

Tutorial
Content Management
System



Table of contents

Introduction	3
Plan the Content Management System	4
Add, Edit and Delete Parts and Articles	6
Create navigation menu	7
Display article lists	9
Display article	12
Create home page	14
List articles and implement multiple delete	19
Add and edit articles using the same form	23
Restrict part delete if it has associated articles	28
Improve the Content Management System	31
Auto-archive old articles	32
Create rich content visually: KHTML Lite	36
Improve administration forms: MX Widgets	39
Optimize content for search engines	41
Other Resources	43
Copyright	44

Introduction

In this tutorial, you will create a **C**ontent **M**anagement **S**ystem (CMS). A CMS is a web application that allows you to create articles or any other pages directly online, from a secured administration area.

This tutorial will not present how to implement user authentication, as this is described in another tutorial (the Job Site tutorial). You must however create the login and registration pages before creating any other sections of the site.

This application will allow you to:

- create, update, and delete articles.
- display a list of articles for the site administrator.
- create a navigation menu and display articles in the front-end.
- auto-archive articles older than one month.

To complete this tutorial, you will make use of features from **MX Kollection 3**. If you do not have the **MX Kollection 3** bundle, then the following separate products must be installed:

- **MX User Login**
- **ImpAKT**
- **MX Includes**
- **NeXTensio 3**
- **MX Widgets**
- **KTML Lite**

To complete this tutorial, it will take about 80 to 100 minutes, depending on your web authoring level.

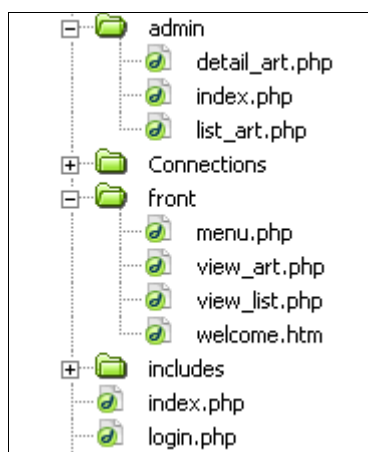
The first thing to do when starting to create an application, is to take some time and plan it out. Decide on the files to create, and the database structure you should use.

Plan the Content Management System

Before you start building this application, make sure you have a correctly configured Dreamweaver site, and a working database connection. For more instructions regarding these actions, consult the Getting started book help file, which can be found in **Help -> InterAKT -> Getting Started**.

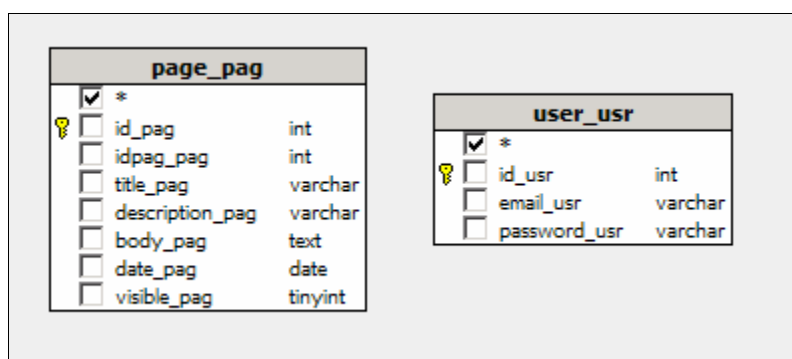
Through the tutorial, you will have to create several files and folders in your site's root. You can create them at the very beginning, so that you will not waste time with this operation again. To create files and folders in the site's root, use the corresponding options in the **File menu** of the **Files tab**. All files created in this tutorial can also be found inside the downloaded package, and you can use them to compare your work with what has been already done.

The file structure will look as in the example below, and you can create it easily by unpacking the ZIP file corresponding to your server model from `\tutorials\Content Management System\` in your site root:



After having created the files for your pages, it is time to set up the database that will hold the information to display. You can find the scripts needed to create an identical table structure inside the downloaded package, in the `\tutorials\Content Management System\db\` folder, as an *sql* or *mdb* file, depending on the database server you intend to use.

For this tutorial, you will use two tables: one for the pages, and one for the users. The fields' names are self-explanatory, as you can see in the following image:



Note: The database diagram in the image above was built with **MX Query Builder** (also referred as **QuB**) to better illustrate the database structure. You do not need to build it in order to complete this tutorial.

Here's a listing of the tables and columns used in this database:

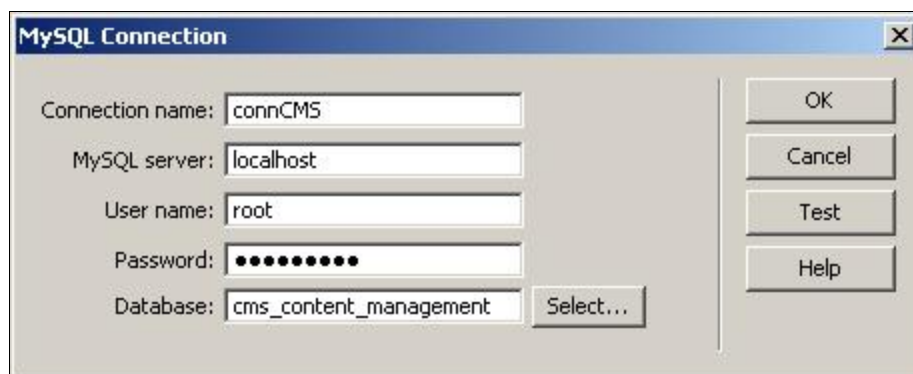
1. *page_pag* - the page table stores all the site pages and their information
 - **id_pag** - the primary key for the page table. No two pages will have the same id.
 - **idpag_pag** - this field serves as a self-foreign key. This is a reference to another page within the page table. Here this field differentiates between article lists, and the articles themselves. The idpag_pag field for a regular article will contain the value of it's parent article. If any of this is confusing just check out the Database concepts page for a little help.
 - **title_pag** - the title of the page.
 - **description_pag** - A brief text description of the page's contents.
 - **body_pag** - the actual content of the page (HTML) is stored in this field.
 - **date_pag** - stores a date value to simplify updating.
 - **visible_pag** - stores the status for each page. If set to 0, the page is invisible.
2. *user_usr* - the user table stores all users registered to the site, and some simple information about them.
 - **id_usr** - the primary key for the user table. No two users can have the same id.
 - **email_usr** - the user's email address (rob_anderson@somedomain.org).
 - **password_usr** - the user's password for logging in.

You can find the scripts needed to create an identical table structure inside the downloaded package as a *SQL* or *MDB* file, depending on the database server you intend to use. Import them in your database server management software (e.g. **PHPMyAdmin**, **Microsoft Access** etc).

Sample data is already entered in the database. To login into the administrative section of the site, the default account is:

- username: **admin**.
- password: **root**.

Open the main *index.php* page and create a new connection named *connCMS* and configure it to connect to your newly created database:



Before moving on, remember to create the login page for your site. First configure your login settings, and then apply the Login Form Wizard on the login page in the site root.

Add, Edit and Delete Parts and Articles

In this section of the tutorial you will learn how to use **MX Kollection 3** features to build the main pages of your content management system, both in the front-end, and the back-end. You will create the following elements:

1. For the Front-end:
 - A menu for the front-end, linking to pages that present lists of articles
 - The actual pages that display the lists, and the article.
 - A main page which ties together the pages and functions listed above
2. For the Back-end:
 - You will create the page that lists the articles, also allowing you to delete multiple items, as well as displaying links to edit or add an article.
 - A page that contains the insert/update form, allowing you to enter the actual article.

To build these pages, you will use the following products:

- MX User Login - to prevent users from accessing the administration pages
- MX Includes - to combine all front-end pages in a single file.
- NeXTensio - to build the back-end list and form of article.

Completing the first, and largest section of the tutorial will take about 40 to 50 minutes.

Create navigation menu

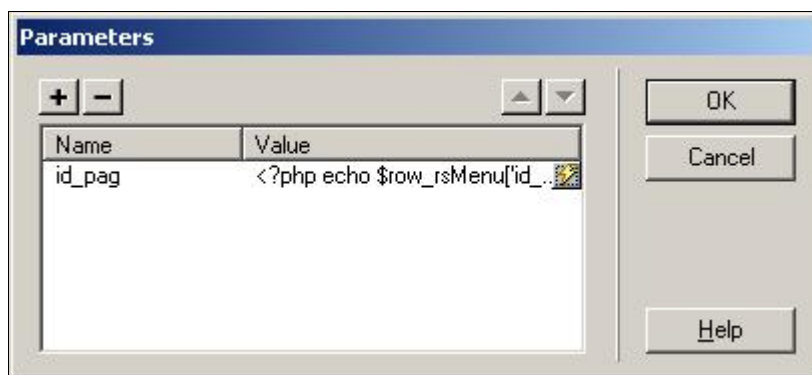
To allow the users browse the articles in an easy manner, you have to add a navigation menu. This will be stored in its own page, in the *front* folder. The menu will display links only to pages that contain a list of articles, as each list will point to the articles it contains, allowing the users to read them.

To create the menu, follow the next steps:

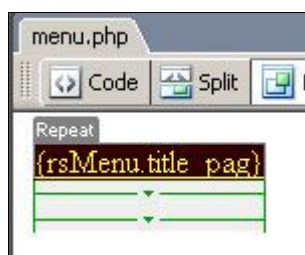
1. Open the *front/menu.php* page.
2. Create a recordset with the titles of the articles on the first level.

