

Tutorial
Contact Management
Application



Table of contents

Introduction	3
Plan the Contact Management Application	4
Add, Edit and Delete Companies and Contacts	6
Display companies and contacts: Nested Repeated Region	7
Add companies: Insert Record Form Wizard	14
Creating the page design	14
Configure the Insert Record Form Wizard	15
Add companies page: browser behavior	18
Edit companies: Update Record Form Wizard	22
Creating the page design	22
Configure the Update Record Form Wizard	22
Delete companies: Delete Record Wizard	26
Create the page design	26
Add, edit and delete contact persons: Interface Persistence	30
Building the insert page	30
Building the update page	33
Building the delete page	34
Validate Contact Information	36
Check e-mail and phone: Validate Form	37
Remove associated contacts: Delete Detail Records	39
Verify associated company: Check Master Record	41
Improve the Contact Management Application	43
Insert both company and main contact	44
Add more contacts after company insert	49
Alter the page design	49
Add application logic	50
Send automatic birthday greeting	55
Display number of contacts per company	62
Other Resources	68
Copyright	69

Introduction

In this tutorial you will use **MX Kollection 3** features to build a web application for managing the contacts of a person or company. Multiple companies or individuals can be created, each with one or more contacts. The application will let you:

1. Display a full list of companies and their corresponding contacts.
2. Add a new company to the database.
3. Delete an existing company, together with all associated contacts.
4. Add a new company and its main contact simultaneously.
5. Add, modify or delete contacts for a company.
6. Validate entered e-mail addresses and phone numbers.
7. Send automatic birthday greetings by e-mail.

This tutorial contains four main sections, starting from the basic functionality, and enriching it with different features:

1. Planning out your application.
2. Building pages that allow basic functionality.
3. Adding features to improve safety and usability.
4. Enhance the Contact Management Application by adding some extra options.

If you have the **MX Kollection 3** bundle installed, then you have all the needed tools. Otherwise, the following separate products should be installed on your computer in order to complete the *Contact Management Application* tutorial:

- ImpAKT
- MX Form Validation
- MX Looper
- MX Send E-mail
- MX Query Builder

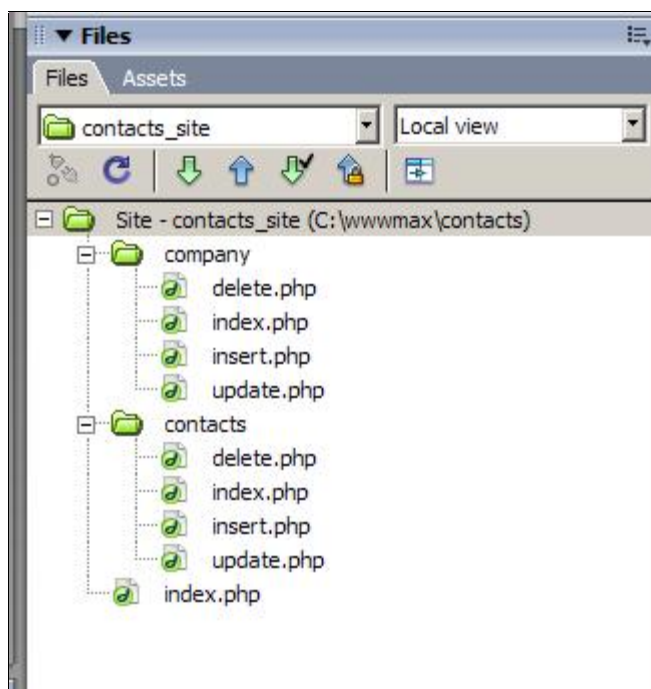
The estimated completion time for this tutorial (considering the initial planning, the mandatory sections and the proposed improvements) is about 100 minutes. It also depends on your authoring knowledge with **Macromedia Dreamweaver** (MX or MX 2004) and **MX Kollection 3**.

Plan the Contact Management Application

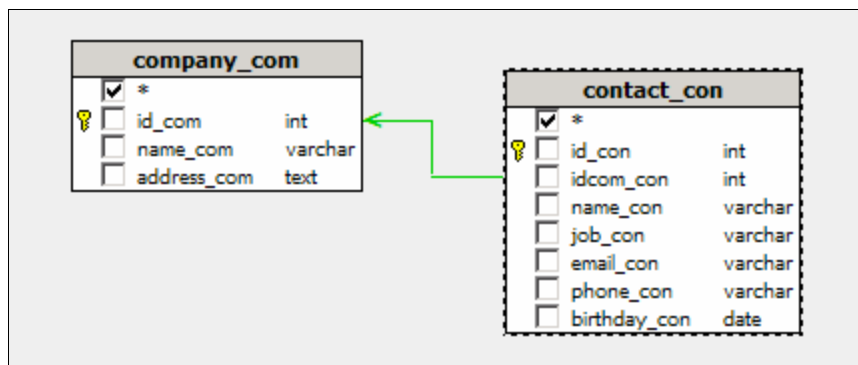
Before you start building this application, make sure you have a correctly configured **Dreamweaver** site, and a working database connection. For more instructions regarding these actions, consult the *Getting started* help file, which can be found in **Help -> InterAKT -> Getting Started**.

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\Contact Management Application\` 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. For this tutorial, you will use two tables: one for the companies, and one for the contacts. The field names are shown 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.

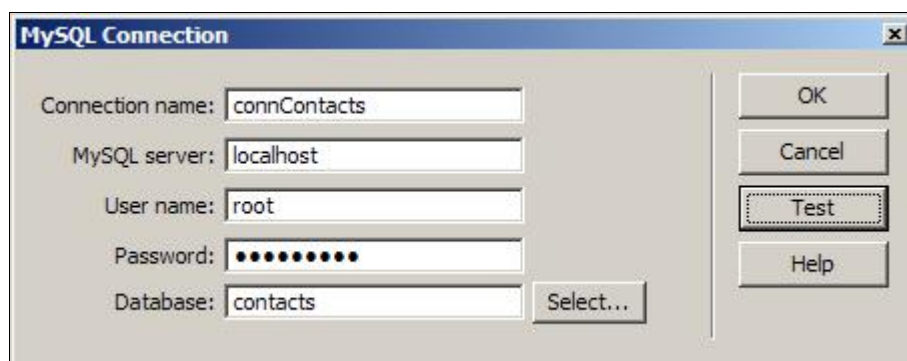
Here are the database tables and their columns:

1. *company_com* - the company table stores one entry for each registered company.
 - **id_com** - the primary key for the company table. No two companies will have the same ID.
 - **name_com** - the company name (Amazon, Best Buy etc.).
 - **address_com** - this field stores the address of the company, (ex: 123 Main St., Chicago, IL).
2. *contact_con* - the contact table stores a list of contents for each company. It is linked to the company table by the **idcom_con** foreign key, which stores the id of the company associated to each contact record.
 - **id_con** - the primary key for the contact table. No two contacts will have the same ID.
 - **idcom_con** - the foreign key containing the company id (**id_com**).
 - **name_con** - the contact name (Bill Jones, Jason Smith etc.).
 - **job_con** - the job or position of each contact (Developer, Manager etc.).
 - **email_con** - the email address of the contact.
 - **phone_con** - contact's phone number.
 - **birthday_con** - contact's date of birth.

You can find the scripts needed to create an identical table structure inside the downloaded package, in the `\tutorials\Contact Management Application\db\` folder, as an *sql* or *mdb* file, depending on the database server you intend to use. Import them in your database server management software (e.g. **PHPMysqlAdmin**, **Microsoft Access** etc.).

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

Note: If using the package provided files, you must replace the existing database connection, user name and password from the connection already created with your particular data.



Once the connection is configured, you can move on to the next section, where you will create the pages for displaying, adding, editing, and deleting companies and contacts.

Add, Edit and Delete Companies and Contacts

By default, the tutorial files are already configured to list your companies and contacts, and are also linked using URL parameters.

When opening the `/company/index.php` page, you should see something similar to this:

Companies:			
Repeat	any ID	Name	Address
	{rsCompanies.id_com}	{rsCompanies.name_com}	{rsCompanies.address_com} Update - Delete
			New

In this section you will build the basic elements of your web application: pages that will allow basic operations on companies and contacts: insert, update and delete. Also, a listing of the companies together with their associated contacts will be created to show a report.

By following this tutorial section, you will create:

- Pages to insert, update and delete companies.
- Pages to insert, update and delete contacts.
- The site's report page that will list companies and contacts in a grouped manner.

If you have the **MX Kollection 3** bundle installed, then you have all the needed tools. Otherwise, the following separate products should be installed on your computer in order to complete this tutorial section:

- ImpAKT
- MX Looper

The estimated completion time for this section is about 40 minutes. It depends on your authoring knowledge with **Macromedia Dreamweaver** (MX or MX 2004) and **MX Kollection 3**.

Display companies and contacts: Nested Repeated Region

Here you will create a listing of all companies and their associated contacts. Next to each entry, you will add links for updating or deleting an item.

Since the list will be created in the site's main *index* page, open the file in **Dreamweaver**.

There are four steps to complete:

1. Create the page design using Dreamweaver's tools.
2. Use the Nested Repeated Region wizard to display the data.
3. Add new, update and delete links next to each page.
4. Save the page and preview it in the browser.

To easily display the companies and their associated contacts, you will use the **Nested Repeat Region** wizard provided by **MX Looper**. Click at the end of the first row in page, after the title, and access this command from the **MX Kollection** tab of the **Insert** bar.

Configure the user interface that opens as follows:

1. In the first step, define the tables and fields to use:
 - In the **Connection** drop-down menu select the database connection you've created at the beginning of this tutorial: *connContacts*.
 - In the **Master table** drop-down menu select the table that contains the main (master) elements: *company_com*.
 - In the **Primary key** drop-down menu select the master table's primary key: *id_com*.
 - In the **Display Value** drop-down menu select the master table's field storing the labels that will be displayed on page: *name_com*.
 - In the **Detail table** drop-down menu select the table that contains the detail elements: *contact_con*.
 - In the **Foreign key** drop-down menu, select the field storing the association between the two tables: *idcom_con*.
 - In the **Display Value** drop-down menu select the detail table's field storing the labels to display: *name_con*.

Create Nested Repeat Region Wizard

Step 1/2: Select the connection and configure the master and detail tables

Specify the database connection

Connection: Define...

Specify master element information

Master table:

Primary key:

Display value:

Specify detail element information

Detail table:

Foreign key:

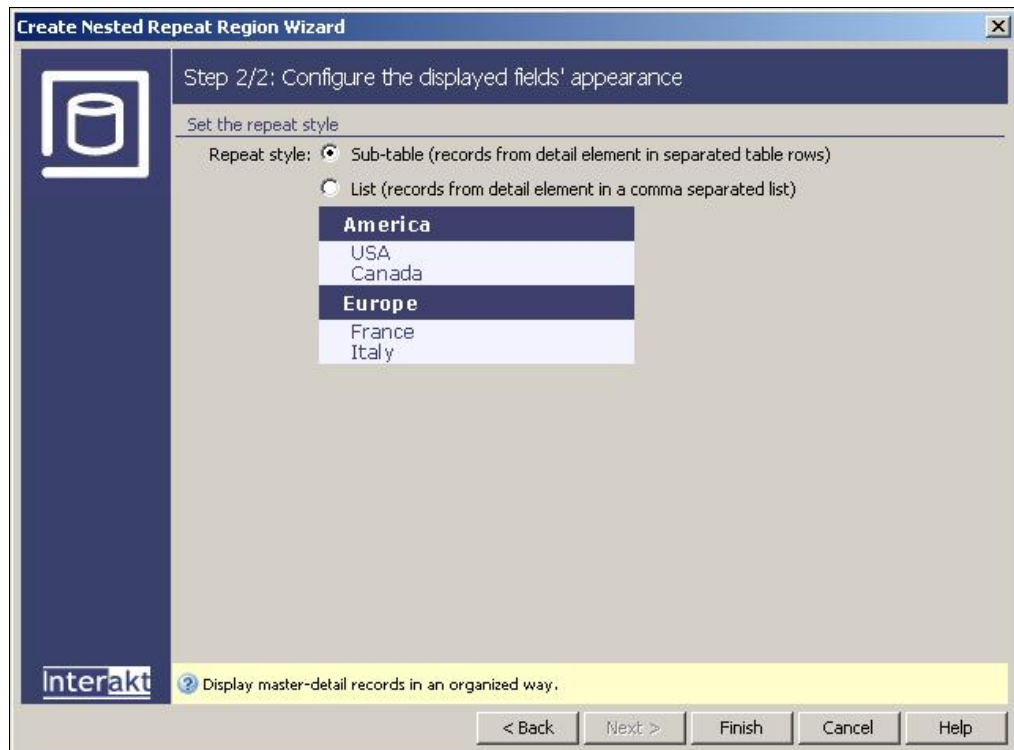
Display value:

Select the field that will be displayed from the detail table.

Interakt

< Back Next > Finish Cancel Help

2. In the second step, decide what **Repeat style** to use:
 - **Sub-table** - each detail element is displayed in a separate row, with indentation. This is the one to select.
 - **List** - this displays the detail elements as a comma separated list.



When you're done configuring the wizard, click the **Finish** button to close it and add all elements to the page: the **HTML** table, the recordsets and the server behaviors.

If you save and preview the page in the browser at this point, it will display the list of companies, and for each, the existing contacts.

Contacts Management

InterAKT Online
Alex Colorado
Chris Benton
Nikita Jelinskis
Macromedia
George Palmer
Tony Norberto
Oracle
Adrian Segata
Mark Johnson

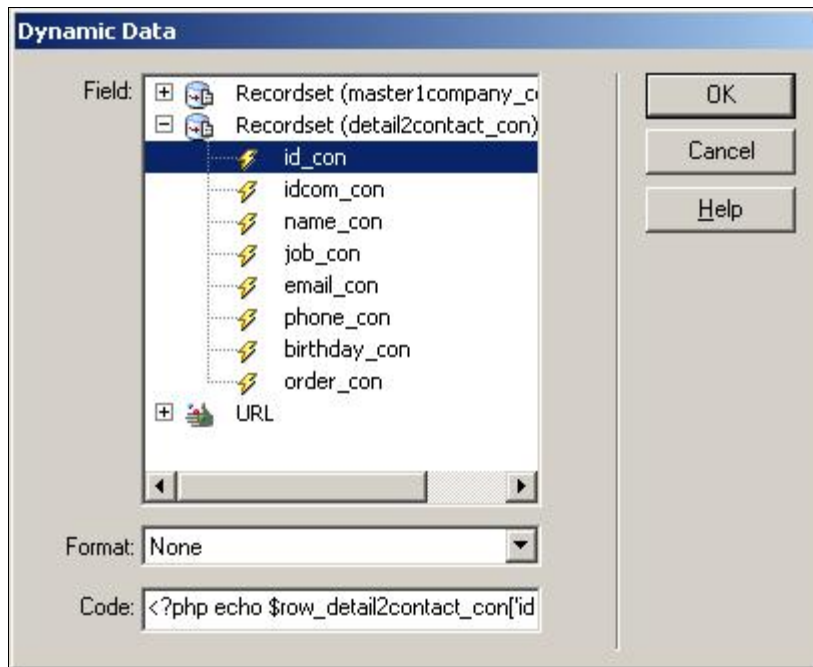
[Manage companies](#)

[Manage contacts](#)

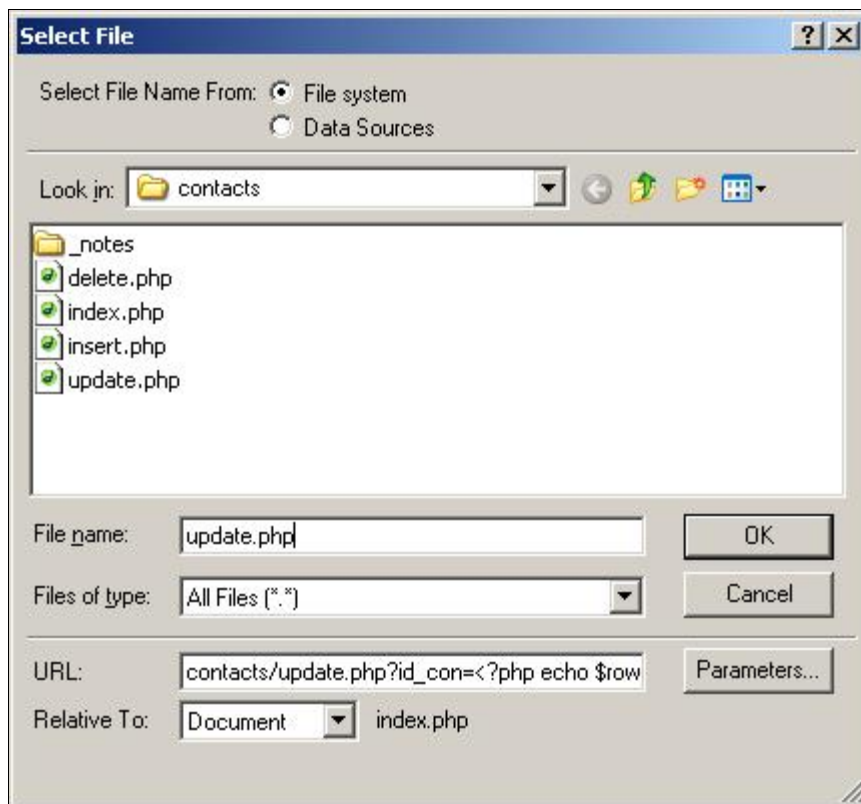
Now, add a new column at the end of the generated table. In the second row's cell type the link names: **Update - Delete**. You will have to turn each of them into links to the actual update and delete pages.

