**.** Use the Parameters:

Name: KTParam1

Default Value: -1

Runtime Value: $_GET['id_prd']

The recordset will be filtered after the ***id_prd*** field. See image below:

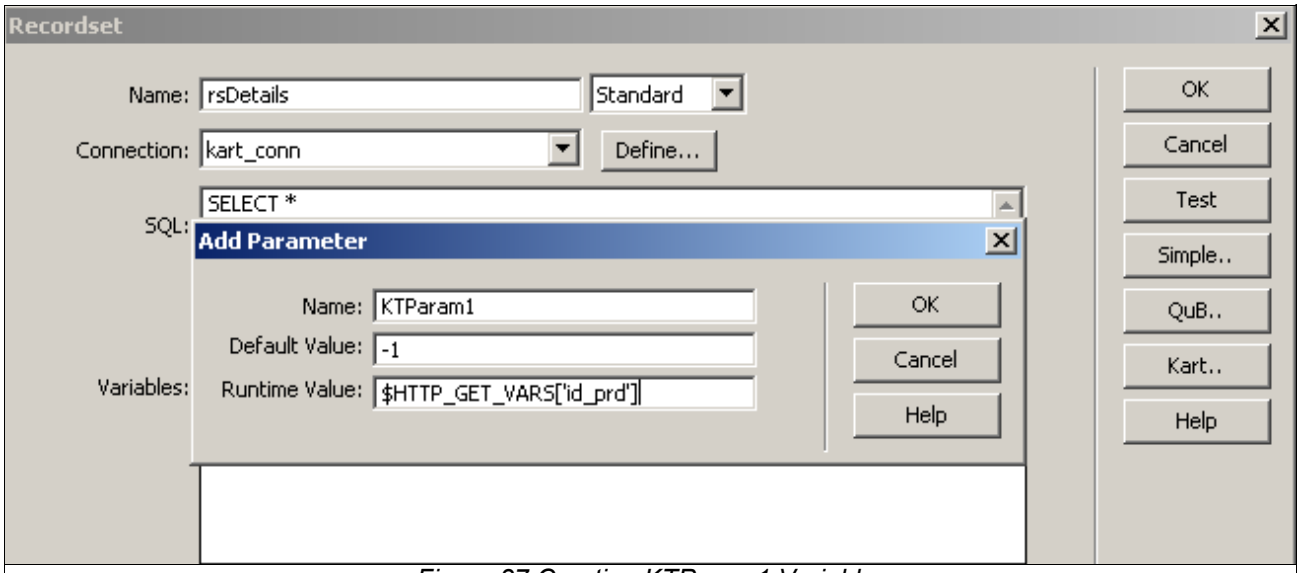


Figure 37 Creating KTParam1 Variable

The next step is to insert a table where all the product details will be displayed. Insert a table having 6 rows and 2 columns. In the first column on each row write: "Category", "Product name", "Description", "Price" and "Options". You could select the entire column and set the ***Horiz*** field to ***Right*** and check the ***Header*** box.

For the first four rows drag-and-drop the corresponding dynamic values from the *rsDetails* recordset in the **Application** panel-> **Bindings** tab. For the ***Price*** row insert two price dynamic values as we did on the `index.php` page. The old price will be displayed with a strikethrough only if the current product is discounted This should be formatted to display currency. To do this you should apply the **Show Product Price** AND **Show If Discounted Product** server behaviors in the same manner as explained above (Figures 14 and 16).

The last row will remain empty for now, as it will contain a link to the product list.

The table should look as in the following image:

Category	{rsDetails.name_ctg}
Product Name	{rsDetails.name_prd}
Description	{rsDetails.description_prd}
Price	Show if Discounted Product ~~{Show price}~~ {rsDetails.price_prd}
Options	

Figure 38 The Product Detail Table

Save the `prodDetail.php` file.

Applying the Add To Kart By Form Command

Below we explain the reasoning behind implementing the **Add To Kart By Form** Command. The reasons for having the **Add To Kart By Link** and **By Form** behaviors are as follows:

- If the customer uses an "Add to Cart" link that was implemented with the **Add To Kart By Link** Server Behavior from the `index.php` page the **Redirect If Properties Not Set** trigger (included on the `addToKart.php` page from the **MX Kart** folder) will check if the selected product has properties.
- If the product has properties the customer will be redirected to the product detail page where he will be able to

choose the product properties and then use the **Add To Kart By Form** Button.

- If the product does not have properties the "Add to Cart" transaction will continue.
- If the customer uses an "Add to Cart" button that was implemented with the **Add To Kart By Form** Command from the *prodDetail.php* page the trigger will add the selected properties and will modify the product price according to the selection. Then the "add to cart" transaction will continue.

In the *prodDetail.php* page click in the cell corresponding to "Options" and apply the **Add To Kart By Form** Command from the **Insert** Panel->**MX Kommerce** tab.

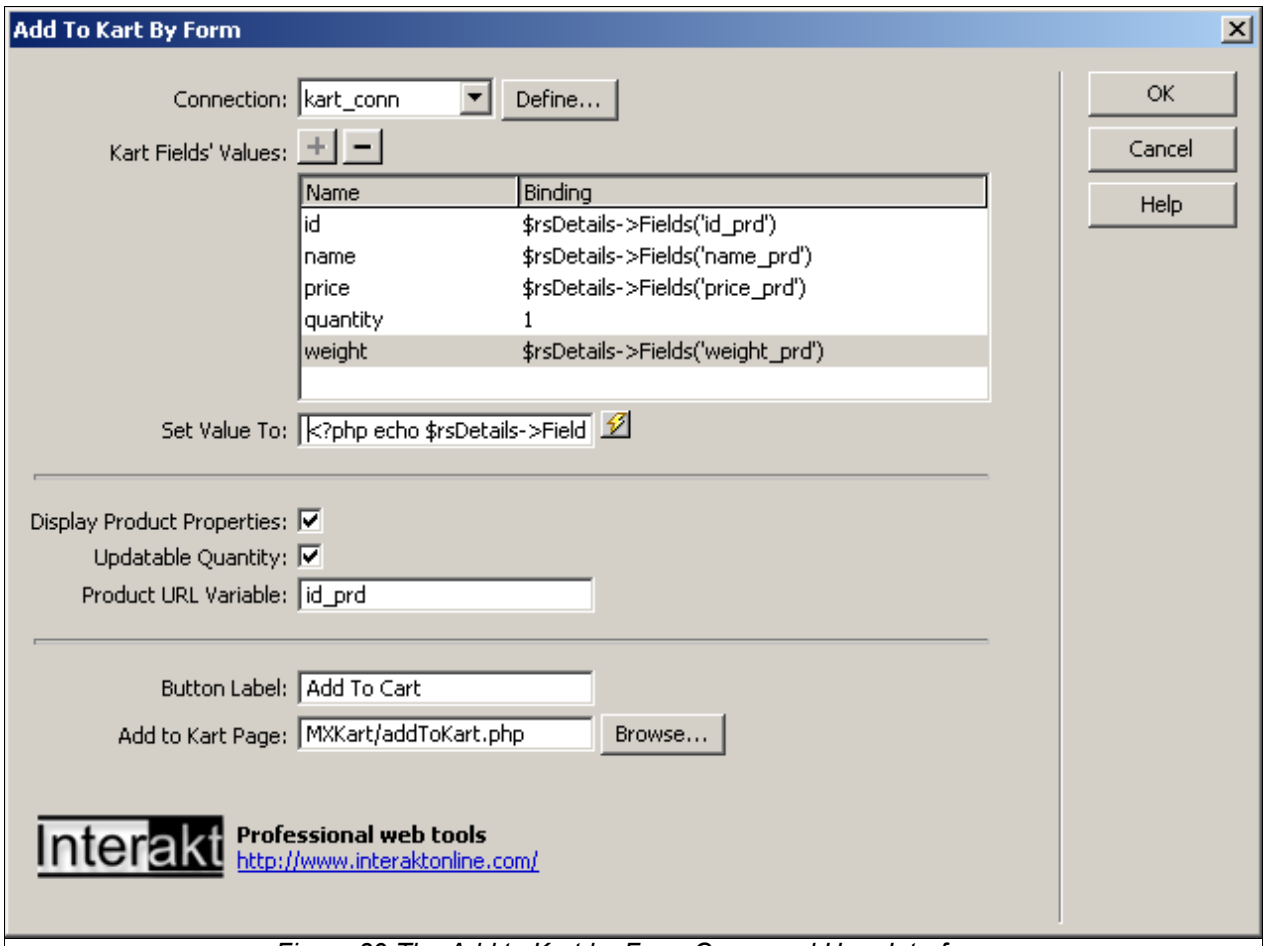


Figure 39 The Add to Kart by Form Command User Interface

- ***Connection*** – select the name of the database connection to which the transaction generated by this command is registered.
- a list with the default parameters that are to be stored in the session recordset
- ***Set Value To*** – this field will allow setting a dynamic value for each parameter. You will simply select the parameter and chose a dynamic value for it by clicking on the "lightning bolt" icon. For the "***quantity***" parameter the value to be entered should be numeric. This will be the default quantity that will appear in the "add to cart" form.
- ***Display Properties*** – when checked this box allows the product options (properties) to be displayed
- ***Updatable Quantity*** – when checked this box enables the ***quantity*** field to be editable
- ***Product URL Variable*** – in this text-field you should enter name of the variable that was sent by URL (the same one that was used as filter when creating the recordset) – in this case, choose ***id_prd***
- ***Button Label*** – in this drop-down menu you should enter the add to cart button label
- ***Add to Kart Page*** – in this text-field you should enter the name of the page that actually executes the add to cart transaction (in this case, you should not change the link that is already entered –

MXKart/addToKart.php). The **Browse** button will facilitate the page selection.

Click **OK**.

On the last row write ***Product list*** and make a link to the *index.php* page.

The Dreamweaver *prodDetail.php* page should look similar to the one in the image below:

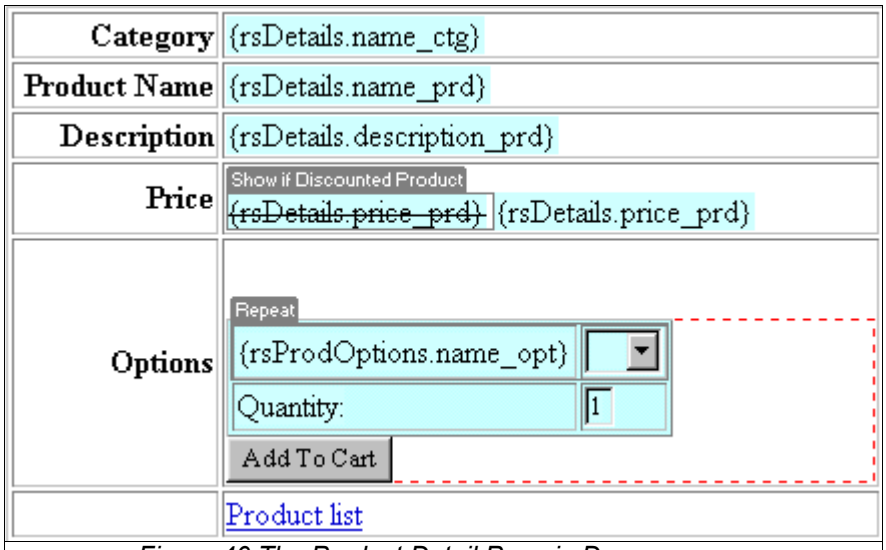


Figure 40 The Product Detail Page in Dreamweaver

Save the *prodDetail.php* file and upload it to the server.

Open the *index.php* page and add another column to the products table. Write "Details" in the newly created column.

For those using the **PHP_ADODB** server model, **s**elect the word and apply a **Go To Detail Page** server behavior from the **Application** panel -> **Server Behaviors** tab as shown below:

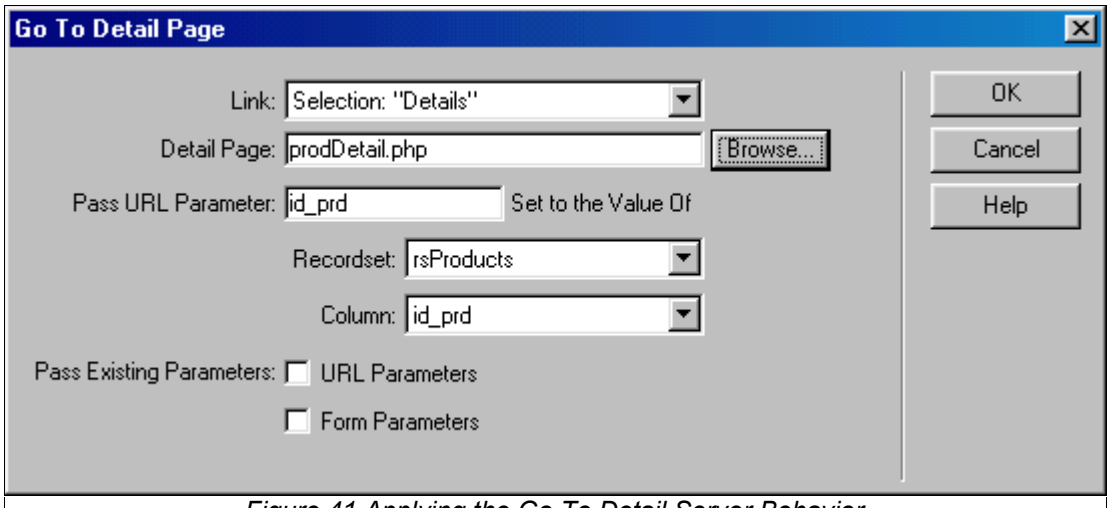


Figure 41 Applying the Go To Detail Server Behavior

For those of you using the PHP_MySQL server model:

1) Select the "Details" text.
2) Make a link to the *prodDetail.php* page by going to the properties inspector and clicking on the ***Browse by folder*** folder icon next to the "Link" field.
3) Select the *prodDetail.php* file.

4) Before you click **OK**, click on the **Parameters** button.
5) Type ***id_prd*** next to lightening bolt icon under ***Name.***
6) Click in ***Value*** row.
7) Click on the lightening bolt icon select ***id_prd*** from the *rsProducts* recordset.

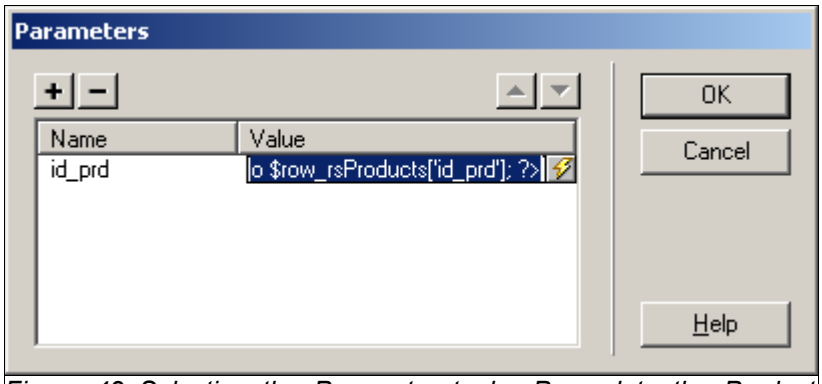


Figure 42 Selecting the Parameter to be Passed to the Product Detail Page

8) Click **OK.**
9) Click **OK** again.