3. Create a dynamic table (Insert bar -> Application tab) with all the records in the *rsMenu* recordset.

4. Delete the first row, to remove the column headings.
5. Delete the columns containing the primary key and the self foreign key.
6. Link the dynamic text to the *view_list.php* page and pass it the **id_pag** URL parameter:



7. Here it is how the menu looks in **Dreamweaver** design view:



8. Hit F12 to preview the menu in your browser.

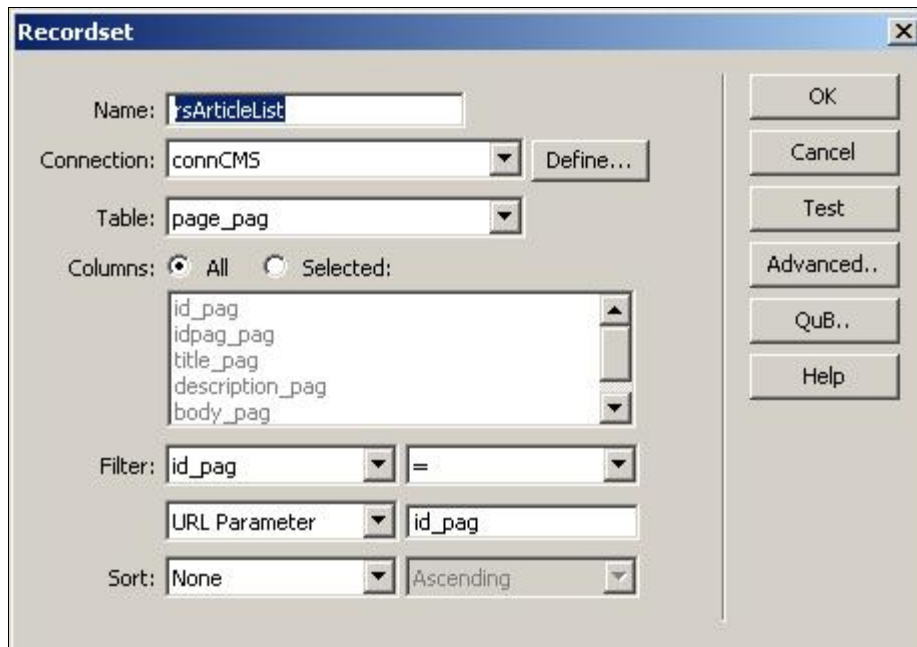


Display article lists

In this section of the tutorial you will build a page that displays article lists. What this page will display is the article referenced by the URL parameter `id_pag` (on top of the page), together with all articles that have the self-foreign key equal to the ID passed through the URL parameter (the articles that belong to the list). For each article, only the title and description are displayed. In order to read the entire content, a link will be displayed.

To build this page, follow the next steps:

1. Open the `\front\view_list.php` page. This is where you will display article lists.
2. Create a recordset with the article list whose URL parameter is passed from the men



The image shows a 'Recordset' configuration dialog box. The 'Name' field is set to 'rsArticleList'. The 'Connection' is 'connCMS'. The 'Table' is 'page_pag'. The 'Columns' section has 'All' selected, and the list of columns includes 'id_pag', 'idpag_pag', 'title_pag', 'description_pag', and 'body_pag'. The 'Filter' is set to 'id_pag = URL Parameter id_pag'. The 'Sort' is set to 'None' and 'Ascending'. On the right side, there are buttons for 'OK', 'Cancel', 'Test', 'Advanced..', 'QuB..', and 'Help'.

3. Create another recordset with all the articles belonging to the current list.

The screenshot shows a 'Recordset' dialog box with the following configuration:

- Name: rsArticles
- Connection: connCMS
- Table: page_pag
- Columns: All (selected), listing id_pag, idpag_pag, title_pag, description_pag, and body_pag.
- Filter: idpag_pag = (dropdown) with URL Parameter and id_pag selected.
- Sort: None (dropdown) and Ascending (dropdown).

4. Drag dynamic text from the first recordset (*rsArticleList*) in your page and format it to look like this:

{rsArticleList.title_pag}

{rsArticleList.description_pag}

5. Drag and drop the **title_pag** and *description_pag* fields from the rsArticles recordset in the Bindings tab onto the page, in a new paragraph below the first section. Also add a link to the *view_art* page, named "Read more" and make sure to pass the **id_pag** URL parameter retrieved from the *rsArticles* recordset. This will display one of the records from the second recordset, but in order to display them all, you need to apply a Repeat region on the entire area:

Repeat

{rsArticles.title_pag}

{rsArticles.description_pag} [Read more](#)

6. Save the page and preview it in the browser. Pass it an ID as the **id_pag** URL parameter manually, or use the menu created in the previous part. The result should look like the following:



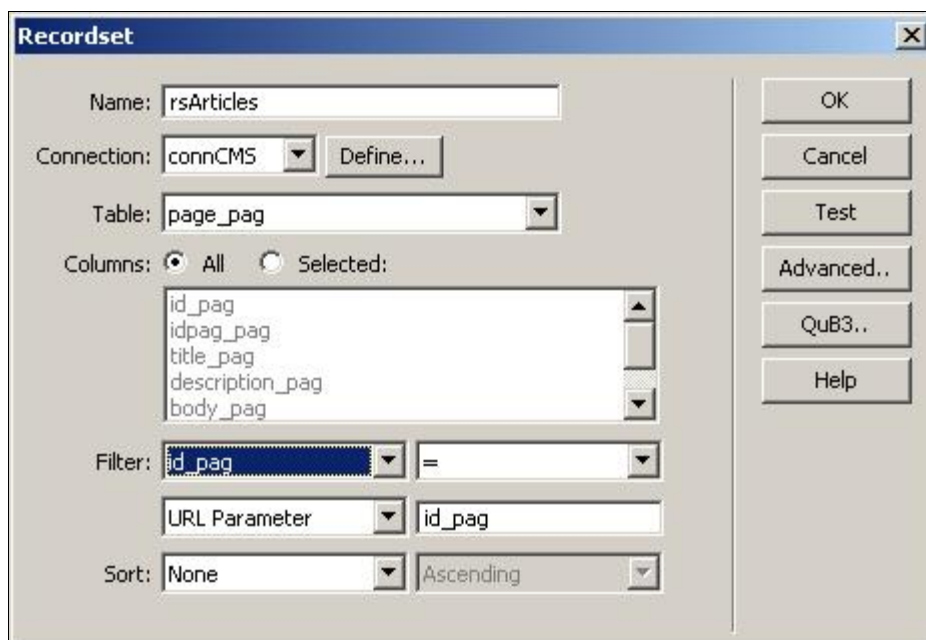
Display article

In this section of the tutorial you will build the page that actually displays the content of an article. As this page will be included in the main index of the site, you can skip adding links that will change the article, or point towards another page.

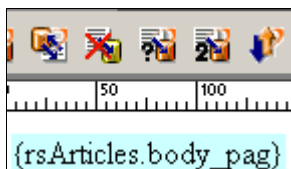
The purpose of this page is to retrieve an article from the database, and display its contents on page. The article ID to be retrieved is passed to the page through an URL parameter, which is fed into a filtered recordset. This way, only the selected article will be displayed.

To create this page, follow the next steps:

1. Open the `front/view_art.php` page in Dreamweaver.
2. Create a new recordset into the page, by clicking on the Plus (+) button of the Bindings tab. From the pop-up menu, select Recordset (Query).
3. The Basic dialog box opens. Configure it, by selecting the database connection you are using for the site, and in the table drop-down menu, select `pages_pag`. Now the recordset retrieves all records from the database, regardless of the `id_pag` parameter that is passed to it.
4. To filter the recordset, select in the **Filter** drop-down menu the `id_pag` table field, and for the condition set it to be equal to an URL parameter called `id_pag`.
5. At this point the page displays only the recordset that is required through the parameter, and you can finish configuring the recordset. Just click on the Ok button to add it into the page. Also, don't forget to enter a name for your recordset: `rsArticles`:



6. Now the `rsArticles` recordset appears in the Bindings tab. Since all this page will do is display the content of the retrieved article, you can simply drag & drop the `body_pag` field from the Bindings tab, onto the page:



7. At this point, the page to display the content of an article is finished, and you can save it and upload it to the server. There is no point in previewing it yet, unless you know the exact URL parameter for an existing article.

When loaded in the browser with a correct *id_pag* value, it will display similar to the image below:



The next step is to integrate the three pages you've created until now (the menu, article lists and article page) into a single file, with the help of **MX Includes**.

Create home page

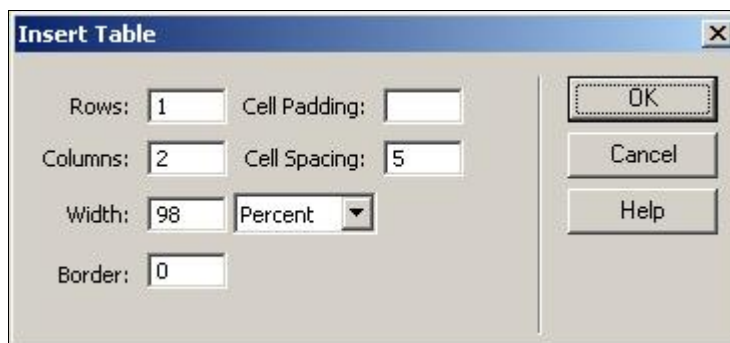
Before starting to build the main page for your site's front-end, you should already have created the pages explained above.

If you skipped one of these sections, you should do it now, as the pages created in each section will now be combined to form the site's index.

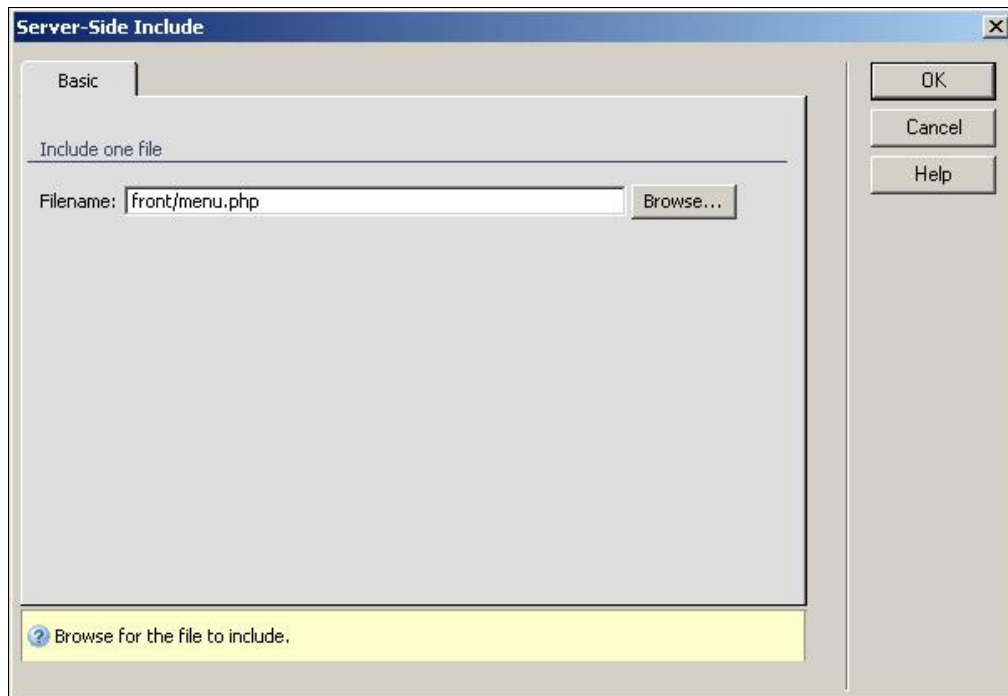
To unite these pages and avoid **Dreamweaver** problems with the included files, you will use another **InterAKT** product: **MX Includes**.