To do so, follow the next steps:

1. Select the **update** text, and click on the **Make link** options from the **right-click** menu.
2. For the page file, select the update file in the contact folder (e.g. *contact/update*).
3. You will also need to pass it the contact unique identifier. Click on the **Parameters** button and add a parameter named **id_con** to the link. For its value, use the lightning icon and select the **id_con** field of the *detail2contact_con* recordset:



4. At this point, the **Make link** dialog box should look as follows:



Repeat the same steps for the **delete** link, but remember to point the link to the *delete* page, instead of the update one (e.g. *contact/delete.php*).

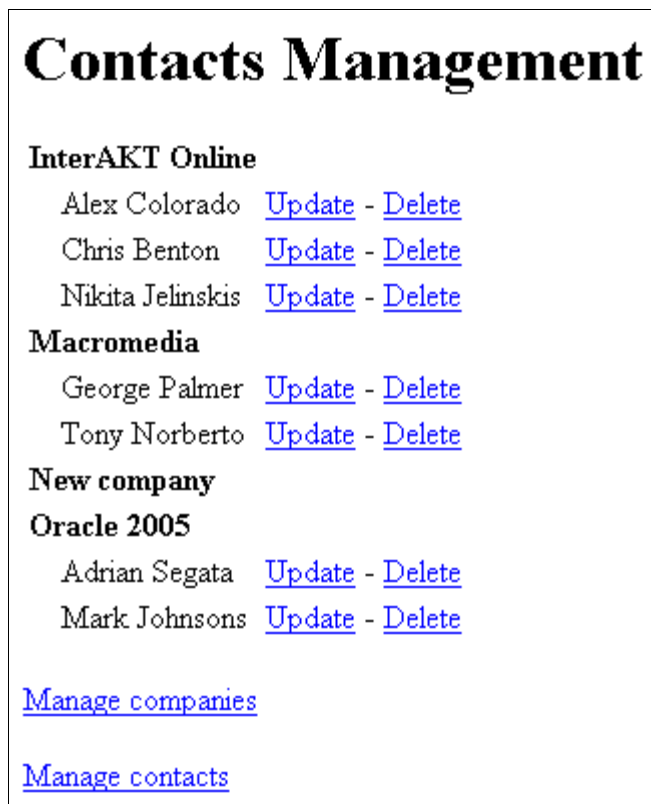
Now, the table has a nice addition, allowing it to actually perform management actions on the contacts:



As you already noticed when you opened this page from the *zip* package, there are two links that will allow you to manage companies and contacts. The **Manage companies** link points to the *company/index* file, and the **Manage contacts** link points to the *contacts/index* file.

Another small improvement for the overall design of the page, is to remove the border surrounding the generated table. To do so, select the table in design view, and, in the **Property Inspector**, set the **Border** to 0.

Now, when you preview the page in the browser, it will look like the following image:



When you click on the **Update** or **Delete** links next to each contact, the update and delete pages will open.

Now the listing page is completed, and you can start putting content into the company and contact's insert, update and delete pages.

In the next part you will create the company *insert* page using one of **MX Kollection 3**'s new features: the wizards.

Add companies: Insert Record Form Wizard

This page offers the possibility to add a new company to the database . The actual insert action and form will be generated automatically through the use of the **Insert Record Form Wizard (Insert bar -> MX Kollection tab)**.

To build the insert page, you will have to follow the next steps:

1. Open the *company/insert* file from the company folder in **Dreamweaver's** editable area.
2. Design your page using **Dreamweaver's** tools. Create a page to suit your taste.
3. Click in an empty place where the insert form will be placed.
4. Use the **Insert Record Form Wizard** to add both an HTML form and application logic to the page. Configure the user interface for your own table and fields.
5. When done configuring, click the **OK** button to add the elements into the page.
6. At this point, the company insert page is fully functional, and you can test it, by previewing it in the browser. Just press the **F12** key.

Creating the page design

Since creating web page designs is not the goal of this document, the page will have a pretty basic look, using only a header to quickly identify the page and the actual form added by the wizard.

First off, add some text to the page, stating its purpose: **"Add company"** (without the quotes). Set the text's format to Heading 2, so that it will become clearer to read, while maintaining reasonable proportions. Select the text and use the **Format** drop-down menu in the **Property inspector** to change its style.

Second, you'll have to add a horizontal rule after the text. This allows a neater separation between the page title and the actual insert form. To insert it, use the horizontal rule button on the **Common** tab (if using **Dreamweaver MX**) or the **HTML** tab (for **Dreamweaver MX 2004**) of the **Insert** bar.

The last thing to do when it comes to the page design, is to create a new paragraph after the horizontal ruler. This is where the insert form will be added. Just press the **ENTER/RETURN** key to create the new paragraph.



Now start the **Insert Record Form Wizard**. Either click its associated button on the **MX Kollection** tab of the **Insert** bar, or select it from the **Server Behaviors** tab -> **MX Kollection** -> **Forms** -> **Insert Record Form Wizard**.

Configure the Insert Record Form Wizard

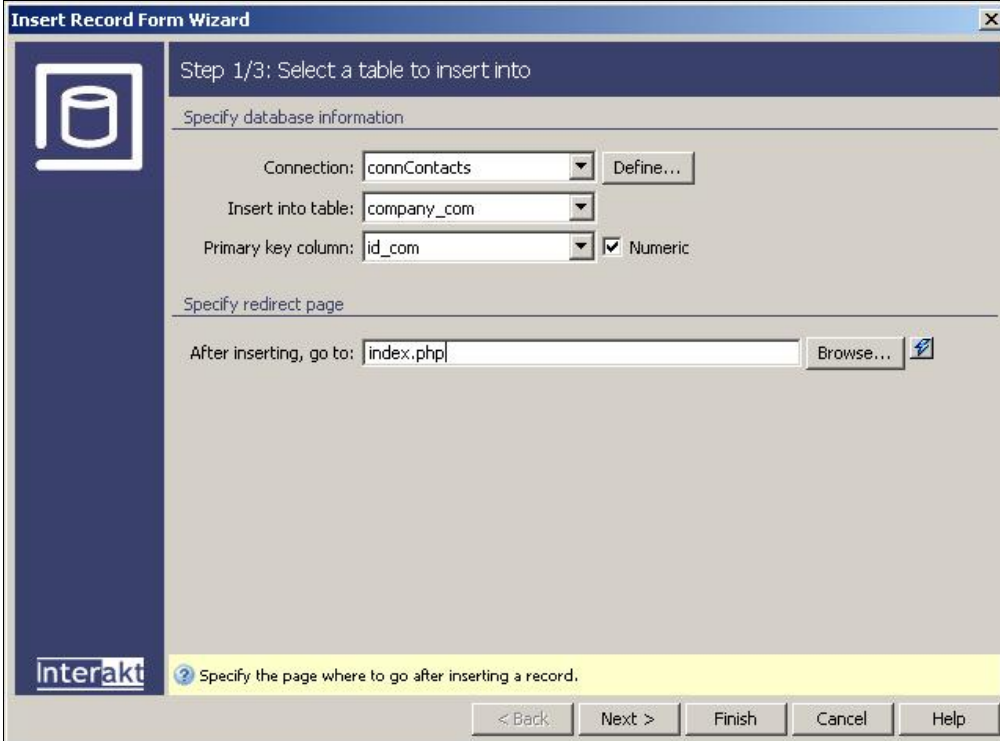
The **Insert Record Form Wizard** will have 2 or 3 steps, depending on if you have **MX Form Validation** installed. For this tutorial the validation step is not important. If you have installed the **MX Kollection 3** bundle, then **MX Form Validation** is installed too, and you will have the third step in the wizard.

After opening the **Insert Record Form Wizard** (either from the **Insert** panel -> **MX Kollection** tab, or from the **Server Behaviors** tab -> **MX Kollection**), you will have to configure the first two steps:

1. In the first step, enter your database connection, table name, primary key and redirects.
2. In the second step you will have to set up the fields to use in the insert transaction : what fields to display in the form, and how.

For the first step, you will have to use the following options:

1. In the **Connection** drop-down menu, select the database connection defined at the beginning of this tutorial - *connContacts*.
2. In the **Table** drop-down select *company_com*; we want to add companies.
3. In the **Primary key column** drop-down menu you must select the field containing the primary key for the table. By default the *id_com* field is selected.
4. In the **After inserting, go to** text field, enter the page to display after the insert operation is successful. Use the **Browse** button to select the *index* page located in the *company* folder.
5. The completed dialog box should resemble the example below:



The screenshot shows the 'Insert Record Form Wizard' dialog box at Step 1/3: 'Select a table to insert into'. The dialog is titled 'Insert Record Form Wizard' and has a close button in the top right corner. On the left side, there is a blue sidebar with a database icon and the 'Interakt' logo. The main area is divided into two sections: 'Specify database information' and 'Specify redirect page'. In the 'Specify database information' section, there are three dropdown menus: 'Connection' (set to 'connContacts'), 'Insert into table' (set to 'company_com'), and 'Primary key column' (set to 'id_com'). There is a 'Define...' button next to the 'Connection' dropdown and a checked 'Numeric' checkbox next to the 'Primary key column' dropdown. In the 'Specify redirect page' section, there is a text field labeled 'After inserting, go to:' containing 'index.php' and a 'Browse...' button with a folder icon. At the bottom of the dialog, there is a yellow status bar with a question mark icon and the text 'Specify the page where to go after inserting a record.' Below the status bar are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

6. You can press the **Next >** button to proceed to the wizard's next step.

In the second step you have to decide which fields to display, and under which form:

1. In the **Form fields** grid, all fields in the current table are initially displayed (except for the primary key, which is usually auto-generated from the database). You will have to keep/remove the fields that don't need to have an **HTML** form counter-part (e.g. an auto-incremented primary key). Use the **+/-** buttons to add/remove fields from the grid. The order for the form fields is established through the **^** and **v** buttons.
2. For each of the two fields you must select the labels, the way it will be displayed (the **Display as** drop-down menu) and how to submit it into the database (the **Submit as** drop-down menu).
 - The Labels will be automatically recognized from the field names as the wizard knows the **InterAKT** database notations.
 - For the **name_com** field you can leave the display and submit options at their defaults: **Text field / Text**
 - For the **address_com** field you should change the **Display as** option, and set it to **Text area**. This is done because an address contains a larger number of characters, and is easier to read with a text field.
 - Leave the **Default value** text field empty.
3. The completed dialog box looks like in the image below:

Step 2/3: Configure the form fields

Set form fields properties

Form fields: + -

Column	Label	Display as	Submit as
name_com	Name:	Text field	Text
address_com	Address:	Text area	Text

Label: Address:

Display as: Text area

Submit as: Text

Default value:

Interakt

The list of fields associated to the insert transaction.

< Back Next > Finish Cancel Help

The last step in the wizard concerns validation for the form elements. For the two existing fields there are no validation rules that can be applied to restrict bad input, and therefore the last step can be skipped. Click on the **Finish** button to exit the wizard's user interface and add it into the page.

Note: for the **ASP** and **ColdFusion** server models, the approach is similar; just replace the page extension with the one required by your server model .

You should now see the following code generated in the **Dreamweaver** design view:

Add company

{Display error message.}

Name: {rscompany_com.name_com} {Hint} {Error}

{rscompany_com.address_com}

Address: {rscompany_com.address_com} {Hint} {Error}

Insert record

There is a **Display error message** server behavior that will be in charge of showing any handled error message. Also field hints and field error message are generated in the form.

The company insert page is now finished, and you can save and preview it in the browser. Let's see now how the generated page behaves in a browser.

Add companies page: browser behavior

Usually, you will reach the insert company page from the companies list:

Companies:

id_com	name_com	address_com	
1	InterAKT Online	1-11 Economu Cezarescu ST, AYASH Center, Bucharest, Romania	Update Delete
2	Macromedia	Macromedia, Inc., 600 Townsend Street, San Francisco, CA 94103, USA	Update Delete
3	Oracle	Oracle Parkway, Thames Valley Park, London, United Kingdom	Update Delete
			New

The insert page looks like this:


The screenshot shows a web browser window with the title "Add company". The browser's address bar and toolbar are visible at the top. The main content area contains a form with the following elements:

- A "Name:" label followed by a text input field.
- An "Address:" label followed by a large text area with a vertical scrollbar.
- An "Insert Record" button located at the bottom right of the form.

If you type data in the form fields, and then click on **Insert Record**, a new entry will be added to the company's table and the browser will be redirected back to the list.

Now let's see how the generated page behaves when inserting companies.

When the **form is submitted empty**, you will see how **tNG3** manages SQL error messages:


Error:
General error: Column 'name_com' cannot be null

Developer Details:
 INSERT INTO company_com (`name_com`,`address_com`) VALUES (null,null)
[Online error troubleshooter](#)

tNG Execution Trace - [VIEW](#)

Name:	<input style="width: 95%;" type="text"/> General error: Column 'name_com' cannot be null
Address:	
<input type="button" value="Insert Record"/>	

A lot of improvements have been brought to the error reporting engine:

1. First, errors are now shown per field, together with a global page error. In our case, we will receive an error for the **name_com** field, saying that the field cannot be null. In order to understand what field has an error, the **tNG3** engine will parse the error message and try to detect the problematic field.
2. Second, you will see several links that will show you various debug information.
 - When clicking the **View** link, a trace of the executed transactions and triggers in the page will be displayed:


Error:
SQL Error: Column 'name_com' cannot be null.

Developer Details:
 SQL Error: INSERT INTO `company_com` (`name_com`,`address_com`) values (null, null). (SQL_ERROR)
[Need help?](#)

tNG Execution Trace - [VIEW](#)

```

      • tNG_insert.executeTransaction
        o STARTER.Trigger_Default_Starter
        o tNG_insert.doTransaction
          ■ BEFORE.Trigger_Default_FormValidation
          ■ tNG_insert.prepareSQL
          ■ tNG_insert.executeTransaction - execute sql*
          ■ AFTER.Trigger_Default_Insert_RollBack
      • tNG_insert.getRecordset
      • tNG_insert.getFakeRsArr
      • tNG_insert.getFakeRecordset
    
```

As you can see, the page section that threw the error will be highlighted in red, and you'll also see the **tNG3** execution flow - starter triggers, before trigger, the actual SQL transaction, after triggers and finally end triggers.

- When clicking the **Need Help** link, you will be redirected to the **InterAKT** site, and the error will be understood by our server in order to provide you with instant troubleshooting:

Developer Error Troubleshooting

The error message submitted was registered into our logging system. The error message was anonymously registered. This future more suggestions on how to prevent or to correct these errors.

OS: WINNT
Server Model: PHP MySQL
Web Server: Apache/1.3.28 (Win32) PHP/5.0.2
User Error: SQL Error: Column 'name_com' cannot be null.
Development Error: SQL Error: INSERT INTO `company_com` (`name_com`, `address_com`) values (null, null). (SQL_ERROR)

What can I do?

Suggestions not available for the moment.

You may visit also the following discussion groups:

- [MX Kollection beta general discussions](#)
- [MX Kollection report bug](#)

You can also check what happens when an error is found for a field. In our case, if we leave the Name field empty, but enter a valid Address, when submitting, because the page will catch the error. The Address field value will be preserved, to prevent the huge annoyance of losing the submitted data on error:



Error:
SQL Error: Column 'name_com' cannot be null.