Save the `prodDetail.php` file and upload it to the server.

This way when clicking on the ***Details*** link the customer will be redirected to the `prodDetail.php` page which should look as follows:

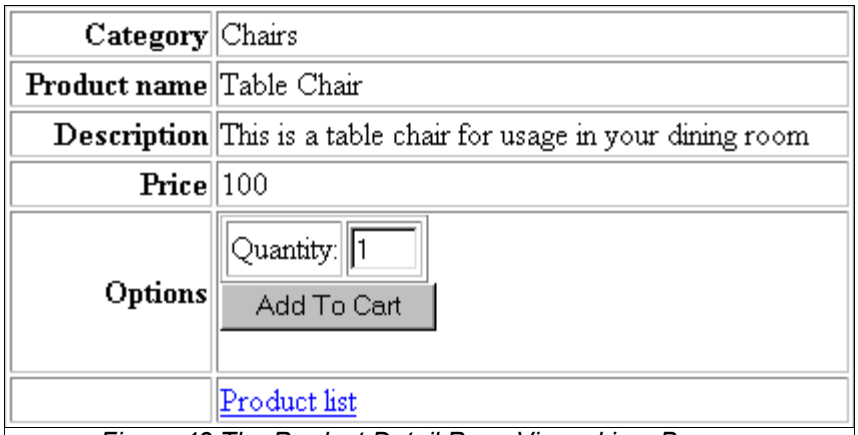


Figure 43 The Product Detail Page Viewed in a Browser

Configuring the addToKart.php page

First an explanation about the `addToKart.php` page included in the **MX Kart** folder. When loaded this file will execute the "Add to Kart" transaction. This transaction will add the selected product to the cart or will redirect the customer to the product detail page if the product has options and have not been chosen yet.

The triggers registered to this transaction are the following:

- **Starter** - the transaction will start if the customer clicks an "Add to Kart" link generated by **MX Kart**
- **Redirect If Properties Not Set** If the customer used an "Add to Kart By Form" button, the trigger will add the selected properties and will modify the product price according to the selection. Then the transaction will continue
- If the customer used "Add to Kart By Link" the trigger will check if the selected product has properties

- If the product has properties the customer will be redirected to the product detail page where he will be able to choose the product properties and then use the Add To Kart By Form Button
- If the product does not have properties the transaction will continue.

◆ **Redirect** - after the product is added to the Kart recordset, the customer is redirected to the page he came from (to the referrer).

Open the `addToKart.php` page (inside the `MXKart` folder) and double click on **Redirect If Properties Not Set** from the server behavior list. In the interface the only change must be made is the one related to the ***Product Page***. In the ***Product Page*** text-field browse to the page where you applied the **Add To Kart By Form** command. In this case you will browse to the `prodDetail.php` page.

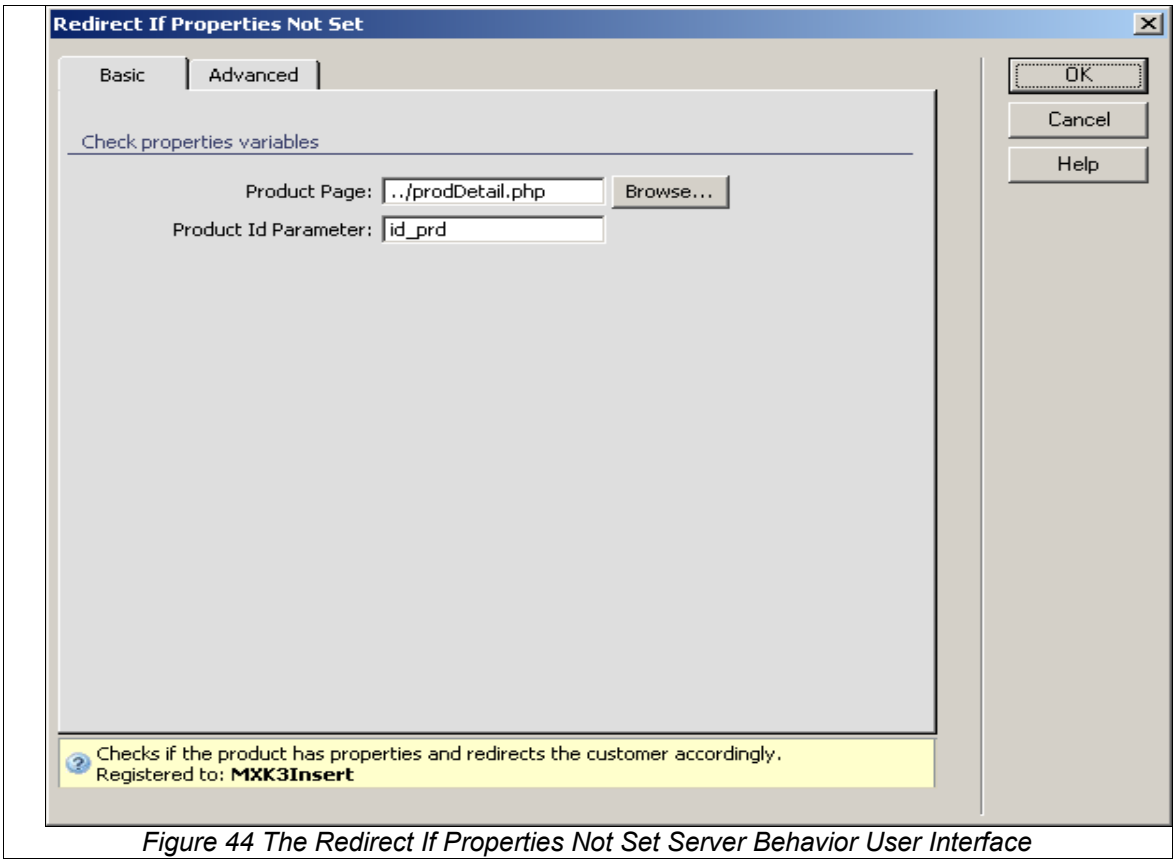


Figure 44 The Redirect If Properties Not Set Server Behavior User Interface

To change the page that will be opened when the product is added to the kart with its options, simply edit the InsertintoKart transaction:

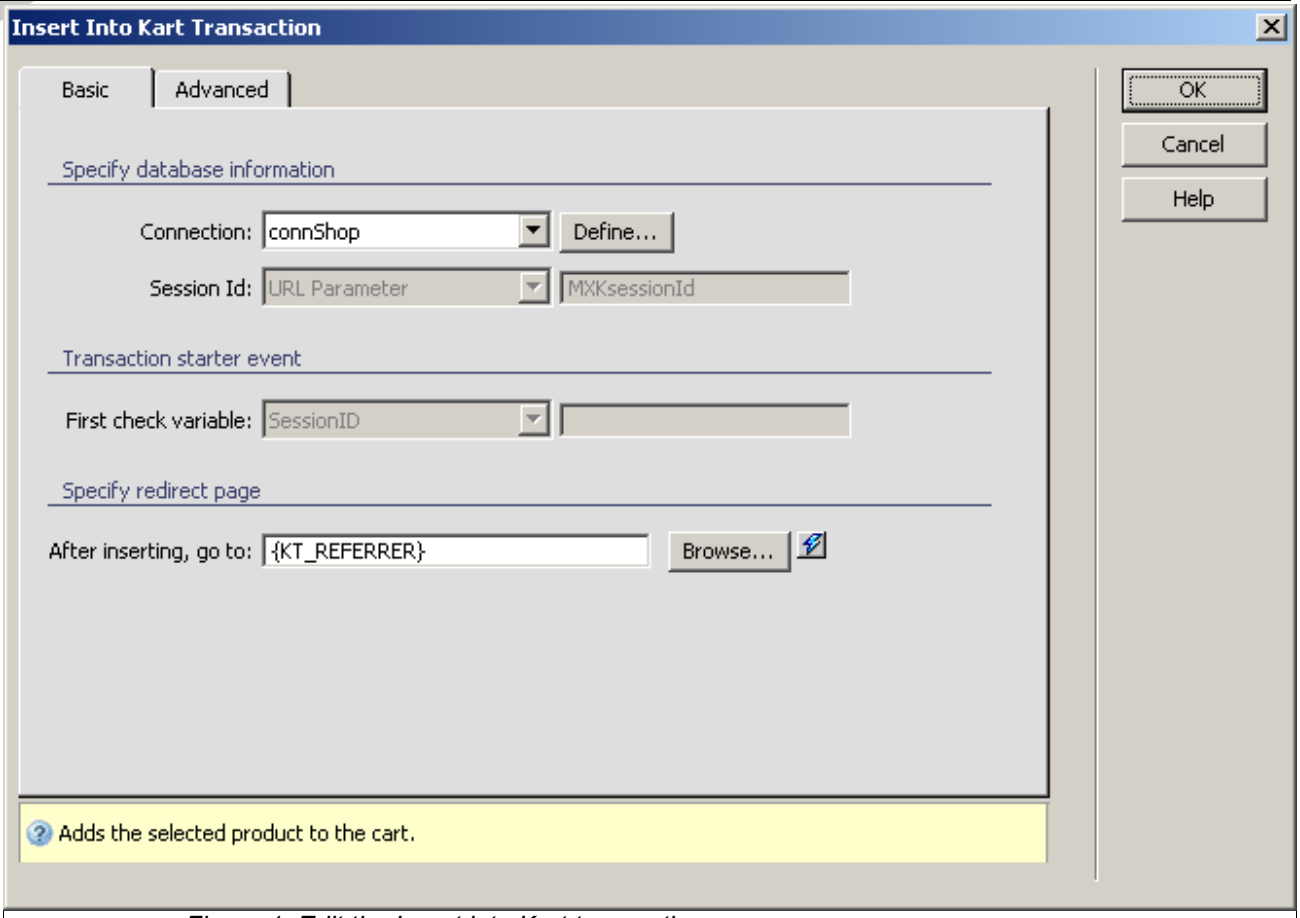


Figure 1: Edit the Insert into Kart transaction

You can change the page your customer is redirected to by browsing to a different page in the ***After inserting, go To*** field. This will be the page loaded after an item is added to the shopping cart. The default redirect is to the page the "add to kart" transaction was applied but it could instead be to the products list page.

Applying the Shipping Rates

When selling physical products over the Internet, shipping is a key component in the sale process. A flexible shipping mechanism can add value to the whole process.

In **MX Kart** there are four types of shipping rates implemented:

- **Shipping Rates per State** – will apply a shipping rate based on the location where the order will be shipped
- **Handling Fee** – will apply a fixed shipping fee to an order
- **Shipping-Method Rate** – will apply different rates for different shipping methods

All the shipping rate server behaviors are already applied on the `MXKart/myShippingRates.inc.php` page.

To apply a shipping rate you need to activate the Shipping Rate by double clicking on the ***Shipping*** entry from the **Kart Properties** command user interface.

You will have the total shipping costs in **Bindings**/***KartRecordset***->***MXK_Shipping***.

The computation method for final shipping costs is very similar to the Discount Engine one. We have individual shipping rate functions that are grouped together to compute the final shipping cost. The grouping method is a functional composition:

$$F_{total} = f_{sn}(...(f_{s2}(f_{s1})))$$

where:

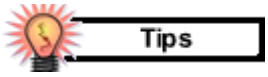
n - number of active shipping rates

s1- the first active shipping rate in the list

sn - the last active shipping rate in the list

When all the shipping rates supported by **MX Kart** are applied, the total shipping costs will be:

$$Ship\,Costs = Handling\,Fee + Total\,Weight * Ship\,Rates\ per\,State + Total\,Weight * Ship\,Method\,Rate$$



If you want to configure the Shipping Rates Server Behaviors for yourself, check the **MX Kart User Manual** for guidelines.

Activating one or more Shipping Rates

You can decide to apply one or more shipping rates.

First open any site page.

In the **Kart Properties** command user interface (the **Insert** pannel -> **MX Kart** tab) double click on ***Shipping*** and select the desired shipping rate component: ***handlingfee*** (for **Shipping - Handling Fee**), ***destinationrate*** (for **Shipping - Rates per State**), ***shiptype*** (for **Shipping-Method Rate**).

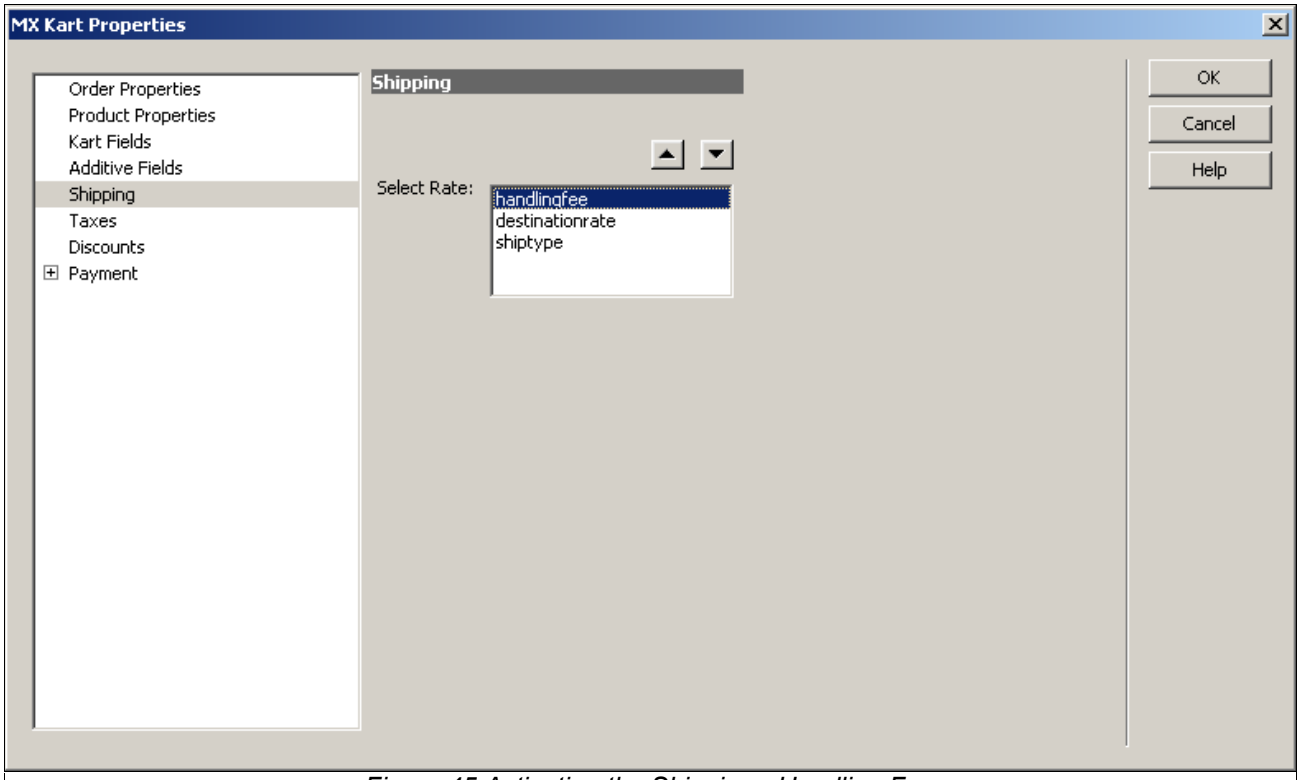


Figure 45 Activating the Shipping - Handling Fee

Use the arrows display in the upper right corner in order to set the processing order.

