

Overcoming ‘The Fear Factor’: Creating a Dynamic Web Site

DIANA FARMER¹ and YONGLI ZHOU²

¹ Hale Library, Kansas State University, Manhattan, Kansas, USA

² Morgan Library, Colorado State University, Fort Collins, Colorado, USA

In this information-rich age, agriculture related information is abundant and updated quickly. Collaborations between agriculture specialists and information specialists are increasing. The question has become how can agricultural information be presented in a more collaborative and timely manner?

This paper documents the process that Kansas State Libraries (KSL) followed to build a dynamic Agriculture Network Information Center (AgNIC) Web site. It discusses how and why librarians collaborated with research and teaching faculty and other agriculture specialists, examines the pros and cons of building dynamic vs. static Web sites, and documents the steps to create a basic dynamic Web site without extensive programming skills.

KEYTERMS AgNIC, agriculture information, collaboration, dynamic Web site, grain milling and processing, Kansas State University

Published as: Overcoming the Fear Factor: Creating a Dynamic Web Site", Journal of Agricultural & Food Information, vol. 10, no. 3, 2009, pp. 198-229.

Received 23 January 2009; accepted 23 March 2009.

All URLs last accessed on 20 September 2008.

Address correspondence to Diana Farmer, Hale Library, Kansas State University, Manhattan, Kansas 66506-1200, USA. E-mail: dmfarmer@ksu.edu.

BACKGROUND

The curriculum of Kansas State (K-State) University's Department of Grain Science and Industry is unique in the United States. K-State offers undergraduate and graduate degrees in Bakery Science and Management, Feed Science and Management, and Milling Science and Management. The program also offers training and support in bio-processing and bio-materials development from grain production. Additionally, there is the International Grains Program (<http://www.k-state.edu/igp/index.htm>), which is designed to educate foreign business leaders and government officials about U.S. grains and oilseeds through technical-training and assistance programs in storage and handling, milling, marketing and processing. In 2005, as part of its commitment to the Agriculture Network Information Center Alliance (AgNIC), KSL began to build the AgNIC site, Grain Milling and Processing (<http://www.grainmilling.org/>). (According to the website (<http://www.agnic.org/about/facts/faq.pdf>), AgNIC is a “voluntary alliance and partnership of nearly 60 member institutions and organizations working to offer quick and reliable access to quality agricultural information and sources”.) The Grain Milling and Processing site provides electronic access to selected, high-quality resources related to cereal grains and their use in baking, cereals, snacks, animal feeds, and industrial applications, as well as grain milling and processing.

The initial design of the site called for static pages. It became apparent in 2006 that maintaining the pages (e.g., updating links, maintaining and updating page formats) was extremely labor intensive and inefficient. As an experiment, K-State science librarians chose to design and build a simple dynamic Web site, using built-in functionalities offered by software available within the KSL. The goals of the redesign were to reduce site maintenance efforts, to offer a collaborative environment, to utilize affordable technologies, and to build a site with

minimal programming skills. The experience gained has been applied to other Web projects such as Kansas Wildflowers and Grasses (<http://www.kswildflower.org/>) and Beef Cattle (<http://www.agnicbeef.org/>).

Maintaining the Grain Milling and Processing site is a cooperative and on-going effort. New content is continuously added and previous content is reviewed on a regular basis. Three K-State science librarians participate in maintaining the site. Two librarians do the content selection and maintenance, while the third provides the technical expertise to update the style sheets, the database design, the site design and other programming functions. Reference questions are routed to only one librarian. If the question cannot be answered by the librarian, it is routed to K-State Grain Science professors.

This paper documents the collaboration between K-State science librarians and Grain Science and Industry Department faculty and the decisions made on Web structure, content and selected technologies. Finally, it provides step-by-step instructions for the software packages used to create this dynamic Web site.

PREPARATION PHASE

KSL science librarians first consulted faculty and students in the Department of Grain Science and Industry on the design and content of the proposed AgNIC site in 2005. The initial and subsequent consultations resulted in the selection of five subject areas for the site—milling, baking, cereal foods, feeds, and industrial uses. The meetings also resulted in valuable information regarding content, design, and organization of the site (standardization of the categories and subcategories that were eventually used), the identification of professors from the Department as the specialists to whom questions could be referred, etc. Meetings with the Grain Science and Industry faculty continue to be held as necessary.

CHOOSING SITE FORMAT: STATIC OR DYNAMIC?

The decision to have static or dynamic pages was addressed first. The science librarians identified the advantages and disadvantages of both, which influenced the final decision:

The advantages of a static site:

- Shorter initial time investment.
- Many agricultural information specialists and librarians have basic skills in Web authoring tools such as Adobe® Dreamweaver® and Microsoft Frontpage.

The disadvantages of a static site:

- Need to obtain and install a Web editor for each librarian working on the pages. This involved requesting at least two additional copies of Adobe® Dreamweaver® or three copies of another software package.
- Need for additional training—either more advanced training in Adobe® Dreamweaver® or both basic and advanced training for any other Web editor package.
- Increased errors due to multiple authors—increased possibility that an error on one page will not be corrected on another page in a timely fashion.
- Increased time adding content. Given the relationships among the basic subjects, it is possible that a Web resource could be entered on multiple pages. Each page on which a resource is listed would need to be updated to reflect any changes in the entry for that resource.
- Lack of easy portability. Content on static pages cannot be automatically transferred to a database.
- Changes are not reflected immediately. The revised page must be uploaded after corrections are made and saved.

The advantages of a database-based dynamic site:

- Less time and effort creating and updating content. Adding new content and updating content can be done via Web forms, so editing individual Web page(s) is unnecessary.
- Only one copy of a Web editor is needed to update the site's structure, interface, images and menu.
- No need for extensive training on Web authoring tools for all staff working on the project. Web forms are used to add and update content.
- Newly-added or updated content displays instantly.
- Data can be transferred to different software, for example from a MySQL database to a Microsoft Access database.

The disadvantages of a database-based dynamic site:

- Longer lead-time.
- Requires basic knowledge of database design, SQL and Web programming.

After much discussion and reflection, the team decided to build a dynamic site with a relational database. The advantages of a dynamic Web site better fulfill the needs and expectations of both the science librarians and the faculty of the Department of Grain Science and Industry.

DESIGN OF THE DATABASE

Initially, Access was chosen as the software for the database as it was already available at KSL and at least one librarian was familiar with its use. The redesign process began by looking at the information to be presented—its purpose, user expectations for content, expectations for future information, and the overall informational and operational needs of the site. Redundant or duplicate data or the absence of required data needed to be avoided in the final product.

The review resulted in five tables for storing data: Content, Profiles, Subjects, Categories, and Subcategories. The tables collect similar types of data into “groups” with similar uses and characteristics. Other specifications detail how a table is used in conjunction with another table. The site map diagram in Figure 1 illustrates this usage. (Compare Figure 1 with Figure 2, the initial site map, to see the need for the review and revision of the site mapping.) To provide consistency, the Subjects, Categories, and Subcategories tables are restricted to specific data only. Only the system administrator may add or delete data in these tables. The tables control which page(s) display a selected Web site/information entry:

1. Subjects Table. This table is limited to the five subjects selected in consultation with faculty in the Grain Science and Industry Department: Milling, Baking, Cereal Foods, Feeds, and Industrial Uses.
2. Categories Table. This table was initially limited to 6 subdivisions for entries on a given subject page. The categories identified are Associations & Organizations, Commercial Sites, Education & Careers, Marketing, Patent Resources, and Research & Statistics. The first five categories use the first three subcategories (United States, Other National, and International) enumerated below, leaving the category Research & Statistics as the only category to be subdivided by the remaining subcategories.
3. Subcategories Table. This table initially provided a further 10 subdivisions. An 11th subdivision (History) was added in 2007. Currently, the subcategories used are United States, Other National, International, Magazines/Newsletters, Full Text Resources, Print Resources, Statistics, Databases, Research Projects, Research Laboratories, and History.

4. Content Table. This table is the heart of the AgNIC Grain Milling and Processing Web site and contains one entry for each resource displayed on the various pages of the site. There are 12 fields in this table, based on selected Dublin Core elements: Content ID, Title, Creator, Description, Publisher, Date, Resource Type, Resource Identifier, Language, Coverage, Format, and Resource Size. Only 3 of these fields are required for the information to display on the public view of the site: Content ID, Title, and Description. Instructions on what and how information should be entered in each of the 12 fields is recorded in the use manual. For example, the Language and Coverage fields use the codes in the MARC Code List for Languages (<http://www.loc.gov/marc/languages/>) and the MARC Code List for Countries (http://www.loc.gov/marc/countries/cou_home.html#top), respectively. Since the data in these fields is comma-delimited, if the resource is available in more than one language or the coverage is for more than one geographic area, the codes are separated by a comma in the appropriate field.
5. Profiles Table. This table controls the display(s) of each item in the Content Table. Each item in the Content Table may have multiple profiles, one profile for each page on which it displays. Each profile consists of a unique combination of Content, Subject, Category, and Subcategory. The table is limited to five fields: Profile ID, Content ID, Subject ID, Category ID, and Subcategory ID. Only the Profile ID field is unique to the database. The data in the remaining fields comes from the table corresponding to the name of the field.

CHOOSING TECHNOLOGIES

Initially, two sets of technologies were considered:

1. Microsoft Access and ASP
2. MySQL and PHP

Both sets include popular Web programming technologies. Adobe® Dreamweaver® creates basic ASP and PHP pages and could be used for either set. Set 1 required Microsoft Access, which is fairly complicated software. While available on all of the science librarians' computers, some of the librarians needed additional training on its use. The software for set 2 was all open source. To create a MySQL database, phpMyAdmin (an open source software) can be used. The phpMyAdmin software allows users to create, manage, and update a MySQL database via a Web interface. At the time of the decision, none of the software for set 2 was available on the science librarians' computers.

Initially, the test database was implemented using the components of set 1. This test database was housed on a local computer that did not have the capacity to host the live site. After consulting with the University's computing support staff, it was discovered that MySQL, PHP, Access, and ASP were not supported on campus. After discussing the available options, the decision was made to move to MySQL and PHP. These technologies fit the smaller scale of the Grain Science and Milling Web site.