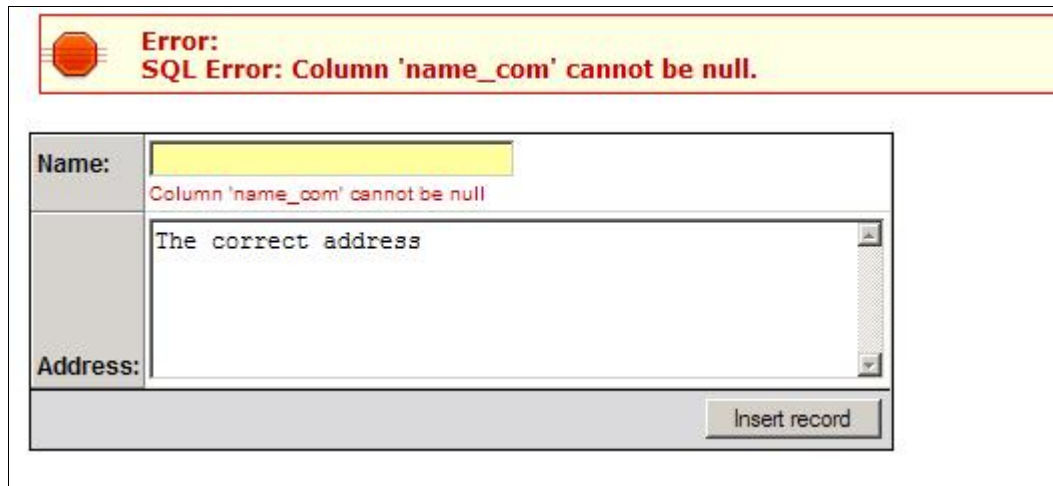
Developer Details:
SQL Error: INSERT INTO `company_com` (`name_com`, `address_com`) values (null, 'The correct address'). (SQL_ERROR) [Need help?](#)

tNG Execution Trace - [VIEW](#)

Name:	<div style="border: 1px solid gray; padding: 2px;"> <p>Column 'name_com' cannot be null</p> </div>
Address:	<div style="border: 1px solid gray; padding: 2px;"> <p>The correct address</p> </div>
<input type="button" value="Insert record"/>	

You should feel pretty confident with these professional features for error handling that will help you build your web applications much faster than before, and that will also help your clients understand what went wrong when an error happens on the server.

While you will surely appreciate the detailed error messages in this form, you should also know that you can move a site in Production mode and thus avoid showing all those errors to the final client (you will be able to either log the detailed errors in a file or send them by e-mail to you). To find out more about this matter, check the **Control Panel** settings here. And this is how the errors is displayed in a production page:



The screenshot shows a web form with a yellow error banner at the top. The banner contains a red octagonal icon and the text: **Error: SQL Error: Column 'name_com' cannot be null.**

Below the banner, the form has two main sections:

- Name:** A text input field with a yellow background. Below it, the error message "Column 'name_com' cannot be null" is displayed in red text.
- Address:** A text area containing the text "The correct address".

At the bottom right of the form, there is a button labeled "Insert record".

After having completed the company insert page, you can move on and learn how to create the update page. It will allow you change information already entered for a company.

Edit companies: Update Record Form Wizard

The Edit Companies page is for modifying a company record in the database . The actual update action and form will be generated automatically through the use of the **Update Record Form Wizard**.

Building the update page will consist of the following steps:

1. Open the *company/update* file from the company folder in **Dreamweaver's** editable area.
2. Design your page using **Dreamweaver's** tools. Create a page to suit your taste.
3. Click in an empty place to add the update form.
4. Use the **Update Record Form Wizard** to add both an HTML form and application logic to the page. Configure the user interface for your own table and fields.
5. When done configuring, click the **Finish** button to add the elements into the page.
6. At this point, the company update page is fully functional, and you can test it, by previewing it in the browser.

While creating the update page, you will notice and take advantage of a new feature in **MX Kollection 3**: UI persistence. This means that you only have to configure some fields once, and the **Dreamweaver** interfaces will remember it. The next time you attempt to use a wizard, it will be already configured the way you did it last time. If you have disabled the UI persistence, you will have to go through all steps of the wizards.

Creating the page design

As mentioned when creating the company insert page, the page design used is kept pretty basic, as it is not the goal of this tutorial. For the update's page design, you should repeat the same steps used for the insert page, only replacing the text with "Modify company data".

Configure the Update Record Form Wizard

The **Update Record Form Wizard** has 2 or 3 steps, depending if you have **MX Form Validation** installed. For this section, you can ignore the validation since it will not be used. If you have installed the **MX Kollection 3** bundle, then **MX Form Validation** is installed too, and you will have the third step in the wizard.

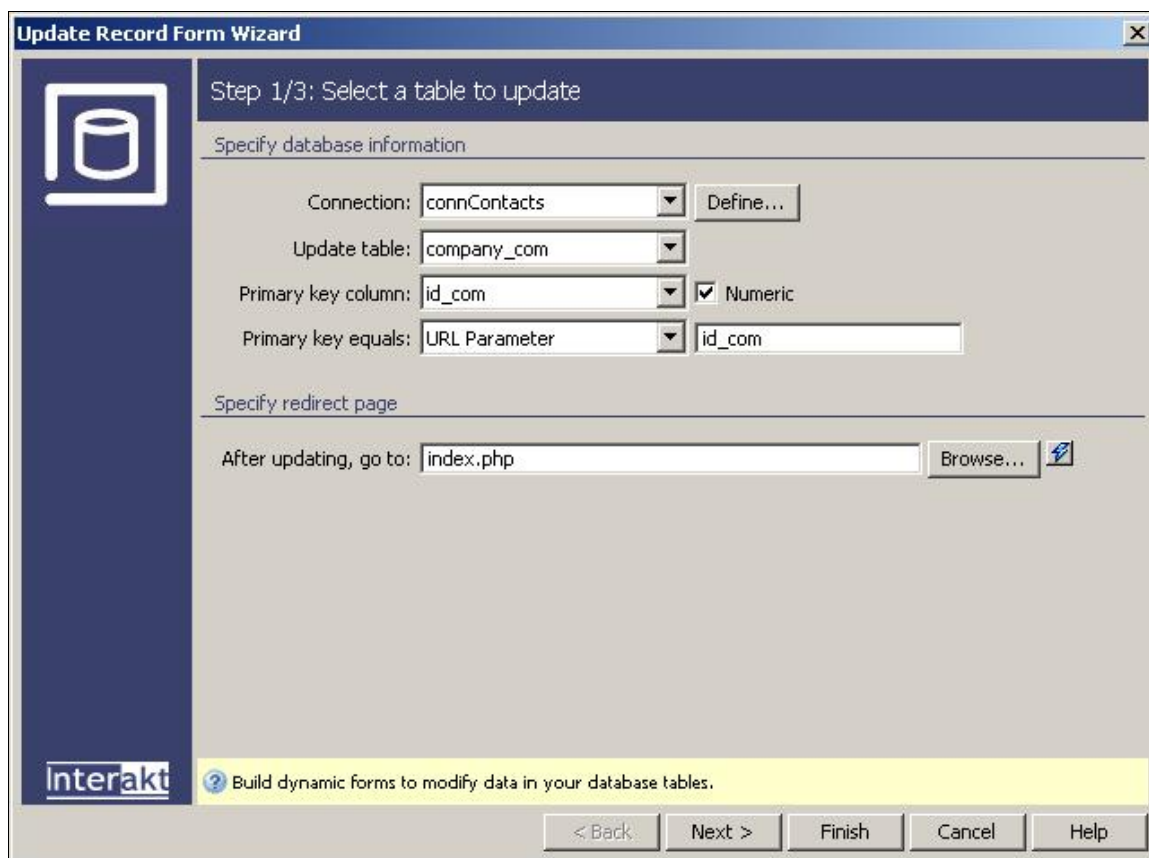
This part of the tutorial will take full advantage of the user interface persistence provided by **MX Kollection 3**. As you open the **Update Record Form Wizard**, you will notice that this time the **Connection, Table, Primary key column** and even the After updating, go to text box are automatically selected with the values used in the **Insert Record Form Wizard**.

Note: It is assumed that you followed this tutorial in order and have already applied and configured the **Insert Record Form Wizard**.

Configuring the **Update Record Form Wizard** can be considered from the start completed, but let's still browse through the two steps to see if all is fine. If you are in a hurry, you can just hit the **Finish** button. The update operation will work nonetheless.

The first step, containing the connection and table information, now has two new fields: the Primary key equals drop-down menu and text field. This is where you decide which way will the company's unique identifier be passed. By default, the wizard considered that the method will be **URL Parameter**, and its name is **id_com**. Leave the default values.

The user interface looks as follows:




The screenshot shows a dialog box titled "Update Record Form Wizard" with a close button (X) in the top right corner. The main title bar is dark blue with the text "Update Record Form Wizard" and a small icon of a database cylinder. The dialog is divided into two sections: "Specify database information" and "Specify redirect page".

Specify database information:

- Connection:
- Update table:
- Primary key column: Numeric
- Primary key equals:

Specify redirect page:

- After updating, go to: 

At the bottom left is the "interakt" logo. A yellow banner at the bottom contains the text: "Build dynamic forms to modify data in your database tables." At the bottom right are five buttons: "< Back", "Next >", "Finish", "Cancel", and "Help".

Click on the **Next >** button if you want to continue checking the dialog box, or **Finish** to apply the wizard.

The second step of the wizard also has all options filled by the user interface persistence. There are no new fields on this dialog box, so after taking a look, you can directly click on **Finish**.

The already filled dialog box looks like:

Update Record Form Wizard

Step 2/3: Configure the form fields

Set form fields properties

Form fields: + -

Column	Label	Display as	Submit as
name_com	Name:	Text field	Text
address_com	Address:	Text area	Text

Label:

Display as:

Submit as:

interakt

The list of fields associated with the Update Record transaction.

< Back Next > Finish Cancel Help

At this point, the update page is completed and can be previewed in the browser. No record will be available for update however, until a URL parameter is passed to the page.

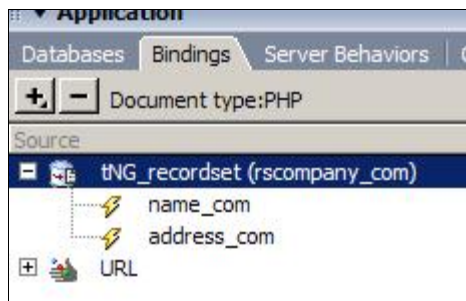
To send the correct URL parameter to this page, the best way involves on clicking the **Update** link in the companies list folder (*/company/index.php*):

Companies:

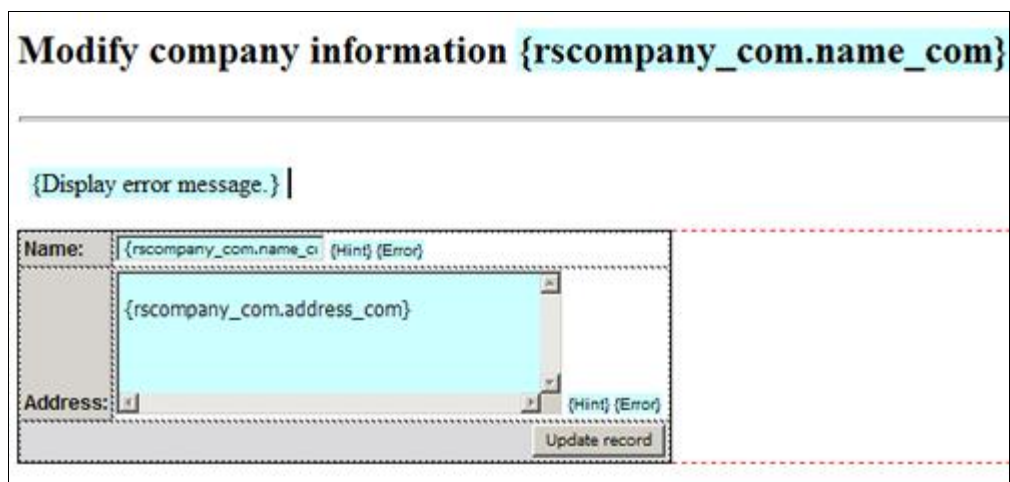
id_com	name_com	address_com	
1	InterAKT Online	1-11 Economu Cezarescu ST, AYASH Center, Bucharest, Romania	Update Delete
2	Macromedia	Macromedia, Inc., 600 Townsend Street, San Francisco, CA 94103, USA	Update Delete
3	Oracle	Oracle Parkway, Thames Valley Park, London, United Kingdom	Update Delete
			New

The wizard adds an **HTML** form, an **Update Record** server behavior and a transaction recordset . This recordset will contain upon page initialization data retrieved from the company table, related only with the company to update. You can use this recordset to ease understanding of which company is being edited, by adding the company's name to the title. To do so, you must follow the next steps:

1. In the **Bindings** tab open the *tnq_Recordset*. If the **Bindings** tab is not visible, you can open it from **Window -> Bindings**:



2. Drag and drop the **name_com** field from the bindings tab onto the page, right after the "Modify company information" text.
3. The page should now look like the following, when viewed in **Dreamweaver**:



Delete companies: Delete Record Wizard

The last page in the company folder will allow you to delete companies from the database. To implement the actual delete operation, the **Delete Record Transaction** will be used. It is similar to the Insert/Update wizards and it also uses the user interface persistence to fill in its fields. At the same time, it is different from the Insert and Update wizards as it does not add any HTML elements into the page.

To create the company *delete* page, follow the next steps:

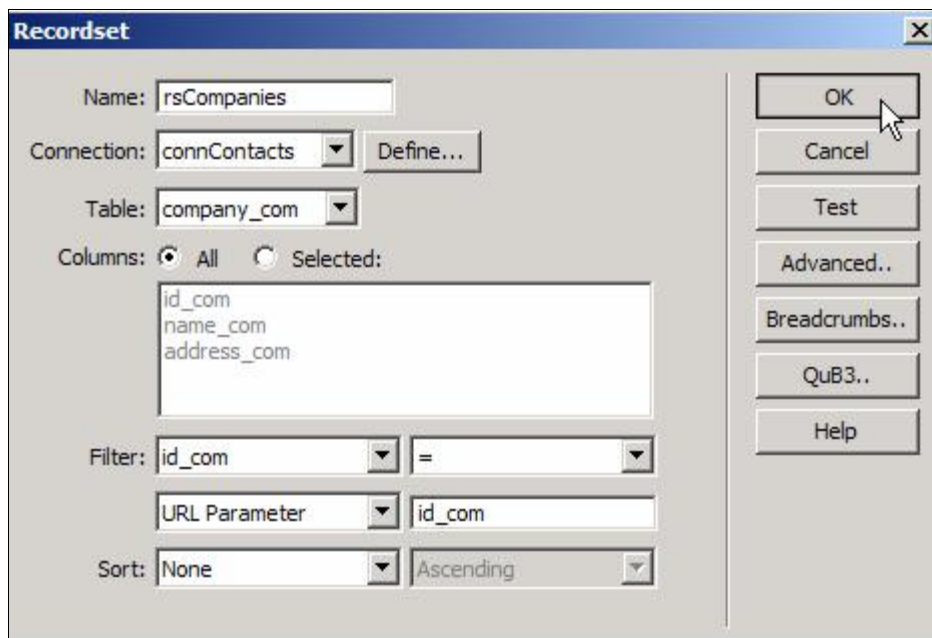
1. Open the *companies/delete* page in **Dreamweaver**.
2. Design your page using **Dreamweaver's** tools. Create a page to suit your taste.
3. Click in the design view.
4. Use the **Delete Record Transaction** to add the application logic that will delete the desired record to the page.
5. When done configuring, click the **OK** button to add the **Delete Record Transaction** server behavior .

Create the page design

Unlike for the other pages, the delete page's design section will help you set up a safety net for the delete operation. More exactly, you will create a page section which will display details of the company to delete, and ask for confirmation. This way, you will not be able to delete a record accidentally, simply by clicking on the delete link in the main page.