You can activate all or more shipping rates by keeping the SHIFT or CTRL key pressed while selecting with the mouse.

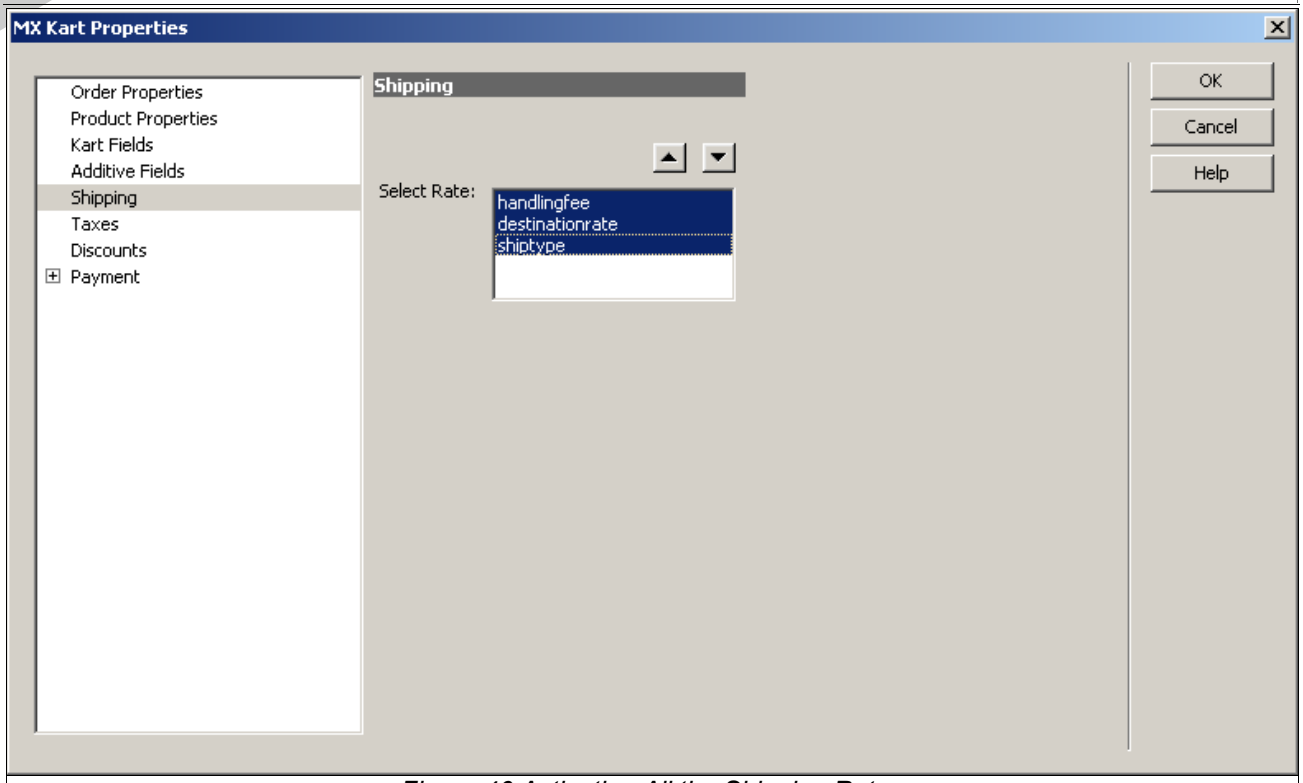


Figure 46 Activating All the Shipping Rates

Click **OK** and save the page.

Applying Taxes

The logic behind the Tax support in **MX Kart** is very similar with the one for discounts and shipping rates. The total tax rate for a specific product is composed (mathematical function composition) from individual tax types:

$$F_{total}(p) = f_{tn}(\ldots f_{t2}(f_{t1}(p)))$$

n - number of active taxes

t1 - the first active tax in the list

tn - the last active tax in the list

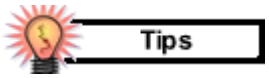
p – product price

Again, the composition order is important.

In **MX Kart** there is only one tax rate function implemented. This rate depends on the tax zone and the product tax category.

Applying Taxes Per Tax-Zone

The **Taxes Per Tax-Zone** server behavior is already applied on the `MxKart/myTaxes.inc.php` page. All you have to do is to activate the **Taxes Per Tax-Zone** server behavior from the **Kart Properties** command user interface by clicking on the ***Taxes*** entry.



However, if you want to configure the **Taxes Per Tax-Zone** Server Behavior yourself, check the **MX Kart User Manual** for guidelines.

Activating the Taxes per Tax-Zone Component

Open any site page.

In the **Kart Properties** command user interface, double click on ***Taxes*** and select the ***zonetax*** component.

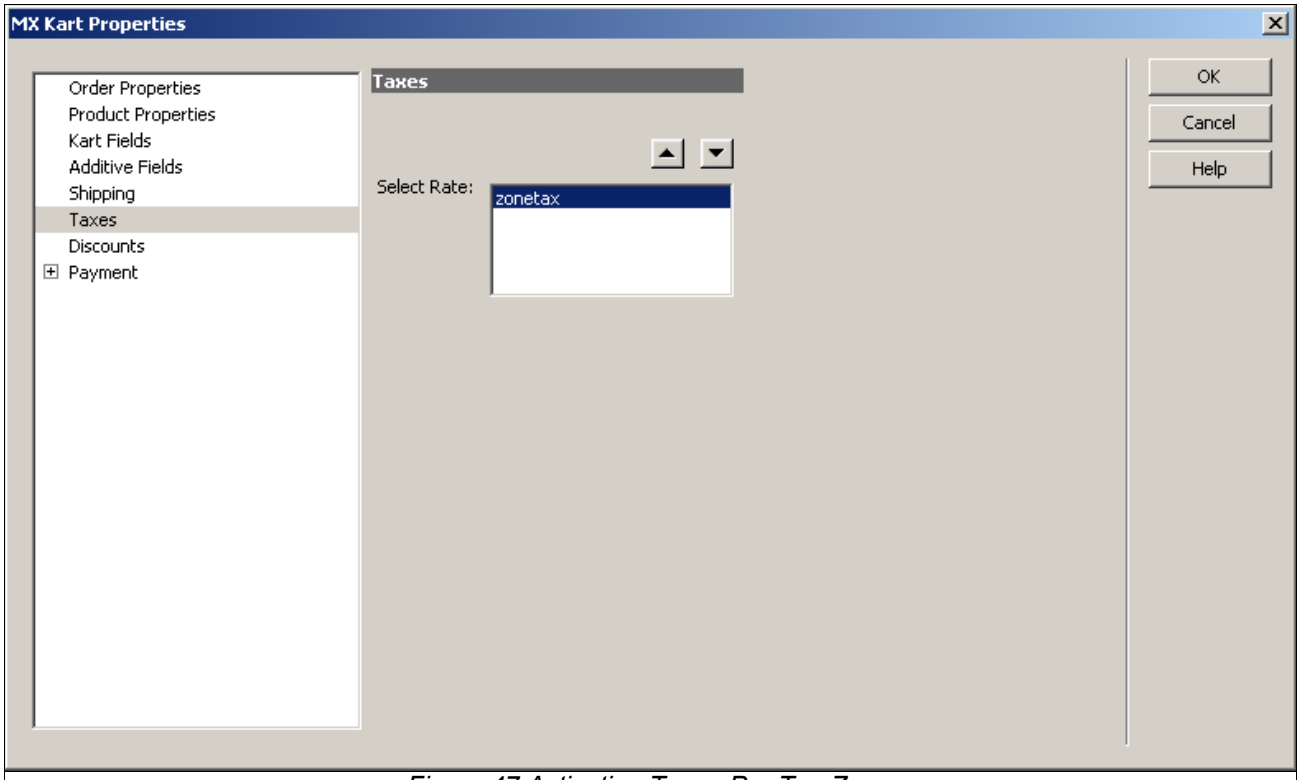


Figure 47 Activating Taxes Per Tax-Zone

Click **OK** and save the page.

Creating the Checkout Pages

The checkout process includes a four step wizard. Only three of the steps will be visible to the customer. On these pages the customer will enter his personal and shipping details and will verify the order totals before checking out through the Payment Gateway.

Creating the step0.php page

The `step0.php` page will not be visible to the customer. On this page, you will apply a **Save Kart To Database** trigger that will insert into the *order_ord* table a record that contains all the kart fields. In the `step1.php` and `step2.php` pages the customer will add the other data to fill in this record.

Applying the Save Kart To Database Trigger

Open the `step0.php` page and apply an empty **Custom Transaction** from the **Application** panel -> **Server Behaviors ->MX Kollection** -> **Forms -> Advanced -> Custom Transaction.** The custom transaction is needed because a trigger must register to a transaction, even an empty one.

In the custom transaction user interface, select the kart_conn database connection in the drop-down menu. The custom transaction will not perform any operation on the tables. The connection is needed for error handling purposes. In order to execute the transaction as soon as the page loads, and not wait for the user to press a button, in the First check variable drop-down menu select the Entered value method of passing the starting command. For the reference (the text-field located next to the drop-down menu) enter 1.

In the same interface of the Custom transaction you can define the page to redirect to when its execution is completed. This will be used instead of a redirect trigger, to forward the customer to step2 of the checkout wizard. Click the Browse button and select the step1.php page.

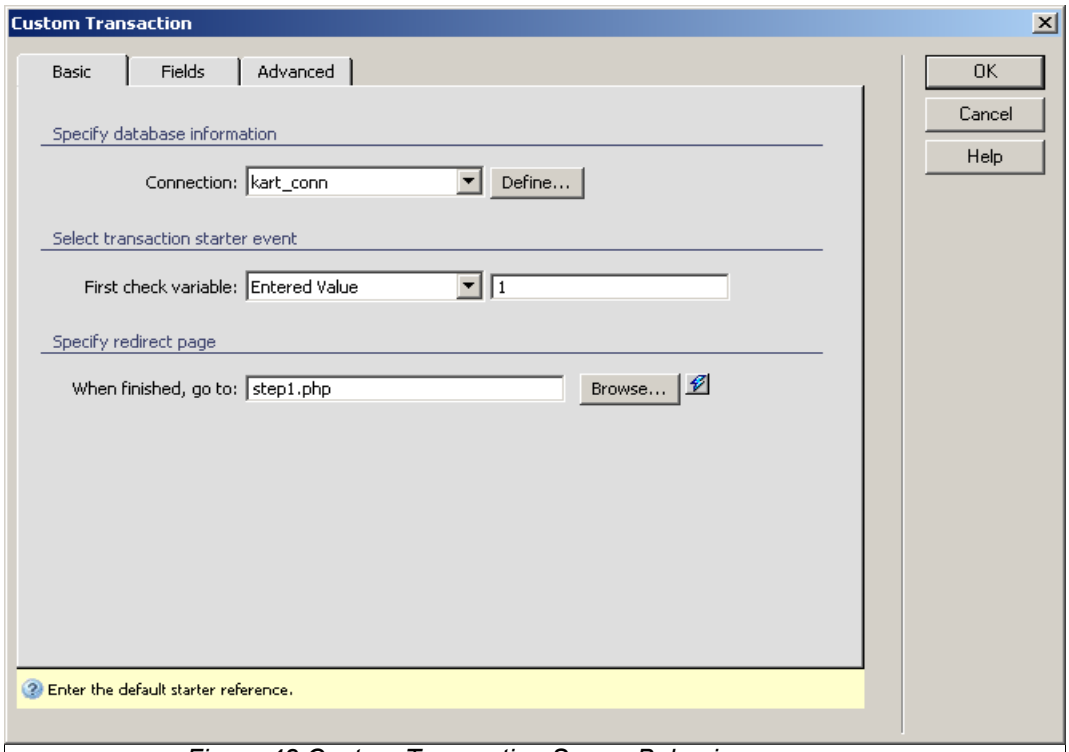


Figure 48 Custom Transaction Server Behavior

Press the OK button to close the dialog box. Once the Custom transaction has been applied to the page, it is time to add the triggers that will perform the various task. All of these triggers will register to the Custom transaction in an automatic manner.

First, the Save Kart To Database trigger must be applied. It will insert into the order_ord table a new record, containing all of the cart elements. This way, all orders get saved to your database table, and can be used later on.

This trigger can be applied by accessing the **Application** Panel->**Server Behaviors**->+->**MX Kommerce**->**Triggers**->**Save Kart To Database trigger**. The trigger auto-registered to the custom transaction on page. Its type is BEFORE, and its priority 1, which means that it will be the first one executed on page (after the STARTER).

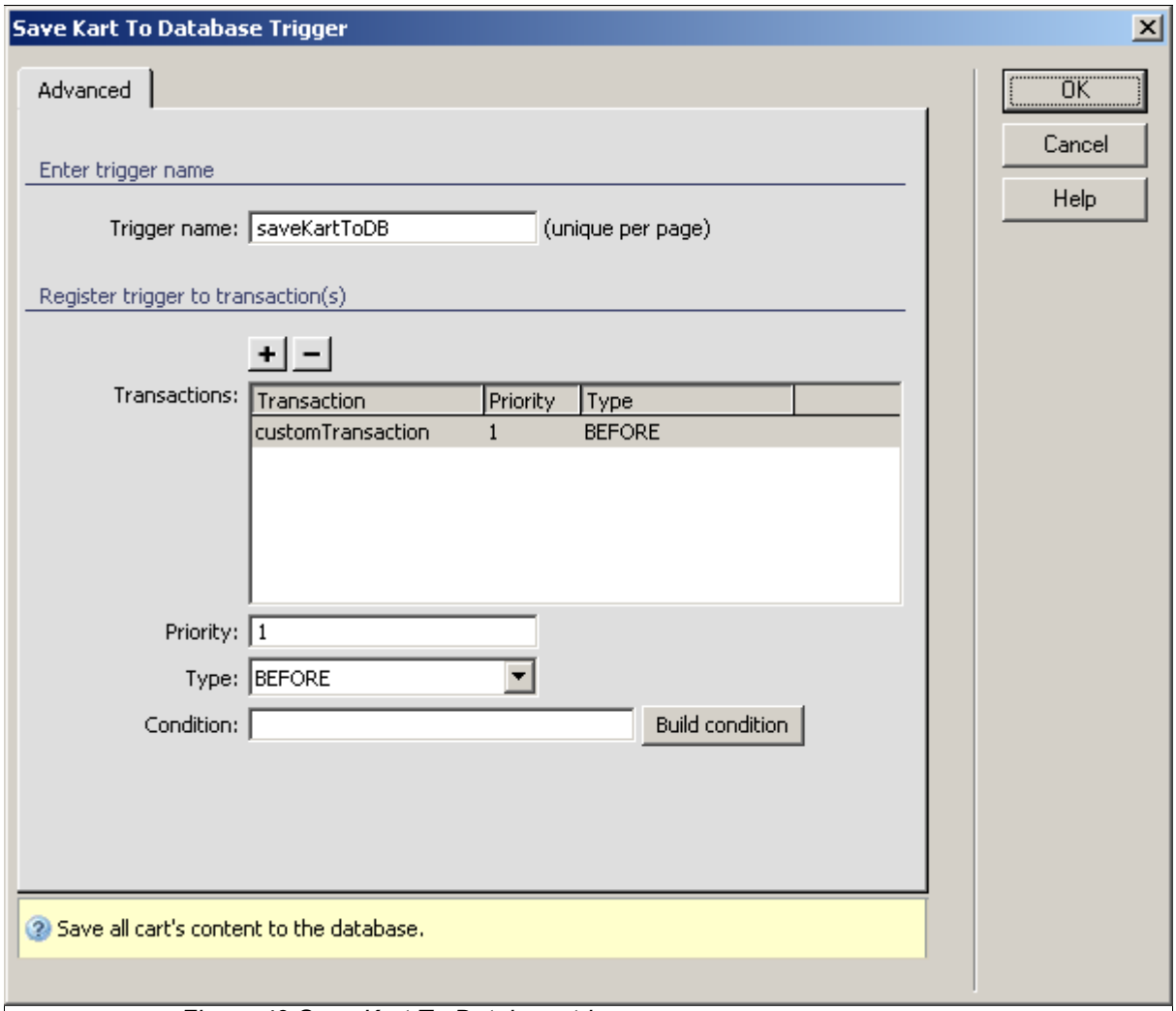


Figure 49 Save Kart To Database trigger

Applying the Prefill Order Details Trigger

The **Prefill Order Details** trigger retrieves the billing and shipping information related to a previous order from the database and will save it into the current order.