The decision to go with commercial hosting was also made at this time. The costs of commercial hosting were fairly low and included tech support. At the time, the cost was only \$5/month, including 24/7 tech support.

The following tools were used to create the current site:

1. Web authoring: Adobe® Dreamweaver®

2. Database management: phpMyAdmin (<http://phpmyadmin.net/>, open source software)
3. Graphic design: Adobe® Photoshop®

At KSL, a local computer was set up as a test server. All dynamic pages are created and tested locally before being uploaded to the production server.

CONTENT DESIGN

User View

The Grain Milling and Processing site contains Web links to associations and organizations, research publications, statistics, educational and career opportunities, government agencies, and relevant international resources. As seen in Figures 3 and 4, each entry contains:

1. A site title which is linked to an actual site
2. A short description
3. The file format and file size (if available) for full text documents.

Staff View

To allow librarians to manage content by Web forms instead of working directly with phpMyAdmin, a staff interface (shown in Figure 5) was created. The staff interface contains brief and detailed table views. It also includes forms that allow content creators to create and update the entries in the Content and Profiles tables.

Each resource has an entry in the Content table that contains a resource's title, URL, description, etc. Since a resource can belong to one or multiple categories, it may have multiple profiles. For example, in Figure 6, ABA Labeling Manual displays only on the *Baking > Research & Statistics > Print Resources* page, so it has only one profile. ABNA: Associated British Nutrition & Agriproducts has two profiles; it displays on both the *Feeds > Commercial*

Sites > International page and the *Industrial Uses > Commercial Sites > International* page.

When content creators update a resource's information, they need to know which record or records need to be updated in both the Content and the Profile tables. The content and profile brief view lists all entries alphabetically by title. Content creators can then select an entry on the brief view page to access its matching detailed view page (see Figure 6 and Figure 7). The content and profile detailed view page has links to update, add, or delete profiles and records.

CREATING A DYNAMIC WEB SITE

The remainder of this paper documents the steps followed by K-State Libraries in creating the Grain Milling and Processing site using PHP and MySQL and will serve as a useful guide to others seeking to create a dynamic site.

I. Set up a Local Test

To test PHP/MySQL pages on a local computer, install the following software:

1. Apache HTTP: allows a local machine to be designated as a local Web server.
2. MySQL: establishes a local computer as a MySQL database server.
3. PHP: allows a local Web server to interpret PHP codes.
4. phpMyAdmin: manages a MySQL database.
5. Adobe® Dreamweaver®: creates Web pages.

The tutorial, "How to Install Apache, MySQL, PHP, and phpMyAdmin on a Windows PC," can be found at <http://www.wikihow.com/Install-Apache,-MySQL,-PHP,-and-phpMyAdmin-on-a-Windows-PC>.

Setting up a Mac computer as a Web server can be done by installing MAMP. MAMP is a package that comes with Apache, PHP, and MySQL. There are two versions, MAMP and

MAMP Pro. For testing purposes, MAMP is sufficient. More information can be found at <http://www.mamp.info>.

II. Create and Maintain a Database Using phpMyAdmin

At KSL, phpMyAdmin (http://www.phpmyadmin.net/home_page/index.php) was used to create the databases for all AgNIC projects. Written in PHP, phpMyAdmin is a tool to handle the administration of MySQL over the Web; it allows the creation and updating of a database by filling out Web forms. Most commercial Web hosting companies provide phpMyAdmin. When ready to move a database from your local test server to the actual production server, export it as a SQL file and then upload it to the production server via a phpMyAdmin utility. Currently, phpMyAdmin can create and drop databases, create/drop/alter tables, delete/edit/add fields, execute any SQL statement, manage keys on fields, manage privileges, and export data into various formats and is available in 55 languages. The software can be downloaded at http://www.phpmyadmin.net/home_page/downloads.php.

a. Create a Database

phpMyAdmin provides an easy-to-use Web interface for database creation and updating. Go to the phpMyAdmin home page, click *Databases*, click *Create new database*, enter a database name, and then click the *Create* button, as shown in Figure 8.

For security purposes, many commercial Web servers do not allow users to create a database directly under phpMyAdmin. Instead, the creation of a new database is handled via a Web administrator management interface, called the cPanel (Figure 9). cPanelR is a trademark of cPanel, Inc.

b. Create a Table

After the database is created, a new table can be generated in two steps:

1. Select the database and click *Create new table*. Name the table and specify the number of fields (Figure 10).

2. Specify the details for each field. Normally the first field is the primary key, a unique ID, with a data type of integer. When a field's data type is set to integer (INT) or variable character (VARCHAR), the Length/Values field must have a value. For example, the Title field in Figure 11 can have up to 256 characters including spaces. Any titles that are longer than 256 characters will be truncated.

After the *Save* button is clicked, phpMyAdmin generates a SQL query to build the table shown in Figure 12. Both query view and table view are available. Without phpMyAdmin, the SQL query needs to be hard-coded by a programmer.

c. Add Content to a Table

Click the *Insert* tab and the insert record form (Figure 13) appears. In most cases, Function fields can be left blank. When a value is added, its value type must correspond to the specified data type. For example, if the ID field is specified as integer, an alphabetic character in the ID is not acceptable.

While adding content to a table or tables via phpMyAdmin is simple, this practice is not recommended. Rights to access phpMyAdmin should be limited to the Web administrator. The administrator should create the database and tables first, so that others can create or update tables via Web forms. In this way, content contributors may not alter or delete tables accidentally. The process to create forms in order to update a database will be discussed in a later section.

d. Edit or Delete an Entry

After a record is entered, it can be edited or deleted via phpMyAdmin (Figure 14) only by the Web administrator, as indicated above.

e. Backup the Database

A database should be backed up on a regular basis so that, if the database on the production server is lost or corrupted, the most recent backup file can be used to restore information. phpMyAdmin provides different options for backup files. At KSL, databases are normally exported to a SQL file (a plain-text file with a file extension of .sql). Figure 15 illustrates the procedure for creating a backup database:

1. Select the database to back up
2. Click the *Export* tab
3. Check SQL under *Export* section and check the *Save as file* box
4. Leave the other options with their default settings
5. Click the *Go* button

III. Create Dynamic Web Pages Using Adobe® Dreamweaver®

Adobe® Dreamweaver MX® was used at KSL to create the original Grain Milling and Processing site in 2005. However, Adobe® Dreamweaver CS3® was used for the examples in this paper, as that is the current version and what is available on the author's computer. Most of the features discussed in this paper are available in other versions of Adobe® Dreamweaver®.

a. Work Flow in Building a Dynamic Web Site

There are five steps to follow in creating a dynamic page:

1. Define a site in Adobe® Dreamweaver®

2. Create templates for the site
 3. Create a new PHP page based on a template
 4. Create a database connection
 5. Retrieve and display records
 - a. Create a Recordset (query)
 - b. Create a place holder for dynamic text
 - c. Display dynamic text for one record at a time
 - d. Add repeat region to display all records associated with a page
- b. Define the Site with Local, Remote and Test Servers in Adobe® Dreamweaver®

Before a dynamic site can be created and tested locally, three servers need to be created in Adobe® Dreamweaver®. A more detailed tutorial, “Installing a Local Web Server,” can be found at the Adobe Help Resource Center

(http://livedocs.adobe.com/en_US/Dreamweaver/9.0/help.html?content=WS28E9562B-D031-4eb0-8011-36C04AFEEBF7.html).

1. Local Server

To define a local server, go to *Site > New Site > Advanced > Local Info*. Only the site name and local root folder fields are required. For the Grain Milling site, the root folder contains three HTML pages and nine subdirectories. Five of these subdirectories correspond to the five subjects: milling, baking, cereal food, feeds, and industrial uses (Figure 16). There are two Adobe® Dreamweaver® generated subdirectories, Connections and Templates. The Connections folder holds PHP files that create connections to the SQL database, while the Templates folder

holds Adobe® Dreamweaver® template files. The images folder is the default folder that holds all images used for the site.

2. Remote Server

From *Site > New Site > Advanced > Local Info*, in the *Category* options on the left side of the page, select the *Remote Info* tab. The display will change. Select *FTP* as the type of access for the Access field. The display will change to present the fields for FTP access. In the *FTP host* field, type the URL for the site.

Complete the field by providing the address and name of the Host directory.

Finish by adding the login and password in the appropriate fields. The other fields can be left with their default values.

3. Test Server

After going to *Site > New Site > Advanced > Local Info*, select *Testing Server* from the *Category* options on the left side of the page. The display will change. Choose *PHP MySQL* for the *Server model* field and *Local/Network* for the *Access* field. Click the yellow folder icon to find the root folder for the test server. Next complete the *URL prefix* field and click the *OK* button at the bottom of the window. Figure 17 displays these steps.

A local test server normally has a prefix of `http://localhost/` or `http://localhost:[aPortNumber]/`. If the *URL prefix* field is set to an incorrect location, a database connection cannot be set up in Adobe® Dreamweaver®. For example, all of K-State Libraries' AgNIC sites were placed in the `C:\www\` directory on the test server. Under the `C:\www\` directory, there is one folder named GM. All Grain Milling and

Processing test files are in the GM folder; therefore, the URL prefix for the test server for this site is `http://localhost/GM/`.

IV. Pre-requisites for Creating PHP/MySQL Pages

The following sections use examples to demonstrate how to create PHP/MySQL pages.

Requirements to test these examples are:

1. Installed and tested the Apache Server
2. Installed and tested the MySQL server
3. Installed the PHP engine
4. Created a database and tables with sample data
5. Established local, remote, and test servers

a. Create a Database Connection

A database connection allows a PHP page to “talk” to a database. To create a database connection in Adobe® Dreamweaver®, as shown in Figure 18, select *Window > Databases* from the menu bar (Note: If the *Database* tab is grayed out, your computer has not met the pre-requisites.). The *Application* panel will display. Select the *Databases* tab. Click the “+” sign and select *MySQL Connection*.

All the fields in the *MySQL Connection* dialog (Figure 19) are required fields. Give the connection a name (spaces are not allowed in the name). Use “localhost” for the *MySQL server* field if a local computer is used as the test server. Click the *Select* button to select an existing database (Note: If an error occurs at the *Select a database* step, check the test server’s URL prefix and make sure it is set to the correct path.). Click the *OK* button. The

newly created connection name should display under the *Databases* tab, as shown in Figure 20.