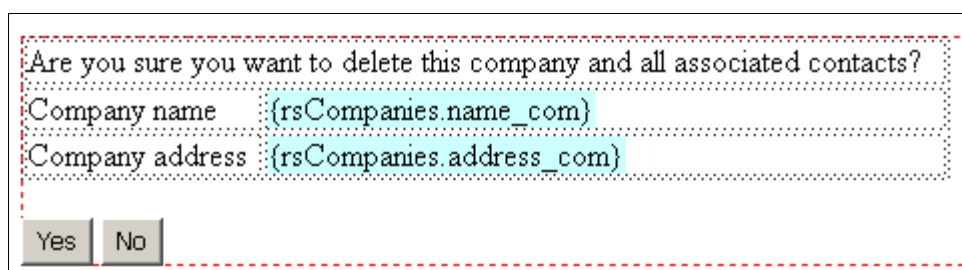
To create the page design, follow the next steps:

1. First off, add a **table** to the page. Set it to have **4** rows and **2** columns. Make its **width** 500 pixels and set the **border**, **cell spacing** and **cell padding** to 0.
2. Merge the first row's cells. Type the delete question in the new cell: "**Are you sure you want to delete this company?**" (without the quotes).
3. On the second and third row you will display the company details: its name and address. But in order to retrieve these details, you must first create a filtered recordset . This recordset will use the *company_com* table, filtered after the **id_com** URL parameter. The recordset creation dialog box should look like the following:



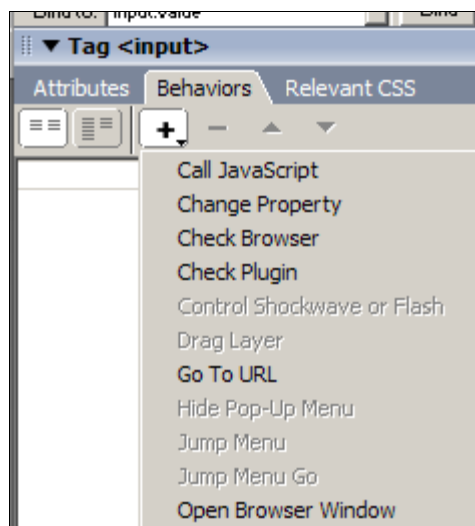
4. Now type in the first cell of the second row "**Company name:**"; drag and drop the field **name_com** from the **Bindings** tab into the second cell of the second row.
5. In the third row's first cell type "**Company address:**", and then drag and drop the field **address_com** from the **Bindings** tab into the row's second cell.
6. At this point, you are displaying all company details in the *delete* page. The last action concerning the *delete* page design is to add the buttons that will continue or stop the delete transaction. To add them, follow the instructions given below.
7. Merge the cells on the last row. Also add a form submit button. When adding the button from the **Forms** tab of the **Insert** panel, a form will be added as well. If a message requesting confirmation to add the form tag appears, click **Yes**.
8. As this will be the confirmation button, you have to change its **Name** to **KT_Submit** and the **Label** to **Yes**. Note down the exact name you've given the button as it will be needed as the starting condition for the delete transaction.
9. Add another button at the right. Change its display value to **No**. Leave the button name to its default state, as you will apply a simple **Dreamweaver** behavior, which opens another URL on click. This change will be done later in the tutorial.
10. Select the form around the buttons, and using the **Property inspector** make sure that the text box corresponding to its **Action** is blank.

Now, the page design should display the company details and the **Yes/No** buttons:

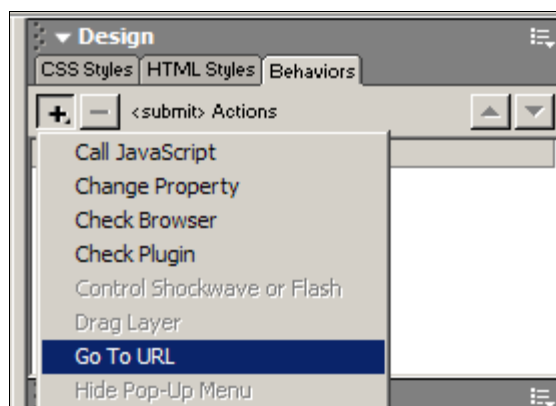


Let's first make the **No** button redirect the page to the main *index* page. Select the **No** button.

For **Dreamweaver MX 2004** users - open the **Tag Inspector** Panel, the **Behaviors** tab, and select **Go To URL**.



For **Dreamweaver MX 6.1** users - open the **Design** panel, the **Behaviors** tab, and select **Go To URL**.



Select the *index* page.

Next, you'll have to apply the **Delete Record Wizard**. You can find its button on the **MX Kollektion** tab of the **Insert** panel (in the upper left side of the **Dreamweaver** window, below the menu). After the dialog box opens, you'll notice that most of the fields are already correctly completed: **Connection**, **Delete from table**, Primary key **column** and **Primary key equals**. If left like this, the delete operation would occur immediately when the page loads, making our safety net completely useless. There is however one dialog box option that lets you define a start condition: the **First check variable** field.

To start the delete action only after pressing the **Yes** button, select in the **First check variable** drop-down menu the **Form Variable** option. Type in the corresponding text field the exact name of the **Yes** button, as defined earlier (at step 8 of this section of the tutorial) - **KT_Submit**.

Delete Record Transaction

Basic | Advanced

Specify database information

Connection: Define...

Delete from table:

Primary key column: Numeric

Primary key equals:

Select transaction starter event

First check variable:

Specify redirect page

After deleting, go to: Browse...

Enter the default starter reference.

OK
Cancel
Help

Since all other fields are already set up, click the **OK** button to add the server behavior to the page. If you simply preview the page in the browser, you will only see the empty placeholders for the dynamic data. This happens because the page is expecting a value for the **id_com** URL parameter. Please click the **Delete** link from the *index* page.

This is the last page to be created in the company folder. Through the existing pages, all needed operations on companies are covered: adding, editing and deleting are all implemented. It is time to move on to the second section of the site: the contacts.

Add, edit and delete contact persons: Interface Persistence

After having created the company management section, it is time to build the other half of this application: the contact management section.

This section's pages are stored in the *contacts* folder, and will perform the addition of new contacts, editing existing contact data and finally deleting a contact. The actions themselves will be added by using the **Insert Record Form Wizard**, **Update Record Form Wizard** and the **Delete Record Transaction**, same as for the companies. There are however some differences, apart from the table and fields name to use, due to the fact that a contact cannot exist without a company. Thus, when adding a contact you must specify which company he stands for.

Note: Before proceeding with the wizards, notice that the *index* page inside the *contacts* folder already contains a dynamic table that lists all the current contacts, no matter the company.

Building the insert page

For the *insert* page, follow the same steps as for the company insert page: page design and wizard configuration.

In terms of design, the same layout will be used: the page header, a horizontal rule, and the wizard generated elements. Use the instructions presented for the company section to create the design for this page too. Remember however to change the "Add new company" text into "Add contact".

When it comes to the **Insert Record Form Wizard**, there are some minor changes. Start the **Insert Record Form Wizard** from the **MX Kollection** tab of the **Insert** bar to add the HTML form. At first, the dialog box's fields are filled with the user interface persistence cache options, using the *company_com* table and its related fields. You will have to change these settings only for the first wizard, as the rest will use the user interface persistence.

In the **Insert into table** drop-down menu, select the *contact_con* table, instead of *company_com*. If the correct *id_con* column is not selected by default in the **Primary key column** drop-down menu, select it.

The page to redirect to will be the main *index* file, situated in the site root. If the correct file is not selected, use the **Browse** button to navigate through your site's folders.

When all options are set, click the **Next >** button to proceed to step two of the wizard.

Insert Record Form Wizard

Step 1/3: Select a table to insert into

Specify database information

Connection: Define...

Insert into table:

Primary key column: Numeric

Specify redirect page

After inserting, go to: Browse...

interakt

Specify the page where to go after inserting a record.

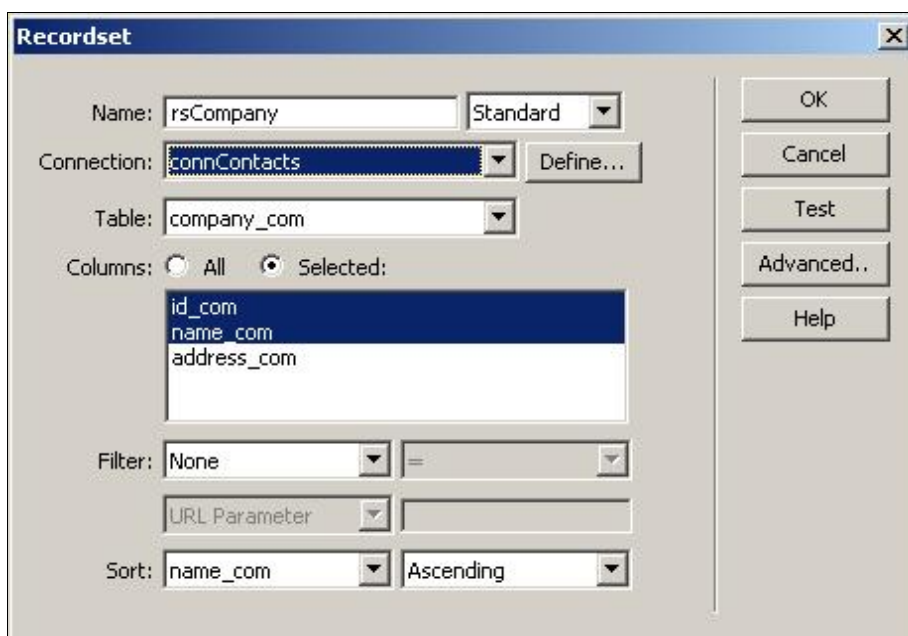
< Back Next > Finish Cancel Help

The second step of the wizard allows you to select what fields will be displayed in the page, their order and input method. For most fields you can leave them at their defaults: the names to be displayed next to the form controls are the field names, without the three letter suffix. The type of content to submit (the **Submit as** drop-down menu's content) is retrieved based on the table meta information.

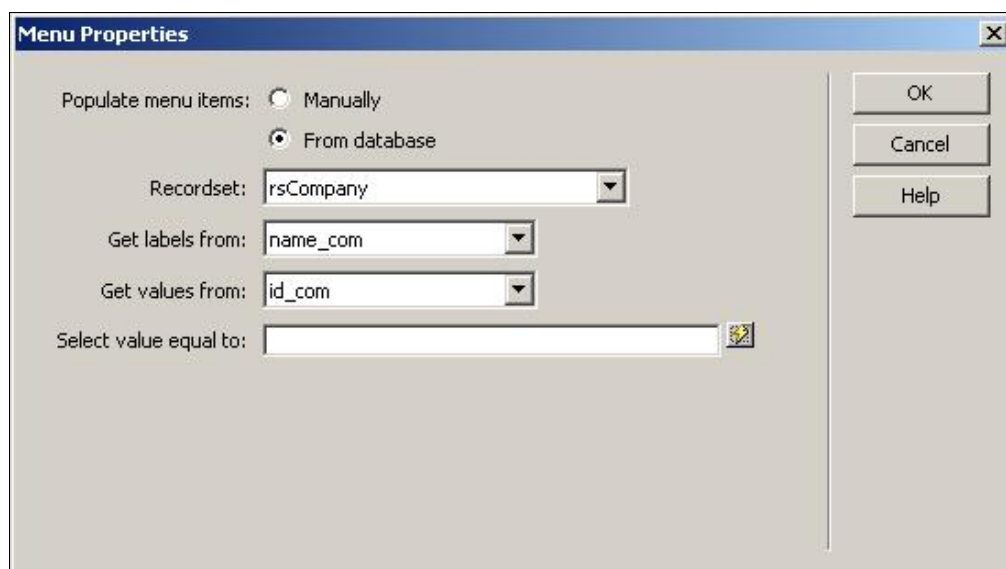
The only element that needs to be changed is the **idcom_con** field. This field is the foreign key, storing the link to the company table. If left as a text field, the company's identifier (a numeric value) would need to be entered by hand. A much better way to select the company is to have a drop-down menu listing all the companies by name and when selected, automatically inserts the company ID.

To use the latter method, follow these instructions:

1. Select the **idcom_con** field in the grid.
2. First change its name to something more human - readable, like **Company**:
3. Second, change the element used to display it, by selecting in the **Display as** drop-down menu the **Menu** option.
4. Two new buttons have appeared: **Menu properties** and **Add Recordset**. Usually, to display dynamic data in the drop-down menu you had to cancel the wizard, create a recordset, and then re-start the **Insert Wizard**. The **Add Recordset** button solves this problem, as it will open the **Recordset** dialog box without leaving the wizard.
5. After clicking on the **Add recordset** button, configure a new recordset, **rsCompany** that will retrieve all companies from the table:



6. After you've added the recordset, you can use the **Menu Properties** button to instruct it to display data from the database:
- Click on the **From database** radio-button, so that elements will be retrieved in a dynamic manner.
 - In the **Recordset** drop-down menu select the one you've just created.
 - In the **Get labels from** and **Get values from** select the table fields containing the company names, respectively ID's.



7. After you've set the menu properties, you can click on the **Finish** button to complete the wizard.

Note: If you are using a **Microsoft Access** database, in the **Submit as** drop-down

menu, for the birthday_con field, another option will be available: **Date MS Access**. Select this option when submitting the date.

The screenshot shows the 'Insert Record Form Wizard' dialog box, Step 2/3: Configure the form fields. The dialog is titled 'Set form fields properties' and contains a table of form fields. The table has columns for Column, Label, Display as, and Submit as. The fields listed are idcom_con, name_con, job_con, email_con, phone_con, and birthday_con. The idcom_con field is selected, and its properties are shown in the fields below the table: Label: Company:, Display as: Menu, and Submit as: Numeric. There are also buttons for 'Menu Properties...', 'Add Recordset', '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Column	Label	Display as	Submit as
idcom_con	Company:	Menu	Numeric
name_con	Name:	Text field	Text
job_con	Job:	Text field	Text
email_con	Email:	Text field	Text
phone_con	Phone:	Text field	Text
birthday_con	Birthday:	Text field	Date

Label: Company:
 Display as: Menu
 Submit as: Numeric

Buttons: Menu Properties..., Add Recordset, < Back, Next >, Finish, Cancel, Help

Interakt logo and status bar: The list of fields associated to the insert transaction.

Now the contact *insert* page is complete, and you can preview it in the browser by pressing the **F12** key. Select a company name from the drop-down menu and enter the contact details. When you click the **Insert Record** button, a new contact is added, having the *idcom_con* field equal to the selected company's ID.

Note: Make sure you have chosen the desired database and screen date formats from the **InterAKT Control Panel -> Date formats** category. For more information on configuring and working with date formats, please read Date formats.

Building the update page

This page will allow you to update details regarding a contact. The update form is similar to the insert one, but it is by default populated with the selected contact's data. The most common way to select the contact to update is to pass its unique identifier as an URL parameter. Therefore, when calling the update page, you will need to pass it the **?id_con=value** part. (assuming that *id_con* is the URL parameter's name).

For the page design, follow the same steps as for the company update page. Add the page header, changing the text to "Modify contact information". Add the horizontal rule too, and after creating a new paragraph below, start the **Update Record Form Wizard**.

From the first look, you can see that all fields are now filled for the *contact_con* table. The user interface persistence remembered that the last table used was *contact_con* and it automatically loaded the settings for it. Also, the additional field, Primary key *equals* is set to URL Parameter, with the name *id_con*. These are the default options, but they are perfect for this page. You can simply click on **Finish** to complete the wizard as the fields are remembered from the **Insert Record Form Wizard**.

Update Record Form Wizard

Step 1/3: Select a table to update

Specify database information

Connection: Define...

Update table:

Primary key column: Numeric

Primary key equals:

Specify redirect page

After updating, go to: Browse...

Interakt

Build dynamic forms to modify data in your database tables.

< Back Next > Finish Cancel Help

To enhance the page, add the contact's name in the page header. First add the "for" text to the header, and then drag and drop the *name_con* field from the **Bindings** tab into the page, after the header text.

Building the delete page

This page will allow you to delete contacts from the database. The logic is implemented through the **Delete Record** server behavior. Similar to the *update* page, the contact's unique identifier must be passed to the page through an URL parameter. Also, a safety net, similar to the one created through page design for the company's delete page will be implemented (the table displaying details for the contact to be deleted - name and job - and the two buttons).

Create the page design following the same steps as for the company delete page; display the contact name and email address in the table, and use the *contact_con* table when creating the recordset. The URL parameter passed will be **id_con**.

Once the page design is finished, complete with the **Yes/No** buttons and apply the **Delete Record** server behavior. Since the user interface persistence is in effect for this wizard too, all you need to specify is the starting condition: the **Form variable** *KT_Submit*:

Delete Record Transaction

Basic | Advanced

Specify database information

Connection: Define...

Delete from table:

Primary key column: Numeric

Primary key equals:

Select transaction starter event

First check variable:

Specify redirect page

After deleting, go to: Browse...