Apply it in the `step0.php` page from the **Application** Panel->**Server Behaviors**->+->**MX Kommerce**->**Triggers**->**Prefill Order Details Trigger**. Similar to the Save Kart To Database trigger, this one also registered autoamtically to the Custom Transaction (it is the only one on page), has its type set to BEFORE, which means it will get executed before the actual transaction, and its priority is set to 2. This means that it will be executed after the Save Kart To Database trigger.

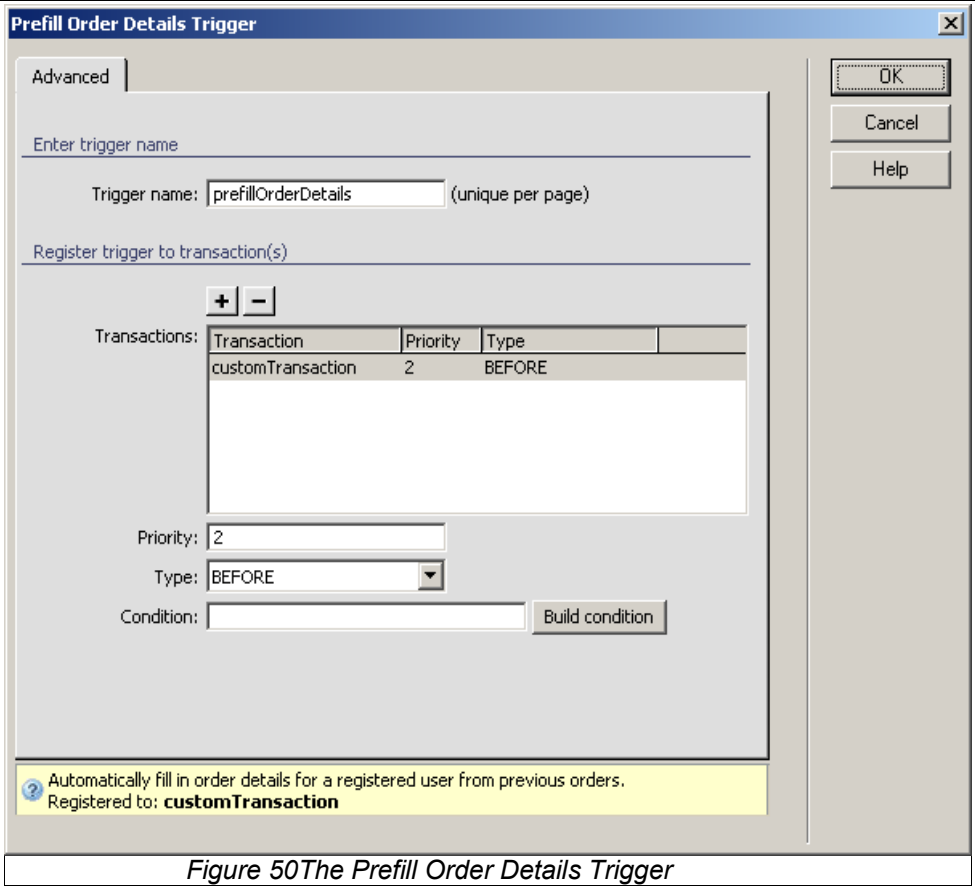
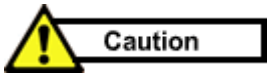


Figure 50The Prefill Order Details Trigger



The **Prefill Order Details Trigger** only works if the user is logged in. Otherwise, it blocks the custom transaction on the page and the redirection to step1 does not take place anymore. So, you should do the following:

Into the `step0.php` page, write the text:

"Please go back and use the Login Nugget to log into the system if you are a registered user. You will then have the information auto-completed in the wizard when you checkout. Otherwise, click Next to proceed to the next step."

Select the word "*back*" from the text and make a link to the `index.php`. Do the same with the "*Next*" word and make a link pointing to the `step1.php`.

Creating the step1.php page

The `step1.php` page will update the *order_ord* table with the customer's billing information. The recordset that will be used in order to insert the update form will have to be filtered after the kart order ID variable. Therefore you should first add a Kart Recordset on the page that will export the ***KT_kartOrderId*** variable into the session.

Adding a Kart Recordset

Open `step1.php`, in the **Application** panel -> **Bindings** and add a **Recordset(query)**. In the displayed interface click on the button for **Kart** you will be able to configure the Kart Recordset:

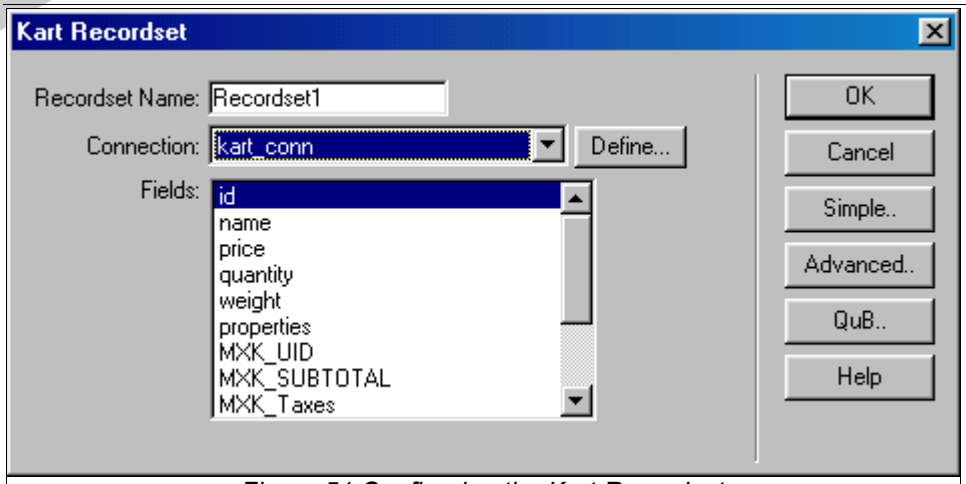


Figure 51 Configuring the Kart Recordset

All you have to do is to select the connection to the database and click **OK**.

Next add another recordset that will be filtered after the order ID field as in the following image (change table to *order_ord* and the filter to ***id_ord*** & ***KT_kartOrderId***) Click **OK**:

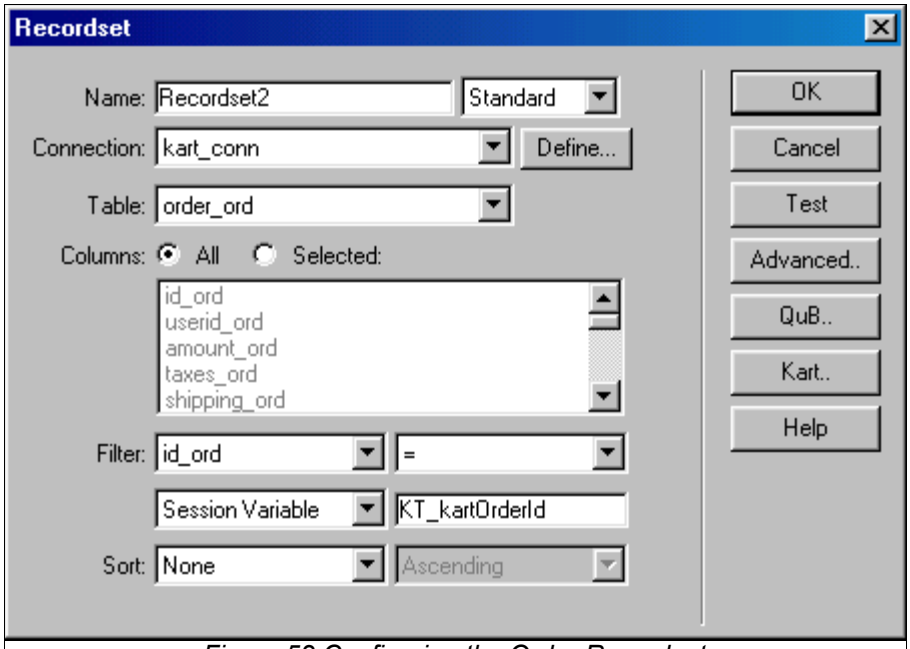


Figure 52 Configuring the Order Recordset

Now we can apply the ***Update Record Form Wizard*** from the **Insert** panel (near the top of the Dreamweaver interface)->MX Kollection tab that will update the *order_ord* table displaying a form where the customer should enter his personal details. The wizard is composed of 2 steps (3 if you have MX Form Validation Installed), each gathering some of the required information.

The first step requires you to enter database information:

- The database connection – kart_conn.
- The database table to update - order_ord.
- The primary key column - id_ord.
- The primary key value for the record to edit is passed through the KT_kartOrderId session variable.

The first step of the wizard, once configured should look like the image below:

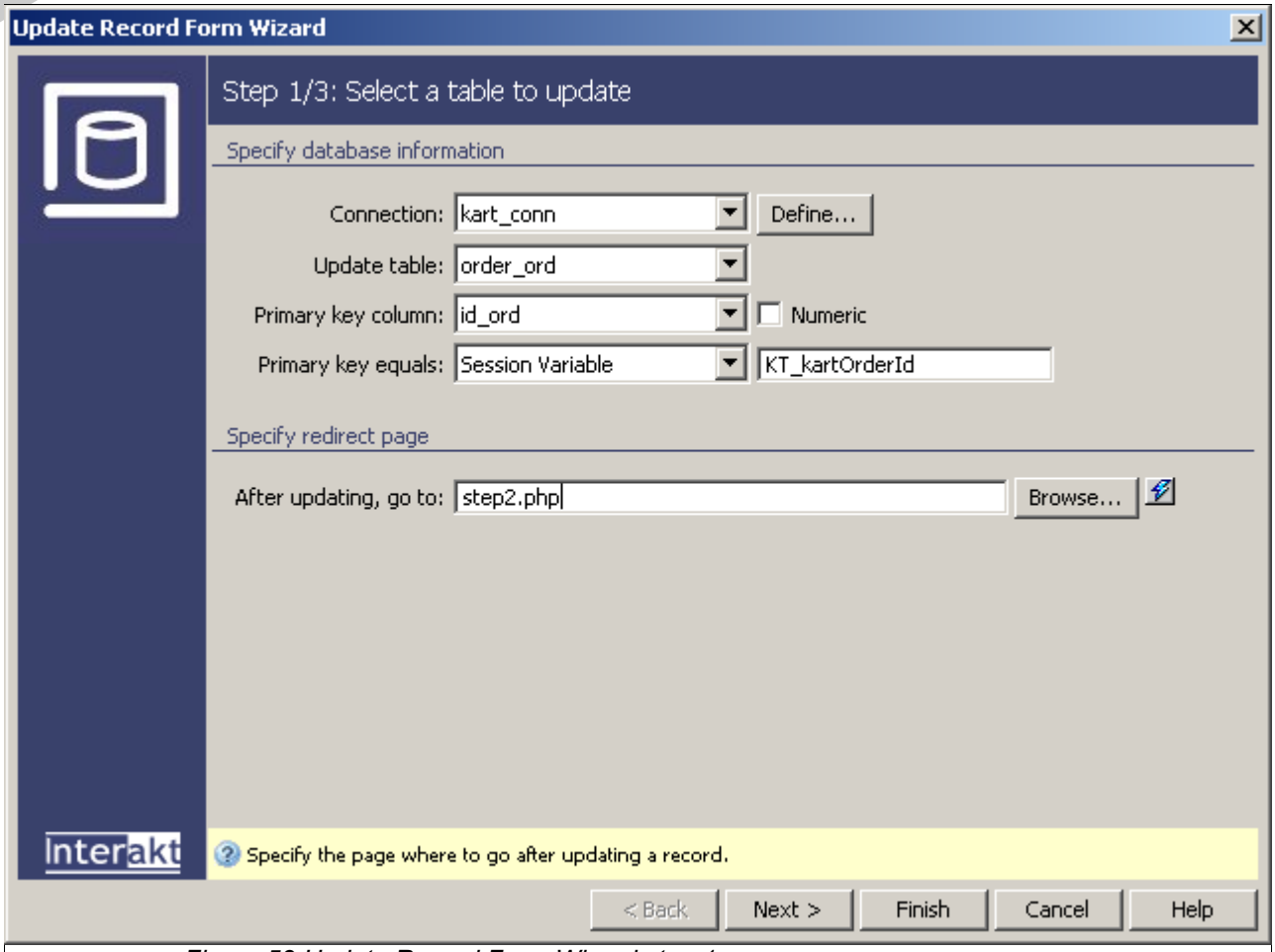


Figure 53 Update Record Form Wizard step 1

The wizard's second step allows you to define the fields to display in the form, the element to use and the type of data to submit to the table.

In the ***Form Fields*** list you should remove some fields (use the – button) in order to display only the following textfields: the first name, the last name, the email, the phone number, the street address, the city, the state, the country and the zip code.