By default, Adobe® Dreamweaver® creates a folder named *Connections* and places “myConnection.php” in it. When uploading the site to its production server, make sure to upload the entire *Connections* folder. Also remember that the log in and password in this file must be changed to match the production server login information. Below is the source code for myConnection.php on the test server:

```
<?php
$hostname_myConnection = "localhost";
$database_myConnection = "grainmilling";
$username_myConnection = "localAdminLogin";
$password_myConnection GM = "localAdminPassword";
$myConnection = mysql_connect($hostname_myConnection,
$username_myConnection, $password_myConnection) or
trigger_error(mysql_error(),E_USER_ERROR); ?>
```

The production server copy of myConnection.php should have the database name and login information for that server. Below is the source code for the production server copy:

```
<?php
$hostname_myConnection = "localhost";
$database_myConnection = "grainmilling";
$username_myConnection = "serverLogin";
$password_myConnection = "serverPassword";
$myConnection = mysql_connect($hostname_myConnection,
$username_myConnection, $password_myConnection) or
trigger_error(mysql_error(),E_USER_ERROR); ?>
```

b. Display Dynamic Content

1. Create a blank PHP page or create a PHP page based on a template.
2. Create a query (called a Recordset in Adobe® Dreamweaver®).
 - a. Click the *Bindings* tab on the *Application* panel and the *Recordset* dialog box pops up.
 - b. Give the Recordset a name (do not use spaces in the name).

- c. In the *Connection* field, select a connection from the drop-down menu. For the example in this paper, *myConnection* (which was created in the previous step) is selected.
- d. In the *Table* field, select a table which contains the information to be displayed. By default, Adobe® Dreamweaver CS3® displays the Simple Recordset which allows you to query only one table at a time. If more than one table needs to be queried, click the *Advanced* button to manually create the query. Figure 21 is a sample SQL query from the Grain Milling and Processing database. The *Advanced query* panel was used to create this query that pulls information from two tables, *Content* and *Profiles*:

```
SELECT *
FROM content, profiles
WHERE profiles.content_id=content.content_id AND profiles.subject_id LIKE
'2'
AND profiles.category_id LIKE '1'
AND profiles.subcategory_id LIKE '1'
ORDER BY content.title
```

Parsing the above query:

- **SELECT * FROM content, profiles:** This tells the system to select all data fields from the tables, *Content* and *Profiles*.
- **WHERE profiles.content_id=content.content_id:** This statement requires the system to match the content IDs from the *Content* table with the content IDs in the *Profiles* table
- **AND profiles.subject_id LIKE '2':** This statement tells the system which subject ID from the *Profiles* table to use. In this instance, Subject ID 2, Baking, is used.

- **AND profiles.category_id LIKE '1'**: This tells the system which category ID from the *Profiles* table to use. In this instance Category ID 1 is Associations and Organizations.
- **AND profiles.subcategory_id LIKE '1'**: This statement tells the system which subcategory ID from the *Profiles* table to use. In this instance Subcategory ID 1 is United States.
- **ORDER BY content.title**: This statement instructs the system to display the returned text in alphabetic order by the title field of the content table.

Click the *Bindings* tab in the *Application* panel to view the returned dynamic query (Figure 21). Adobe® Dreamweaver® automatically inserts some PHP and MySQL code into the HTML source code (See Appendix for details). This code does not display when users view the source code using a Web browser. The code is interpreted by a Web server before it is displayed to users.

3. Display Dynamic Text

A SQL query retrieves data from a database. PHP is used to display the retrieved information on a Web page. At the time a PHP page is created, it does not know how many entries will be returned by the query. In order to display an unknown number of entries, the returned entry is placed into a repeat region. The repeat region contains programming code (called a loop) that will run repeatedly until the last entry is retrieved.

In Adobe® Dreamweaver® the simplest way to create dynamic content is to use a Dynamic Table (Figure 22). Since a machine-generated dynamic table is generally difficult to format and KSL's steps pre-date the release of Adobe® Dreamweaver CS3®, KSL developed (and offer below) steps that some may find easier to use than the

instructions in the Adobe® Dreamweaver CS3® Help Manual under “Create a Dynamic Table.”

A customized table is created first and then dynamic text is added to it. Each customized table contains one returned record. For the Grain Milling and Processing site, only an entry’s title, brief description, format and resource size fields (if available) are retrieved and displayed. The following instructions were used to create the *Baking > Associations and Organizations* page.

- Create a new PHP page based on the baking template.
- Create a Recordset (see the previous section) to retrieve records profiled for *Baking > Associations and Organizations*.
- Insert a blank table into the Content editable region.
- Highlight the entire table and add a repeat region (Go to *Insert > Data Objects > Repeat Region > select a Recordset > All records*).
- Drag and drop dynamic text from the *Bindings* panel to the table which is enclosed in a repeat region (see Figure 22).
- Save the page and preview it in a Web browser. If the page is previewed on a local test server, the page’s URL normally is [http://localhost\(:portNumber\)/yourSite/newPage.php](http://localhost(:portNumber)/yourSite/newPage.php). The URL for the foregoing example is <http://localhost/GM/baking/test.php>.

4. Add a URL to a Title

PHP code is used to attach a retrieved URL to its corresponding title when it is necessary or desirable to hyperlink text. For example, on the Grain Milling and Processing pages, a

title is also a hyperlink. In the Content Table, the title is stored in the title field and the URL is stored in the resource_identifier field:

```
$aLink = $row_bakingAssociations['Resource_Identifier'];  
$aTitle = $row_bakingAssociations['Title'];  
printf (<a href=\"%s\">%s</a>\", $aLink, $aURL);
```

\$aLink = \$row_bakingAssociations['Resource_Identifier'] This code retrieves a URL from the content table;

\$aTitle = \$row_bakingAssociations['Title'] This code retrieves the corresponding title field;

printf (%s\", \$aLink, \$aTitle) This instructs the system to replace the first %s with \$aLink, the URL; and to replace the second %s with \$aTitle, the title of the entry.

The following example shows the source code generated by PHP for the U.S. Wheat Association:

```
$aLink = "http://www.uswheat.com";  
$aTitle = "U.S. Wheat Association";  
printf (<a href=\"%s\">%s</a>\", "http://www.uswheat.com\", "U.S. Wheat  
Association");
```

When the page is displayed on a browser, the PHP code will display only as HTML:

```
<a href="http://www.uswheat.org/">U.S. Wheat Associates</a>
```

c. Create a Staff Web Interface

1. Create a Data Insertion Form

For the Grain Milling and Processing site, Web forms were created to allow content contributors to add data. Adobe® Dreamweaver® has built-in functions that help you to create such forms. Figure 23 is a screenshot of the form used to create new content in the Content Table.

A detailed tutorial on how to create an Insert form can be found in Adobe® Dreamweaver CS3® Help Manual under the section, “Build the Insert Page in One Operation.” For the reasons stated earlier, the steps KSL followed to create the “Create New Content” form for the Grain Milling and Processing site are given below:

- a. Create a PHP page and save it in the test GM directory.
- b. Select *Insert > Data Objects > Insert Record > Record Insertion Form Wizard*. The dialog box shown in Figure 24 appears.
- c. Select a connection. For small-scale sites, only one database is usually queried, so you only need to create one database connection. For example, *myConnection* is shared with all other Grain Milling and Processing PHP pages.
- d. Select the name of the table into which you will insert new data.
- e. Specify a redirect page once the data entry form is complete (This will tell the system which page to display after the form is entered.).
- f. Add requirements for fields as needed. For example, specify that the field, *Content_id*, will be an automatically-generated number.
- g. Click the *OK* button when you have added all the necessary field requirements.

Figure 25 shows the default form for adding new content. The left column contains labels that are static text which can be formatted as HTML text.

2. Create a Form to Update Records

Update forms can also be created using Adobe® Dreamweaver® built-in functions. The instructions below were used to create a form to update the *Content* table for the Grain Milling and Processing site.

- a. Create a blank PHP page and save it under the test site.
- b. Create a Recordset by opening the *Application* panel and clicking the *Bindings* tab (Note: you must have finished the first three steps as indicated in Figure 26 before you can create a Recordset.). Next, click the *plus (+)* button to choose a Recordset. Give the Recordset a name; choose a Connection from the *Connection* drop-down menu; select a table to be updated from the *Table* drop-down menu; indicate if you want to retrieve all columns or selected columns from the *Columns* option. Use a filter to associate this form with a specific field in the record to be updated. Click the *Filter* dropdown menu and select *content_id* and the “=” sign (The *content_id* is the primary key of the Content table. The primary key is unique and that makes it a perfect candidate for a filter.). Next set the *URL Parameter* to *content_id*. The *Sort* option is left blank because only one record is displayed and updated at a time in this example. The primary key is used as the filter to link the *Update Content* button (see Figure 7) to the Update form. When the *Update Content* button is clicked, a content ID is passed to the Update form and the form will retrieve and display only the matching record. Figure 27 illustrates the process for creating a Recordset for an Update form.

c. Create an Update Form. Go to *Insert > Data Objects > Update Record > Record Update Form Wizard*. In the *Record Update Form* dialog (Figure 28), choose *myConnection* from the *Connection* drop-down menu; select the *Content* table from the *Table* drop-down menu; choose *updateContentTable* from the *Select record from* menu; select the query name just created in the preceding step. Now specify which page is to display once the update is complete. Leave the default settings for all other options.

d. Format Update Form. The Update form (Figure 29) has two columns. The left column contains labels and the right column contains dynamic text. Both the static labels and the dynamic text can be formatted.

e. Create a Link Button. The above steps have created an Update form, which accepts a content ID and then retrieves information accordingly. The next step is to create a link, which is on another page, to link to and pass an ID to the Update form. In Figure 7, the *Update Content* and *Update Profile* buttons serve this purpose. The source code for the *Update Content* button is displayed in Figure 30.

