# Table of Contents

# Overview

This tutorial is designed to help you understand how to use the MX Widgets to create powerful HTML forms. We will present all the Server Behaviors included in the package, together with the generated widget code.

We will create a small dynamic PHP site that will be used to manage a travel journal. The journal will store product shipments in various cities, made by specific products with the car driven by different drivers. Looking at a real case problem, we see this kind of application being used by shipment companies like DHL or Fedex, which ship merchandise using cars (for the last mile) in the whole world.

To exemplify all our widgets, we have chosen to manage in our journal various dynamic properties: the shipped product (a simple dynamic dropdown), the product measurement unit (which is a control that manage an “n to 1” database relation), the product price (a control that will demonstrate the numeric data format), the delivery car number (a control that will enforce a specific mask for a field), an entrance date (a textfield that restrict the input format to date), the exit date (displays a calendar to allow easy date selection), the continent-country-city cascaded property (a cascaded master/detail menu) and the car driver (a dropdown menu allowing you to select and add a driver name).

We'll use MySQL as database server and the PHAkt2 server model. However, the same behaviors will function in a similar way with the PHP_MySQL server model.

The generated site will benefit from the widgets power, to allow the site users to enter information into the HTML form with ease.

# Prerequisites

## *Intended audience*

This tutorial is intended for the Dreamweaver MX developers interested in easing their work when designing dynamic web sites. You will need a basic understanding of PHP development using Dreamweaver MX and a limited knowledge of SQL, including how to create a database and a query.

## *Requirements*

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

In order to use the MX Widgets Suite, you'll have to install the following software programs :

| | |
|---|---|
| Macromedia Dreamweaver MX | http ://www.macromedia.com/ |
| | Dreamweaver MX 6.1 updater |
| MX Widgets Suite 1.0 | http ://www.interakt.ro/products/MXWidgets/ |
| PHAkt2 (only for the PHP_ADODB server model) | http ://www.interakt.ro/products/PHAkt/ |
| PHP | http ://www.php.net |
| Apache | http ://www.apache.org/ |
| A MySQL database | http ://www.mysql.com/ |

**NOTES :**

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

## *Typographic conventions*

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

- database name will be displayed using bold font "**`Widgets_database`**"
- database table : using a italic font "*continents_con*"

- database field will be displayed using a bold italic font "***id_con***"
- site name : underlined font "Widgets"
- site page : mono spaced italic "`index.php`"
- recordset name : underlined italic "*recordset*"
- application button, menu or panel : bold font "**Button**"
- source code : monospaced font "`<?php echo "MXWidgets tutorial"?>`"

### *Files*

The MX Widgets tutorial package includes the SQL script used to create the tutorial database, the final site files (the PHP_MySQL version and PHAkt version) and this document. You can use these files to overcome potential blocking problems or to compare them with your results.

| | |
|---|---|
| The database generation script : | **mxwidgets.sql** |
| The MXWidgets website : | **MXWidgets_site_ADOdb.zip** |
| | **MXWidgets_site_MySQL.zip** |
| This tutorial : | **MX Widgets Suite Tutorial.pdf** |

# Functional overview

### *Widgets Explained*

A widget is an element of a graphical user interface that displays information or provides a specific way for an user to interact with the application. Widgets include icons, pull-down menus, buttons, selection boxes, progress indicators, on-off check marks, scroll bars, windows, window edges (that let you resize the window), toggle buttons, forms, and many other devices for displaying information and for inviting, accepting, and responding to user actions.

In the web browsers there is a limited number of widgets: checkbox, radio button, radio group, textfield, textarea, dropdown menu, hidden field. In order to extend these default controls, we have created the MX Widget package with seven more very useful widgets included:

- **Editable Dropdown** – to allow you to type in a HTML menu to quickly locate a record
- **Dependent Dropdown** – allows users to cascade master/detail menus
- **Calendar** – allow you to select a date from a calendar pop up
- **Smart Date** – can be used to control the date format into an HTML `<input>` tag
- **Masked Input Field** – users will be able to enter content in fields using specific formats
- **Numeric Input Field** – users will be able to enter only numbers in a textfield
- **Dynamic Search** – allows users to type in a search substring having the rest of the options suggested
- **n..1 Dependent Field** – will implement a "n-1" relationship between a menu and a textfield.