To create the index page, follow the next steps:

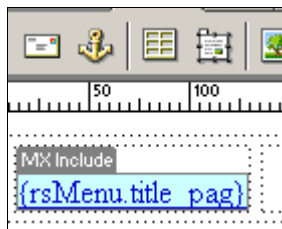
1. Open the *index* page in Dreamweaver.
2. Add a table containing 1 row and 2 columns. Set the cell spacing to 5, in order to separate the cells easier.



3. In the first cell of the table, you will display the menu. Since the menu has been created as a separate page, you can use it with the help of an MX Includes server behavior, namely the Server-Side Include. It acts similar to the standard Server-Side Include (accessed from the Dreamweaver menu: Insert -> Script Objects -> Server-Side Include), but without the inherent problems that are caused by repeating the `<head>` tags.
Place the cursor in the table's first cell, and then apply this server behavior.
4. To access it, click the Plus (+) button in the Server Behaviors tab, and then select MX Kollection -> Server-Side Includes -> Server-Side Include.
5. In the text field of the user interface, select the *front/menu.php* file, either by entering its path, or by using the Browse button to select it from the local folders.



- When you click the OK button, the server behavior will be added to the page, and the *menu* file's content will be displayed:



- In the second table cell, the page content is to be displayed. But this content varies, depending on what is selected:

A welcome page, when the index is viewed for the first time. This will be created at the end of this section, as it will contain only some welcome text.

An article list page, *front/view_list*, when a link in the menu is clicked

The actual article, when the read more link is pressed.

- To solve this problem, MX Includes offers another server behavior: the Server-Side Includes From List. What this does is allow you to define an URL parameter and what pages to load, depending on the values it receives. The URL parameter used with this server behavior is by default named *mod*. You can use any URL parameter name you wish. The only rule is that it has to be consistent throughout the pages (e.g. the same URL parameter name must be used in the Server Side Includes definition, as well as in the menu that passes the links). The three cases mentioned earlier, translate into the following combinations that can be used with the dynamic includes server behavior:

for the welcome page: mod value: none, file to include: *front/welcome.htm*

for the article listing page: mod value: 1, file to include: *front/view_list.php*

for the article page: mod value: 0, file to include: *front/view_art.php*

9. Now open the Server-Side Includes From List server behavior from Server Behaviors -> + -> MX Kollection -> Server-Side Includes, and fill in the fields with the values listed below:

Server-Side Includes From List

Basic | **Parameter**

Specify files to include

Include files: + - ▲ ▼

URL parameter value	Filename
1	front/view_art.php
0	front/view_list.php
0	front/welcome.htm

URL value:

Filename: Browse...

Page title:

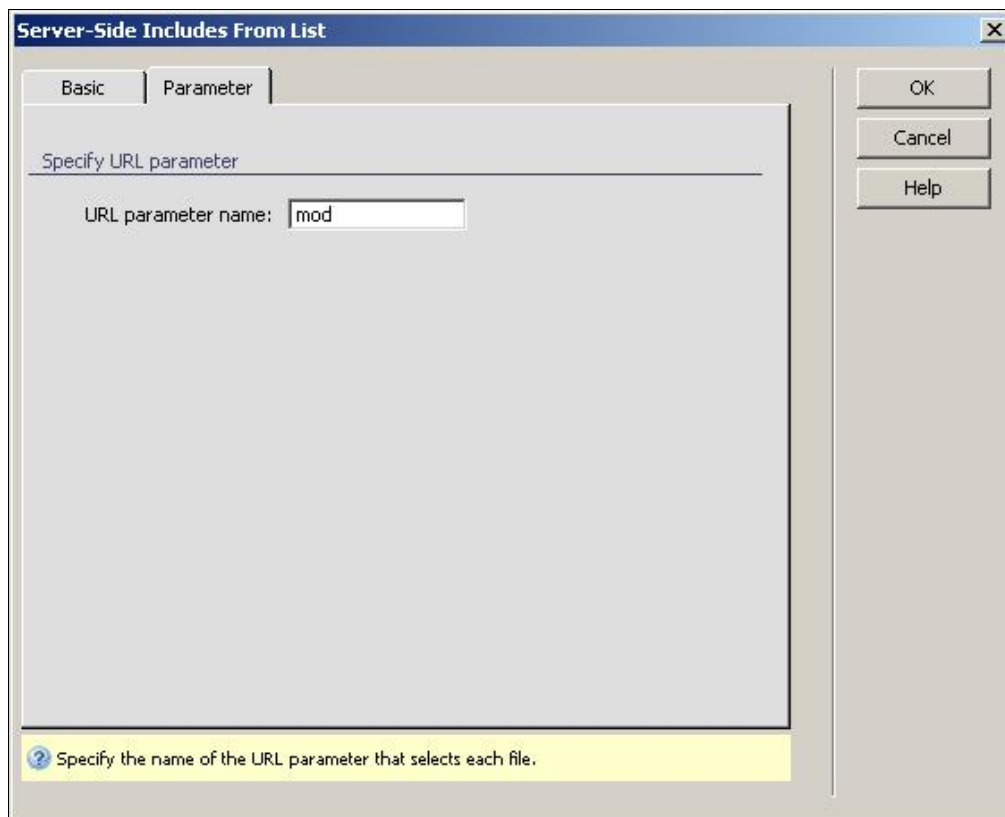
Meta keywords:

Meta description:

Type the corresponding HTML title of the included file.

OK
Cancel
Help

Don't forget to specify your parameter name in the **Parameter** tab. The same parameter name must be passed by the links in the menu, in order to display the correct pages.

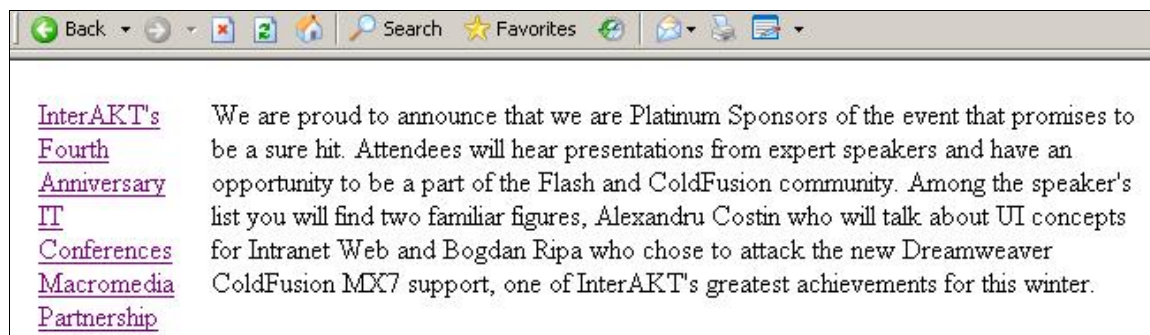


10. Adjust the links in the *menu* and *view_list* files: make them point to the *index* page and make sure they pass, besides the existing *id_pag* URL parameter, a new one, *mod*, with the values set to 0, respectively 1. The *mod* parameter is used by the Server-Side Includes from List server behavior to identify the pages to load.

Let's not forget the *welcome.htm* file. Create this file in the *front* folder, and type in (with heading 1 formatting) Welcome to my CMS. This should do it. Once you've altered the links, and created the welcome page, you can test-drive the site's main page. Save it, and preview it in the browser. Since this is a public page, you didn't apply the **Restrict Access To Page** server behavior, and as such you do not need to login to view it:



If you select an item from the menu, and then an article from the list, it will load the *view_art* page displaying the selected article:



List articles and implement multiple delete

In this section of the tutorial, you will build the article list. From the list you can add an article, and edit or delete an existing article. The easiest way to implement this is to use a NeXTensio list. The page that will contain this list is the *admin/list_art* page.

Before creating the actual list, you should remember that this page belongs to the administrative section of the site, and as such, not everybody should be allowed to access it. This is why the first thing to do in regards to the *list_art* page (besides opening it in Dreamweaver, of course) is to apply a **Restrict access to page** server behavior. You can open the Restrict Access to Page server behavior from the **Server Behaviors tab >+ > MX Kollection > User Login**. Configure it to restrict access based on username and password alone.

First make sure that you already have configured your Control Panel -> Login Settings to use the **user_usr** table for the content management system tutorial. If not, you should do so before continuing.

Now apply the **Restrict Access to Page Server Behavior** from the **Server Behaviors tab > MX Kollection > User Login**. Configure it similar to what is shown in the User Authentication tutorial, but using only the username and password.

Once you've secured the page, you can move on and add the actual article listing. To do so, you will have to start and configure the **NeXTensio List Wizard**.

Click its button from the **MX Kollection** tab of the **Insert** bar, and configure it as follows:

- In the first step, select the database connection you've created at the beginning of this tutorial.
 - For the **Get data** field, select the Table option.
 - Use *connCMS* for the **Connection**
 - Select the *page_pag* table, as it contains all articles.
 - Set the **Primary Key Column** to use the *id_pag* field.
 - Change the Default detail location from form.php to detail_art.php.

The screenshot shows the 'Create NeXTensio List Wizard' dialog box at Step 1/4: 'Select the table to display'. The dialog is divided into several sections:

- Specify the method to retrieve data:** 'Get data from:' is set to 'Table'.
- Specify data source information:** 'Connection:' is 'connCMS', 'Table:' is 'page_pag', and 'Primary key column:' is 'id_pag'. There is a checked 'Numeric' checkbox.
- Specify the detail page:** 'Detail page:' is 'detail_art.php'.
- Number of records:** Set to '10'.