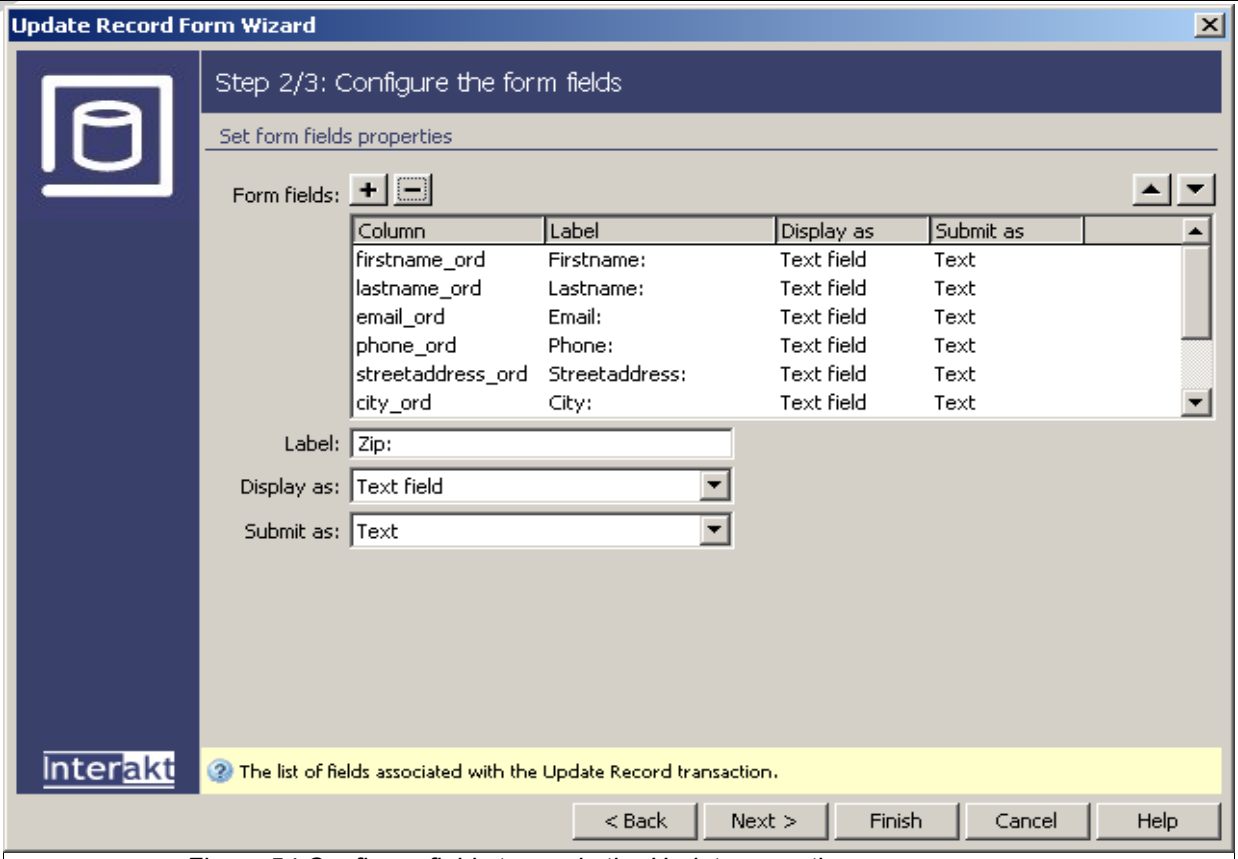


Figure 54 Configure fields to use in the Update operation

The wizard's third step allows you to define validation rules on each field. You should apply a validation format and use the Required Checkbox where appropriate (e.g. use the ZIP Code validation format for the Zip column).

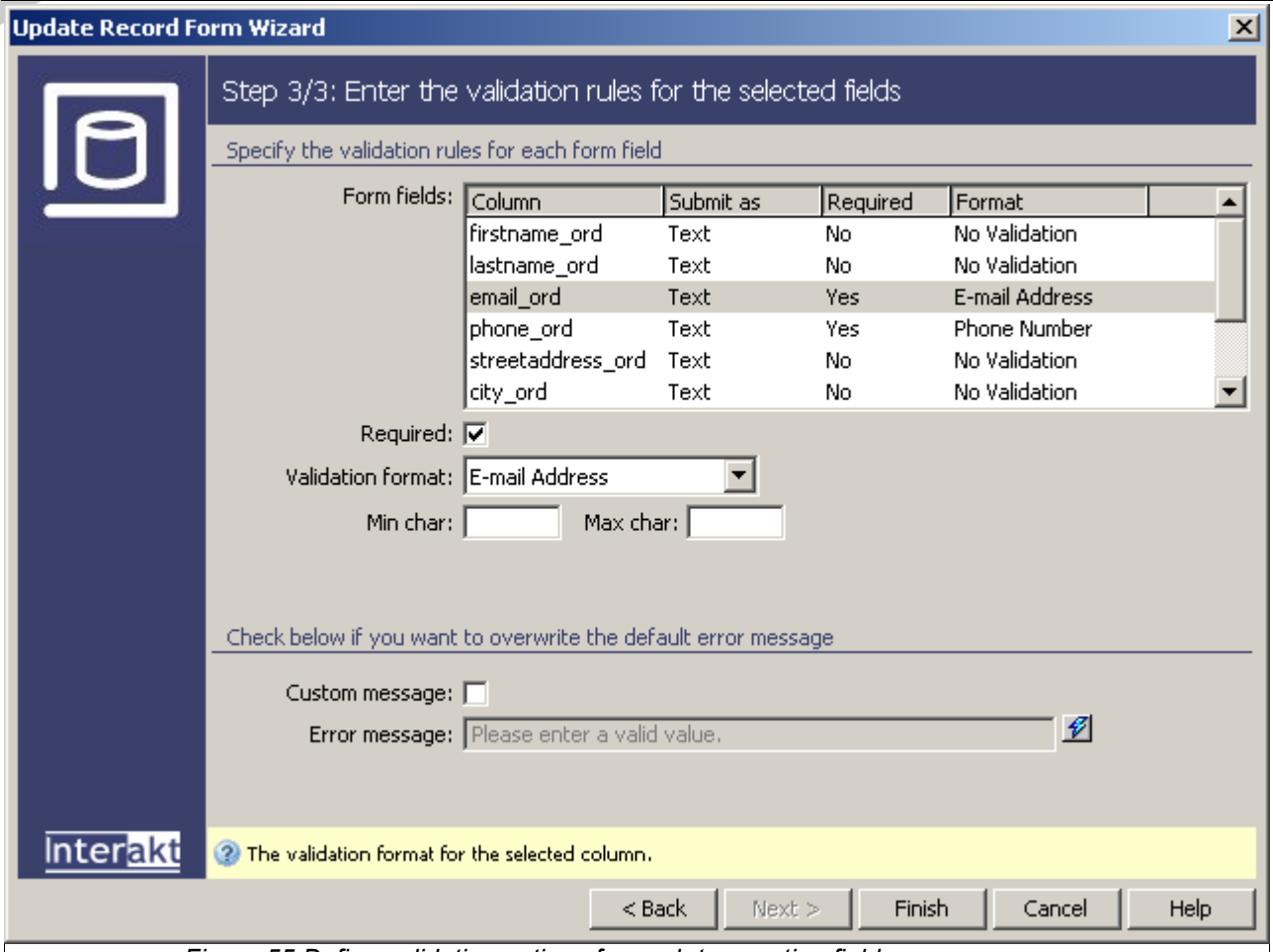


Figure 55 Define validation options for each transaction field

The Country and State fields are used in the taxes costs calculation. Leaving off the Country or State fields will result in the taxes NOT being calculated.

In Dreamveawer MX the `step1.php` page should look as follows:

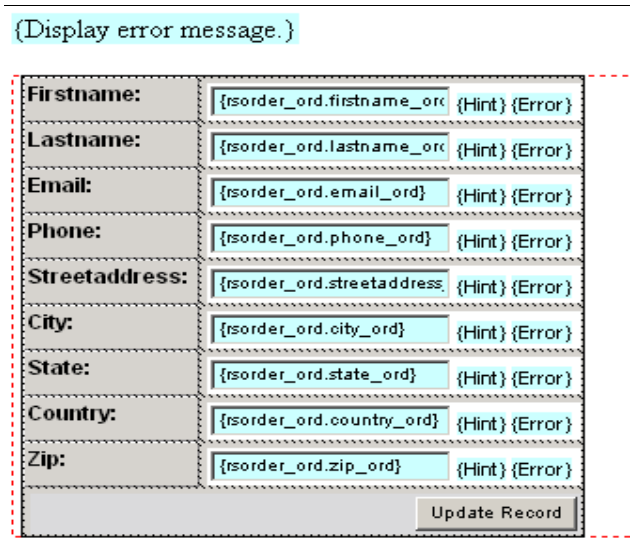


Figure 56 Dreamweaver view of the step1 page

Save the `step1.php` page, upload it to the server and test it with the browser by using the F12 key.

The page should look as shown below:

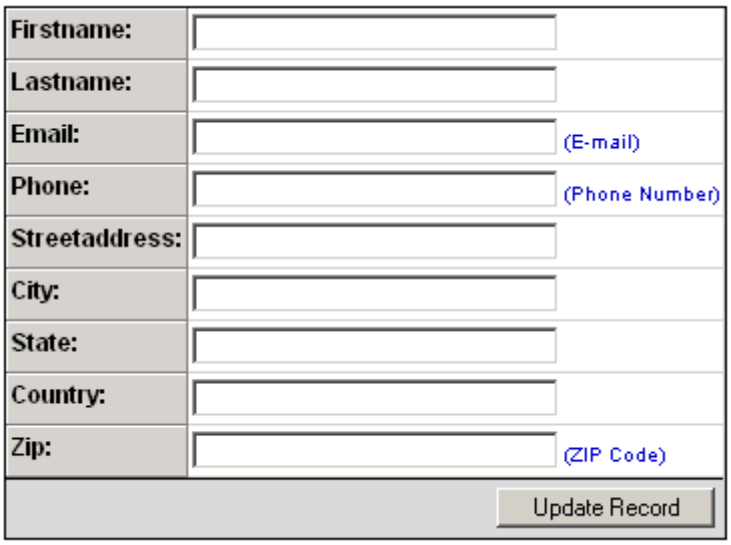
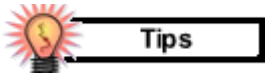


Figure 57 Browser view of the step1 page



If you want to submit the billing country name as a full name or an ISO3 abbreviation, you must go to the `MXKart/myShippingRates.inc.php` page, edit the **Shipping Rates per State** server behavior and set ***Country Name Column*** to either ***"name_cnt"*** or ***"iso3_cnt"***. If you have the taxes calculation feature activated, do the same thing in `MXKart/myTaxes.inc.php` page, editing the **Taxes Per Tax-Zone** server behavior.

If you want to submit the billing state name as a full name you must go to the `MXKart/myShippingRates.inc.php` page, edit the **Shipping Rates per State** server behavior and set ***State Name Column*** to ***"name_sta"***. If you have the taxes calculation feature activated, do the same thing in `MXKart/myTaxes.inc.php` page, editing the **Taxes Per Tax-Zone** server behavior.

Creating the step2.php page

The `step2.php` page will update the *order_ord* table with the customer shipping information. This page is created similar to the `step1.php` . To do this first open `step2.php.` Add a Kart Recordset on the page that will export the ***KT_kartOrderId*** variable into the session. This action is required in order to provide the Update Record Form Wizard a value for the record to edit primary key.

After adding the Kart recordset, apply an Update Record Form Wizard from the Insert Panel > MX Kollection Tab. Configure the wizard in a similar manner to the one above (in the step1.php page).

In step one of the wizard, set the rediret page to step3.php. Use the same order_ord table and the same STARTER method and value.

In the second step of the wizard remove some fields in order to display only the shipping details: the shipping method, the shipping name, the shipping street address, the shipping city, the shipping state, the shipping country, the shipping zip. Except for the ***shipmethod_ord***, all the other fields will be displayed as textfields. The ***shipmethod_ord*** field should be a drop-down menu allowing the customer to select one of the shipping methods available (express or standard for example). Set ***Display As:*** to ***Menu*** for ***shipmethod_ord***.

Two new buttons become available when highlighting the shipmethod_ord field: Add recordset and Menu Properties. Since the drop-down menu must display values from the shipmethods_smt table, you must make it dynamic.

Click the Add recordset button to create a new recordset that will retrieve all data from the shipmethods_smt table. Configure the recordset creation dialog box as shown in the image below:

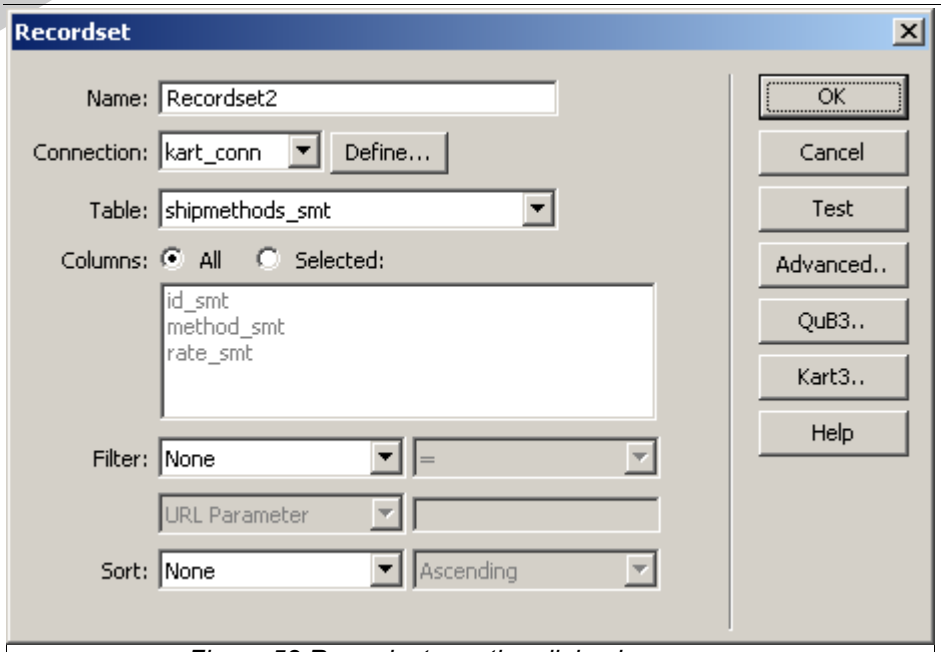


Figure 58 Recordset creation dialog box

Once the recordset is configured, click the OK button to add it to the page. Next you must configure the menu to use the newly created recordset. Press the Menu Properties button and configure the user interface to use the method_smt field for both the value and label:

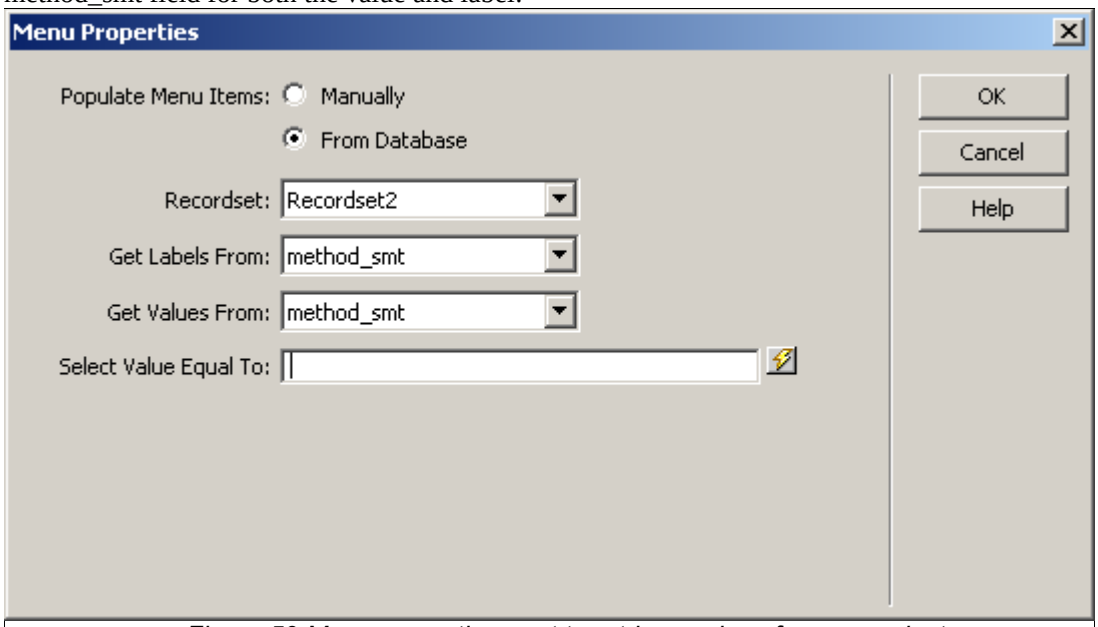
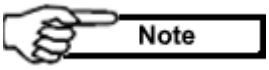


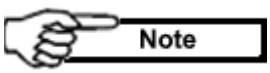
Figure 59 Menu properties - set to retrieve values from recordset



The recordset field chosen to contain the submitted values (the one set for ***Values*** ui field) must be the same as the field from shipping method table (in our case - *shipmethods_smt*) set for ***Shipping Method Column*** field in ***Shipping-Method Rate*** server behavior interface (`myShippingRates.inc.php` page).