First create an image as a hyperlink, using the code in Figure 31. Then attach a dynamically-generated content ID to the link (Figure 30). The line “`updateContent.php?content_id=`” tells the system to link to the *updateContent.php* page and insert the Content ID of the record to be updated. The code after the “`=`” tells the system to print out a dynamically-generated Content ID. `<?php ?>` is the syntax for imbedding PHP code into HTML. The “`echo`” is PHP syntax for displaying text. The code “`$row_conent_profiles['content_id']`” is a dynamically-generated ID.

The Update and Insert New forms are created to allow easy Web access. For the Grain Milling and Processing site, these forms are placed in a password-protected directory.

Multiple authors can log in and update the site simultaneously. This practice allows those working on the site to do so while maintaining independent work schedules.

CONCLUSION

Many agricultural information specialists and librarians do not have Web programming expertise and are new to dynamic Web programming. However, many of them have a basic knowledge of database structure and can build static Web sites using Web authoring tools such as Adobe® Dreamweaver®. There are many tools, including open source software, that can help build dynamic Web sites easily.

APPENDIX

PHP AND SQL QUERIES GENERATED FROM CREATING A DATABASE CONNECTION AND A RECORDSET

After a connection and a Recordset are defined, Adobe® Dreamweaver CS3® automatically inserts the following PHP code at the top of the page. If the Connection and the Recordset were defined correctly, the system knows what it has been instructed to do beginning with “if (!function_exists ..” through “... return \$theValue;”. You do not need to understand all the code; however, it is useful to understand the code from “mysql_select_db ...” to “...\$totalRows_bakingAssociations”.

```
<?php require_once('../Connections/myConnection.php'); ?>
<?php

if (!function_exists("GetSQLValueString")) {
function GetSQLValueString($theValue, $theType, $theDefinedValue = "",
$theNotDefinedValue = "") {
{
    $theValue = get_magic_quotes_gpc() ? stripslashes($theValue) :
$theValue;
    $theValue = function_exists("mysql_real_escape_string") ?
mysql_real_escape_string($theValue) : mysql_escape_string($theValue);

    switch ($theType) {
        case "text":
            $theValue = ($theValue != "") ? "'" . $theValue . "'" :
"NULL";
            break;

        case "long":
        case "int":
            $theValue = ($theValue != "") ? intval($theValue) :
"NULL";
            break;
        case "double":
            $theValue = ($theValue != "") ? "'" . doubleval($theValue)
. "'" : "NULL";
            break;
        case "date":
            $theValue = ($theValue != "") ? "'" . $theValue . "'" :
"NULL";
            break;
        case "defined":
            $theValue = ($theValue != "") ? $theDefinedValue :
$theNotDefinedValue;
    }
}
}

```

```

        break;□ }□

        return $theValue;□
    }□
}□□
mysql_select_db($database_myConnection, $myConnection);□

$query_bakingAssociations = "SELECT * FROM content, profiles WHERE
profiles.content_id=content.content_id AND profiles.subject_id LIKE '2' AND
profiles.category_id LIKE '1' AND profiles.subcategory_id LIKE '1' ORDER BY
content.title";□

$bakingAssociations = mysql_query($query_bakingAssociations, $myConnection)
or die(mysql_error());□
$row_bakingAssociations = mysql_fetch_assoc($bakingAssociations);□
$totalRows_bakingAssociations = mysql_num_rows($bakingAssociations);□

?>□

```

Explanatory Notes:

1. `<?php require_once('../Connections/myConnection.php'); ?>` When you create a Recordset, you need to choose a connection. For this step, Adobe® Dreamweaver® automatically creates a PHP code which links to a connection file which is in the Connections folder. The Connection file contains log-in information for a database. Without this line, the PHP page would not know which database to access.
2. `mysql_select_db($database_myConnection, $myConnection);`□ This sets the current active database on the server that's associated with the specified link identifier which is `$myConnection`. `$myConnection` is the MySQL connection.
3. `$query_bakingAssociations` holds the MySQL query.
4. `$bakingAssociations = mysql_query($query_bakingAssociations, $myConnection) or die(mysql_error());` `mysql_query()` sends the query, `query_bakingAssociation`, to the currently active database. The returned value, which is a resource, is assigned to `$bakingAssociations`. A resource is a special variable, holding a reference to an external resource. Resources are created and used by special functions.
5. `$row_bakingAssociations = mysql_fetch_assoc($bakingAssociations);` `mysql_fetch_assoc()` fetches one row of data from the database at a time and moves the internal data pointer ahead. Note, this only fetches one row of data, which is in an array. You need to use a loop to fetch all matching data.
6. `$totalRows_bakingAssociations = mysql_num_rows($bakingAssociations);`□ `mysql_num_rows()` retrieves the number of rows from a result set. `$totalRows_bakingAssociations` holds the total number of matching records returned by the query.

FIGURE 1 Grain Milling and Processing’s Current Site Map. Copyright K-State Libraries. Reproduced with permission.

FIGURE 2 Grain Milling and Processing’s Initial Site Map. Copyright K-State Libraries. Reproduced with permission.

FIGURE 3 Sample Site Entry. Copyright K-State Libraries. Reproduced with permission.

FIGURE 4 Sample Full Text Entry. Copyright K-State Libraries. Reproduced with permission.

FIGURE 5 Staff Interface. Copyright K-State Libraries. Reproduced with permission.

FIGURE 6 Content and Profile Brief View. Copyright K-State Libraries. Reproduced with permission.

FIGURE 7 Content and Profile Detailed View. Copyright K-State Libraries. Reproduced with permission.

FIGURE 8 Create a Database Under phpMyAdmin. Note: Spaces and special characters should be avoided in a database name. For example, use myDatabase or my_database as the database name, instead of “my database.” Reproduced with permission of phpMyAdmin.

FIGURE 9 Create a Database Using cPanel. Copyright cPanel, Inc. Reproduced with permission.

FIGURE 10 Create New Table Step 1. Reproduced with permission of phpMyAdmin

FIGURE 11 Create New Table Step 2. Reproduced with permission of phpMyAdmin

FIGURE 12 Created Table View. Reproduced with permission of phpMyAdmin

FIGURE 13 Insert a Record to a Table. Reproduced with permission of phpMyAdmin

FIGURE 14 Edit or Delete an Entry. Reproduced with permission of phpMyAdmin

FIGURE 15 Back Up a Database. Reproduced with permission of phpMyAdmin

FIGURE 16 Local Server.

FIGURE 17 Definition of a Test Server. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 18 Create a MySQL Connection. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 19 MySQL Connection Dialog. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 20 Connections on the Application Panel. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 21 Create a Query/Recordset Dialog. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 22 Adding Dynamic Text. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 23 Create New Content Form for the Grain Milling and Processing site.
Copyright K-State Libraries. Reproduced with permission.

FIGURE 24 Record Insertion Form Dialog. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 25 Insertion Form Preview. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 26 Bindings Panel. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 27 Create a Recordset for an Update Form. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 28 Record Update Form Dialog. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 29 Update Form Preview. Adobe product screen shot(s) reprinted with permission from Adobe Systems Incorporated.

FIGURE 30 Pass an ID to an Update Form.

FIGURE 31 Add a Link to an Image.

FIG 1

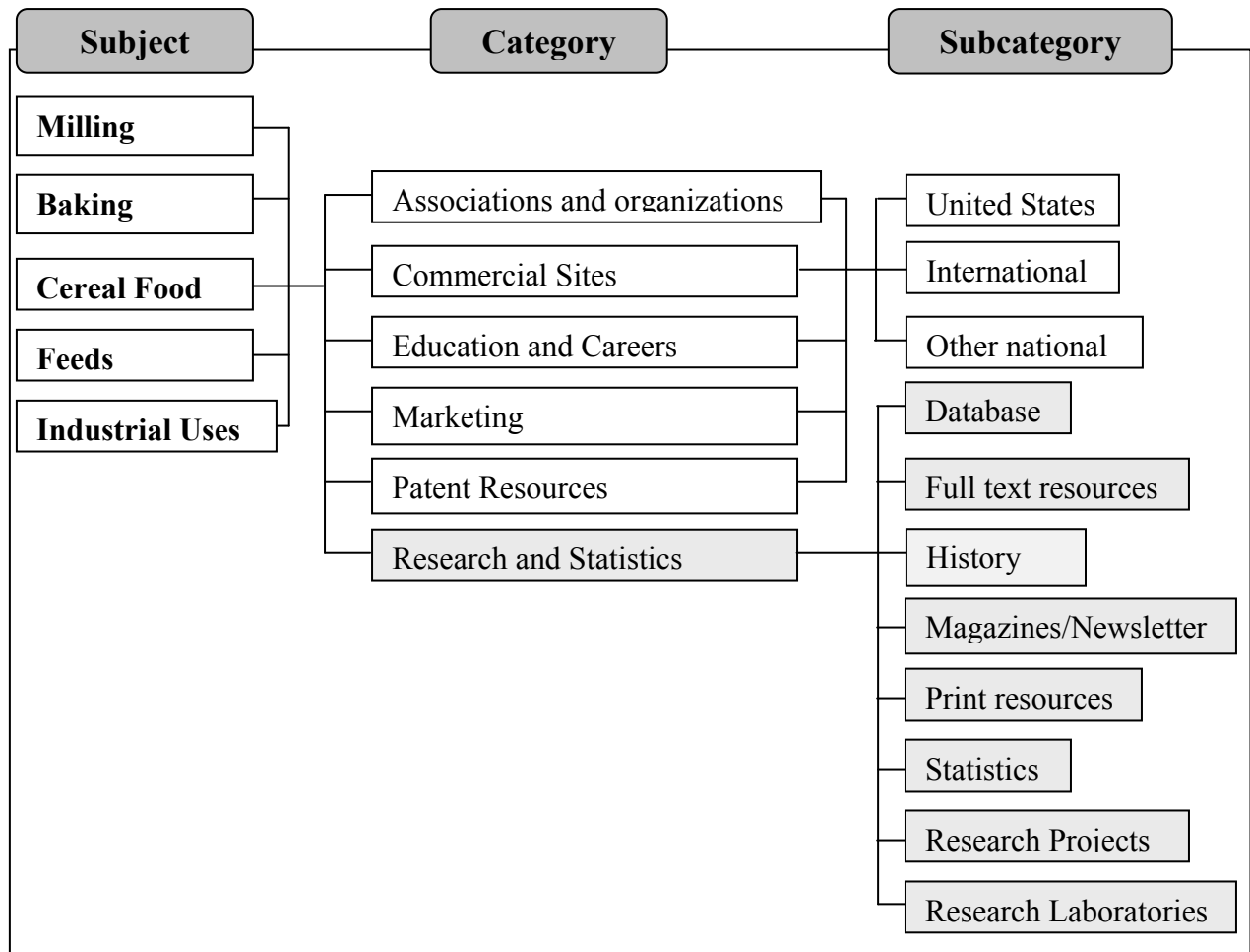


FIGURE 1 Grain Milling and Processing's Current Site Map.