Enter the default starter reference.

OK
Cancel
Help

Now you have the entire suite of insert, update and delete operations completed and ready to use. Except for the *insert* page, which you can already test, for the others you have to create a way to pass them the correct identifier.

The simplest way to access the *update* and *delete* pages, is to place them in a list, together with two links: one for editing, and one for deleting.

Validate Contact Information

In the first part of this tutorial, you've created a simple web application that allows you to manage companies and contacts. You can add, edit or delete companies and contacts through HTML forms. These are basic features that you will complete in this section of the tutorial (implementing validation in your pages).

In this section, you will add the following features to your *Contact Management Application*:

- Verifying if the texts entered in the contact e-mail and phone number text boxes are correct.
- Deleting the contacts related to a company, when the company is removed.
- Making sure that the company for which you are adding a contact still exists.

If you have the **MX Kollection 3** bundle installed, then you have all the needed tools. Otherwise, the following separate products should be installed on your computer in order to complete this tutorial section:

- **MX Form Validation**

The estimated completion time for this section is about 20 minutes. It depends on your authoring knowledge with **Macromedia Dreamweaver** (MX or MX 2004) and **MX Kollection 3**.

Check e-mail and phone: Validate Form

When adding data in a table through **HTML** forms, you have no way to control what the user enters. Now, with **MX Form Validation** you can define specific rules for each form element. If you have the **MX Kollection 3** bundle, **MX Form Validation** is included.

When managing contacts for a company or person, you want to make sure that the contact person is added correctly: with a valid e-mail address and phone number. **MX Form Validation** allows you to add rules for the e-mail and phone fields. If **JavaScript** is enabled in the browser, the check takes place on the local machine, saving a round-trip to the server.

To add these rules, you will have to follow the instructions below. First off, open the contacts *insert* page.

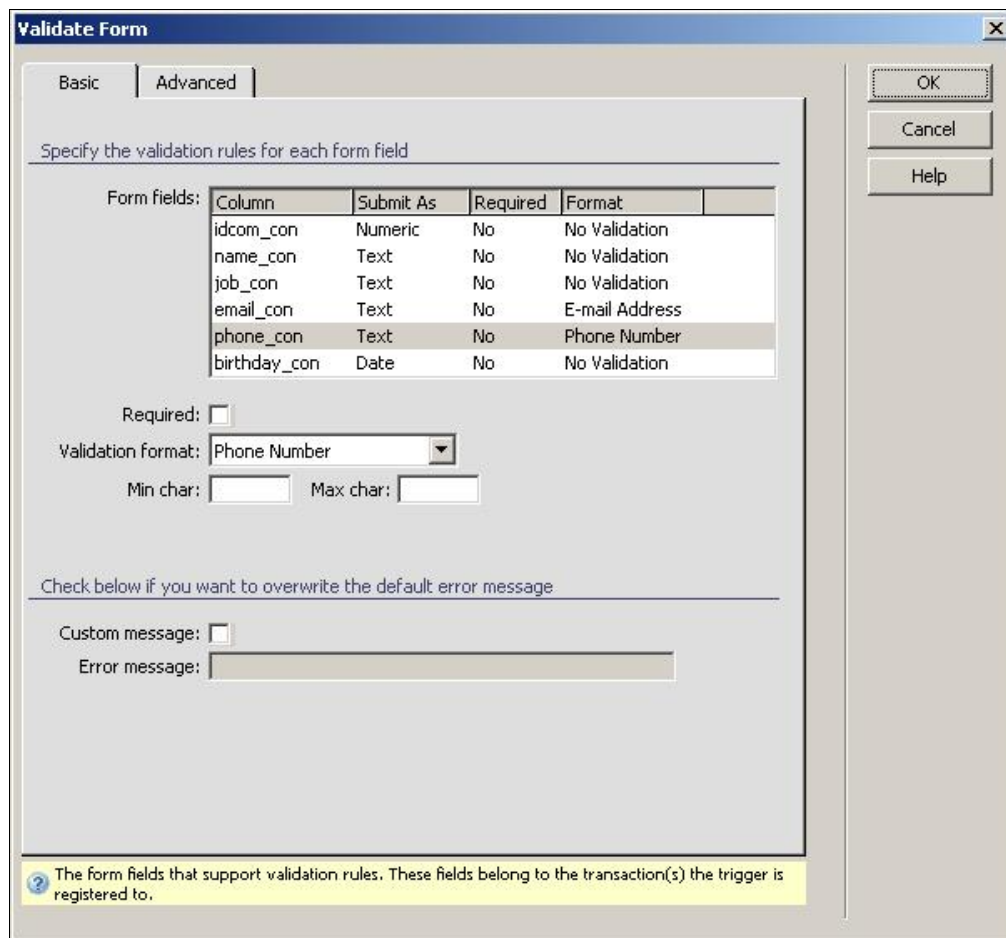
If a Validate Form server behavior has not been added by the wizard - the third step, you will have to apply the **Validate Form** server behavior. To access it, click on the **Plus (+)** button of the **Server Behaviors** tab, then select **MX Kollection -> Form Validation -> Validate Form**. If the **Server Behaviors** tab is not visible, you can open it from **Window -> Server Behaviors**. If the server behavior already existed, simply double click its name to edit it.

The **Validate Form** dialog box will open at the **Basic** tab. The **Advanced** tab is of no interest at this time, as all necessary options can be set here.

In the **Columns** grid, all fields that have a form element counter-part are displayed. Beside the name, the submit type, the required state and the validation format are displayed. You'll have to select each field to modify from the grid, and only then use the fields below to modify its rules.

To add validation for the e-mail address and phone number, follow the next steps:

1. Select the **email_con** field from the grid. Add the following rule:
 - The e-mail address should be mandatory when entering contact details, so check the **Required** checkbox.
 - From the **Validation Format** drop-down menu select **E-mail Address**. This will force the user enter addresses that respect the format [name@server.domain](#)
 - As rule elements are added to a field, the grid updates to reflect the changes.
2. The second element to add validation rules to is the **phone_con** field. Select this field in the grid, and follow the next steps:
 - The phone number should not be mandatory when adding a new contact, so the **Required** checkbox should remain unchecked.
 - From the **Validation Format** drop-down select **Phone number**. This way only phone numbers can be entered. If you need extra options, you can set a minimum and maximum number of characters that the user can enter so that the number is valid.



Form fields:	Column	Submit As	Required	Format
	idcom_con	Numeric	No	No Validation
	name_con	Text	No	No Validation
	job_con	Text	No	No Validation
	email_con	Text	No	E-mail Address
	phone_con	Text	No	Phone Number
	birthday_con	Date	No	No Validation

- Next to the regular validation options, you can customize error messages for each field that has validation rules applied. Just select the field from the grid, check the **Custom message** checkbox and you can enter your own text in the **Error message** text field.

Once you're done defining the validation rules for the desired fields, click on **OK** to apply the server behavior. You will notice some changes in the page design, as next to each field that has a validation rule applied some text is shown. This is where the mandatory format and the required text will be displayed in the browser.

After ensuring that your users can not enter bad information, you can take one more step to make your web application easier: learn how to automatically delete the contacts associated with a company, when the company itself is deleted. Check this out in the next part.

Remove associated contacts: Delete Detail Records

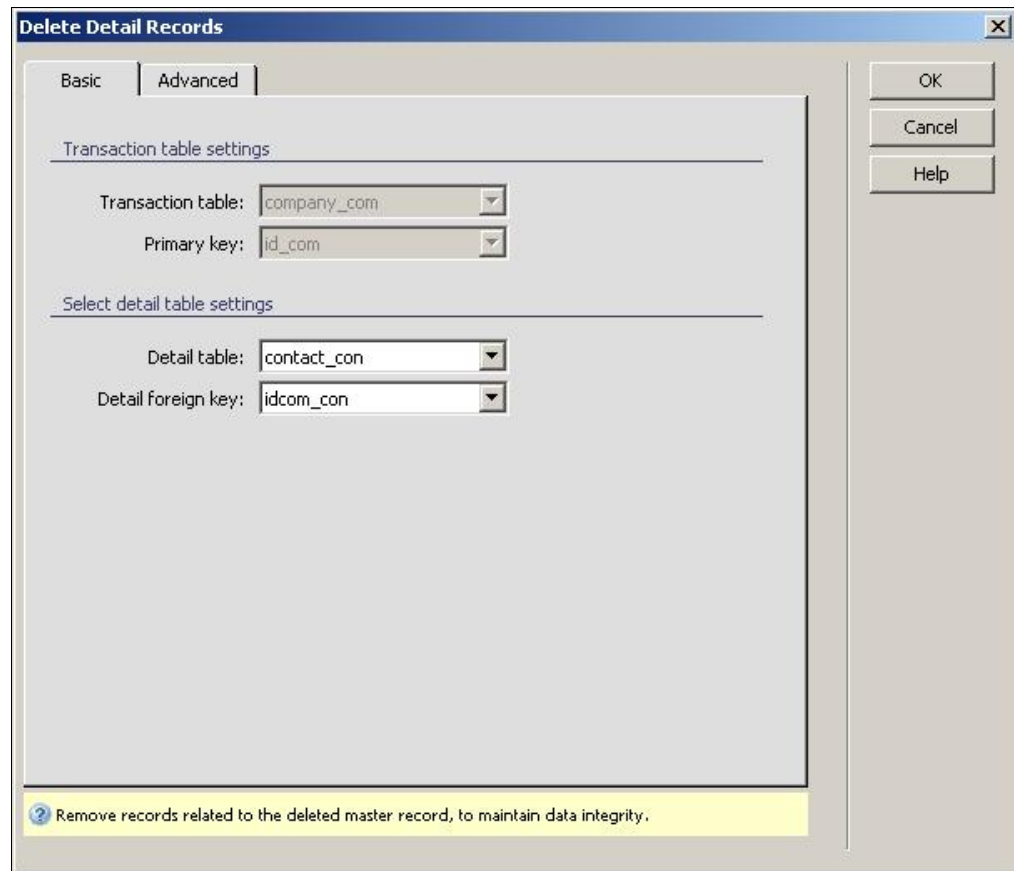
When you've created the company *delete* page in a previous section of this tutorial, the only thing implemented was a simple delete from the company table. Indeed, this removes the selected company from the database, but what happens to the contacts associated to the company? The answer is absolutely nothing. This means you will be stuck with some contact data you actually can't access from the site, and you don't need. These are called orphan records, as their parent company has been deleted.

One way to solve this problem is to fire up your database server management utility (**PHPMYAdmin**, **Microsoft Access** etc) and manually delete the remaining records. But this means an extra operation, one which is unnecessary.

The other (simpler) way is to use **ImpAKT3's Delete Detail Records** trigger. This trigger can only be used on pages that contain a delete operation, and allows implementing a cascade delete: all records in a detail table, associated with the current record that is being deleted, will be erased too.

To add this functionality to your site, follow the next steps:

1. Open the *delete* page from the company folder. Remember that you can only use this server behavior on a page containing a delete transaction for the master table.
2. With the *delete* page opened in **Dreamweaver's** editable area, apply the **Delete Detail Records** server behavior. To access it, click on the **Plus (+)** button of the **Server Behaviors** tab, then open **MX Kollection -> Form Validation -> Delete Detail Records**. If the **Server Behaviors** tab is not visible, open it from **Window -> Server Behaviors**.
3. Once the server behavior's dialog box opens, configure it to work on the desired detail table, in the way you want it to. Setting up the options is quite easy, and you can follow the instructions below:
 - Notice first that the interface is divided into two tabs: **Basic**, which allows you to set the tables, and field keys, and **Advanced**, where more in depth options regarding the trigger's properties can be set. Of interest right now is the **Basic** tab, as all trigger properties are automatically configured.
 - The first section of the **Basic** tab contains fields regarding the master (transaction) table. These fields are automatically retrieved from the delete transaction that already exists on page: the *company_com* table, with the **id_com** primary key field.
 - The second section is where you define the detail table.
 - In the **Detail table** drop-down menu select the table to act as detail; in this case, it's the contacts table: *contact_con*.
 - In the **Detail** foreign key drop-down menu, select the field that acts as a link between the master and detail table: **idcom_con**.
 - If you've used the same names, the dialog box looks like this:



- Now you can click on the **OK** button to add the server behavior to the page.
4. This is all it takes to implement the removal of associated contacts, along with the company they belong to. The final step is to edit the page header, so that it will suggest that contacts will be deleted too. Change the header text to "**Are you sure you want to delete the following company and its associated contacts?**" (without the quotes).
 5. Save your page and upload it to the server. When you will attempt to delete a company in the future, all its contacts will be removed too. You can check this by opening your database server management utility, and browsing through the contacts table, before and after the delete operation.

Now you can move on to the next part, where you will learn how to make your contact insertion page bullet proof, by adding a mechanism to prevent adding a contact for a company that doesn't exist.

Verify associated company: Check Master Record

When adding entries in a detail table where the user must manually enter the foreign key value, accidents are prone to happen. Also, when you use a drop-down menu, or any other way of deciding the foreign key value, there is still the risk that between the page load, with the master record options, and the actual insert, another user will delete the master record. Since the retrieval of the options and the insert operation are two different SQL transactions, and there will always be some time interval between them, something might go wrong.

This is why you should check for the existence of the master record before the actual insert. **MX Kollection 3** offers yet another tool to perform this task: the **Check Master Record** server behavior .

To add this functionality to your site, and thus make the insert contact page bullet proof, follow the next steps:

1. First off, open the *insert* page from the contact folder in **Dreamweaver's** editable area.
2. You should already have the **Insert transaction** on page, from the previous section of this tutorial. This is needed, as the **Check Master Record** uses transaction data for its configuration.
3. Apply the **Check Master Record** server behavior. Access it by clicking on the **Plus (+)** button on the server behaviors tab, and then going to **MX Kollection -> Form Validation -> Check Master Record**.
4. Configure the server behavior, by setting the options in the dialog box that opens. The user interface is divided into two tabs: **Basic**, containing options about the tables and fields to use, and **Advanced**, with options regarding the trigger properties. As the latter is automatically set up, you will only have to deal with the **Basic** options. The instructions below will guide you to quickly configure the trigger:
 - The **Transaction table** drop-down is automatically completed with the table used in the insert transaction (*contact_con* for this case), and disabled.
 - In the Foreign key drop-down menu you must select the transaction table's field that ensures the link to the master table. For this particular case, the field is **idcom_con**.
 - In the **Master table** drop-down menu select the database table that acts as master for the transaction table. Select *company_com*.
 - In the **Master primary key** drop-down menu select the master table's field containing the primary key: **id_com**.
 - You will also have to enter an **Error message**, that will be displayed when the master record check fails. Enter a suggestive text (e.g. "The selected company does not exist!").
 - If you've followed the instructions above, the dialog box should look like this:

Check Master Record

Basic | Advanced

Transaction table settings

Transaction table:

Foreign key:

Select master table settings

Master table:

Master primary key:

Error to be displayed when master record does not exists

Error message:

Enter the message that will be displayed when an error occurs.

OK
Cancel
Help

5. Click on the **OK** button to add the server behavior to the page.
6. You can now save your page and upload it to the server. If you wish to test the newly added server behavior, open the *insert* page in a browser window, and in another window delete the company. When submitting a new contact for that company, you will receive an error message.

Improve the Contact Management Application

In this part of the tutorial, you will enhance the web-application already created by adding features that will help save time and make it more user-friendly and interactive.

The following sections of the tutorial are optional, as they only add features to the already finished application. They will teach you how to:

- Add a base contact together with the company's addition to the database.
- Enhance the company add form so that more contacts can be added right away.
- Keep in touch with contacts by automatically sending birthday greetings.
- Keep count of each company's contacts by displaying their number aside the company name, in the main listings.

If you have the **MX Kollection 3** bundle installed, then you have all the needed tools. Otherwise, the following separate products should be installed on your computer in order to complete this tutorial section:

- ImpAKT
- MX Send E-mail
- MX Query Builder

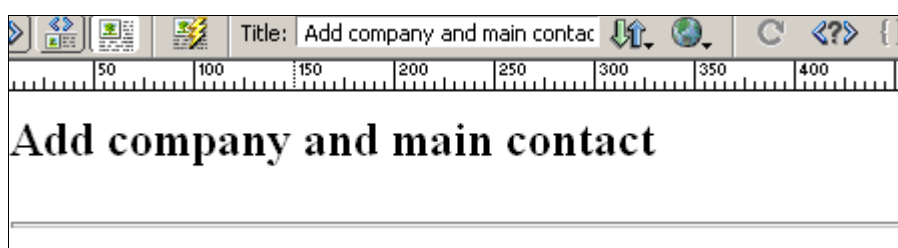
The estimated completion time for this section is about 30 minutes, depending on your authoring knowledge with **Macromedia Dreamweaver** (MX or MX 2004) and **MX Kollection 3**.