Our widgets are designed to work together with the database to allow the easy generation of complex dynamic forms, being real **data binded controls**.

# Configuring Your Site

## Database Design

This tutorial is meant to be a short overview of all controls included in the MX Widgets package. In order to implement the **Dependent Dropdown**, **n...1 Dependent Field**, **Numeric Input**, **Masked Input**, **Dynamic Search**, **Calendar**, **Smart Date** and **Editable Dropdown** widgets we need a database structure that will be used to fill the menu options.

We have created the **`MxWidgets`** database containing seven tables: *continents_con*, *countries_ctr, cities_cit*, *journal_jrn, mesunit_mes, drivers_drv* and *products_prd.*

Please find below the graphical representation of the database structure, made using the InterAKT visual query editor, QuB.

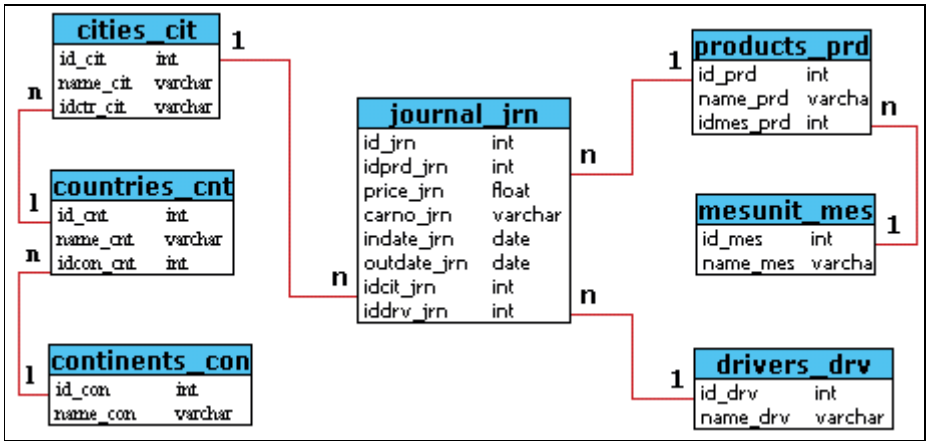


*Figure 1 Visual representation of the "MxWidgets" database using QuB*

The *continents_con* table contains two fields: ***id_con*** - the primary key and ***name_con*** – the name of the continent

The *countries_cnt* table contains three fields: ***id_cnt*** - the primary key, ***name_cnt –*** the country name, and ***idcon_cnt*** - that will act as a foreign key to (*continents_con*, ***id_con)***. This way, we have defined an "one-to-many" relation between these two tables (*continents_con* and *countries_cnt*).

The *cities_cit* table contains three fields: ***id_cit*** - the primary key, ***name_cit –*** the city name, and ***idctr_cit*** - that will act as a foreign key to (*country_cnt*, ***id_cnt***) defining an "one-to-many" relation. This way, we have defined an "one-to-many" relation between these two tables. (*cities_cit* and *countries_cnt*)

The *mesunit_mes* table contains two fields: ***id_mes*** - the primary key and ***name_mes***.

The *drivers_drv* table contains three fields: ***id_drv*** - the primary key and ***name_drv***.

The *products_prd* table contains three fields: ***id_prd*** - the primary key, ***name_prd***, and ***idmes_prd*** - that will act as a foreign key to ***id_mes***. This way, we have defined an "one-to-many" relation between the *mesunit_mes* and *products_prd* tables.

The *journal_jrn* table contains seven fields: ***id_jrn*** - the primary key, ***idprd_cit*** and ***idcit_jrn*** - that will act as a foreign key to ***id_prd*** and ***id_cit*** respectively, ***price_jrn*** – the price of the product, ***carno_jrn*** – the delivery car number, ***indate_jrn*** – the product entrance date, ***outdate_jrn*** – the product exit date.