FIG 2

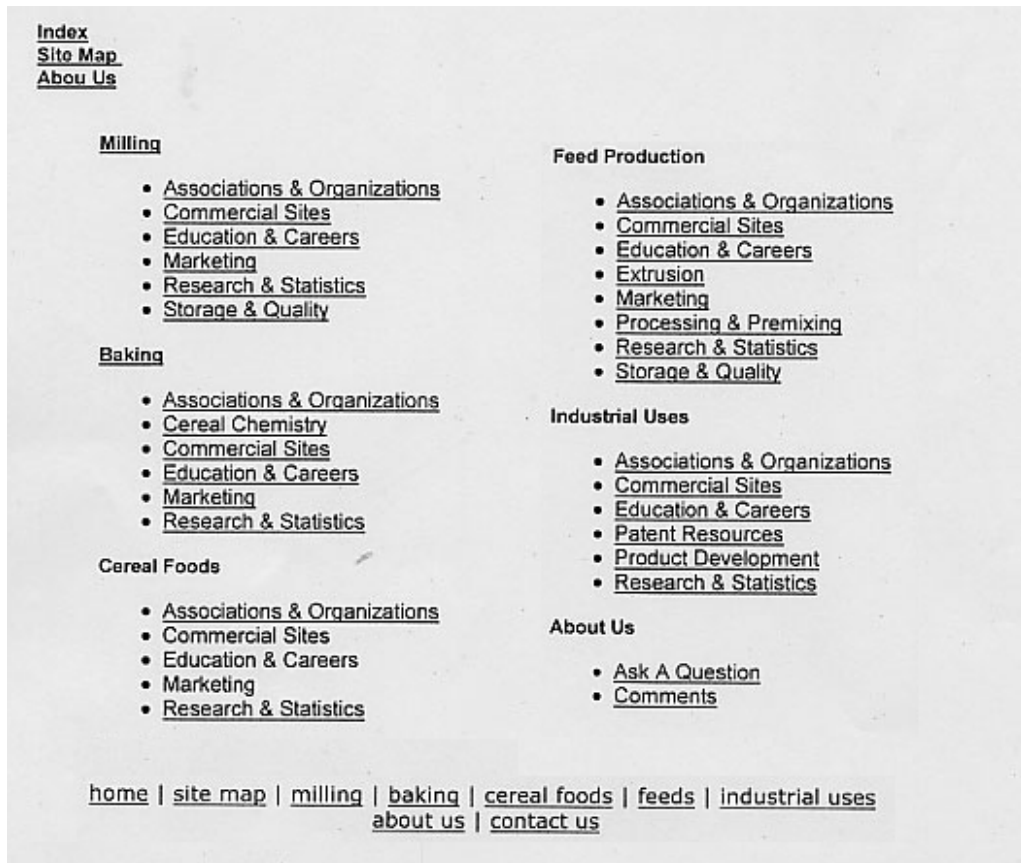


FIGURE 2 Grain Milling and Processing's Initial Site Map.

FIG 3

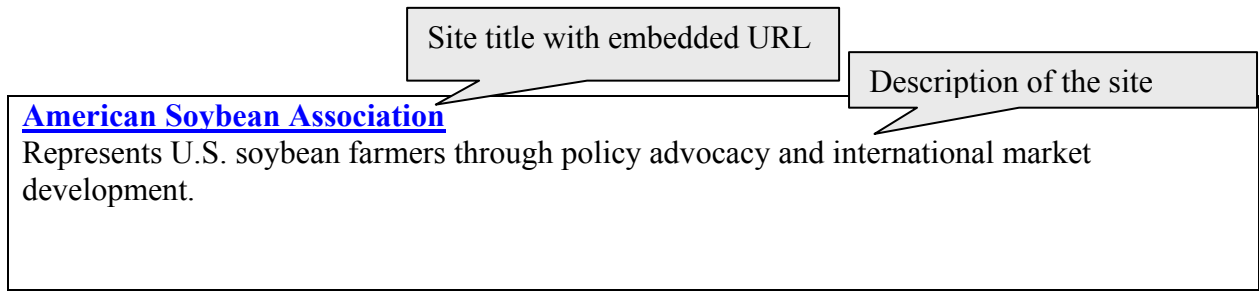


FIGURE 3 Sample Site Entry.

FIG 4

The diagram illustrates a sample full text entry within a rectangular box. The entry consists of a title and a paragraph of text. Three callout boxes with arrows point to specific parts of the entry: 'Site title with embedded URL' points to the title, 'Physical description' points to the first sentence of the paragraph, and 'Description of the site' points to the last sentence of the paragraph.

Site title with embedded URL

Physical description

[Biosecurity Awareness Guide](#) [PDF] 1.26 MB
The Biosecurity Awareness Guide was produced with the participation of the Animal Health Institute (AHI); the Center of Veterinary Medicine (CVM)/Food and Drug Administration (FDA); and the National Renderers Association (NRA) by the American Feed Industry Association. It is available as a [PDF](#) or as a [PowerPoint Presentation](#) file. The PowerPoint Presentation file is available in either [English](#) or [Spanish](#).

Description of the site

FIGURE 4 Sample Full Text Entry.

Fig 5

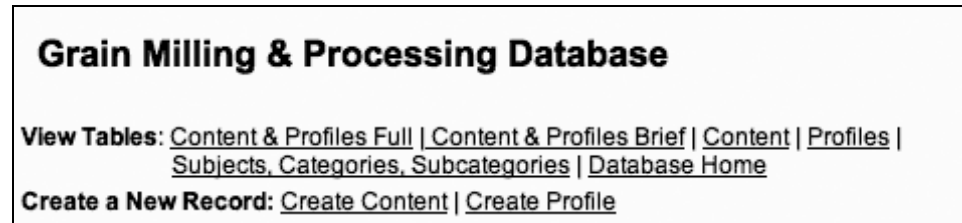


FIGURE 5 Staff Interface.

Fig 6

2	Content ID/Profile ID	248 & 292	
	Title	ABA Labeling Manual	
3	Content ID/Profile ID	79 & 329	↓
	Title	ABNA : Associated British Nutrition & Agriproducts	
4	Content ID/Profile ID	79 & 330	↓
	Title	ABNA : Associated British Nutrition & Agriproducts	

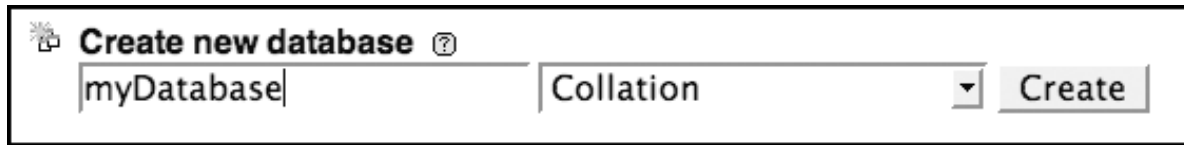
FIGURE 6 Content and Profile Brief View.

Fig 7

Content ID & Profile ID: 79 & 329	
subject category subcategory: feeds-> commercial sites -> International	
Record # 88	Title ABNA : Associated British Nutrition & Agriproducts
	Creator
	Description An international UK business delivering products, technology and services to the food, drink and animal feed sectors.
	Publisher ABNA
	Date
	Resource Type Lookup web page
	Resource Identifier http://www.clickagri.com/page.cfm?LANGUAGE=eng&pageID=852
	Language Eng,
	Coverage
	Format
	Resource Size
Update Content	Update Profile
Delete Profile	Add Profile
Delete Record	▲
Content ID & Profile ID: 79 & 330	
subject category subcategory: industrial uses-> commercial sites -> International	
Record # 89	Title ABNA : Associated British Nutrition & Agriproducts
	Creator
	Description An international UK business delivering products, technology and services to the food, drink and animal feed sectors.
	Publisher ABNA
	Date
	Resource Type Lookup web page
	Resource Identifier http://www.clickagri.com/page.cfm?LANGUAGE=eng&pageID=852
	Language Eng,
	Coverage
	Format
	Resource Size
Update Content	Update Profile
Delete Profile	Add Profile
Delete Record	▲

FIGURE 7 Content and Profile Detailed View.

Fig 8



The image shows a screenshot of the 'Create new database' form in phpMyAdmin. The form is enclosed in a black border. At the top left, there is a small icon of a database and the text 'Create new database' followed by a question mark icon. Below this, there is a text input field containing the text 'myDatabase'. To the right of the input field is a dropdown menu with the text 'Collation' and a downward-pointing arrow. To the right of the dropdown menu is a button labeled 'Create'.

FIGURE 8 Create a Database Under phpMyAdmin. Note: Spaces and special characters should be avoided in a database name. For example, use myDatabase or my_database as the database name, instead of “my database.”