At the bottom, there is a yellow bar with a question mark icon and the text 'Specify the page that contains the NeXTensio form.' Below this are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

- For the second step, remove the *body_pag* field. For the *idpag_pag* field you should display the actual titles instead of ID's. To achieve this, select the element in the grid, enter the

header text (e.g. Parent page) and then select for the value to use a look-up table (the page_pag table, with the title for labels and the ID for values). Leave the rest at their default:

Create NeXTensio List Wizard

Step 2/4: Select the list columns

Specify list columns

List columns: + -

Column	Header	Char width
idpag_pag	Parent page:	20
title_pag	Title:	20
description_pag	Description:	20
date_pag	Date:	20
visible_pag	Visible:	20

Header: Parent page:

Char width: 20

Get value from: Current table Look-up table

Database table: page_pag

Label column: title_pag

Value column: id_pag

Interakt

Select the column that stores the labels of the current field.

< Back Next > Finish Cancel Help

- Each **NeXTensio** List can have a row for filtering the list results. For the third step, you can specify how you want the filters to appear. For some fields you may want a text field, others a drop-down menu, checkboxes for boolean values etc. The only change is made for the visibility field: set it to use a **Checkbox: 1,0** as filter element: when the checkbox is selected, only visible (with the value 1 in the database) elements will be displayed.

Note: If you are using a **Microsoft Access** database, in the **Submit as** drop-down menu, another option will be available: **Date MS Access**. Select this option when submitting the date. See the NeXTensio List Wizard user manual for more details.

Create NeXTensio List Wizard

Step 3/4: Define the filter

Specify the global filter behavior

Use filter: Yes

Specify the filter settings for each column

Column	Field type	Display as
idpag_pag	Numeric	Menu
title_pag	Text	Text field
description_pag	Text	Text field
date_pag	Date	Text field
visible_pag	Numeric	Menu

Display as: Menu

Field type: Numeric

Max chars:

Interakt ? List of fields to display in the filter.

< Back Next > Finish Cancel Help

- The final wizard step is where you will find some display options for the list - skins, row effects, and the possibility to have buttons and navigation above and below the display list. Duplicating is useful if you plan to have a long list, so the user will not have to scroll up or down.

Create NeXTensio List Wizard

Step 4/4: Set up the list layout

Specify NeXTensio List settings

Default order: title_pag Ascending

Move up/down column: None

Change skin parameters

Current skin: **Kollektion** Change skin

Duplicate buttons: Yes

Duplicate navigation: Yes

Table row effects: Yes

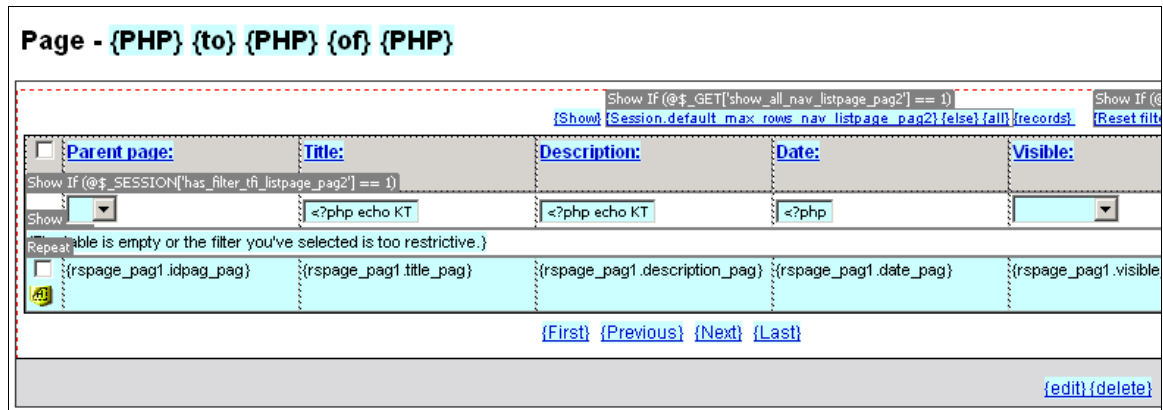
Show links as buttons: Yes

Has row counter: No

Interakt ? Check if you want a row counter to be displayed.

< Back Next > Finish Cancel Help

When you click the **Finish** button, the **HTML** elements, as well as some server behaviors will be added to the page. Next to the regular table cells and links, some form elements are added to allow filtering the list after different criteria. This is how the page should look like in **Dreamweaver**:



If you try to preview the page in a browser, you will be redirected due to the **Restrict Access to page** server behavior. After you login however, you can access it.

While the list was being created, you've already implemented the second objective: being able to delete several articles at a time. When the list is generated, it contains checkboxes and buttons to facilitate multiple deletions. Once the form page will be created, the delete operation will become functional as well.

You can use the list's filter fields to show only elements respecting certain conditions. Through this list, sorting is also possible, by clicking on the column title. Once the list is sorted after a column, visual indicators will be displayed next to the title, to allow changing the sort type (**Ascending** or **Descending**).

After you've created the article listing by using a **NeXTensio List**, you have to create the list's counterpart: the NeXTensio Form which is used when links like add or edit clicked.

Add and edit articles using the same form

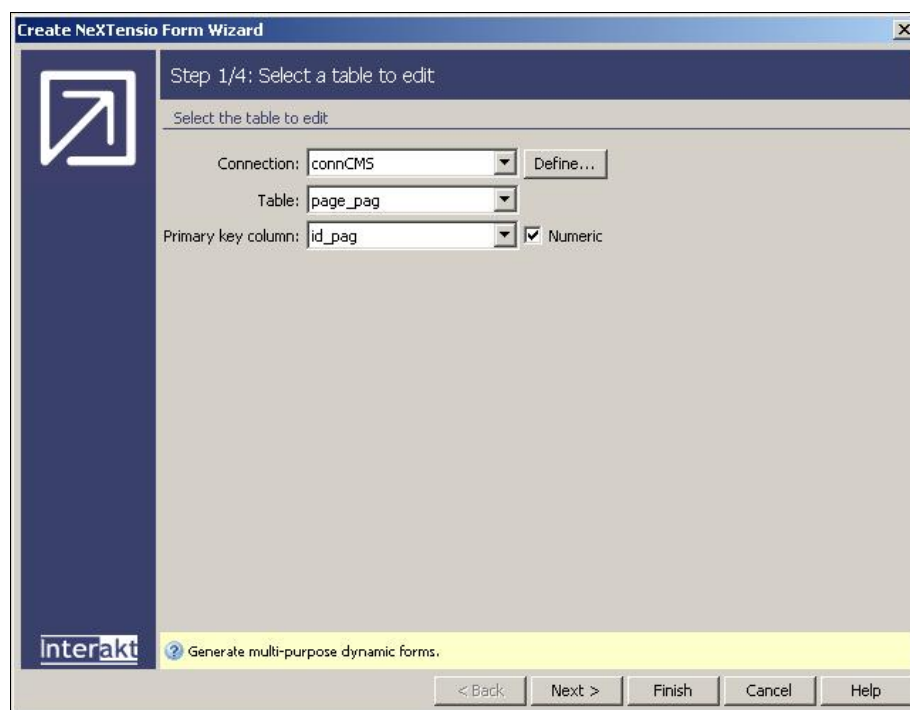
Having a simple list of articles is of no real use in a CMS. You have to be able to add, delete and also edit pages using a simple interface. When you created the listing using the NeXTensio List Wizard in the previous tutorial page, in the first step, you specified a file to use for the form: *detail_art.php*. After testing the page in the wizard, you may have noticed that when you click on the add / edit links, the browser points to the *detail_art* page, which is, for the moment, blank. Also, the delete buttons seem to have no effect whatsoever.

To create this page, you will use the **NeXTensio List's** counterpart: the **Create NeXTensio Form Wizard**, which will create all of the required elements.

Before you start this wizard, you should secure this page, in the same manner as the article listing page: apply the **Restrict Access to Page** server behavior, based on user name and password.

Once you've got your page secured, click on the **Create NeXTensio Form Wizard** from the **MX Kollection tab** of the **Insert panel**, and configure it as follows:

1. In the wizard's first step, enter details regarding the table to use, and the means to access it:
 - a) Select the connection you've defined for your site in the **Connection** drop-down menu.
 - b) Select the *page_pag* table, as the table on which the insert/update/delete operations will take place.



2. In the second step of the wizard, you have to define the columns that will be used in the transactions, and what form fields will be used:
 - For the *idpag_pag*, which is the self-foreign key that determines the parent article or the one that is currently inserted, the user should not be allowed to type values directly. To avoid this, you should set its form element to Menu.
 - By using a menu, you can use dynamic data, and restrict the choices to available ones. Once you select the menu element in the Display as drop-down, two new buttons will become available: Add new recordset and Menu properties.
 - Click on the Add recordset button to create a new recordset, that retrieves all entries having the *idpag_pag* foreign key equal with 0. This is done in a similar manner to the recordset created for the navigation menu.

- Once you have created the recordset, you can click on the **Menu properties** button and define which field is to be used for the labels, and which for the values. Configure the dialog box to use the **title_pag** column as label, and the **id_pag** as value. Also change its label to something more suggestive (e.g. Parent article).
- The second form field to alter is body_pag. Set it to be displayed on page as text area, and change its label to **Page content**.
- The **visible_pag** form field only has two options: visible and invisible. This is stored in the database as 1 and 0. Due to the user interface, the field is configured as a checkbox (as set in the list). To make it more suggestive, you will change the element to a menu, which contains two options: **Visible** and **Invisible**. Choose **Menu** in the **Display as** drop-down menu, and then click the **Menu Properties** button to add the two items: label - Visible, value - 1, and label - Invisible, value - 0.
- The last field, storing the date of the article's insertion should be automatically completed with the current date. To achieve this, first set the Display as option to text, and in the Default value, enter the following mark-up: {NOW}. This will be replaced at runtime with the actual date.