Once the menu properties have been set, press the Next > button to move on to the wizard's last step, or Finish to close it.

In the third step of the wizard, set the appropriate validation format for each field.



The Country and State fields are used in the shipping costs calculation. Leaving off the Country or State fields will result in the shipping costs NOT being calculated.

In Dreamveawer MX, the *step2.php* page should look as follows:

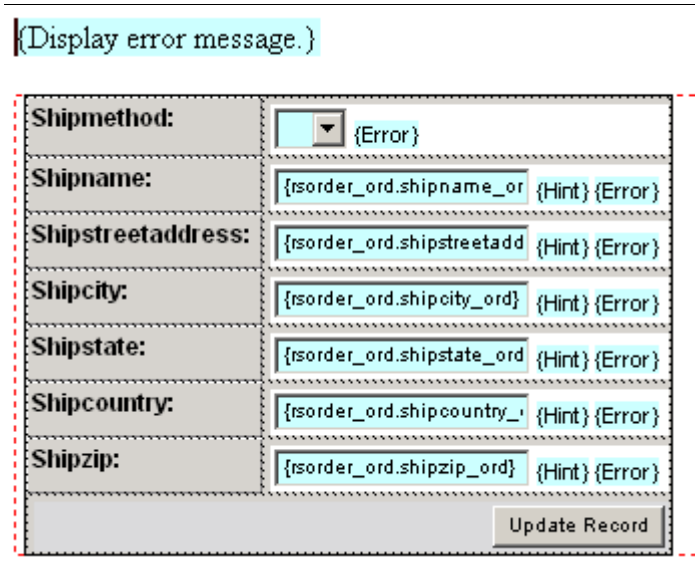


Figure 60 Dreamweaver view of the step2 page

Save the *step2.php* page, upload it to the server and test it into the browser by using the F12 key.

The page should look as shown below:

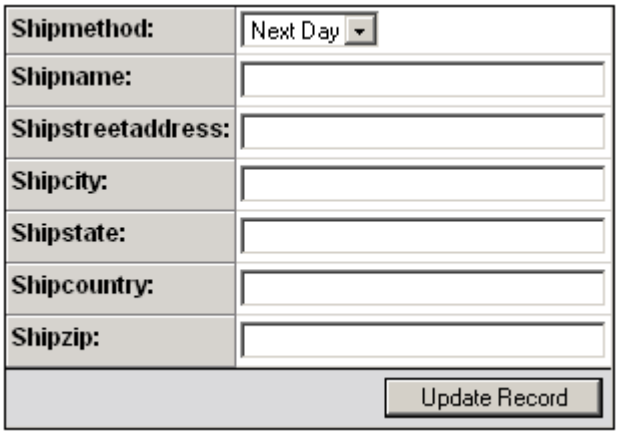


Figure 61 Browser view of the step2 page

If you want to submit the shipping country name as an ISO2 or ISO3 abbreviation, you must go to the *MXKart/myShippingRates.inc.php* page, edit the **Shipping Rates per State** server behavior and set ***Country Name Column*** to either ***"iso2_cnt"*** or ***"iso3_cnt"***.

If you want to submit the shipping state name as a state code, you must must go to the *MXKart/myShippingRates.inc.php* page, edit the **Shipping Rate per State** server behavior and set and set ***State Name Column*** to ***"code_sta"***.

Creating the step3.php page

The *step3.php* page will display the most relevant order information.

First open *step3.php then* add a Kart Recordset click **OK**. Now insert a table with 5 rows and 2 columns. In

the first column in each row type: “Goods Total Value”, “Shipping Costs”, “Tax”, “Weight” and “Total Payable Amount”. Then go to the **Application** panel -> **Bindings** tab and drag-and-drop into the second table column the corresponding dynamic values from the Kart Recordset as shown below:

Goods Total Value	{Recordset1.MXK_TotalPrice}
Shipping Costs	{Recordset1.MXK_Shipping}
Taxes	{Recordset1.MXK_TaxesTotal}
Weight	{Recordset1.MXK_Total_weight}
Total Payable Value	{Recordset1.MXK_TotalPayable}

Figure 62The step3.php Page in Dreamweaver MX

Under the table insert a ***Shop some more*** link to the `index.php` page and a ***Checkout*** link to the `checkOut.php` page (inside the `MXKart` folder).

Configuring the Payment Gateway

Creating a Payment Gateway Account

First lets very briefly explain the checkout process.

After clicking the ***Checkout*** link the customer will be redirected to a Payment Gateway page where they may fill in the requested information. After finishing the order the customer will be redirected to the **MX Kart** site. He will also receive an email in which the Payment Gateway informs that an order has been placed under his name and lists the inserted customer information.

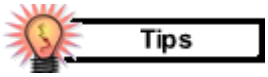
In order to enable this checkout process the merchant should first create an account on one of the Payment Gateways supported by **MX Kart** and then pass the information to the web developer. After this the merchant will be able to receive online credit card payments.

Creating a PayPal Account

In order to register to PayPal, the merchant should access the page: https://www.paypal.com/us/cgi-bin/

After selecting a country from the ***Business Account in:*** drop-down menu a PayPal form will be displayed.

The merchant should fill in all the fields and after agreeing to pay a certain amount, he will have a PayPal Business Account enabling him to accept online credit card payments.



For more detailed information, check the **MX Kart – How to Configure the Payment Gateway** tutorial.

Selecting the Payment Gateway

The **Kart Properties** command also allows the configuration of the payment gateway from the ***Payment*** entry from the command user interface left side menu.

Open any site page.

First we will select the payment gateway that you want to use in order to accept instant online credit card payments. Double clicking on ***Payment Gateway Type*** of the **Payment** component will display the following interface.

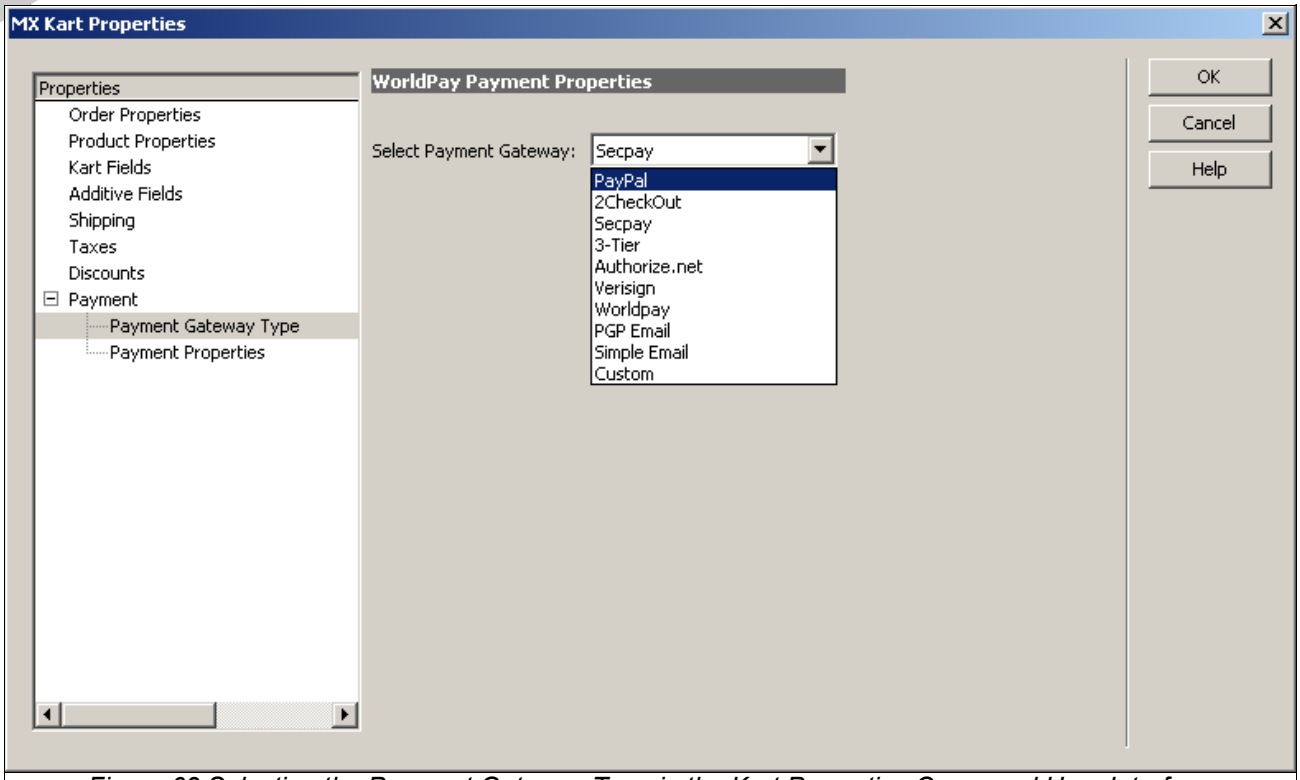


Figure 63 Selecting the Payment Gateway Type in the Kart Properties Command User Interface

- ***Select Payment Gateway*** – this drop-down menu allows the selection of one of the payment gateways supported by **MX Kart**.

Configuring the Payment Gateway Properties

After selecting the ***PayPal*** payment gateway, for example, double clicking on ***Payment Properties*** from the ***Payment*** entry the following interface will be displayed (a different interface with the required options will be displayed depending on the selected payment gateway):

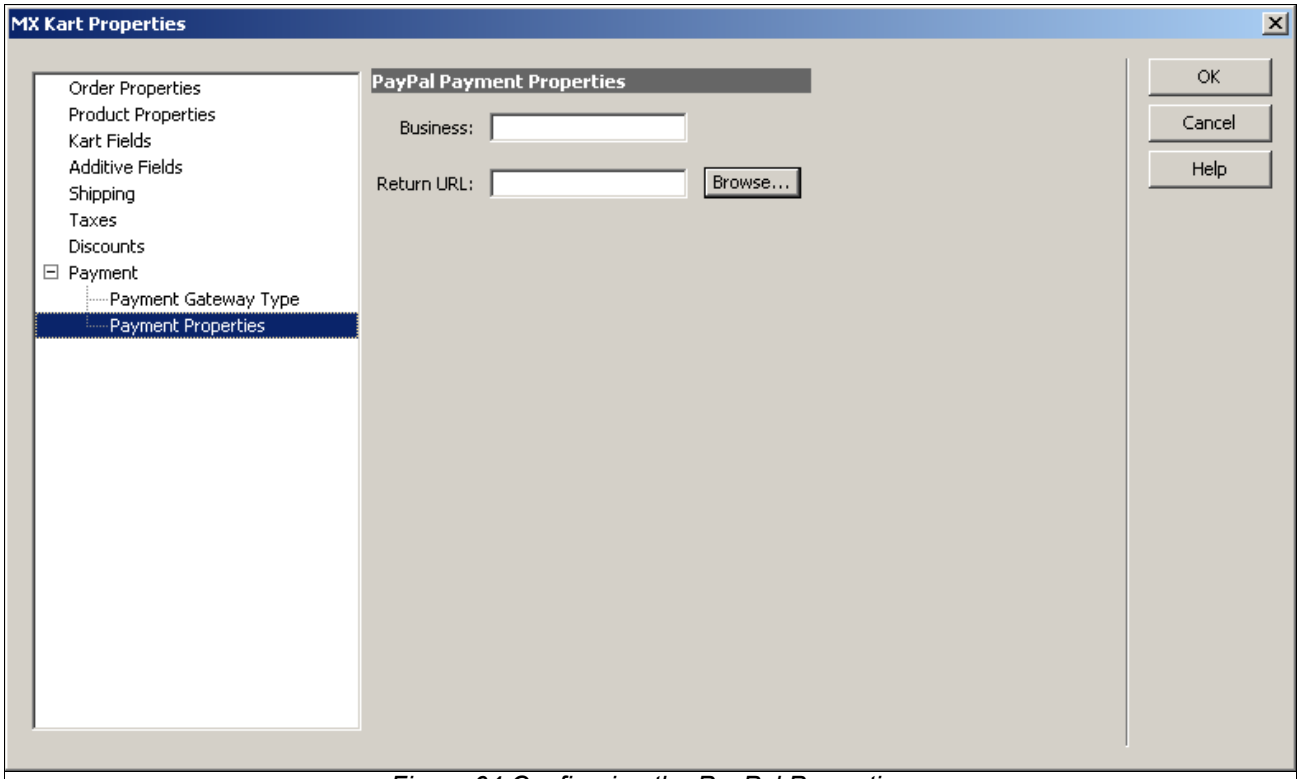


Figure 64 Configuring the PayPal Properties

- ***Business*** – in this text-field enter the the merchant's (site owner's) ID ("business" as it is called on PayPal – is in fact the email registered with PayPal) on the payment gateway.
- ***Return URL*** – in this text-field enter the path to the page that processes the information received from the payment gateway. By default the link is set to a predefined file (`return.php`) which is correctly configured for the provided **MX Kart** installation kit. We recommend you not to change this selection.

Save the page.

Applying the Discounts Server Behaviors

After creating a basic shopping cart application you can also add some advanced features such as discounts.

The **MX Kart** discounts will be configured to retrieve their values from database tables, to allow the site administrator to manage the discount policies after the e-commerce site is published.

In **MX Kart** there are four types of discounts implemented:

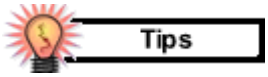
- **User Level Discount** – applies a different discount for each authenticated user level
- **Volume Discount** – applies a discount to the cart's total value, if it exceeds a certain limit
- **Special Offers** – sets a new discounted price for a product for a specific period of time
- **Coupons** – applies a discount to a product when the customer enters a valid coupon code

All the discounts are already applied on the `MXKart/myDiscounts.inc.php` page.

Therefore, adding a discount business logic in a current **MX Kart** application takes 2 steps:

- Activate the discount by double clicking on the ***Discounts*** entry from the **Kart Properties** command user interface.
- Make the discount visible on every price field that is displayed from the original product table recordset by using the ***Show Discounted Price*** Server Behavior.

Of course for some discounts other preparative steps will need to be made in order to make them fully functional in your application. Those steps will usually require you to manage the information in the database tables.



If you want to configure the Discounts Server Behaviors yourself check the **MX Kart User Manual** for guidelines.

Applying the Discounts

You can decide to apply one or more discounts to the products you sell with your e-commerce application.

First open any site page.