Fig 9

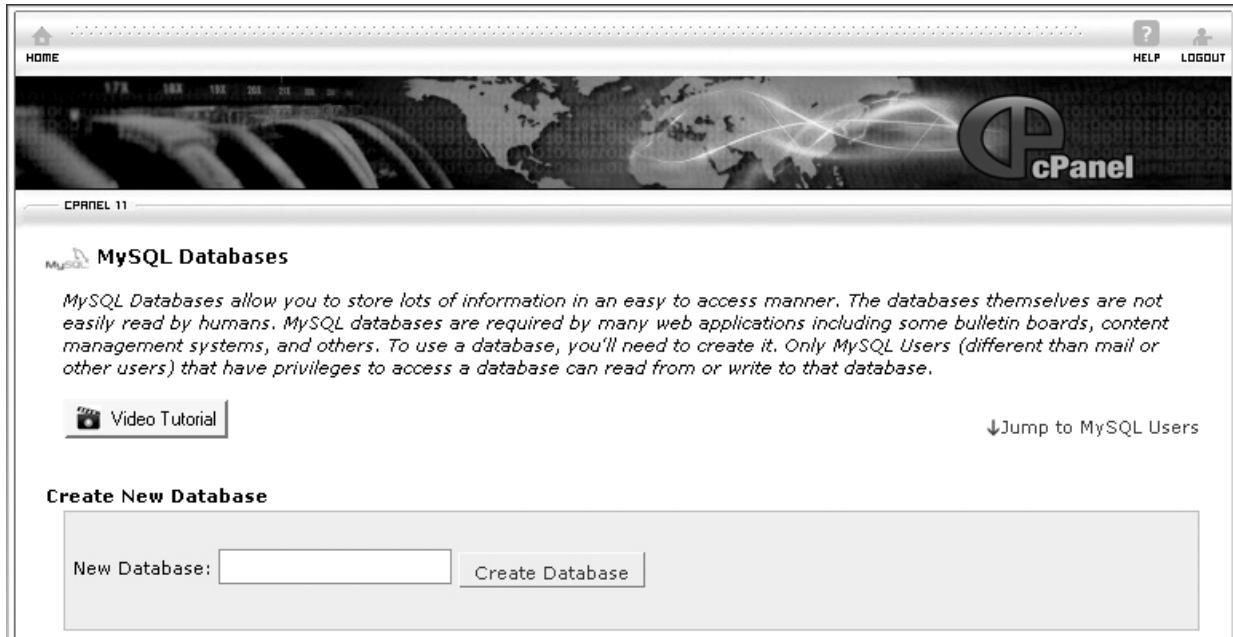


FIGURE 9 Create a Database Using cPanel.

Fig 10

The screenshot shows a database management interface with a table structure overview and a 'Create new table' dialog.

Table Structure Overview:

	Table	Action	Records	Type	Collation	Size	Overhead
<input type="checkbox"/>	categories		6	MyISAM	latin1_swedish_ci	2.2 KiB	-
<input type="checkbox"/>	content		437	MyISAM	latin1_swedish_ci	192.8 KiB	-
<input type="checkbox"/>	login		2	MyISAM	latin1_swedish_ci	2.0 KiB	-
<input type="checkbox"/>	profiles		639	MyISAM	latin1_swedish_ci	22.1 KiB	-
<input type="checkbox"/>	subcategories		11	MyISAM	latin1_swedish_ci	2.3 KiB	-
<input type="checkbox"/>	subjects		5	MyISAM	latin1_swedish_ci	2.1 KiB	-
	6 table(s)	Sum	1,100	MyISAM	latin1_swedish_ci	223.5 KiB	0 B

Check All / Uncheck All With selected:

Print view Data Dictionary

Create new table on database gmilling_grainmilling

Name: Number of fields: ← create a new table within an existing database

FIGURE 10 Create New Table Step 1.

Fig 11

Server: localhost ▶ Database: gmilling_grainmilling ▶ Table: test

Field	ID	Name a field	Title	Comment
Type ②	INT	Specify data type	VARCHAR	TEXT
Length/Values ¹	4	Specify data length	256	
Collation				
Attributes				
Null	not null		not null	not null
Default ²				
Extra				
<input checked="" type="radio"/>	MUST choose one		<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	field as the primary key		<input type="radio"/>	<input type="radio"/>
<input type="radio"/>			<input type="radio"/>	<input type="radio"/>
---	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>
<input type="checkbox"/>			<input type="checkbox"/>	<input type="checkbox"/>
Comments				

Table comments:

Storage Engine: ② MyISAM

Collation:

Save the new table ➡ Or Add field(s)

FIGURE 11 Create New Table Step 2.

Fig 12

The screenshot shows the phpMyAdmin interface with the following elements:

- Navigation Bar:** Browse, Structure, SQL, Search, Insert, Export, Import, Operations, Empty.
- Buttons:** Drop (with a warning icon).
- Message:** Table `gmilling_grainmilling`.`test` has been created.
- SQL Query:**

```
CREATE TABLE `gmilling_grainmilling`.`test` (
  `ID` INT(4) NOT NULL ,
  `Title` VARCHAR(256) NOT NULL ,
  `Comment` TEXT NOT NULL ,
  PRIMARY KEY (`ID`)
) ENGINE = MYISAM
```
- Table Structure:**

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	<u>ID</u>	int(4)			No			
<input type="checkbox"/>	Title	varchar(256)	latin1_swedish_ci		No			
<input type="checkbox"/>	Comment	text	latin1_swedish_ci		No			
- Options:** Profiling [Edit] [Create PHP Code]

FIGURE 12 Created Table View.

Fig 13

Field	Type	Function	Null	Value
ID	int(4)	<input type="text"/>		<input type="text" value="24"/>
Title	varchar(256)	<input type="text"/>		<input type="text" value="Input a title less no more than 256 letters"/>
Comment	text	<input type="text"/>		<input type="text" value="Input a message here"/>

FIGURE 13 Insert a Record to a Table.

Fig 14







←T→		content_id	title	creator	description	publisher
<input type="checkbox"/>	 	1	Rain damage reduced grain quality	<i>NULL</i>	Grain that is re-wet by rainfall after it is dried...	North Dakota State University Extension Services
 edit an entry						
<input type="checkbox"/>	 	2	American Institute of Baking	American Institute of Baking	A not-for-profit corporation, founded by the North...	American Institute of Baking
 delete an entry						

FIGURE 15 Edit or Delete an Entry.

Fig 15

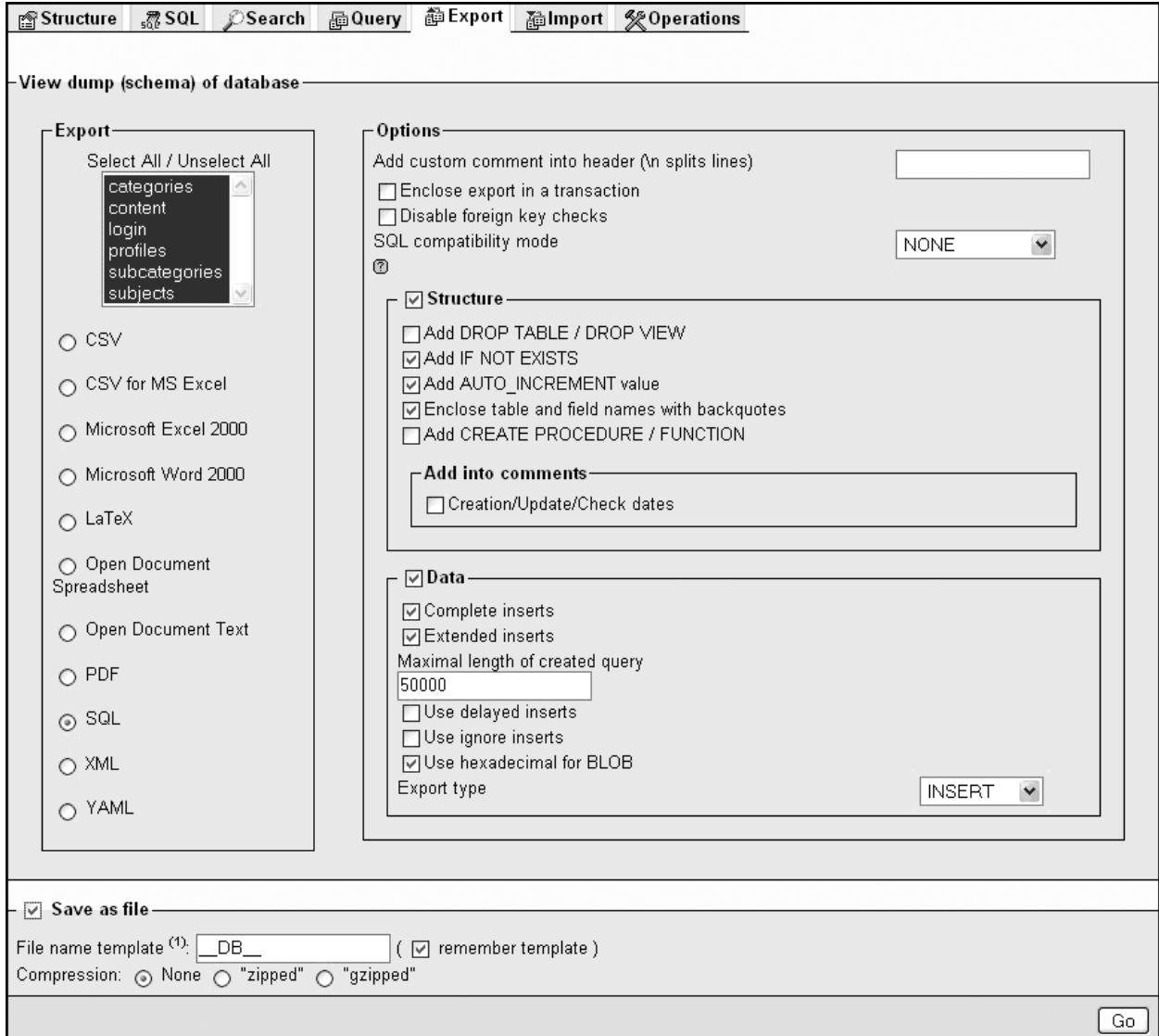


FIGURE 15 Back Up a Database.