Step 2/4: Configure the form fields

Specify form fields properties

Form fields: + -

Column	Label	Display as	Submit as	Show on
idpag_pag	Parent page:	Menu	Numeric	Insert and Update
title_pag	Title:	Text field	Text	Insert and Update
description_pag	Description:	Text field	Text	Insert and Update
body_pag	Page content:	Text area	Text	Insert and Update
date_pag	Date:	Text	Date	Insert and Update
visible_pag	Visible:	Check box	Checkbox...	Insert and Update

Label:

Show on:

Display as:

Submit as:

Char width: Max chars:

Default value:

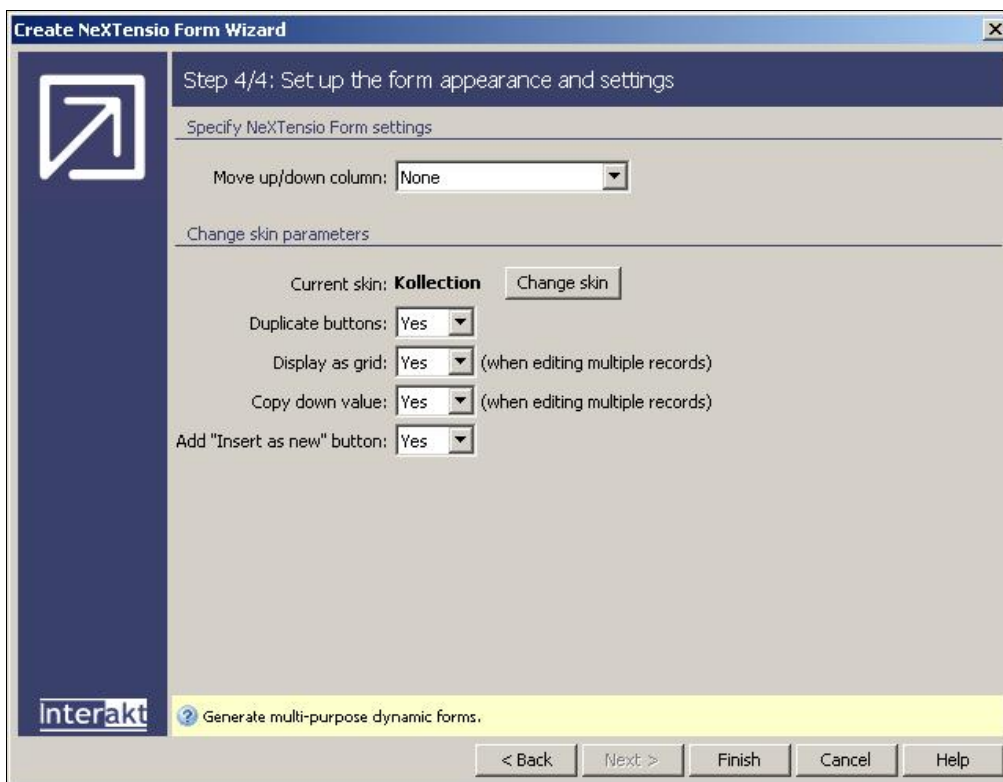
interakt

Select the default value for the selected field.

< Back Next > Finish Cancel Help

3. The wizard's third step deals with applying validation rules on the form fields. As there are no fields with special requirements, you can safely skip this step. Also, if you do not have MX Form Validation installed, there is no third step of the wizard.

- In the final step you can specify some display settings, or leave the fields at their default.



The screenshot shows the 'Create NeXTensio Form Wizard' dialog box, Step 4/4: Set up the form appearance and settings. The dialog box has a title bar with the text 'Create NeXTensio Form Wizard' and a close button. The main area is divided into two sections: 'Specify NeXTensio Form settings' and 'Change skin parameters'. In the 'Specify NeXTensio Form settings' section, there is a dropdown menu for 'Move up/down column:' with the value 'None'. In the 'Change skin parameters' section, there is a 'Current skin:' label with the value 'Kollektion' and a 'Change skin' button. Below this are four dropdown menus: 'Duplicate buttons:' (Yes), 'Display as grid:' (Yes) (when editing multiple records), 'Copy down value:' (Yes) (when editing multiple records), and 'Add "Insert as new" button:' (Yes). At the bottom left, there is the Interakt logo and a help icon with the text 'Generate multi-purpose dynamic forms.'. At the bottom right, there are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Now click on the **Finish** button to close the wizard's dialog box, and add all elements into the page. In Dreamweaver, you will have now all of the HTML form elements and some server behaviors. This page allows you to add, edit and delete entries. It can be called either from the article listing page, to edit or delete a specific record, whose ID is passed through an URL parameter, or directly, in which case, it will act as a normal insertion form:

{Display error message.}

Show If (@\$_GET['id_pag'] == "")

{Insert} {else} {Update} Page

Repeat

{Initialize Counter} {Increment Counter}

Show If (@\$totalRows_rspage_pag > 1)

{Record} {Display Counter}

Parent page:	<input type="text" value=""/> {Error}
Title:	{rspage_pag.title_pag} {Hint} {Error}
Description:	{rspage_pag.description_p} {Hint} {Error}
Page content:	{rspage_pag.body_pag} {Hint} {Error}
Date:	{rspage_pag.date_pag}
Visible:	<input type="checkbox"/> {Error}

Show If (@\$_GET['id_pag'] == "")

Insert {else} Insert as new Update Delete Cancel

Because you've applied the **Restrict Access to page** server behavior, you cannot simply preview the page in the browser, as it would redirect you to the login page. Once you login though, visit the **Article listing** page, and click on the edit link aside one of the articles. A page similar to the following will be displayed for the update operation:

Update Page

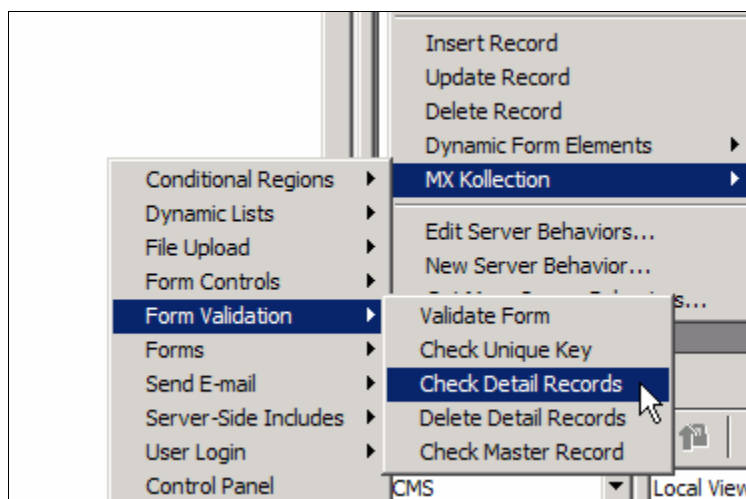
<input type="button" value="Insert as new"/>		<input type="button" value="Update"/>	<input type="button" value="Delete"/>	<input type="button" value="Cancel"/>
Parent page:	<input type="text" value="Select one..."/>			
Title:	<input type="text" value="CeBIT Conference"/>			
Description:	<input type="text" value="Bucharest, February 23, 2005 - InterA"/>			
Page content:	<input type="text" value="During 10-16 March 2005 in the German town of Hannover 6000 companies from the IT industry will join forces into the largest exhibitions of the kind in Europe."/>			
Date:	<input type="text" value="3/20/05"/>			
Visible:	<input type="text" value="Visible"/>			
<input type="button" value="Insert as new"/>		<input type="button" value="Update"/>	<input type="button" value="Delete"/>	<input type="button" value="Cancel"/>

Now, through these two pages, you can completely administer the articles.

Restrict part delete if it has associated articles

In this page you will add an additional check before a part is deleted. The check is designed to prevent a part from being deleted if it has sub-parts underneath it.

1. Open the page containing the **NeXTensio** Form, *detail_art*, and add the **Check Detail Records** behavior from the **Server Behaviors** menu:



2. When the Check Detail Records interface opens, configure the fields as follows:
 - **Detail table:** *page_pag*
 - **Foreign Key:** *idpag_pag* (self foreign key)

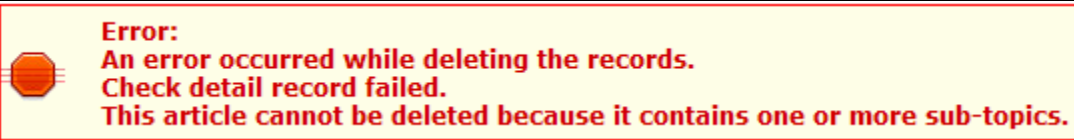
- **Error Message:** type a meaningful error message in the text area.

3. Click the OK button. The server behavior will then be added to the page. Save the page and upload it to the server. You can test the page by logging in to the site, and going to the *list_art* page in the admin section. Try to delete a record, like IT Conferences, which contains sub-articles:

<input checked="" type="checkbox"/>	IT Conferences	Our Agenc
<input type="checkbox"/>	Macromedia Partnersh...	InterAKT h
<input type="checkbox"/>	Web Conference	The sixth T

edit delete

After clicking delete, an error message (the one you defined) should appear on the *detail_art* page:



With the **Check Detail Records** behavior, the only way to delete both an article and its sub-articles is to individually delete the sub-articles first, then delete the main article. If you decide this is too much of an inconvenience, you can delete the **Check Detail Records** behavior, and add **Delete Detail Records** one instead. It is configured in the same way, except without an error message field. In the next chapter, you will improve your Content Management System by adding advanced features, including auto-archiving old articles, creating rich-text content using the KHTML, and search engine optimization.

Improve the Content Management System

In this section you will improve the existing CMS pages by adding new functionality and simplifying certain tasks.

The improvements you will create are:

- An archiving feature, for articles older than a specified date.
- Create rich article content visually, with KHTML Lite.
- Improve the administration pages, by replacing standard form elements with widgets.
- Optimize content for search engines.

The products you will use in order to achieve these goals are:

- **MX Widgets**
- **ImpAKT**
- **KHTML Lite**

As this section of the tutorial is slightly smaller than its predecessor, it should only take 30 to 40 minutes to complete the pages.