Insert both company and main contact

When adding a new company, almost in all cases you already have at least one contact for it. So, wouldn't it be easier if you could add the main contact alongside the company? This means that you will insert information into the company and the contact tables from the same page, using a single page submit operation.

In **MX Kollection 3** you can do this, by using the **Insert Into Two Tables Wizard**. To build such a page, you need to follow the next steps:

1. First create a new file in the site structure, in the company folder. The name used in this tutorial is *insertcomcon*. To create a new file, select the *company* folder in the **Files** tab, and then select the **New** option from the **File** menu. Type in the file's name, followed by the extension required by your server model .
2. Open the newly created file in **Dreamweaver**. The page design will be similar to the one used for the regular company insertion page. Copy the design section and change the header text to "**Add company and main contact**" (without the quotes):



3. Click in the empty paragraph after the horizontal rule, and apply the **Insert Into Two Tables Wizard**. You can open it by clicking on its button, found on the **MX Kollection** tab of the **Insert** bar.
4. The wizard's user interface can have three or five steps, depending on whether you have **MX Form Validation** installed or not. The first three steps define all options for the tables and fields used in the transaction , while the last two steps allow setting up validation rules for each form field.
5. This wizard benefits from the user interface persistence implemented by **MX Kollection 3**. If you haven't already created the insert page for companies and the insert page for contacts, you should do that now. If you already have created them, proceed with configuring the wizard:
 - In the first step you must configure the connection, table and primary key fields to use:

Insert Into Two Tables Wizard

Step 1/5: Select the two tables to insert into

Specify database information

Connection: connContacts Define...

Master table: company_com

Primary key column: id_com Numeric

Detail table: contact_con

Primary key column: id_con Numeric

Foreign key column: idcom_con

Specify redirect page

After inserting, go to: index.php Browse...

Interakt ? Specify the page where to go after inserting the records.

< Back Next > Finish Cancel Help

- The **Connection** drop-down menu already contains the connection you've already used throughout this tutorial.
- In the **Master table** drop-down menu select the `company_com` table. The `id_com` field is selected by default in the **Primary key column** drop-down menu.
- In the **Detail table** drop-down menu select the `contact_con` table. The **Primary key column** and Foreign key **column** will be automatically completed with `id_con` and `idcom_con`.
- Thanks to the user interface persistence, the **After inserting, go to** text field already contains the name of the `index` file.
- At this point you can already click the **Finish** button to exit the wizard's configuration, as the fields to use for each table's form have their properties set from the persistence cache. Browse through the wizard's next steps to make sure everything is in order:

Insert Into Two Tables Wizard

Step 2/5: Configure the master form fields

Set master form fields properties

Form fields: + -

Column	Label	Display as	Submit as
name_com	Name:	Text field	Text
address_com	Address:	Text area	Text

Label:

Display as:

Submit as:

Default value:

The list of fields associated with the transaction and belonging to the master table.

< Back Next > Finish Cancel Help

- the company's name and address fields ...

Insert Into Two Tables Wizard

Step 3/5: Configure the detail form fields

Set detail form fields properties

Form fields: + -

Column	Label	Display as	Submit as
name_con	Name:	Text field	Text
job_con	Job:	Text field	Text
email_con	Email:	Text field	Text
phone_con	Phone:	Text field	Text
birthday_con	Birthday:	Text field	Date

Label:

Display as:

Submit as:

Default value:

Build dynamic forms for inserting records in two related database tables.

< Back Next > Finish Cancel Help

... and the contact's table fields.

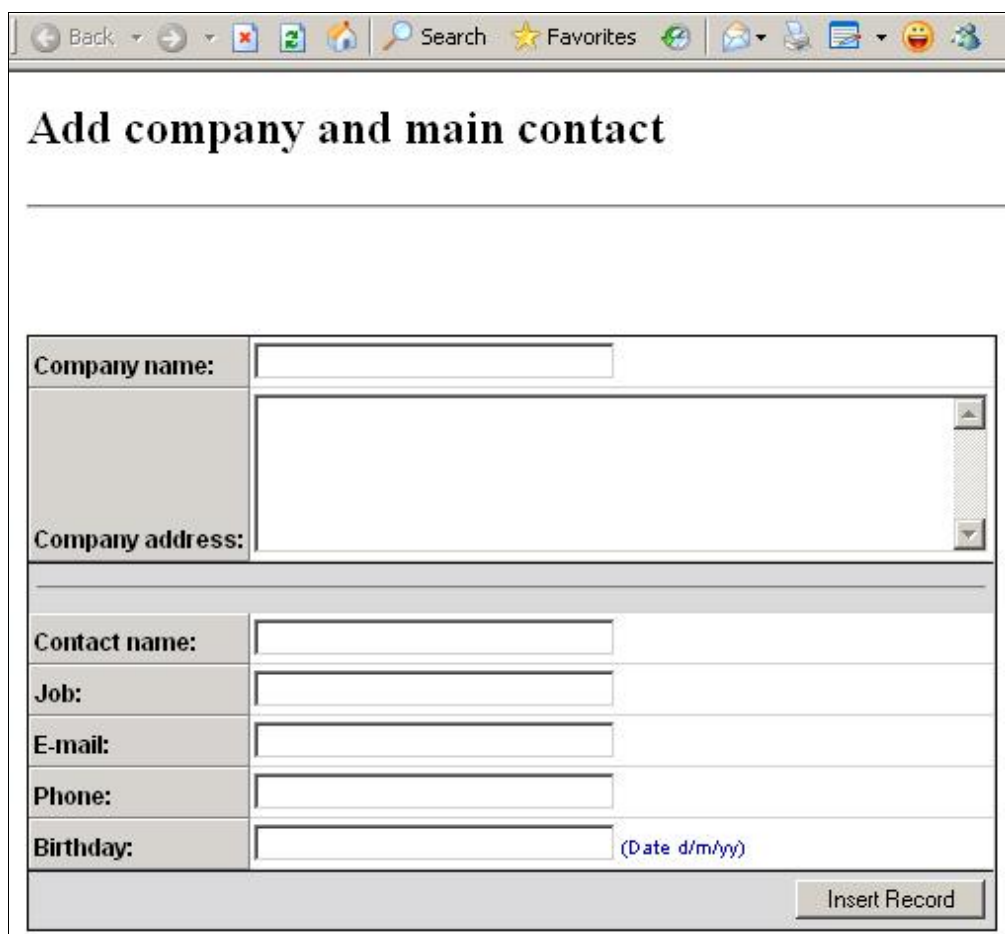
- As you can see, the *idcom_con* field does not appear in the **Form fields** grid of the wizard's third step. That is because the value for the foreign key field is set automatically to the unique identifier of the company that is added.
- At this point click on the finish button to add all elements into the page. The wizard will add the HTML form elements, as well as the server behaviors used to perform the insert operation.

The screenshot shows a web browser window with the title "Add company and main contact". The browser's address bar and navigation buttons are visible at the top. Below the title, there is a horizontal line and a placeholder text "{Display error message.}". The main form area is divided into several sections:

- Company name:** A text input field with the placeholder text "{rscompany_com.name_c}" and a "{Hint} {Error}" label.
- Company address:** A text area with the placeholder text "{rscompany_com.address_com}" and a "{Hint} {Error}" label.
- Contact name:** A text input field with the placeholder text "{rscontact_con.name_con}" and a "{Hint} {Error}" label.
- Job:** A text input field with the placeholder text "{rscontact_con.job_con}" and a "{Hint} {Error}" label.
- E-mail:** A text input field with the placeholder text "{rscontact_con.email_con}" and a "{Hint} {Error}" label.
- Phone:** A text input field with the placeholder text "{rscontact_con.phone_con}" and a "{Hint} {Error}" label.
- Birthday:** A text input field with the placeholder text "{rscontact_con.birthday_c}" and a "{Hint} {Error}" label.

An "Insert Record" button is located at the bottom right of the form.

- Now the *insertcomcon* page is finished, and you can save and test it. Preview the page in the browser by pressing the **F12** key, and add a new company and its main contact. When you return to the listing page, you will notice the new pair being displayed:



The screenshot shows a web browser window with a title bar containing navigation buttons (Back, Forward, Stop, Refresh, Home) and search, favorites, and help icons. The main content area displays a form titled "Add company and main contact". The form is divided into two main sections. The first section is for company information, with a label "Company name:" next to a text input field, and a label "Company address:" next to a larger text area with a vertical scrollbar. The second section is for contact information, with labels "Contact name:", "Job:", "E-mail:", "Phone:", and "Birthday:" next to their respective text input fields. The "Birthday:" field includes a date format hint "(Date d/m/yy)". At the bottom right of the form is a button labeled "Insert Record".

If you prefer this method of adding a company, replace the link to add a company from the *index* page, to point to the new insert page (*insertcomcon*).

In the next part, you will learn how to enhance this insert method by allowing the choice of adding some more contacts after the company insert.

Add more contacts after company insert

In this section of the tutorial, you will learn how to configure the newly created insert page (*insertcomcon*), so that you can decide if you want to add more contacts after the submit or not.

To accomplish this, you will need a way for the user to pass his decision (a menu, radio-button, checkbox), and a way to redirect to a specific page based on a condition. As for the passing the decision part, there is no problem, only the choice of the HTML form elements remains. For the conditional redirect action, however, you would have some hand-coding to do.

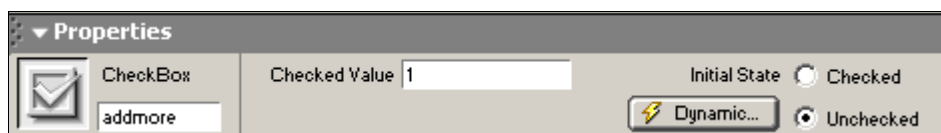
MX Kollection 3 has this need covered too, however, through the **Redirect To Page** trigger. As all the other **MX Kollection 3** triggers, this one can be turned into a conditional trigger, simply by using the expression builder on the advanced tab.

To add this functionality, you will have to alter the design of the insert page (e.g. *insertcomcon*), and only then add the application logic that performs the actual task.

Alter the page design

This alteration concerns adding a new form element, so that the user option of whether to add some more contacts or not can be passed to the trigger. The simplest of elements that can be used effectively is a checkbox. To change the design, follow the next steps:

1. Click on the last row that contains a text box. If you haven't removed any fields in the wizard configuration, it should be the row containing the birthday field.
2. Add a new row below the current selection. To do this, right-click in the cell and select **Table -> Insert Rows and Columns**. Click on **OK** in the dialog box that opens.
3. In the new row, type in the first cell "**Add more contacts**" (without the quotes) as a label for the checkbox.
4. Click in the second cell. Here is where you will insert the checkbox. Add it from the **Forms** tab of the **Insert** bar, by clicking on its button. Configure the checkbox, by changing its name to "**addmore**" and set the **Checked value** field to **1**. Also, leave the **Initial state** on the **unchecked** setting.



5. Now the checkbox appears on page, but to ease future work you should add it to the **Bindings** panel. Click on the **Plus(+)** button of the **Bindings** panel and select **Form Variable**.
Note: For ASP users, you should select the **Request Variable** type, then in the dialog box select Request.Form from the drop down menu.
6. Enter the checkbox name in the dialog box's text field: **addmore**. Click on **OK** to add the new variable to the environment.



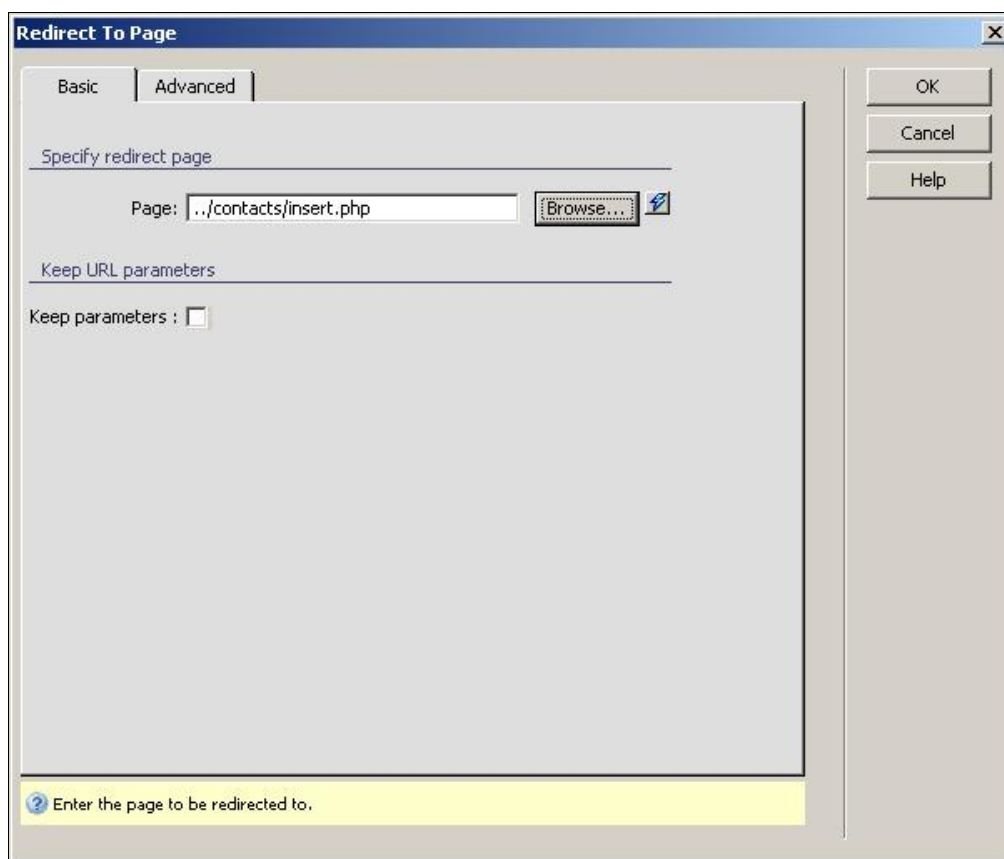
7. These are all the changes you will make in the page design. Save the page, and proceed to adding the trigger for the redirect operation.

Add application logic

The actual operation will be based on the **Redirect To Page** trigger, which will be set to start on a custom condition: that the **Add more contacts** checkbox is checked.

To add and configure the redirect trigger, follow the instructions below:

1. Click on the **Plus (+)** button of the **Server Behaviors** tab, and select **MX Kollection -> Forms -> Redirect to page**. This will open the redirect trigger dialog box.
2. For this trigger you will have to set options both for the **Basic** and **Advanced** tab. On the **Basic** tab, just enter the page to which the user will be redirected. Either enter its name, or use the **Browse** button to locate the *insert* page in the contact folder.



3. Move on to the **Advanced** tab, where you can define a starting condition for the redirect. You can either write the condition in the text-field, or use the condition builder, by pressing the **Build Condition** button. Let's use the latter.

Redirect To Page

Basic | Advanced

Enter trigger name

Trigger name: (unique per page)

Register trigger to transaction(s)

Transactions:

Transaction	Priority	Type (read-only)
ins_company_com	90	END

Priority:

Type:

Condition:

A trigger can be registered to a transaction as a conditional trigger. It will be executed only when the defined condition is met.

OK
Cancel
Help

4. The **Condition Builder** offers an easy way to create starting conditions that use two or more terms, and an operator. Its dialog box is divided into two tabs: **Basic**, where you can define a simple condition based on two terms and the default operators, and **Advanced**, where you can enter more complex conditions in a text-area. Since in this tutorial you'll only need to compare if the **addmore** variable has been checked, the **Basic** tab will suffice.
5. It's time to build the actual condition. Follow the instructions below to construct it:
 - For the first term you will use the **Form Variable** defined when you've added the checkbox into the page. Click on the blue lightning bolt icon to open **InterAKT's** dynamic data selector. Here, select the **addmore** Form Variable.

InterAKT Dynamic Data

Basic

Get dynamic values from the source indicated below

Get values from: Form Variable

Variable: addmore

The corresponding mark-up code

Mark-up code: {POST.addmore}

Automatically generate dynamic data code from various sources.