## Creating the database

To create the database you can either use a database administration tool or you can create it in the command shell. For your convenience please name the database "**`MxWidgets`**" as we have consistently used this name throughout this document. Once the database is created, you should run the "`mxwidgets.sql`" script that will automatically create the tables described above and add some sample data.

The "`mxwidgets.sql`" script used to generate the database tables for the MySQL database server is included in the MX Widgets distribution package in a zip archive.

## *Creating a Dreamweaver MX Site*

The site that we'll use for this tutorial is a sample site that will have two pages named "*index.php*" and "*detail.php*". But, before creating the pages, we have first to properly create and configure a Dreamweaver MX site. Follow the steps below to learn how to use the **Advanced** Dreamweaver MX site configuration tool.

Open the Macromedia Dreamweaver MX and, by using the "**New Site**" option from the "**Site**" menu create a new site named "MXWidgets". Configure your site following the next three steps.

Note that, on our site samples we have used MXWidgets site name for the one created using the ADODB server model and MXWidgets_MySQL for the one created using the PHP_MySQL server model. On this document we'll use only the MXWidgets site name.

### Step 1 : Configuring the Local Information Section

The information requested in the Local info section is about the local configuration setting, that you will use during the development process: local root folder (as you can see, we used the "**C :\work\MXWidgets**" folder) and the URL of the actual site http :// address (in our case, the HTTP address will be http://localhost/MXWidgets/ (this is a local URL, so don't try to load it in your browser unless you have a local Apache configuration)).