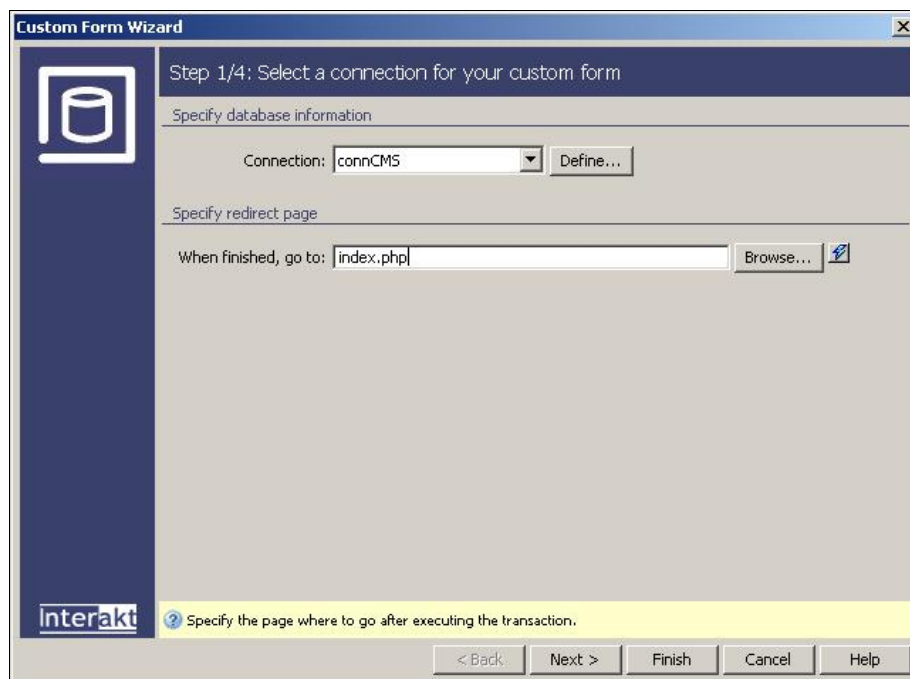
Auto-archive old articles

The first improvement to add to your site is the possibility to archive old articles. If you use your site in a more intensive manner, you may end up having users view articles from months ago, which have no relevance in the present context. Archiving is a more useful option than deletion, because you might need the files for future use.

The ***visible_pag*** field makes the archiving implementation very simple. This field determines whether an article will be displayed or not. You could edit each article and set the field to 0 (not visible), but for a large number of articles this takes a lot of time. This is why you will create a new administration page, allowing you to select a date from a calendar. All records that were posted before that date, are to be set as invisible.

To create this page, follow the next steps:

1. In the *admin* folder, create a new file, called *archive* Open it in Dreamweaver.
2. Since this is an administrative page, you should apply a Restrict Access to Page server behavior..
3. Next, you must insert the field that will allow selecting a date. To implement both the form elements and the SQL logic, use the Custom Form Wizard, that is accessible from the MX Kollection tab of the Insert bar.
4. Once the Custom Form Wizard opens, configure it in the following manner:
 - In the **Connection** drop-down select the database connection used in your site. If you do not have a connection, create a new one, by clicking on the Define button.
 - In the **When finished, go to** text field, enter the page that will be opened when the operation finishes: *index*



- In the second step of the wizard, you must add a **transaction** field that will allow you to enter the date. Click the Plus (+) button of the **Form fields** grid, and enter the label: Archive articles older than. Once added, change its submit as property to date. If left as text, it will provide bad input to the database transaction.

Note: If you are using a Microsoft Access database, in the **Submit as** drop-down menu, another option will be available: **Date MS Access**. Select this option when submitting the date.

Custom Form Wizard

Step 2/4: Configure the form fields

Set form fields properties

Form fields: + -

Field	Label	Display as	Submit as
Archive_articles_...	Archive articles older ...	Text field	Date

Label: Archive articles older than:

Display as: Text field

Submit as: Date

Default value:

Interakt ? Enter the default value for the current form field.

< Back Next > Finish Cancel Help

- If you have MX Form Validation installed, an additional step is available in the wizard: the third, providing a way to validate input. Simply skip this step, as it is not required.
- In the final step of the wizard, you have to write the SQL query that will perform the actual operation. It is a simple update query, involving the field added at the last step as dynamic data:

```
UPDATE page_pag SET visible_pag=0
WHERE date_pag<{Archive_articles_older_than}
```

- Just paste the above code in the SQL area. The dynamic mark-up can be selected by clicking the InterAKT Dynamic Data icon, and choosing the right transaction field.

Custom Form Wizard

Step 4/4: Enter the SQL query for the custom transaction

The SQL query for the custom transaction

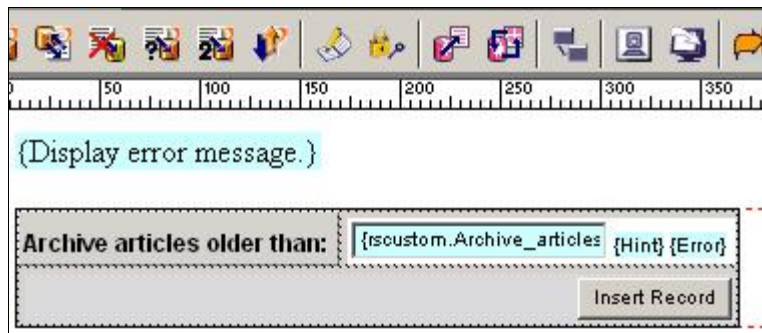
SQL query:

```
UPDATE page_pag SET visible_pag=0
WHERE date_pag<{Archive_articles_older_than}
```

Interakt ? Enter the SQL code that will be executed by the transaction.

< Back Next > Finish Cancel Help

5. When you click the Finish button, a form will be added into the page, along with some server behaviors. If you need to edit the transaction properties, simply double-click the corresponding server behavior in the Server Behaviors tab of the Application panel.



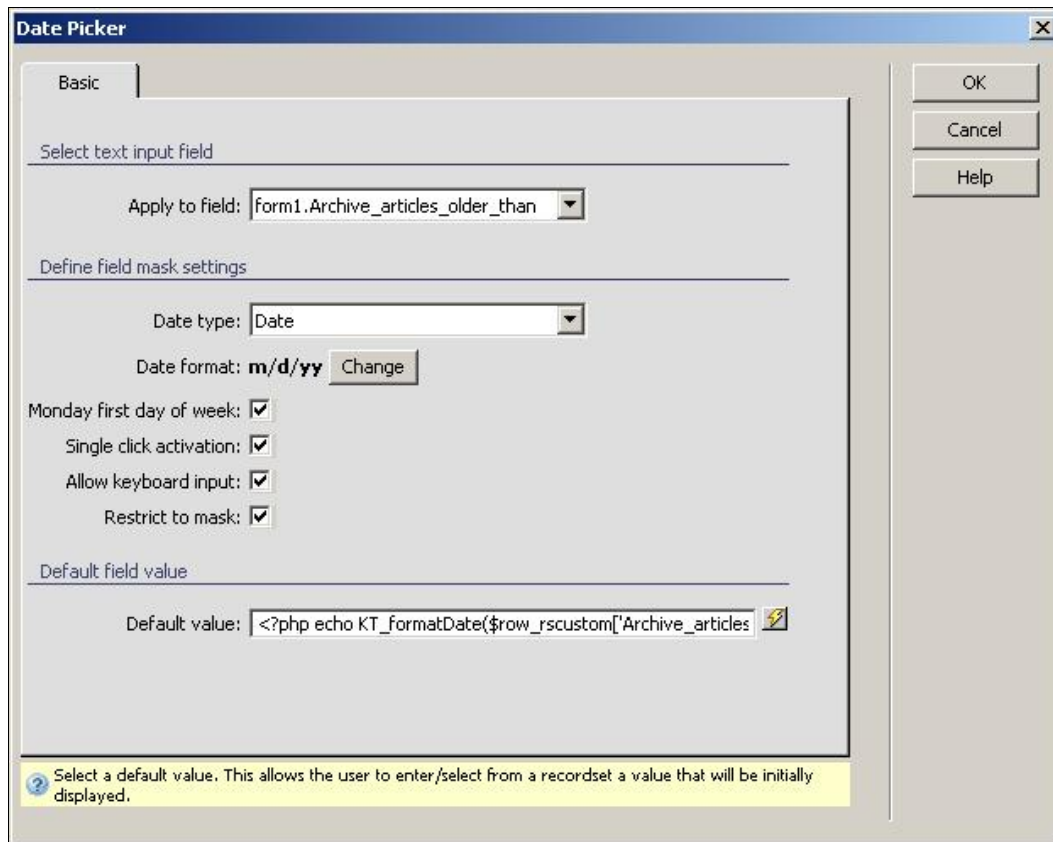
By default, the Custom Form Wizard will name the submit button as Insert Record. Change it to Archive.

If you preview the page in the browser, after you login, a standard text-box where you can enter a date is displayed. If you enter a valid date, it will perform the requested operation and return you to the admin index.

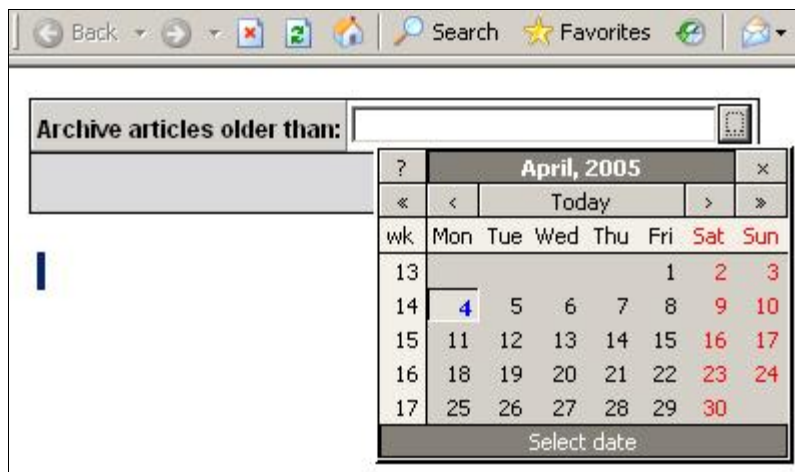
The problem with this approach is that you have to enter the date by hand, and in the correct format. Why not use a calendar, where you can visually select the date?

To improve the standard text-field, you must have MX Widgets installed. If you do, then simply select the text-field on page, and apply the DatePicker text field server behaviors, from the Server Behaviors tab->MX Kollection->Form controls.