OK
Cancel
Help

- Click on **OK** and notice that a dynamic construct has been added into the first element's text field.
- Next comes the operator. From the drop-down list select "==" to check for equality.
- For the second operator, you will use an entered value. Since the **Checked Value** property for the checkbox has been set to 1, this is the value that the condition should use. This way, if the trigger's condition that the checkbox value equals 1 is true, the redirect will take place. Otherwise, the default redirect of the insert operation (to the main *index*) will execute.

Condition Builder

Basic | Advanced

Build simple condition

Expression 1: {POST.addmore}

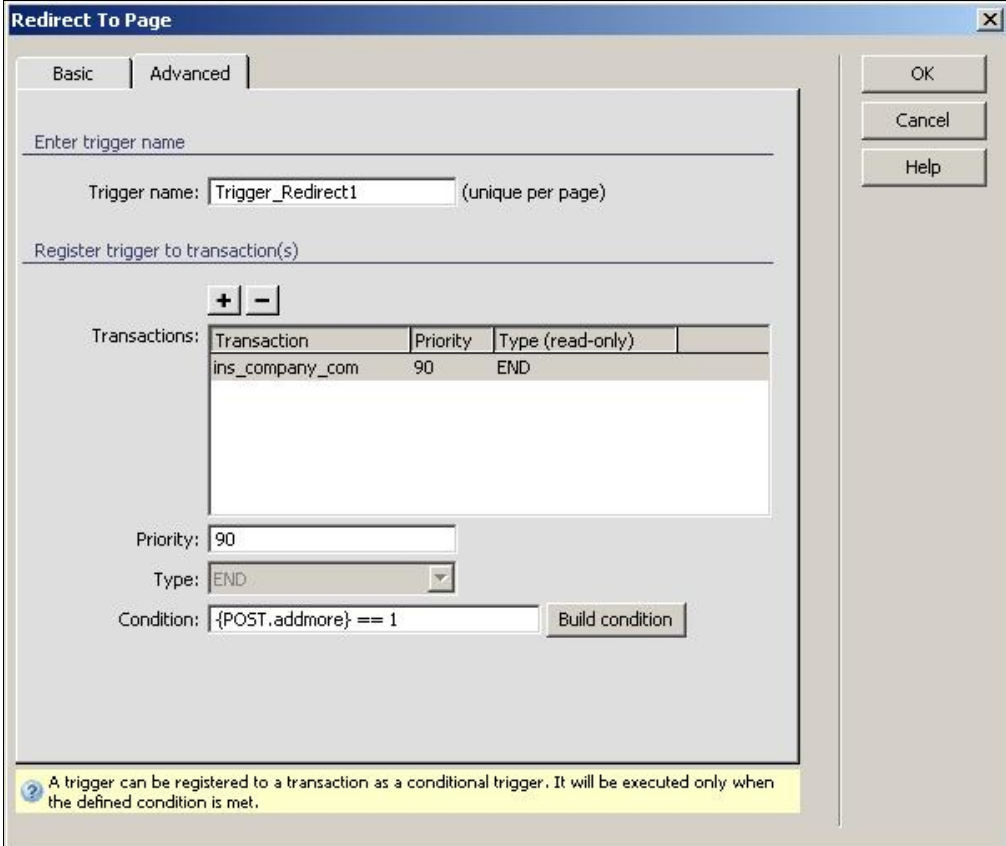
Condition: ==

Expression 2: 1

Enter the second member of the condition.

OK
Cancel
Help

- Click on the **OK** button to accept the condition. The expression will be displayed in the **Redirect To Page** trigger's **Advanced** tab, in the **Condition** text field:



Redirect To Page

Basic | Advanced

Enter trigger name

Trigger name: (unique per page)

Register trigger to transaction(s)

+

-

Transaction	Priority	Type (read-only)
ins_company_com	90	END

Priority:

Type:

Condition:

ⓘ A trigger can be registered to a transaction as a conditional trigger. It will be executed only when the defined condition is met.

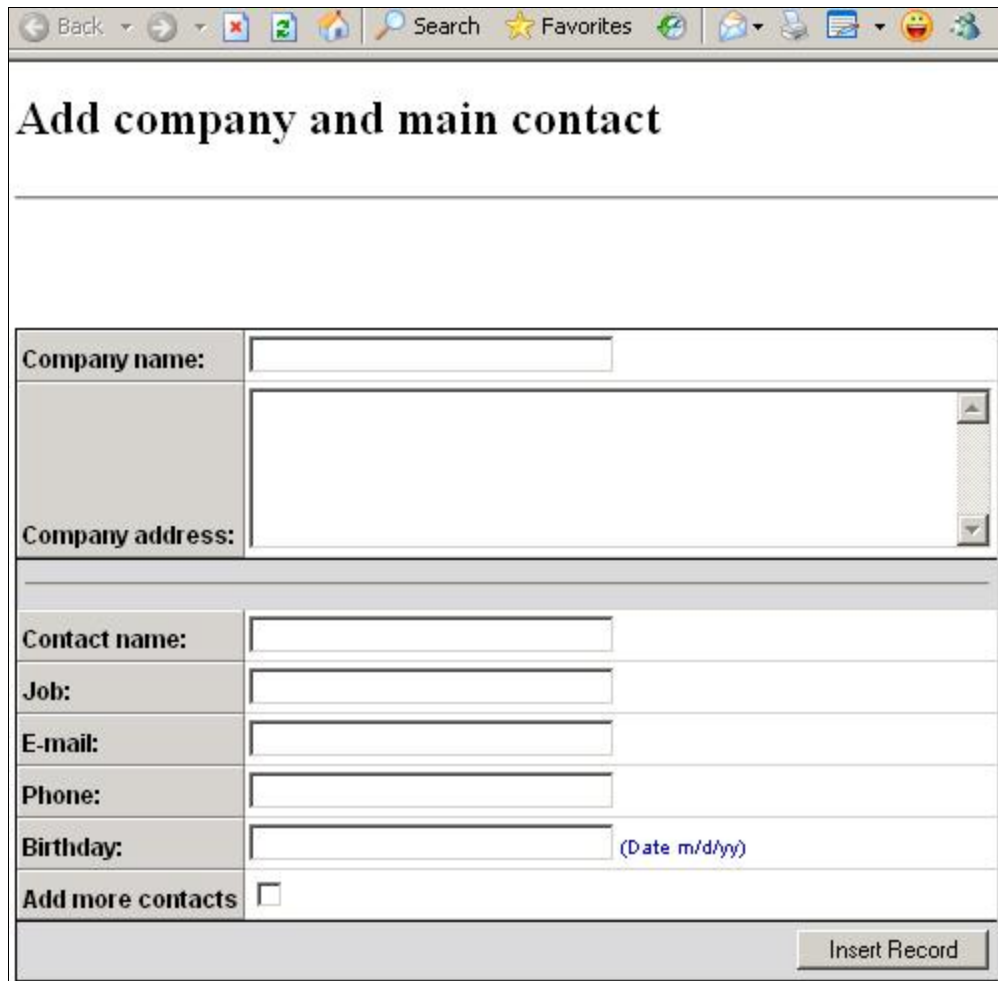
OK

Cancel

Help

6. Now the trigger configuration is finished, and you can click on the **OK** button to add the trigger into the page. Then you can save it, upload it on the server and test it in the browser.

When you add a company through this insert form, if the **Add more contacts** checkbox is checked, when submitting the page, you will be taken to the contact *insert* page, where you can continue adding entries:



The screenshot shows a web browser window with a title bar containing navigation buttons (Back, Forward, Stop, Refresh, Home) and search, favorites, and help icons. The main content area displays a form titled "Add company and main contact".

The form is divided into two main sections: "Company" and "Contact".

Company Section:

- Company name:** A single-line text input field.
- Company address:** A multi-line text area with a vertical scrollbar on the right side.

Contact Section:

- Contact name:** A single-line text input field.
- Job:** A single-line text input field.
- E-mail:** A single-line text input field.
- Phone:** A single-line text input field.
- Birthday:** A date input field with a placeholder "(Date m/d/yy)".
- Add more contacts:** A checkbox.

At the bottom right of the form, there is a button labeled "Insert Record".

Send automatic birthday greeting

The next enhancement to your web application is meant to ease the task of checking daily which contact's birthday it is and use your e-mail client to send a greeting. It will all be done in an automatic manner at the click of a button.

You will have to create a new page in your site's root folder, which will contain the elements that send the e-mail. To create it, select the **New** option of the **File** menu, in the **Files** tab. If the **Files** tab is not visible, open it from **Window -> Files**. Give it a suggestive name (e.g. *birthday.php*). You will later on add a link in the main page, which will point to this one.

Sending the birthday greetings will be done in a semi-automatic manner at first, the page displaying a list of all persons that were born on this day, and allowing you to decide whether to send the e-mail or not. The automatic part is that you'll only have to click one single button to send the greeting to all people in the list.

To start creating the page, open it in **Dreamweaver**, and continue with this tutorial: you will create the design first, and later on, you'll add the code that does all of the actions.

When creating the basic design, the approach is the same as for the contact or company *insert* and *update* pages:

1. On top of the page, using a header 2 format, place the page title: "**Persons born today**".
2. Place a **horizontal rule**, to separate the header from the content.
3. Create a new paragraph where to display the list.

The list of people born on the day the page is being accessed is in fact a dynamic table, as it should retrieve the names from the contacts table. Before using the **Dynamic Table** command however, you have to create a recordset containing the names.

To create the recordset, follow the instructions below:

1. Click on the **Plus (+)** button of the **Bindings** panel. If the **Bindings** panel is not visible, open it from **Window -> Bindings**.
2. Select **Recordset (Query)** from the pop-up menu.
3. The **Recordset** dialog box opens in the **Basic** form. Switch to the **Advanced** form by clicking on the **Advanced** button. In this form, the user interface allows you to enter the SQL query by hand. This way, you can create more complex queries that do exactly what you need.
4. In the **Connection** drop-down menu select the database connection used for your site.
5. In the **SQL** text area, enter the following query:
For **MySQL**:

```
SELECT *  
FROM contact_con  
WHERE date_format(birthday_con, '%m%d') = date_format(now(), '%m%d')
```

For **Microsoft Access**:

```
SELECT *  
FROM contact_con  
WHERE birthday_con = Date()
```

Note: Microsoft Access uses the same date format as your operating system to store date values.

For **Microsoft SQL Server**:

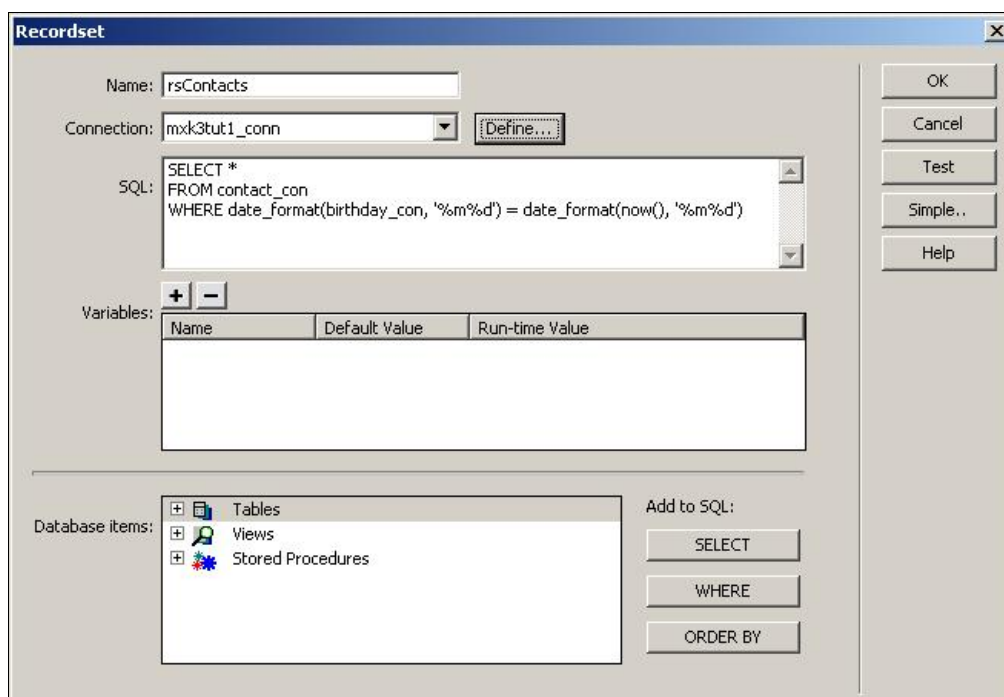
```
SELECT *  
FROM dbo.contact_con
```

```
WHERE DAY(dbo.contact_con.birthday_con) = DATEPART(DAY,GETDATE())
and MONTH(dbo.contact_con.birthday_con) = DATEPART(MONTH,GETDATE()).
```

For **PostgreSQL**:

```
SELECT *
FROM contact_con
WHERE birthday_con = NOW()
```

- The query above is used to retrieve from the `contact_con` table only the entries containing in their birthday field the same day-month combination as for the current date. The **date_format** function retrieves from the data source specified as the first parameter, only the format specified by the second parameter. The call: `date_format("2005-03-15", '%m%d')` will return "03-15", meaning the month and day. The `now()` function returns the current date.
- As you can see, the query above returns only the entries that have the birthday field set on the current day. The SQL query is the last thing done on the new recordset dialog box. Click the **Test** button to see if anyone is born on this day, or the **OK** button to add the recordset into the page.



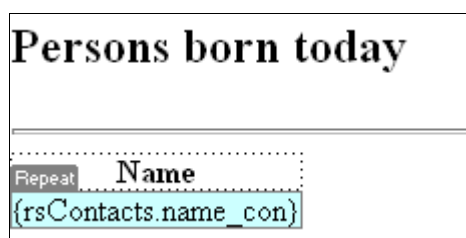
After having created the recordset, you can create the listing of its content, with the use of a dynamic table. To add a dynamic table, click on its button in the **Application** tab of the **Insert** bar. Configure it to display entries from the recordset created earlier, `rsContacts`, and to display all records. Also, set the table border, cell spacing and padding to 0. Clicking on **OK** will add the dynamic table into the page.



At this point, a table with all data concerning the contacts has been added in the last page paragraph. Not all of the entries are useful, or meaningful, as only the actual contact name is of interest. Therefore, delete all table columns except the ***name_con*** one.

For the remaining column, change the ***name_con*** static text into a more understandable one: "**Name**". Also, set the cell as having the header property, from the **Property inspector**.

Now your page should look like this:



Next comes the switch that will start the mail sending operation: an HTML form button. Place one in a new paragraph below the dynamic table. When asked if a form tag should be added, answer **Yes**. To make the button more intuitive, replace the **Submit** label with: "**Send birthday greetings**". Leave the button's name as is, "**Submit**".

The mail sending is implemented through the use of one of **MX Kollection 3**'s trigger: the **Send E-mail to Recipients from Recordset**. However, as any trigger, it needs a transaction on page. As there is no need to add, update or delete anything, the only one that will suit this purpose is the custom transaction. The custom transaction allows you to define the fields to use, and the SQL which will use it. Also, you can just use it as a placeholder, with no **SQL** operation at all.

Add a **Custom Transaction** to the page, by clicking on the **Plus (+)** button of the **Server Behaviors** tab, and selecting **MX Kollection -> Forms -> Advanced -> Custom Transaction**. Configure it as follows:

1. In the **Connection** drop-down menu select the database connection defined for your site. Although there will be no database operation involved, the transaction uses the connection for error handling.
2. The actual mailing operation has to start only when you press the **Send birthday greetings** button. To enforce this, you must set it as a starting rule for the custom transaction. In the **First check variable** drop-down menu select **Form variable** and in the text field that follows, enter the button's name: **Submit**.
3. In the **When finished, go to** text box, enter the page to return to after the e-mails have been sent. Either enter the name of the main page or select it by pressing on the **Browse** button.

- These are all of the options to set for the custom transaction. Click on the **OK** button to apply the configuration.