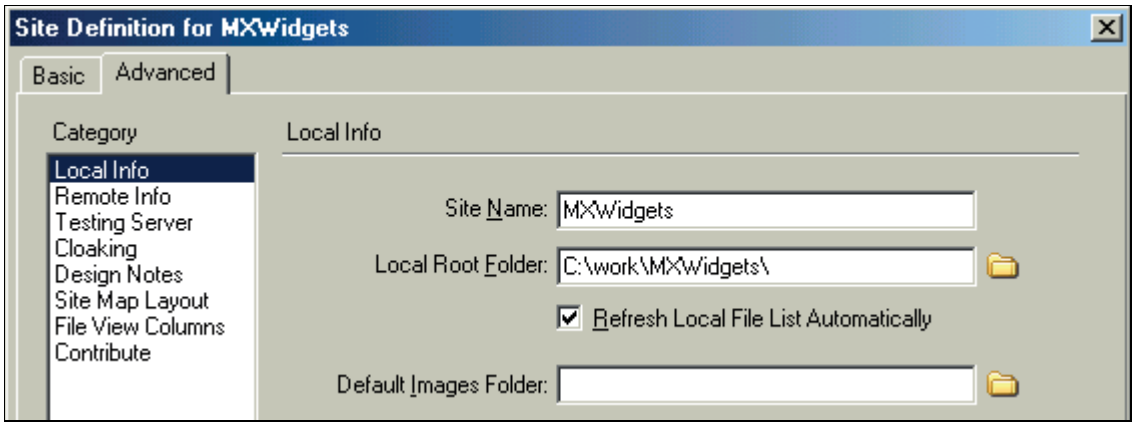


*Figure 2 Configuring the Local Info Section*

### Step 2 : Configuring the Remote Information Section

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

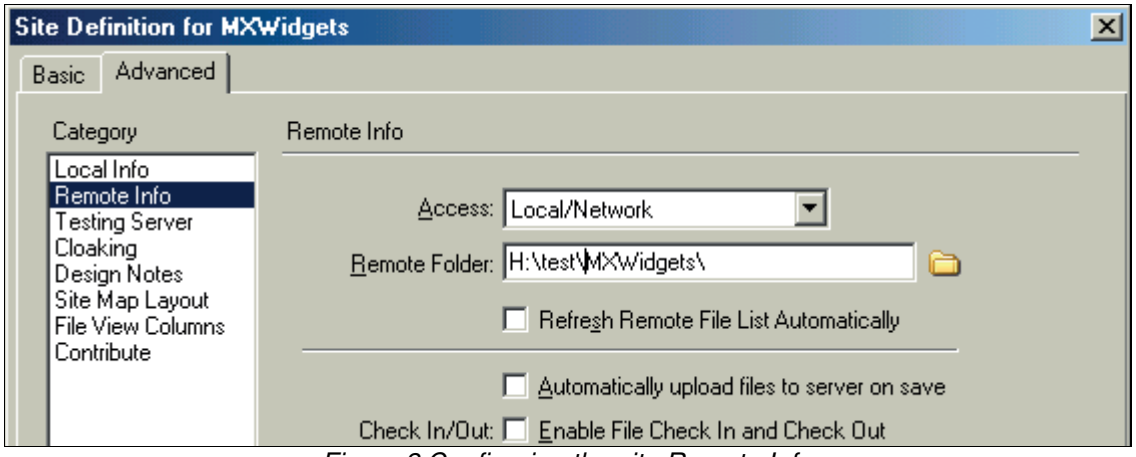


*Figure 3 Configuring the site Remote Info*

## Step 3 : Configuring the Testing Server Section

The Testing Server section refers to the type of connection and protocol used to connect to a test server. PHAkt2 users will have to select PHP_ADODB while Dreamweaver PHP_MySQL users will select PHP_MySQL as server model. The HTTP address of the site and the path of the remote site root folder are also required :

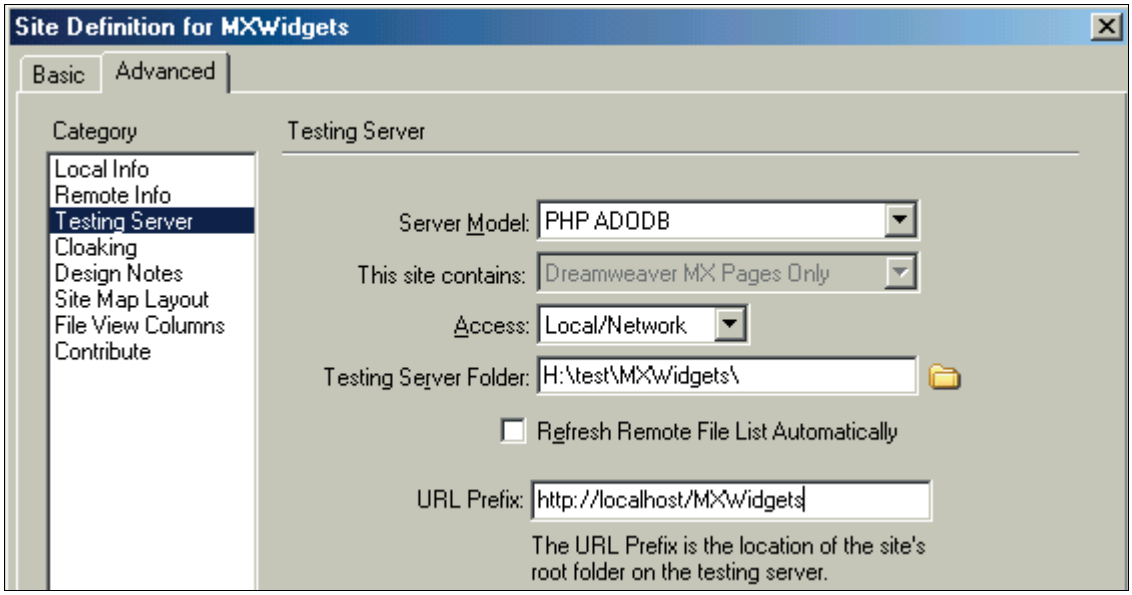


*Figure 4 Configuring Testing Server*

# Using the MX Widgets Suite controls

The goal of this tutorial is to design a simple web application in order to show how easy is to implement complex HTML form controls using the MX Widgets Suite. As we have explained above, the site will contain two pages: `index.php` and `detail.php`. The first page is used to list records from the database, while the `detail.php` page will be used to insert and update database records.

## *Creating the record list*

In the site root folder create the files "`index.php`" and "`detail.php`" by selecting the site name, pressing the mouse right button and then selecting the "**New File**" option.

Once finished, open the `index.php` file and create a database connection that will allow us to view, insert and update records. Go to "**Databases**" tab in the "**Application**" panel, press the "+" button and select "**ADODB Connection**".