Once the dialog box opens, you can configure some properties for the calendar, like whether to use single - click or double-click for activation, and more. Select the options that suits your taste, and click OK to apply it:



In Dreamweaver, there is little modification to be seen, unless you look at the code. When you save and preview the page in the browser, however, a new button is displayed next to the text-field. When you press it, a calendar will open, allowing you to select the date:



When you click the **Archive** button, the SQL query will run, and will update records as necessary. You can verify this behavior, by opening the **Article list** in the administration area, and notice the new values.

The next improvement you'll add to your site, is to replace the text-area form element used to enter the article content with **InterAKT's KTML Lite** control, allowing you to enter rich content visually.

Create rich content visually: KTML Lite

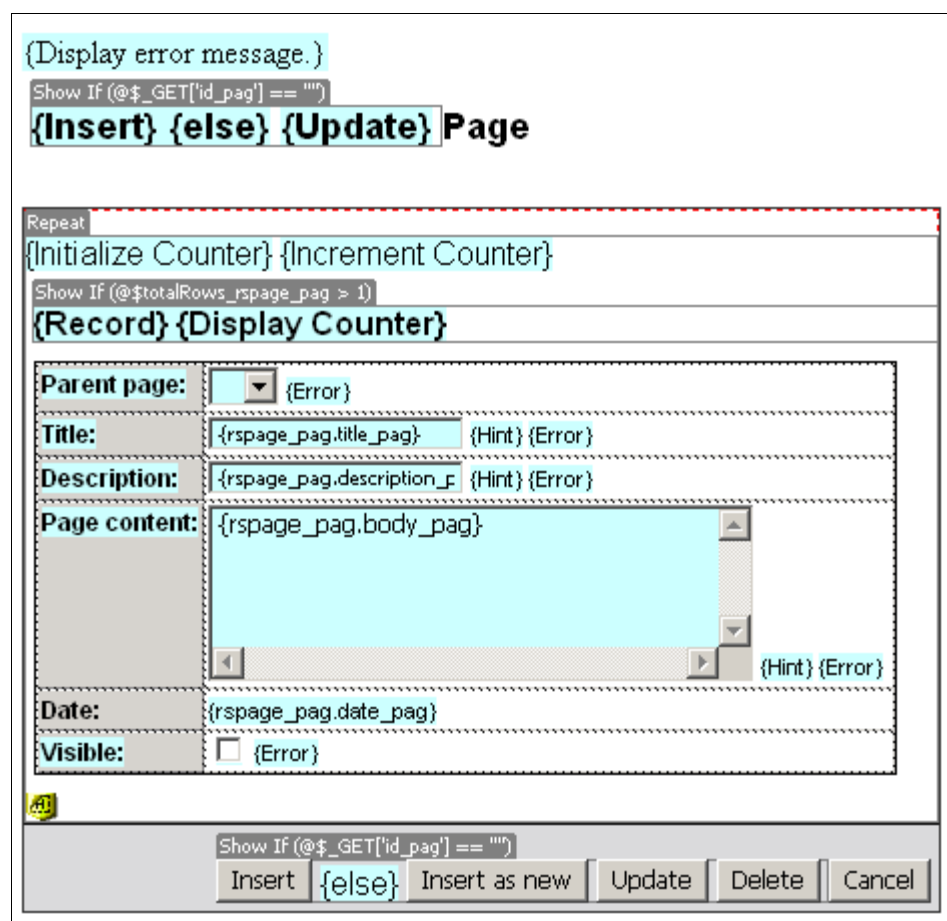
When you create new articles from the administration section, you are limited by the fact that the only element allowing the entry of content is a standard text-box. Of course, nothing stops you from entering **HTML** code into the text area, and it will be rendered just fine, but having to write a lot of tags and properties just to get a simple image next to some text is not the most efficient way to do things.

In this scenario, an online HTML editor, embedded in your page, would be a useful replacement for a standard text area.

Therefore, this page will use **InterAKT's** free editor, **KTML Lite**. Since it comes as a separate extension from **MX Kollection**, you will need to download and install it from the **InterAKT** site.

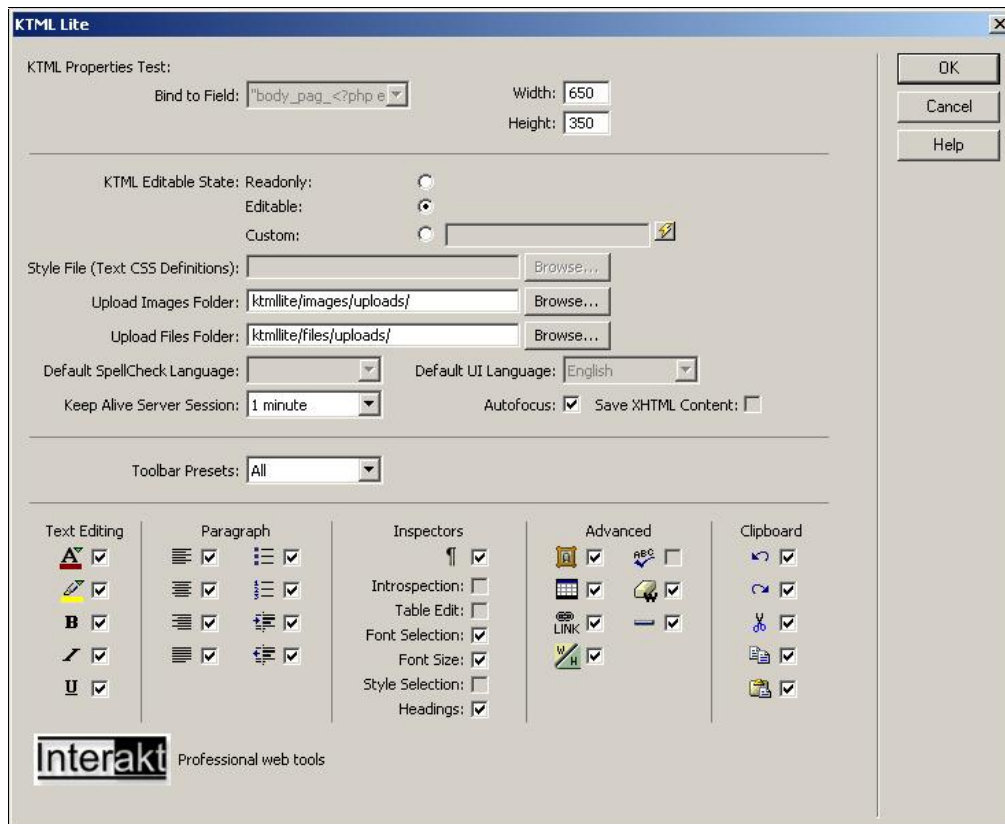
Once installation is complete, open the *admin/detail_art* page in **Dreamweaver**. **KTML** replaces an existing text area, so you do not need to re-create the entire page, and the process is simple, requiring only a few actions:

In the article detail page you have a text-area that is currently used to enter the page content. Click on it to select it.

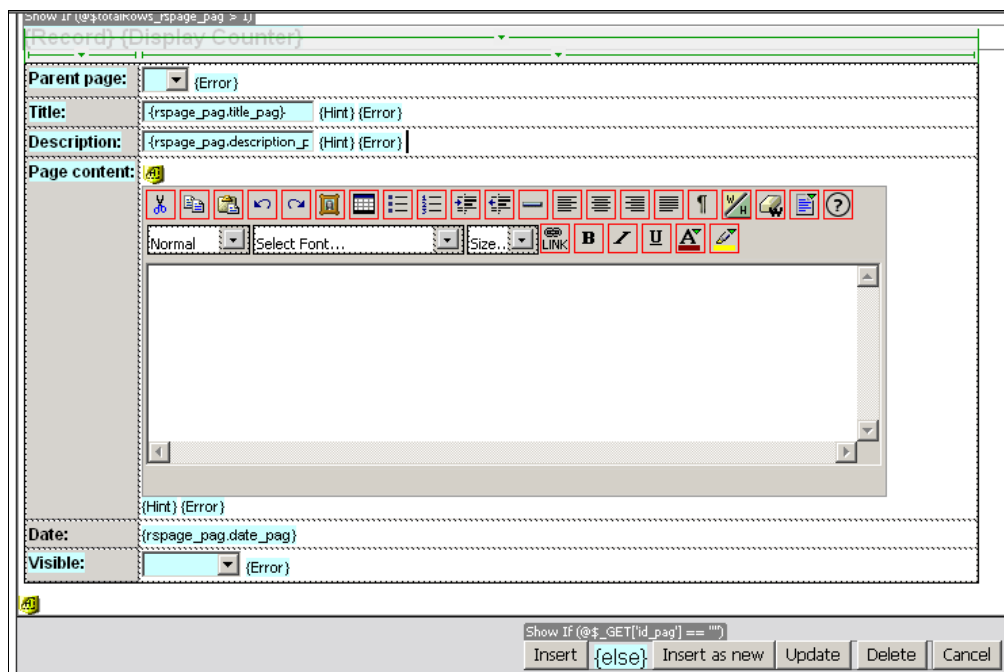


Apply the KTML Lite server behavior: in the Server Behaviors tab, click the Plus (+) button, then select KTML-> KTML Lite.

When the KTML configuration dialog box opens, it will first ask if you want to check the InterAKT servers for a newer version. Assuming you've recently downloaded the extension, you can safely skip this step. Then define the correct options for your particular control: the folders for uploaded images and files, the toolbar buttons to display, the control's size, etc:



Once you decide what size the editor will have, and what tools (e.g. buttons on the toolbar) the administrators will be able to use, hit the OK button to close the interface and apply the server behavior. The content text-area has been replaced in Dreamweaver with a KHTML actual size translator:



Now save and upload the file to the remote server. To test it, open the site login page and login with a valid account. If you haven't changed the users table, you can log in with admin/root. Then click the list articles link, and edit one of the existing articles. Now you can visually create HTML content, that will render exactly as you enter it.