Fig 16

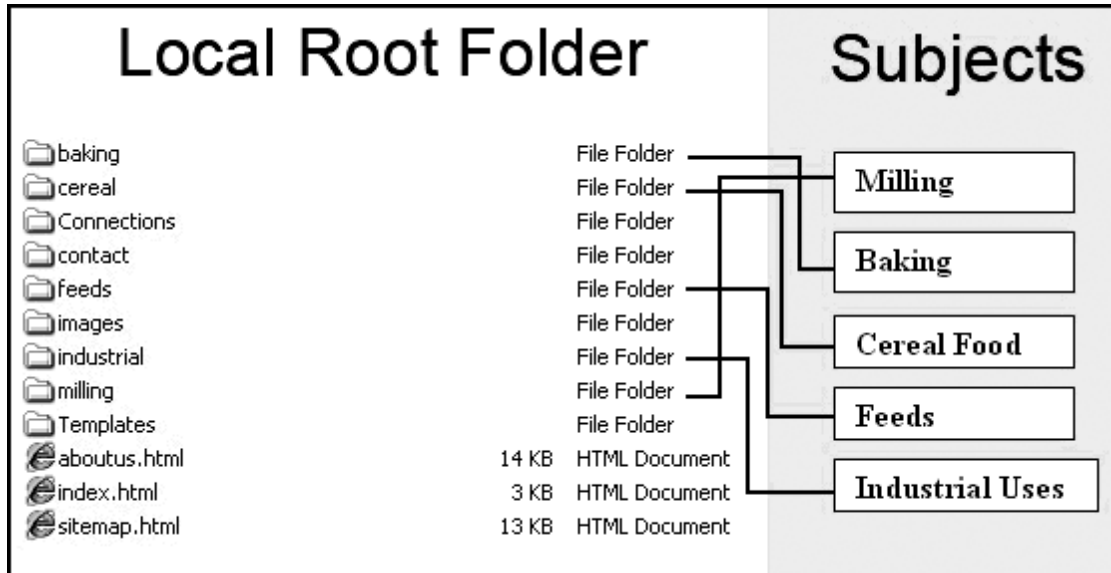


FIGURE 16 Local Server.

Fig 17

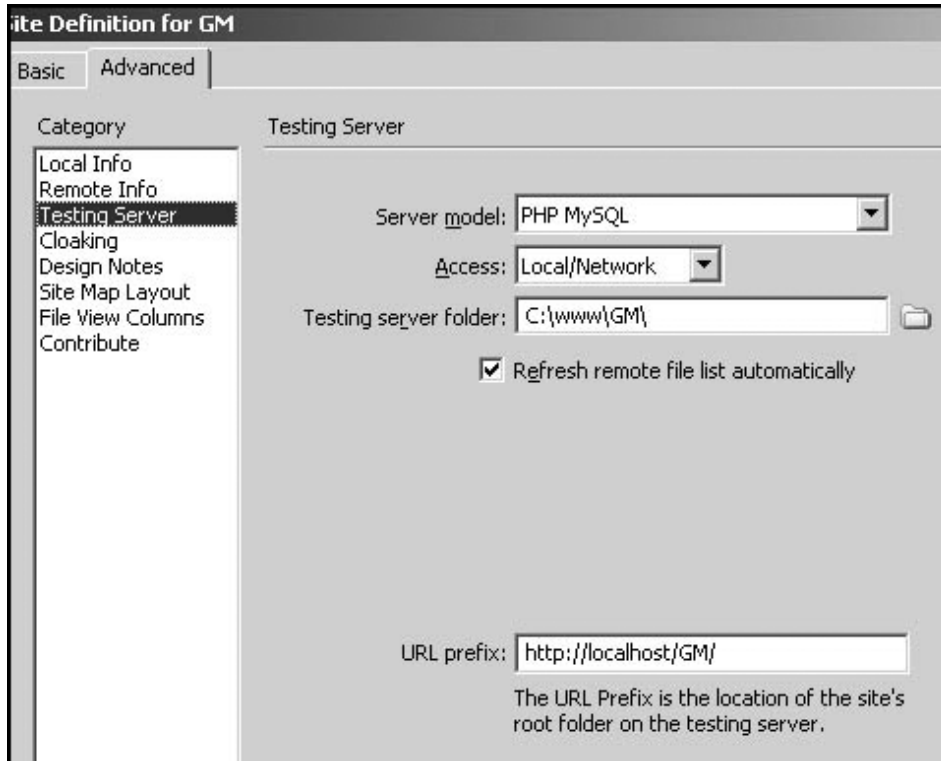


FIGURE 17 Definition of a Test Server.

Fig 18

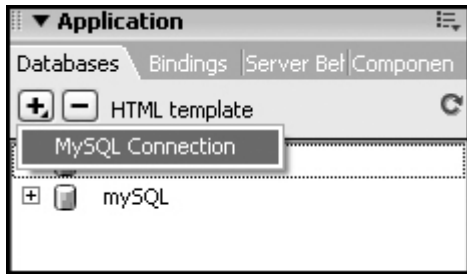


FIGURE 18 Create a MySQL Connection.

Fig 19

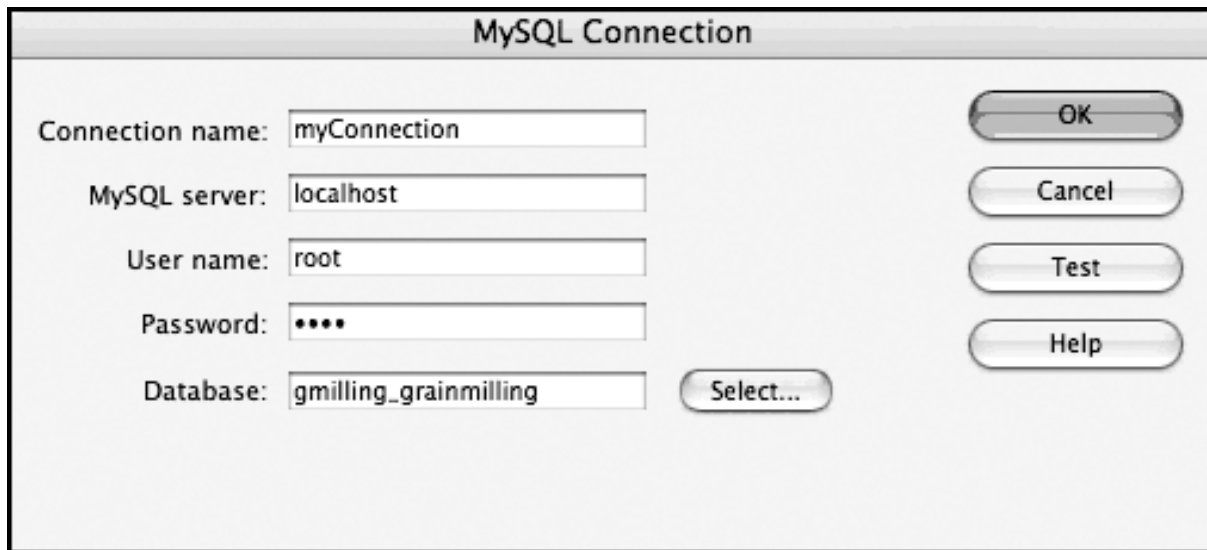


FIGURE 19 MySQL Connection Dialog.

Fig 20

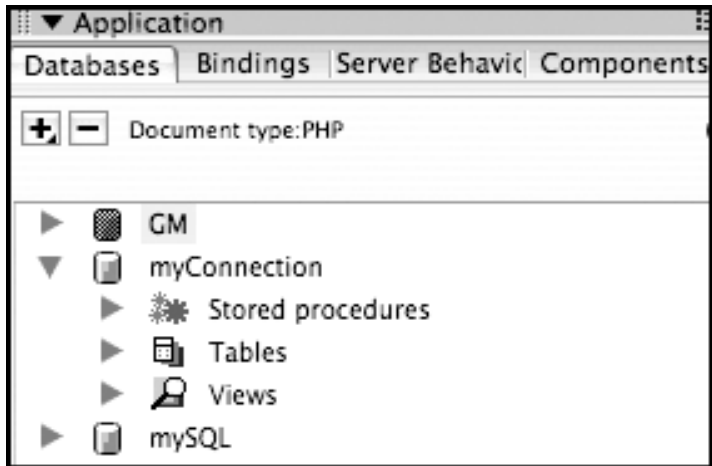


FIGURE 20 Connections on the Application Panel.

Fig 21

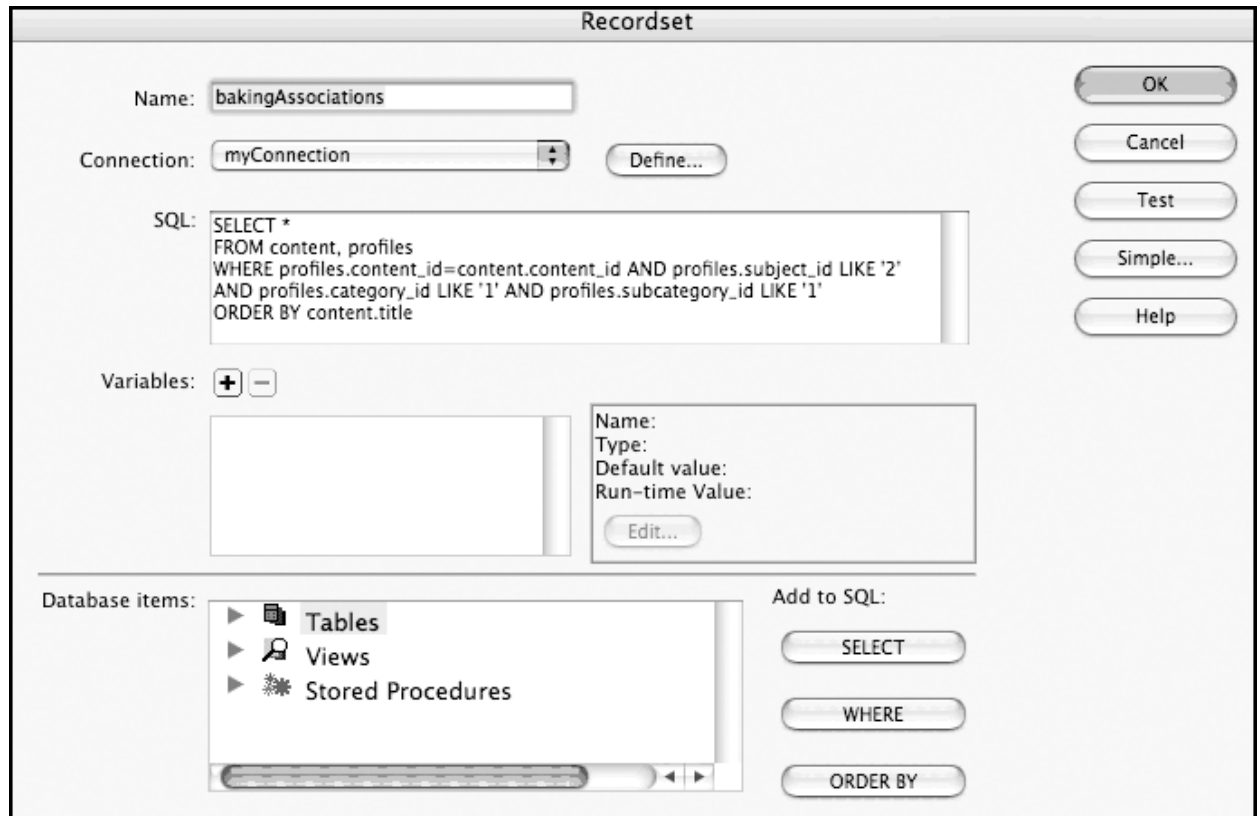


FIGURE 21 Create a Query/Recordset Dialog.