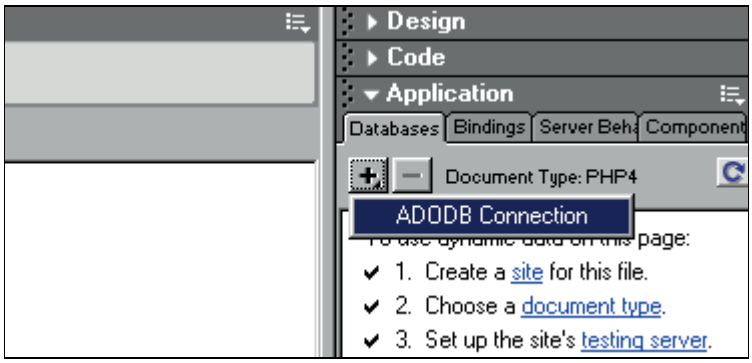


*Figure 5 Creating an ADOdb connection*

Configure the connection like in the below image and press the "**OK**" button:

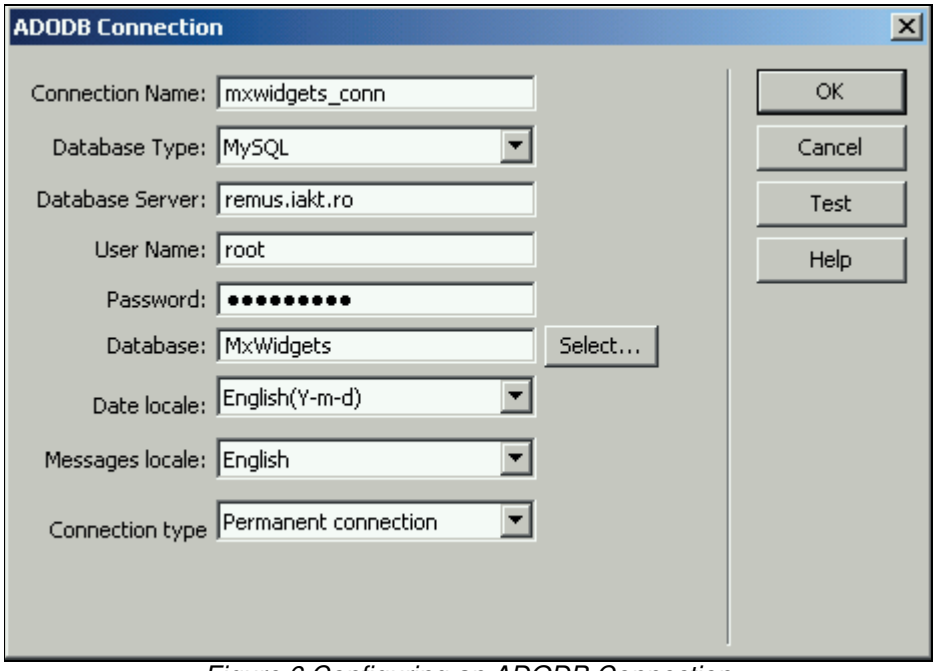


*Figure 6 Configuring an ADODB Connection*

When you are using the MySQL server model, the "***Database Type***", "***Date Locale***", "***Messages Locale***" and "***Connection Type***" fields are not included in the interface. Also the "***Database Server***" field is named "***MySQL Server***".