Activating one or more Discounts

In the **Kart Properties** command user interface from the **Insert** panel -> **MX Kart** tab, double click on ***Discounts*** and select the desired discount component: ***userdiscount*** (for **User Level Discount**), ***specialoffers*** (for **Special Offers**), ***volumediscount*** (for **Volume Discounts**).

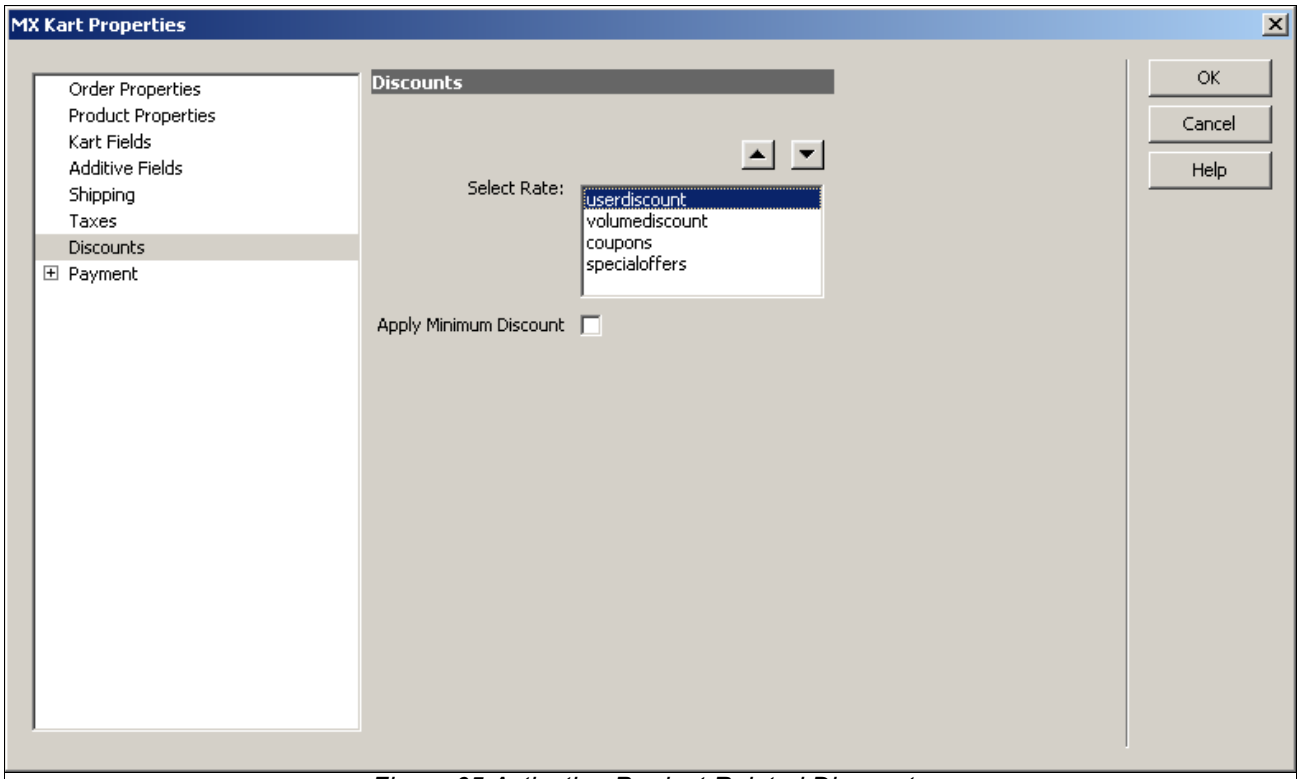
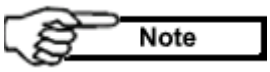


Figure 65 Activating Product Related Discounts

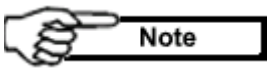


For the moment don't select the ***coupons*** discount because the page where the customer can enter his coupon code will be created later.

Use the arrows display in the upper right corner in order to set the processing order.

As you can see there is an ***Apply Minimum Discount*** checkbox in this user interface. Its purpose is to enable you to select the approach that you want to use when applying several discounts in your e-commerce application.

- when checked, the final discount function is obtained by computing the minimum of all individual discount functions: $F_{total} = min(f_{d1}, f_{d2}, \ldots)$ The order in which individual discounts is processed is not important.
- when unchecked, the final discount function is obtained by computing all individual discount functions: $F_{total} = f_{dn}(\ldots(f_{d2}(f_{d1}(p))))$ The order in which the discount functions are considered is important as the composition is not a commutative operation.



By default, the **Volume Discounts** will be the last one applied.

You can activate all or more discounts by keeping the SHIFT or CTRL key pressed while selecting them with the mouse.

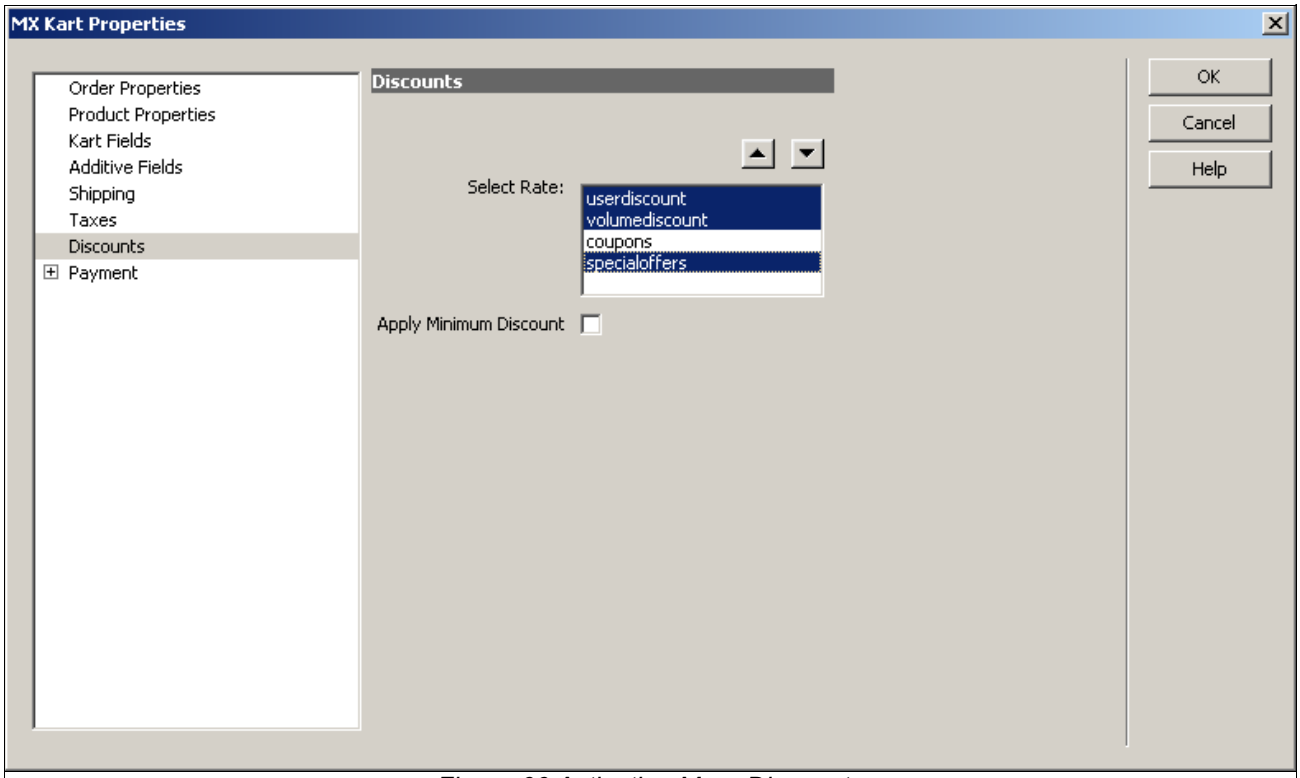


Figure 66 Activating More Discounts

Save the page.

Applying the Show Discounted Price Server Behavior

Finally, you should select the dynamic value and apply the **Show Discounted Price** Server Behavior.

Open the `index.php` page and select the price dynamic value from the products table (the one **not** strikedthrough) and then apply the **Show Discounted Price** Server Behavior from the **Application** Panel -> **Server Behaviors** -> + ->**MX Kommerce** -> **Show Discounted Price**.

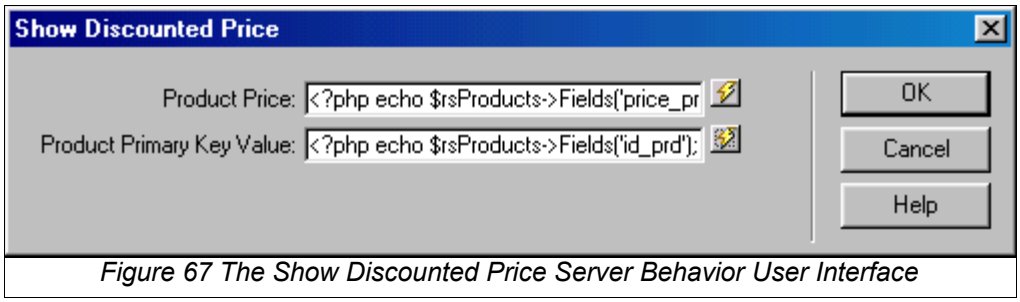


Figure 67 The Show Discounted Price Server Behavior User Interface

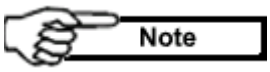
As you can see, the ***Product Price*** field is filled in automatically with the ***price_prd*** parameter of the *rsProducts* recordset. All you have to do is to set dynamic value for the ***Product Primary Key Value*** field by clicking on the "lightning bolt" icon and choosing ***id_prd*** from the recordset. Click **OK**.

Open the `prodDetail.php` page and select the price dynamic value (the one **not** strikethrough) and then apply the **Show Discounted Price** Server Behavior from the **Application** Panel -> **Server Behaviors** -> + ->**MX Kommerce** ->**Show Discounted Price**.

Set the dynamic value for the ***Product Primary Key Value*** field by clicking on the "lightning bolt" icon and

choosing ***id_prd*** from the recordset.

After applying this server behavior all the activated discounts will be applied on these price fields.



You are probably wondering why do we apply the Volume Discount on the individual price fields and not on the Total Amount field. The reason is that in every order, the Total Amount should be the sum of individual products' prices. An invoice cannot display the undiscounted detailed prices, the Total Undiscounted Amount and then the Total Discounted Amount.

Applying the Coupons

By using the **Discounts - Coupons** Server Behavior you will be able to implement a promotional campaign using coupons on your site. In order to take advantage of a discount on a specific product the customer must enter the coupon code in the public site section and this should be stored in the session.

Creating the Add Coupons page

For that on the `addCoupons.php` page apply the **Create Shopping Kart View** command (**MX Kommerce** tab of the **Insert** panel near the top of the Dreamweaver interface):

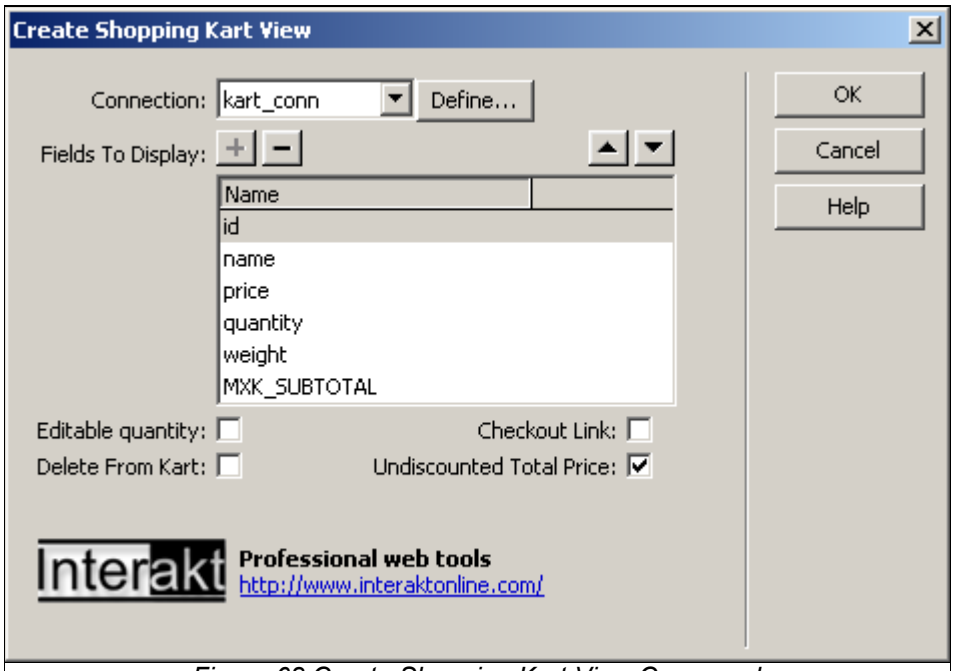


Figure 68 Create Shopping Kart View Command

It is not necessary to check the ***Editable quantity***, ***Delete From Kart*** and ***Checkout Link*** boxes.

Next insert a form by clicking on the **Form** button from the **Insert** Menu -> **Forms** tab (near the top of the Dreamweaver interface). Inside this form enter the text “Enter Coupon Code:”. Next, insert a text-field having the name ***couponNr*** (instead of ***TextField*** from the ***Properties*** inspector) and a Submit button. You can also add a ***Shop some more*** link to the `index.php` page.