Improve administration forms: MX Widgets

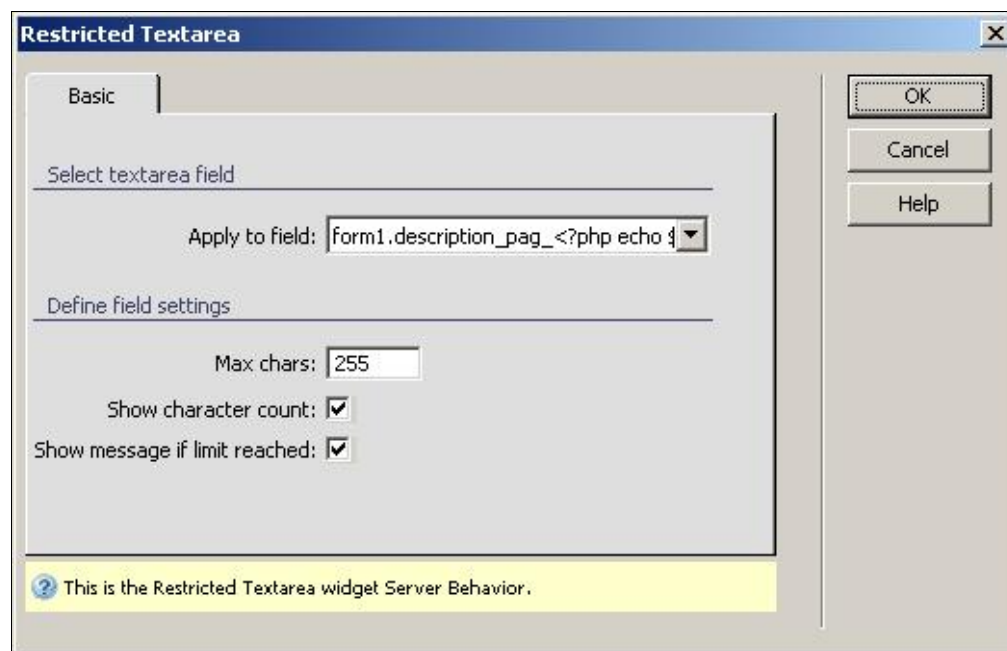
In this section of the tutorial, you will improve the form elements added automatically by NeXTensio. Improvements to be made include allowing the administrator to restrict the number of characters that can be used for the short description, or use a created on / modified on feature.

For the first improvement, open the *admin/detail_art* page in Dreamweaver. This page contains the form that allows editing or inserting new articles, and it contains several form elements: a drop-down menu for the part selection, a text-field for the title, a text field for the short description field, and a KHTML control, added in the previous section, for the page content. Also, you can decide if the article is visible or not, by using a drop-down menu.

The element you'll modify is the short description text-field. To allow entering the description easier, convert the text field into a text area (select the text field and in the Property Inspector, select the Multi Line radio button). As it will be stored in the table as a 255 character length field, there should be a way to stop users from entering more. The Restricted text area widget does just that.

To apply the Restricted Text Area server behavior, select it from the Server Behaviors tab -> MX Kollektion-> Form Controls.

Configure it to display the character counter next to the text area, and set a message to be displayed when the maximum number of characters has been reached:



When done configuring the server behavior, click the OK button to apply the server behavior. Besides the new server behavior in the server behaviors tab, there are no other major changes in the page. When you preview in the browser, a text box displaying the number of characters that you can enter is displayed next to the text area used to enter the short description of the article:

Update Page	
Insert as new	
Parent page:	Select one... ▼
Title:	CeBIT Conference
Description:	Bucharest, February 23, 2005 - InterAKT Online announces its
	▲ 127 ▼

To add the created on / modified on features for articles, you would first have to modified your database, and add a new field which will store the modification data. Then, you must use the Manage NeXTensio Form Wizard to add the new field to the form. Configure the original date to be displayed on Insert only, and the modification date field only on update. To learn exactly how to implement this improvement, check the How To implement created on / modified on section of the manual.

Optimize content for search engines

In order to optimize your site for greater search engine recognition, it is first necessary to become familiar with meta description and meta keywords.

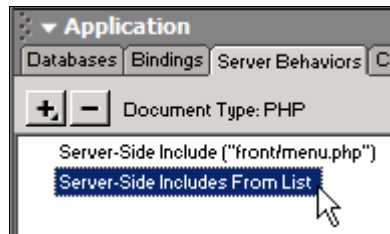
The meta keywords and the meta description contribute significantly to the Search Engine Optimization, helping with the **Google** ranking. When surfing the web looking for specific pages, the search engine will use the keywords of a page to generate the results. The results list will contain the meta description of the pages found.

Note: here is an example of how these HTML tags look like (`<meta>` tags are included within the `<head>` tag along with the `<title>` tag).

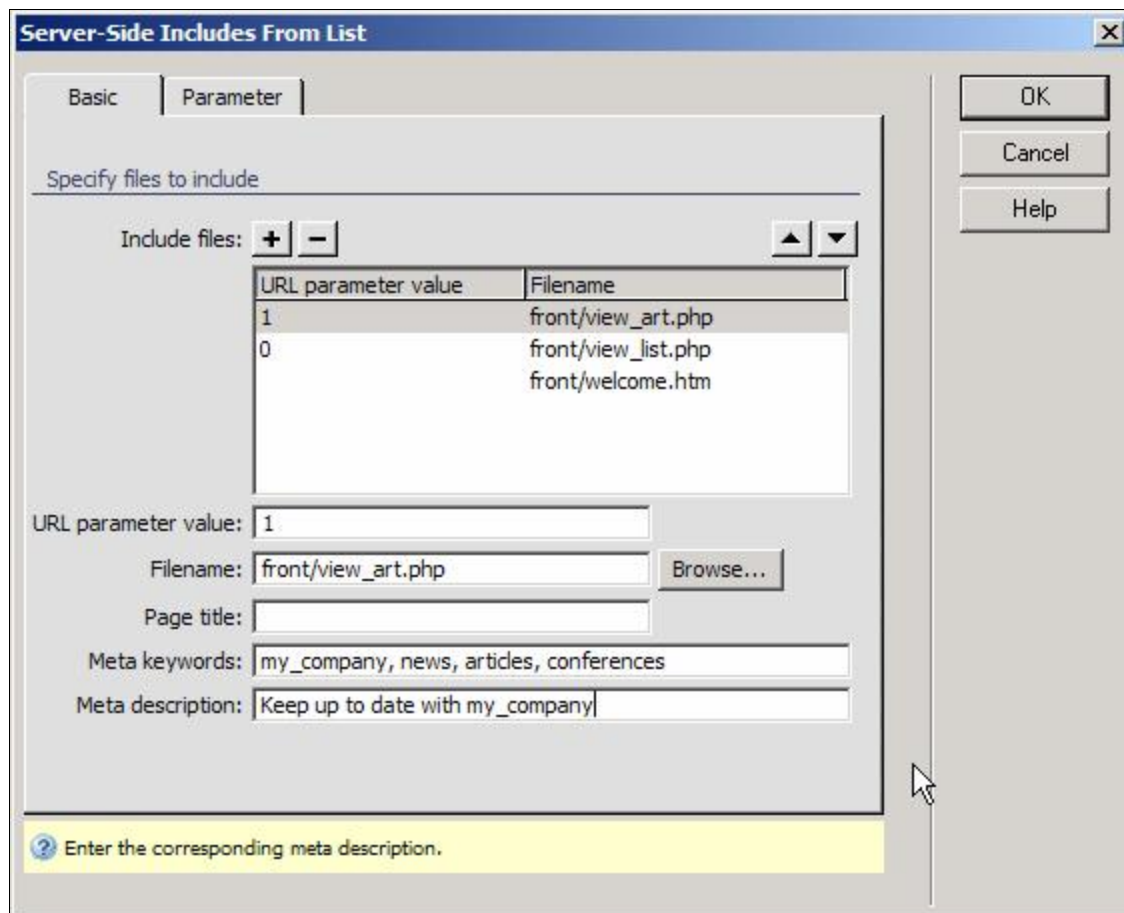
```
<meta name="keywords" content="dreamweaver extensions, web development,
dynamic website" />
<meta name="description" content="Dreamweaver Extensions for Dynamic Web
Sites." />
```

The **Server Side Includes from List** behavior from **MX Includes** lets you specify Meta data about your site.

1. Open the main `index.php` file and click on the Server Behaviors tab.
2. In this list you should see the Server Side Includes from List behavior. Double click on it.



3. In the basic tab, you will see options for entering meta descriptions and keywords:



4. The Meta fields let you specify information to help your site get recognized by search engines. Just enter some appropriate information about your site, and you're done.

Now that you've finished the CMS tutorial you have a working system to help manage your site information.

Other Resources

Other Dreamweaver Extensions from InterAKT

- KTML
- MX Kart
- MX Site Search
- MX RSS Reader-Writer
- MX Dynamic Table Sorter
- MX Coder Pack
- MX Dynamic Charts
- MX CSS Dynamic Menus

Contact Us

- **Address:**
1-11 Economu Cezarescu ST, AYASH Center, 1st floor
Sector 6, ZIP 060754, Bucharest, Romania
- **Web:** <http://www.interaktonline.com/>
- **E-mail:** contact@interaktonline.com
- **Phone:** +4031 401.68.19 or +4021 312.51.91
- **Fax:** +4021 312.53.12

Copyright

Windows is a trademark of Microsoft, Inc.

Dreamweaver MX is a trademark of Macromedia, Inc.

Redhat is a trademark of Redhat, Inc.

Copyrights and Trademarks

Copyright 2000-2005 by InterAKT Online.

All Rights Reserved. This tutorial is subject to copyright protection.

PHAkt, ImpAKT, NeXTensio, MX Query Builder, Transaction Engine, MX Includes, KHTML, MX Kommerce, MX Kollection, MX Widgets, MX Looper, MX Dynamic Charts, MX CSS Dynamic Menus, MX Tree Menu, MX Form Validation, MX File Upload, MX Send E-mail, MX User Login, MX CSV Import-Export, MX Kart, MX Site Search, MX Dynamic Table Sorter, MX RSS Reader-Writer, MX Coder Pack, MX Dynamic Charts are trademarks of InterAKT Online.

All other trademarks are acknowledged as the property of their respective owners.

This document and the product to which it pertains are distributed under licenses restricting their use, copying, distribution, and decompilation.

No part of this document or of the associated product may be reproduced in any form by any means without prior written authorization of InterAKT Online, except when presenting only a summary of the tutorial and then linking to the InterAKT website.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Send comments and suggestions to products@interaktonline.com



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