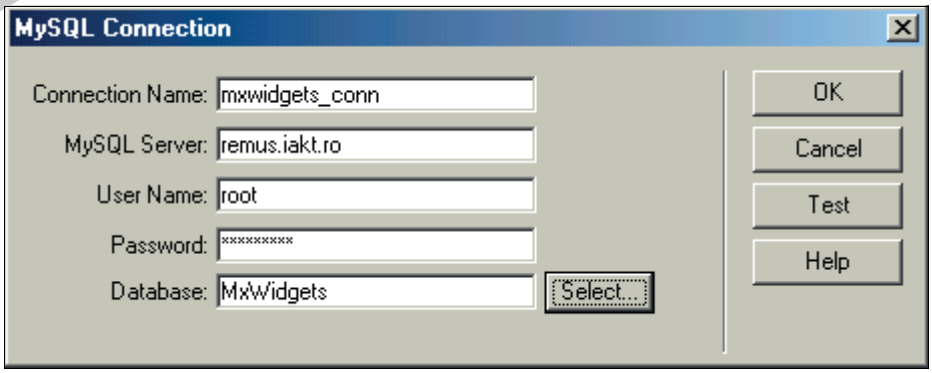


*Figure 7 Configuring a MySQL connection*

The *index.php* page will display a list of products with the related data. This page will have two links to the *detail.php* page: **New** that will allow the insertion of a new recordset, and **Detail** to update an existent record.

Before creating the record list we'll create a recordset that will be used to extract the information from the database. This data will later be displayed on page using a dynamic table.

To create a recordset, follow the steps described below:

- first create a recordset: go to the **Bindings** tab in the **Application** panel, press the "+" button and select Recordset **(Query)** option; in the displayed interface press the "**Advanced...**" button to switch to the advanced recordset configuration mode;
- name the recordset *rsJournals*
- in order to have a table that will display the product name as well as the city name (fields that are not contained in the *journal_jrn* table, but as referred using foreign keys), we should join the following tables : *journal_jrn* with *products_prd* and *cities_cit*. So, the SQL will be:

```
"SELECT journal_jrn.*, products_prd.name_prd, cities_cit.name_cit FROM journal_jrn INNER JOIN products_prd ON idprd_jrn = id_prd INNER JOIN cities_cit ON idcit_jrn = id_cit"
```

- the recordset configuration window should look like in the following image:

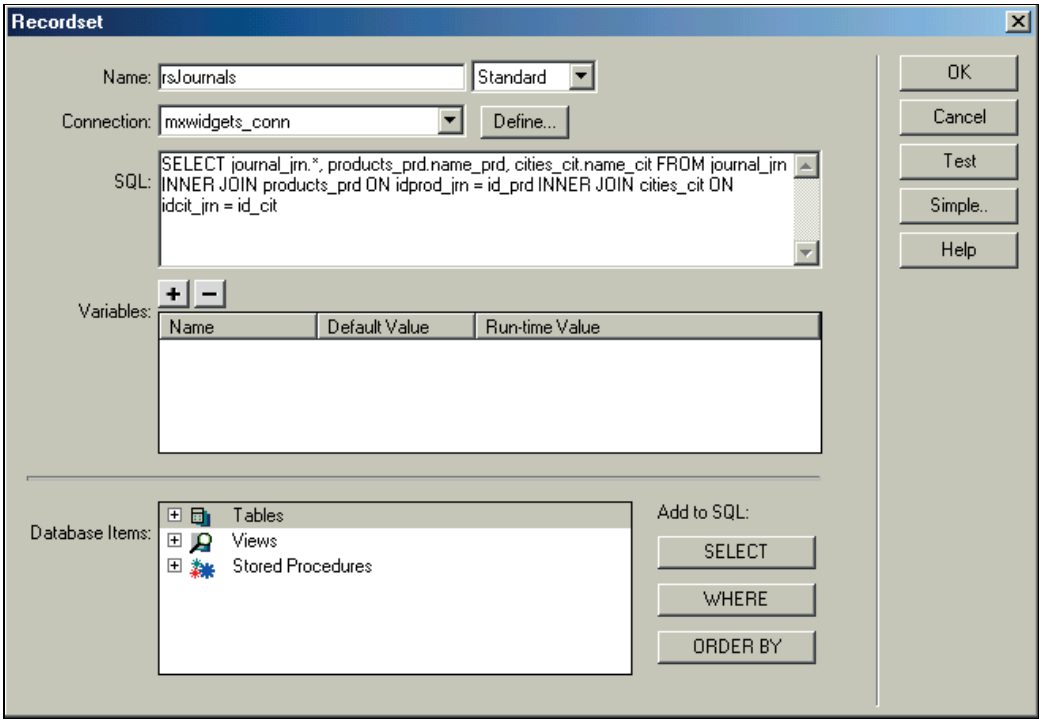


*Figure 8 Configuring the rsJournals Recordset*