Applying the Save Variable To Session

The next step is to transform the ***couponNr*** variable into a session variable which stores the coupon numbers entered by the user. First you will need to apply an empty **Custom Transaction** from the **Application** panel -> **Server Behaviors ->MX Kollection->Forms>Advanced> Custom Transaction.** Configure it to use the **kart_conn** database connection and set the default **STARTER** method to **form_variable** (the submit button). For the method reference enter **Submit.** In the **When finished, go to** text field, enter the `index.php` page**.**

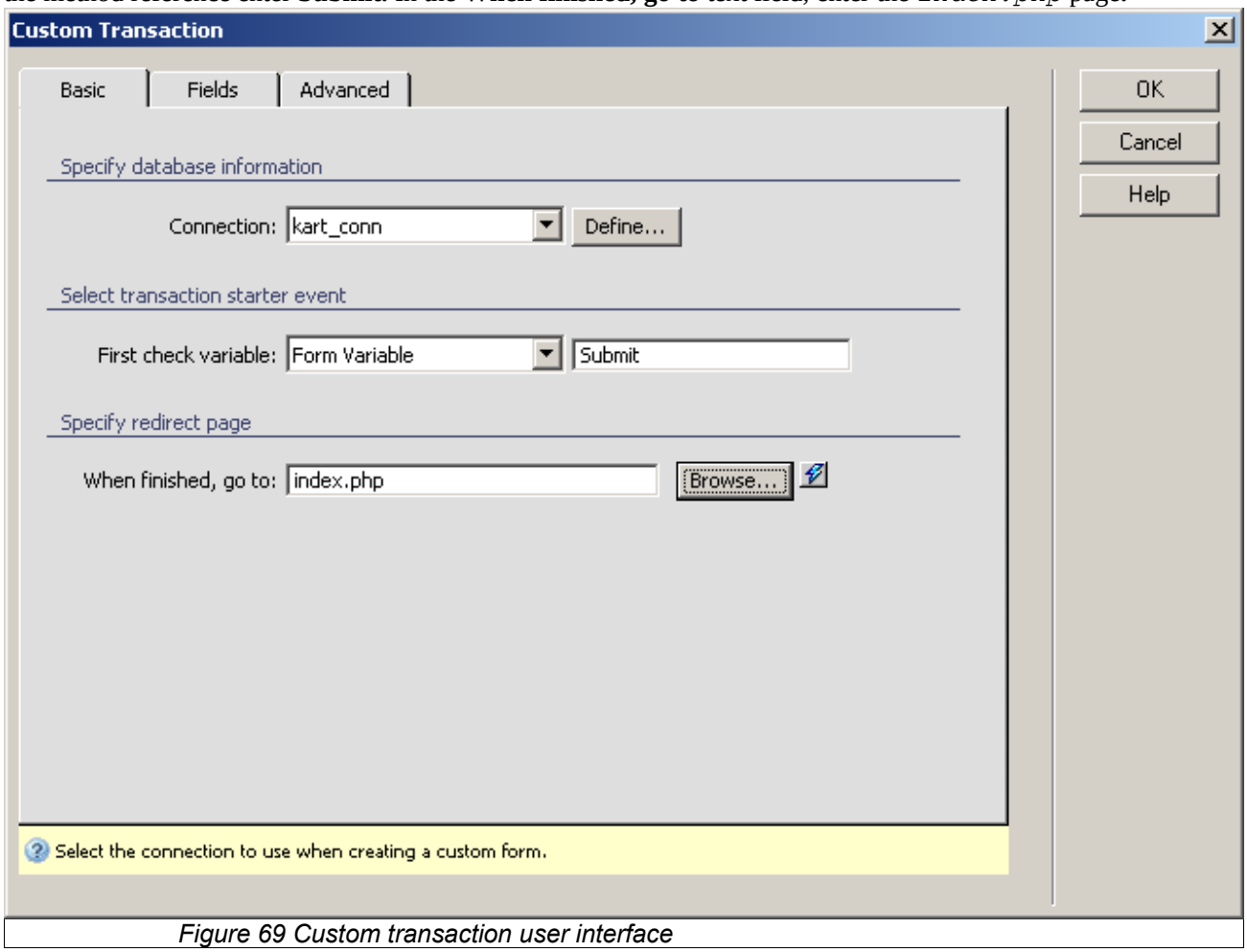


Figure 69 Custom transaction user interface

Next apply a **Save Variable To Session** trigger (**Application-> Server Behaviors-> MX Kommerce->Triggers->Save Variable to Session)**.

The **Save Variable To Session** trigger that will be registered to the custom transaction we just created should be configured as follows:

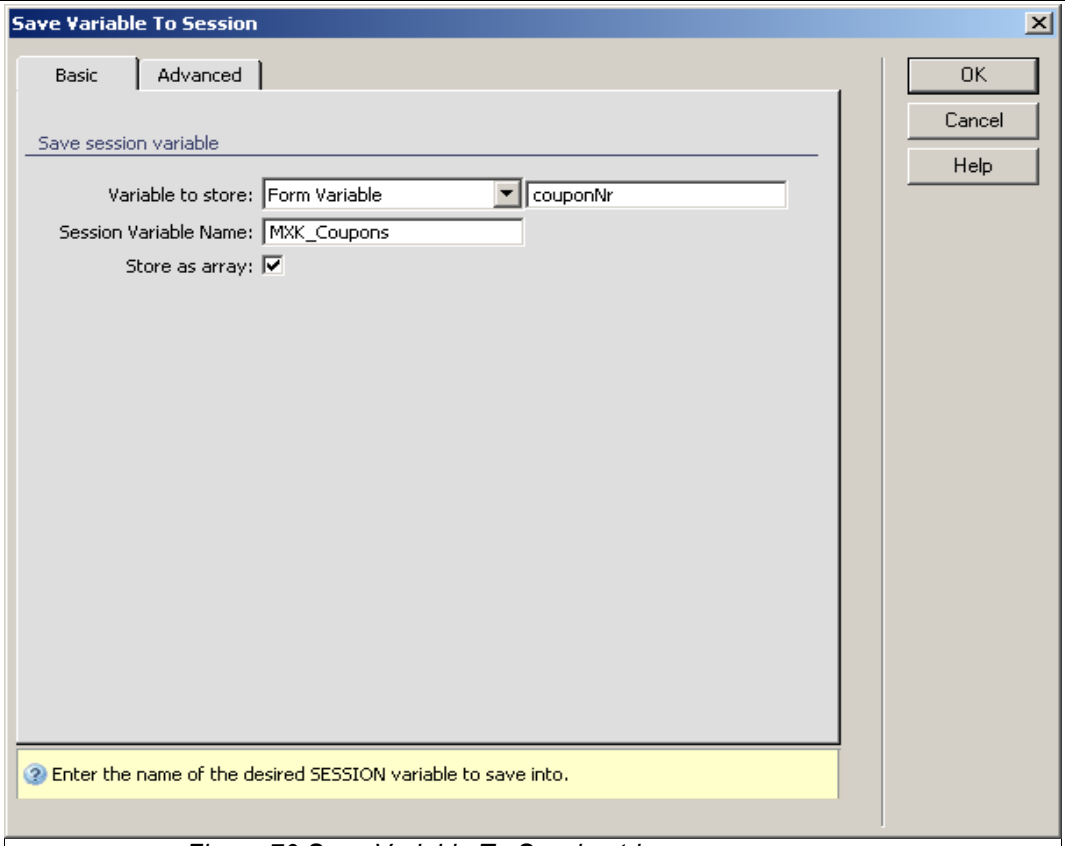


Figure 70 Save Variable To Session trigger

As you can see we transformed the ***couponNr*** variable into a session variable named ***MXK_Coupons***.

Next, you should go to the `MXKart/myDiscounts.inc.php` page and double click on ***Coupons*** from the Server Behaviors list. You should change the ***Session Variable Name*** into ***MXK_Coupons***.

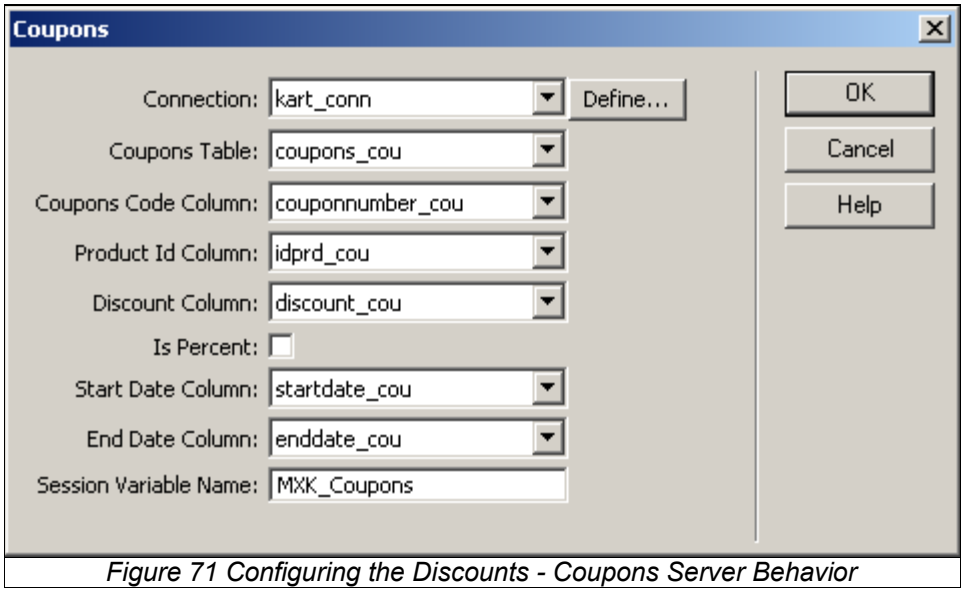


Figure 71 Configuring the Discounts - Coupons Server Behavior

You can choose any name you like for your session variable as long as it is the same as the one entered in the ***Session Variable Name*** in the **Discounts - Coupons** Server Behavior interface (the default value is ***couponcode***) .

In **Dreamweaver MX** the `addCoupons.php` page should look as follows:

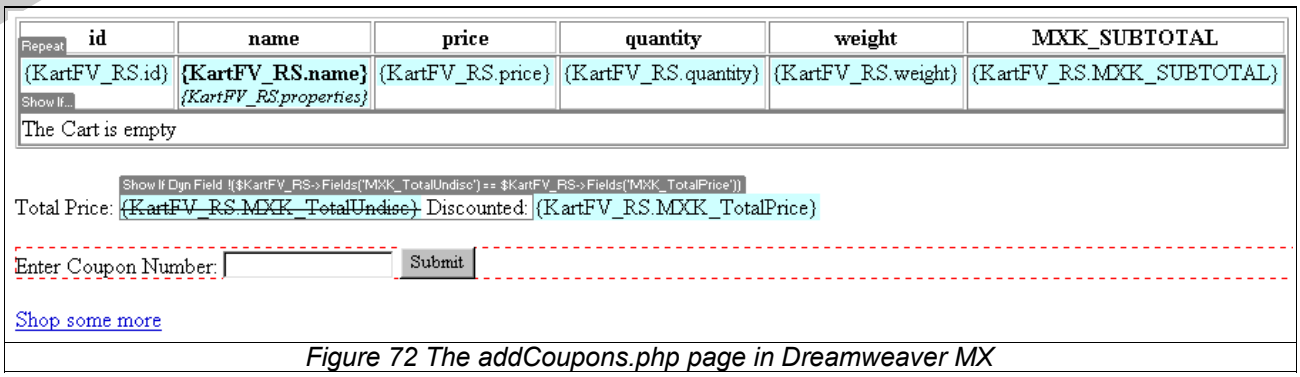


Figure 72 The addCoupons.php page in Dreamweaver MX

Save the `addCoupons.php` page. Open the `index.php` page and add an ***Add Coupon Number*** link to the `addCoupons.php` page (under the "Full Cart View" link in the outside table's second column).

Activating the Coupons Component

In the **Kart Properties** command user interface activate the ***coupons*** component by double-clicking on ***Discounts*** and then shift clicking to select all the options in the ***Select Rate*** section. Click **OK**.

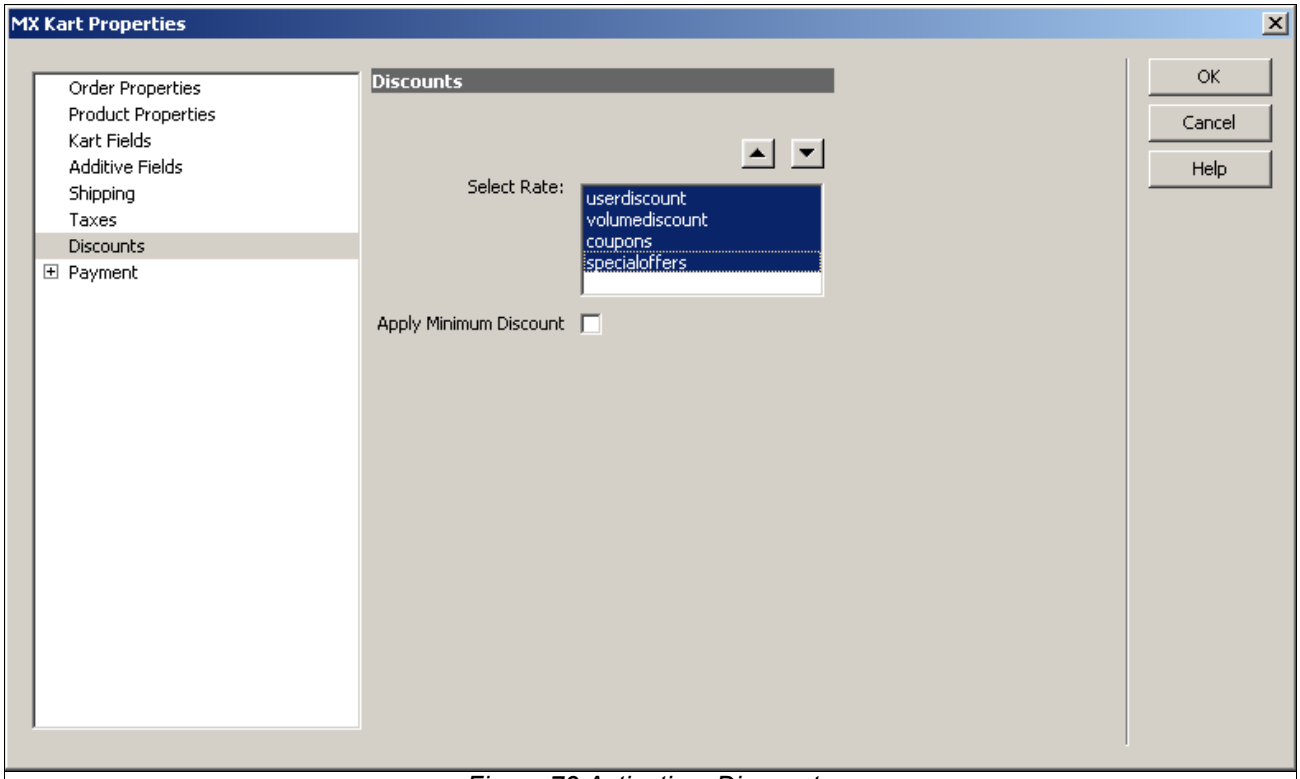


Figure 73 Activating Discounts

Conclusions

Basically **MX Kart** will give you the ability to generate sites with the following capabilities:

- add to cart – enables the customer to add products to the cart either by link or by form (after setting the products properties)
- view cart – enables the customer to visualize the cart content and to update it.
- checkout procedure – enables the customer to checkout to a payment gateway in order to execute an online credit card payment and saving the cart to an orders table
- discounts – enables the developer to apply preset types of discounts.
- taxes – computes taxes automatically
- shipping rates – computes shipping fees automatically

Further Reading

This tutorial can serve as a starting point for future developments.

To find out more about **MX Kart** and its features we recommend reading the following:

- **MX Kart User Manual**

To find out more about **MX Kollection** and its features, we recommend reading:

- **The MX Kollection user manual**
- **The MX Kollection tutorials**

For news and updates, also visit our website at http://www.interaktonline.com

Appendix I - versions

To create this tutorial we have used the following software versions:

Code Generator: Macromedia Dreamweaver MX 6.1 or 2004

Extensions: **MX Kart** 1.0.0

Database server: MySQL 3.23

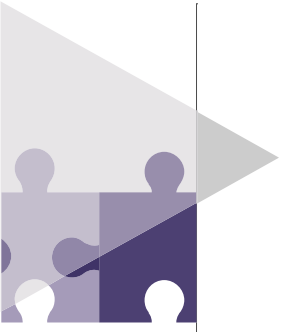
PHP: PHP 4.3.2

Web server: Apache Web Server 1.3.27

Operating systems:

- Workstation: Microsoft Windows 98
- Web server: Redhat Linux 7.2

Copyright



Copyrights and Trademarks

Send comments and suggestions to products@interaktonline.com

Dreamweaver extensions for dynamic websites

InterAKT Online
Web: http://www.interaktonline.com/
E-mail: contact@interaktonline.com
Address: 1-11 Economu Cezarescu ST, AYASH Center, 1st floor
Sector 6, ZIP 060754, Bucharest, Romania
Phone: +4021 312.51.91
Fax: +4021 312.53.12