Now that a transaction exists on the page, you can also add triggers that register to it, as the **Send E-mail** trigger, for instance. Therefore, you can finally complete the last action in this tutorial - apply the **Send E-mail to Recipients from Recordset** trigger:

- To access the trigger, click on the **Plus (+)** button of the **Server behaviors** tab, and select **MX Kollection -> Send E-mail -> Send E-mail to Recipients from Recordset**.
- You can use this option instead of the regular **Send E-mail** trigger, as in the *rsContacts* recordset created earlier, only the contacts which have their birthday today are selected.
- Configure the trigger to send the correct e-mail to the correct addresses:
 - In the **Recordset** drop-down menu select *rsContacts*.
 - In the **Email to Field** select the table field that stores the e-mail address of the contact.
 - In the **From** text-box, enter the e-mail address from which the mail will appear to have originated. Enter an address that will identify you as the sender; e.g. admin@mysite.com.
 - In the **Subject** text-box enter the mail's subject. You can even enter dynamic data, by using the **InterAKT Dynamic Data** icon, and selecting the desired dynamic variable. For this mail, type the static text: "**Happy Birthday**", and then press the blue lightning icon to select the dynamic name of the contact. In the new dialog box set the values source to the *rsContacts* recordset, and the field to the one containing the contact's name:

InterAKT Dynamic Data

Basic

Get dynamic values from the source indicated below

Get values from: Recordset Field

Recordset: rsContacts

Field: name_con

The corresponding mark-up code

Mark-up code: {rsContacts.name_con}

This is the generated mark-up code, based on your dynamic data selection.

OK

Cancel

Help

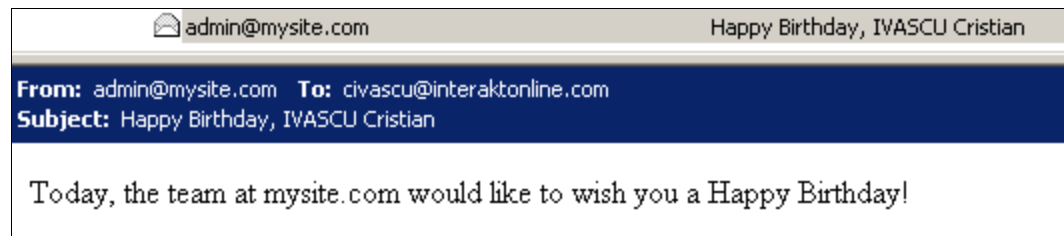
- When you click on **OK** in the **InterAKT Dynamic Data** interface, some code will be added to the **Subject** field. It will be replaced at runtime with its dynamic value.
- In the **Body** drop-down menu, select the **Write content** option to type in the mail's content.
- In the **Content** text area enter the message to pass to the recipient; in this case a **Happy Birthday greeting** from you:

- This is all you have to configure for the trigger, in order for the e-mail sending to work. You should check the E-mail Settings in the **InterAKT Control Panel**, where you can set the main options for your server. If these are not properly set, the messages will not get delivered.
- Click on the **OK** button to apply the **Send E-mail to Recipients from Recordset** server behavior.

Note: If using the **ASP VBScript** server model you must configure the E-mail server settings in the Control Panel. You must fill in the server address (or name), the port (by default it is 25), the user name and the password. Optionally, you can also set the default sender field.

This was the last step to create the birthday greetings page. You can now save, upload and test it. If no names appear in the list, it may be due to the fact that no contacts in your table are born on this day. Either try again another day, or enter a contact whose birthday is today.

When one of the contacts checks his mail, he will have a pleasant surprise:



Note: When using the **Send E-mail to Recipients from Recordset** server behavior to send messages to a large number of recipients, make sure the value of the `max_execution_time` variable in your `php.ini` configuration file is sufficiently large to support the execution of the code. Sending many e-mail messages might take longer than the default setting, depending on your server configuration and specifications. If you encounter an error similar to the following

```
Fatal error: Maximum execution time of 30 seconds exceeded in  
c:\server\site\includes\common\lib\email\Pear\Net\Socket.php on line 241
```

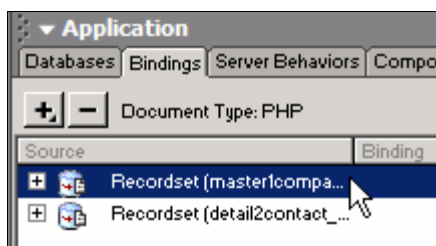
please increase the value of the `max_execution_time` variable in `php.ini`, or contact your network administrator.

Display number of contacts per company

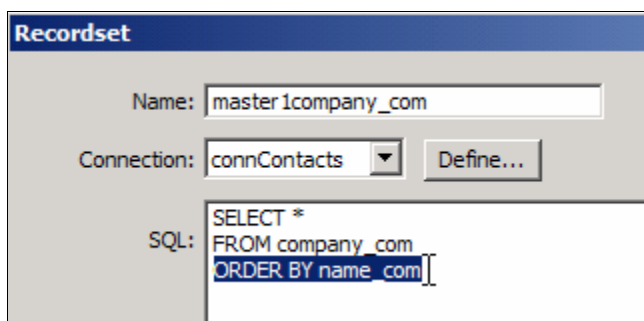
In this page you will use **MX Query Builder** to modify the main *index.php* page so that it will display the number of contacts per company. To get a count of a company's contacts, you will have to modify the *master1company_com* recordset, which was generated by the **Nested Repeat Region** server behavior.

The steps to take in order to accomplish this objective are as follows:

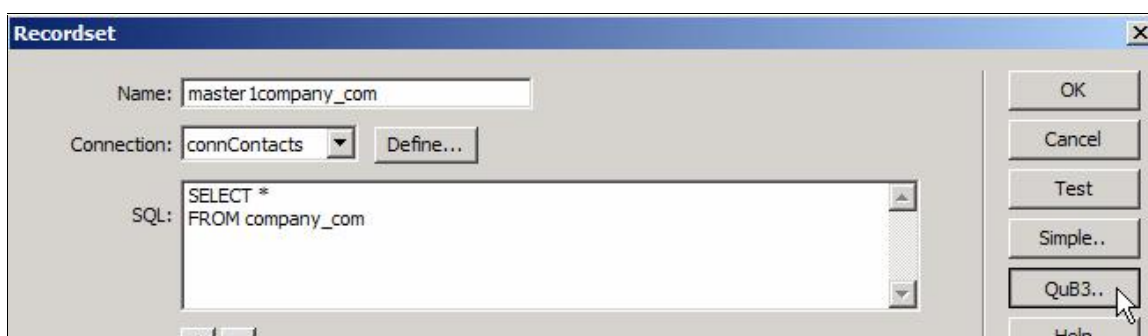
1. Begin by opening the main *index.php* page. In the Application panel, under the Bindings tab, you'll see the two queries which were generated by the Nested Repeat Region earlier in the tutorial.



2. Double click on the first recordset, *master1company_com*, to open it.
3. When the recordset window opens, remove the *ORDER By name_com* line from the SQL query. This step makes the query easier to view when you open **MX Query Builder**.



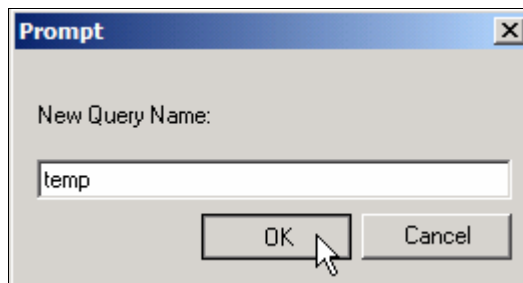
4. Once the query code looks like the one below, click the **QuB3** button on the right:



5. In the **QuB Visual Recordset** window that appears, click the **New Query** button:



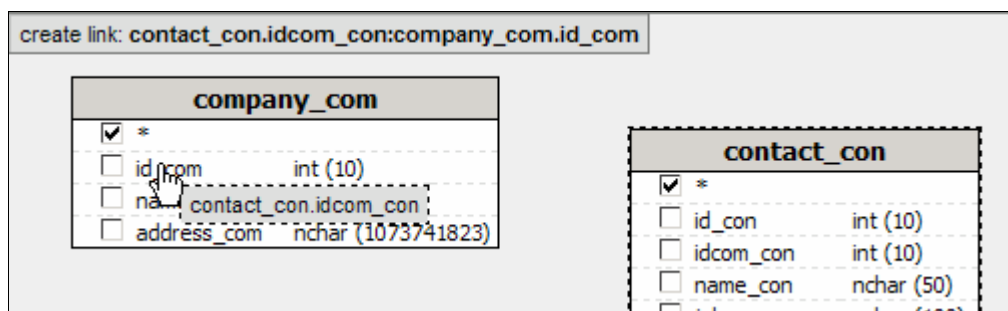
6. Type a temporary name for the query like "temp", then click the **OK** button:



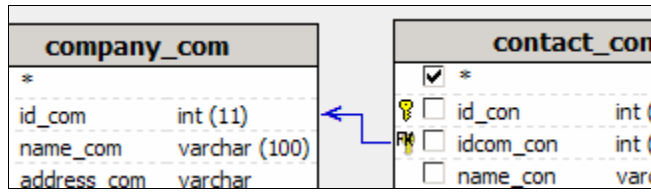
7. After clicking OK, the QuB interface will open in a separate window. On the left side there's a **Tables** panel containing a list of the tables in the current database. Select both tables as shown below:



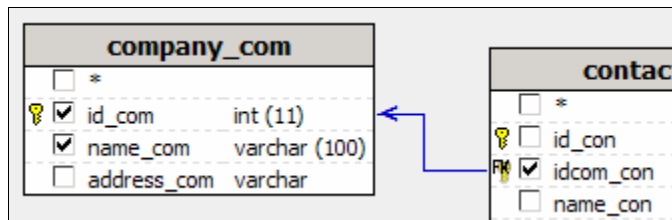
8. Now both tables will be in the **Database Diagram**. Create a table relation by dragging the **idcom_com** foreign key and dropping it on the **id_com** field:



9. A blue line will now appear between the two tables. You don't have to do anything, just look at it.



10. For this query we will use the **id_com**, **name_com** and **idcom_con** fields. Use the checkboxes to select them from the **Database Diagram**:



11. After checking the boxes, the fields will now appear below in the Query Management panel.

Column Name	Alias	Sort
company_com.name_com		
company_com.id_com		
contact_con.idcom_con		

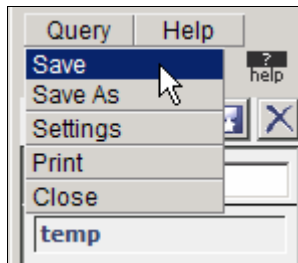
Select DISTINCT

12. With the necessary fields now listed, the various options can be used to further organize the data. On the **contact_con.idcom_con** field, we'll use the count option in the **Grouping** drop-down menu.

13. This last step will give you a count of all the contacts associated with a company. You can verify the result in the **SQL/Results Preview** Panel:

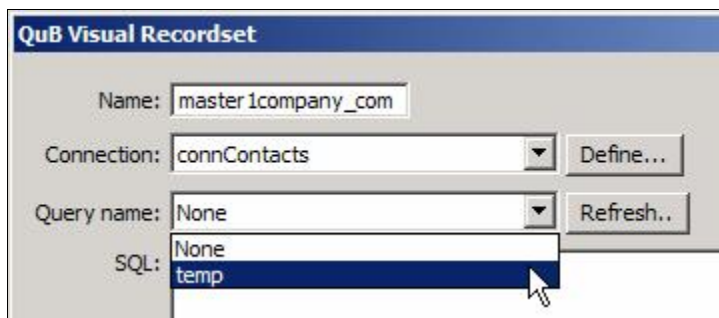
name_com	id_com	count_idcom_con_1
InterAKT Online	1	3
Macromedia	2	2
Oracle	3	2

14. If the results match what you're looking for, then save the Query using the Query menu:

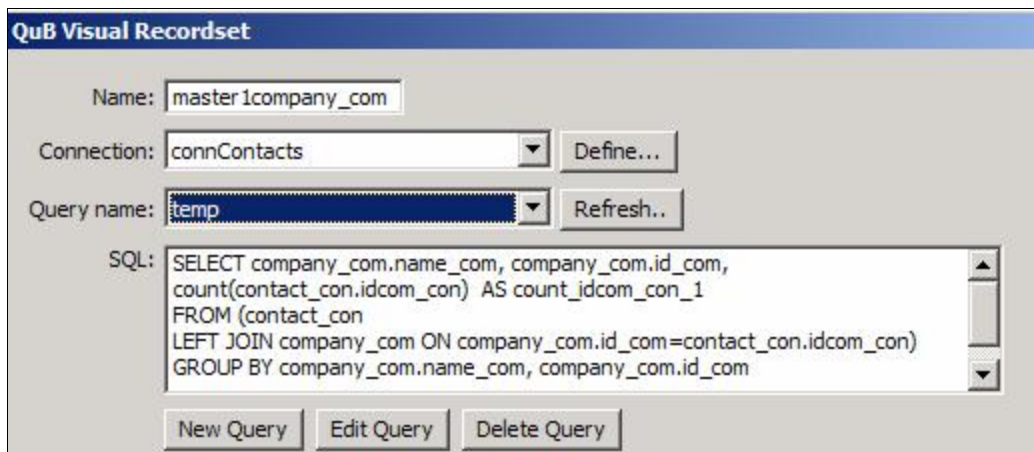


15. Now that the query is saved, you can close the **MX Query Builder** interface and go back into **Dreamweaver**. The **QuB Visual Recordset** window should still be open.

16. In the **Query name** dropdown, select the query you just created:

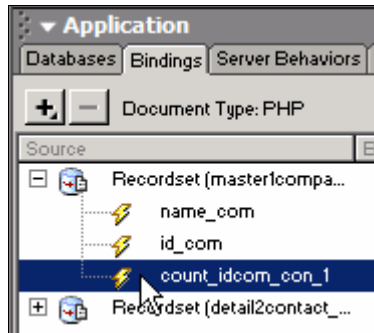


17. After selecting the query name, the **SQL** text area will contain the code built from the **Query Builder** interface.

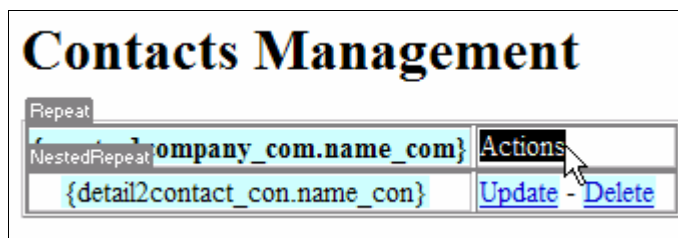


18. Click OK. The new recordset will now appear alongside the others in the Application panel.

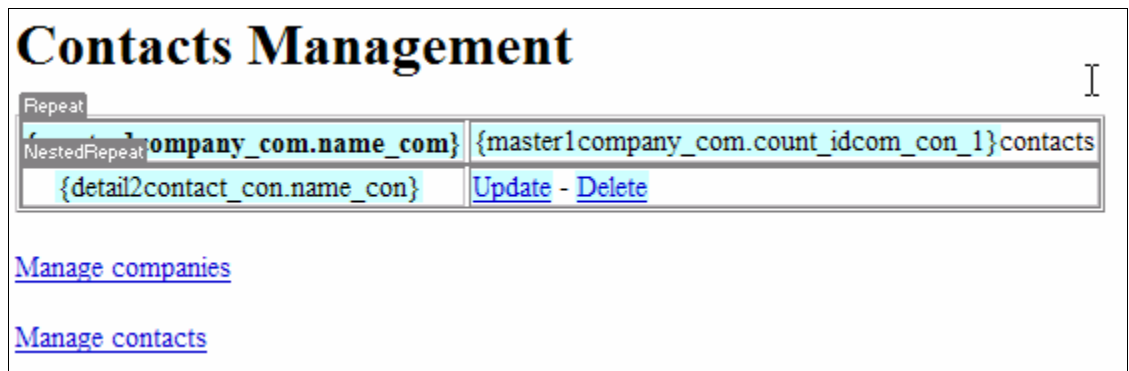
19. Now in the **Application** panel > **Bindings** tab, you can open the recordset and find the **count_idcom_con1** field listed:



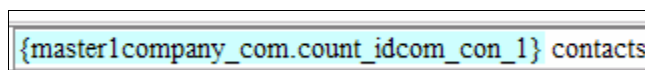
20. Select the text labeled "Actions" in the **Nested Repeat Region** table:



21. Take the **count_idcom_con_1** field (shown in step 19) and drag it over to the **Nested Repeat Region** table. Drop it in the place of the "Actions" text. When you're done, the table should look like the one below:



22. Just type the word "contacts" in the same cell and you're done:



Now you can view the finished page, with the number of contacts in the browser:

Contacts Management

InterAKT Online 3 contactsAlex Colorado [Update](#) - [Delete](#)Chris Benton [Update](#) - [Delete](#)Nikita Jelinski [Update](#) - [Delete](#)**Macromedia** 2 contactsGeorge Palmer [Update](#) - [Delete](#)Tony Norberto [Update](#) - [Delete](#)

This concludes the Contact Management Application tutorial.

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