Fig 22



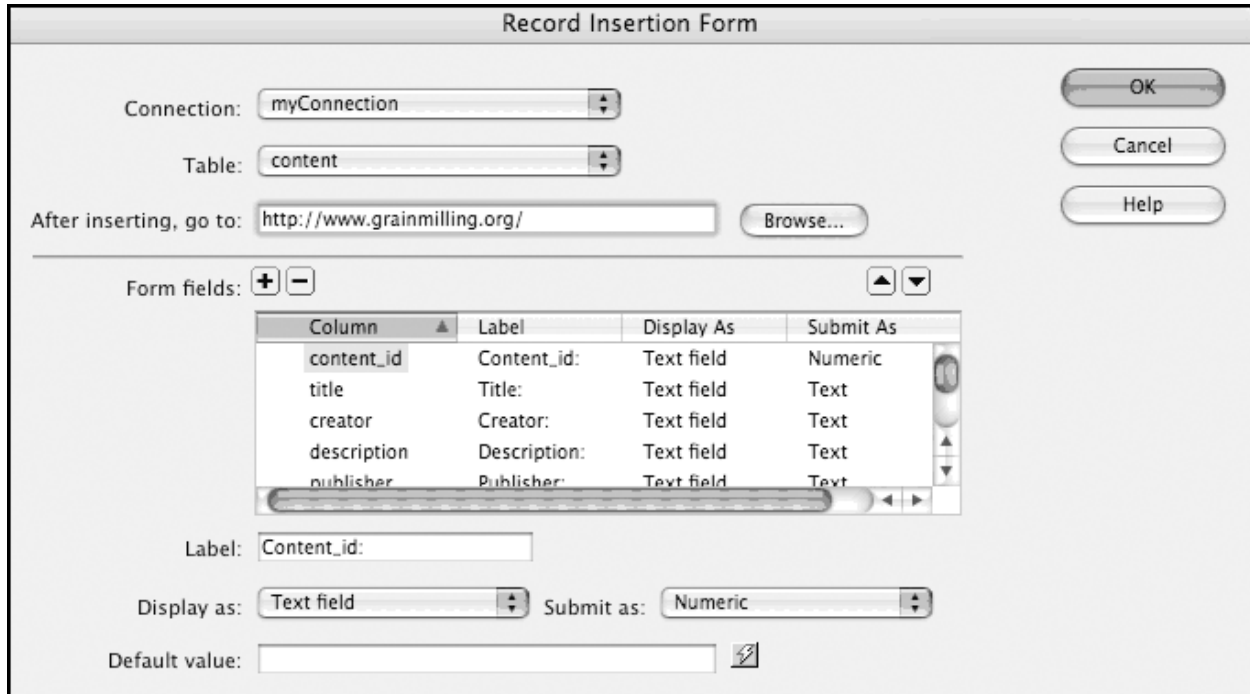
FIGURE 22 Adding Dynamic Text.

Fig 23

The Last Record's Content ID is439	
* Content_id:	<input type="text"/>
* Title:	<input type="text"/>
Creator:	<input type="text"/>
* Description:	<input type="text"/>
Publisher:	<input type="text"/>
Date:	<input type="text"/>
Resource Type Lookup:	<input type="text"/>
Resource Identifier:	<input type="text"/>
include http://	<input type="text"/>
Language:	<input type="text" value="Eng,"/>
Coverage:	<input type="text"/>
Format:	<input type="text"/>
Resource Size:	<input type="text"/>
<input type="button" value="Insert record"/>	

FIGURE 13 Create New Content Form for the Grain Milling and Processing site.

Fig 24



The image shows a 'Record Insertion Form' dialog box. At the top, it has a title bar. Below the title bar, there are several input fields: 'Connection' with a dropdown menu showing 'myConnection', 'Table' with a dropdown menu showing 'content', and 'After inserting, go to:' with a text field containing 'http://www.grainmilling.org/' and a 'Browse...' button. On the right side, there are three buttons: 'OK', 'Cancel', and 'Help'. Below these fields, there is a section for 'Form fields:' with a '+' and '-' button and up/down arrow buttons. This section contains a table with four columns: 'Column', 'Label', 'Display As', and 'Submit As'. The table lists five columns: 'content_id', 'title', 'creator', 'description', and 'publisher'. Below the table, there are fields for 'Label:' (containing 'Content_id:'), 'Display as:' (a dropdown menu showing 'Text field'), 'Submit as:' (a dropdown menu showing 'Numeric'), and 'Default value:' (a text field with a small icon to its right).

Column	Label	Display As	Submit As
content_id	Content_id:	Text field	Numeric
title	Title:	Text field	Text
creator	Creator:	Text field	Text
description	Description:	Text field	Text
publisher	Publisher:	Text field	Text

FIGURE 24 Record Insertion Form Dialog.

Fig 25

Content_id:	
Title:	
Creator:	
Description:	
Publisher:	
Date:	
Resource_type_lookup:	
Resource_identifier:	
Language:	
Coverage:	
Format:	
Resource_size:	
	<input type="button" value="insert record"/>

FIGURE 25 Insertion Form Preview.

Fig 26

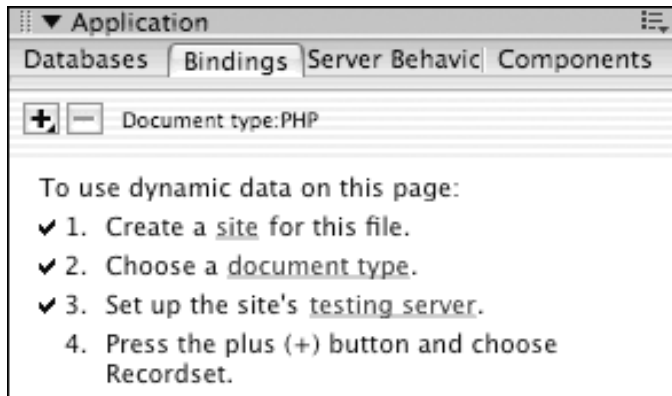


FIGURE 26 Bindings Panel.

Fig 27

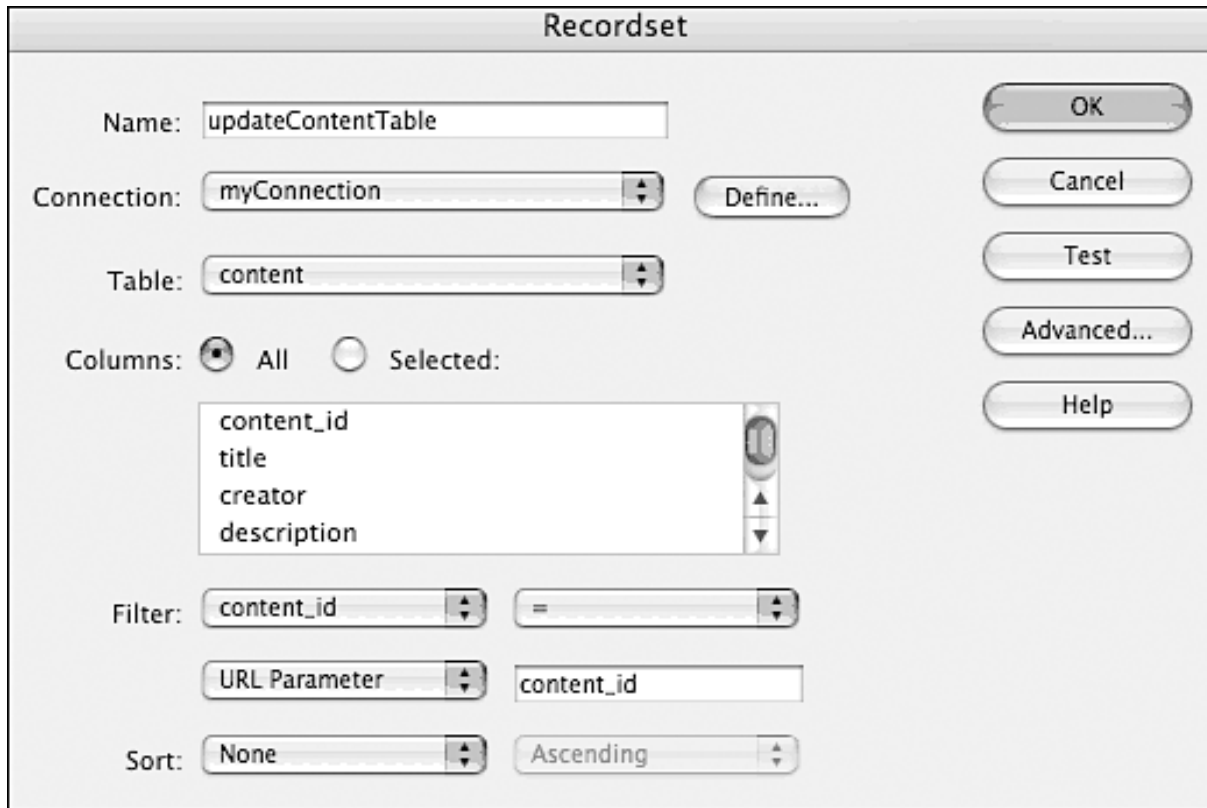
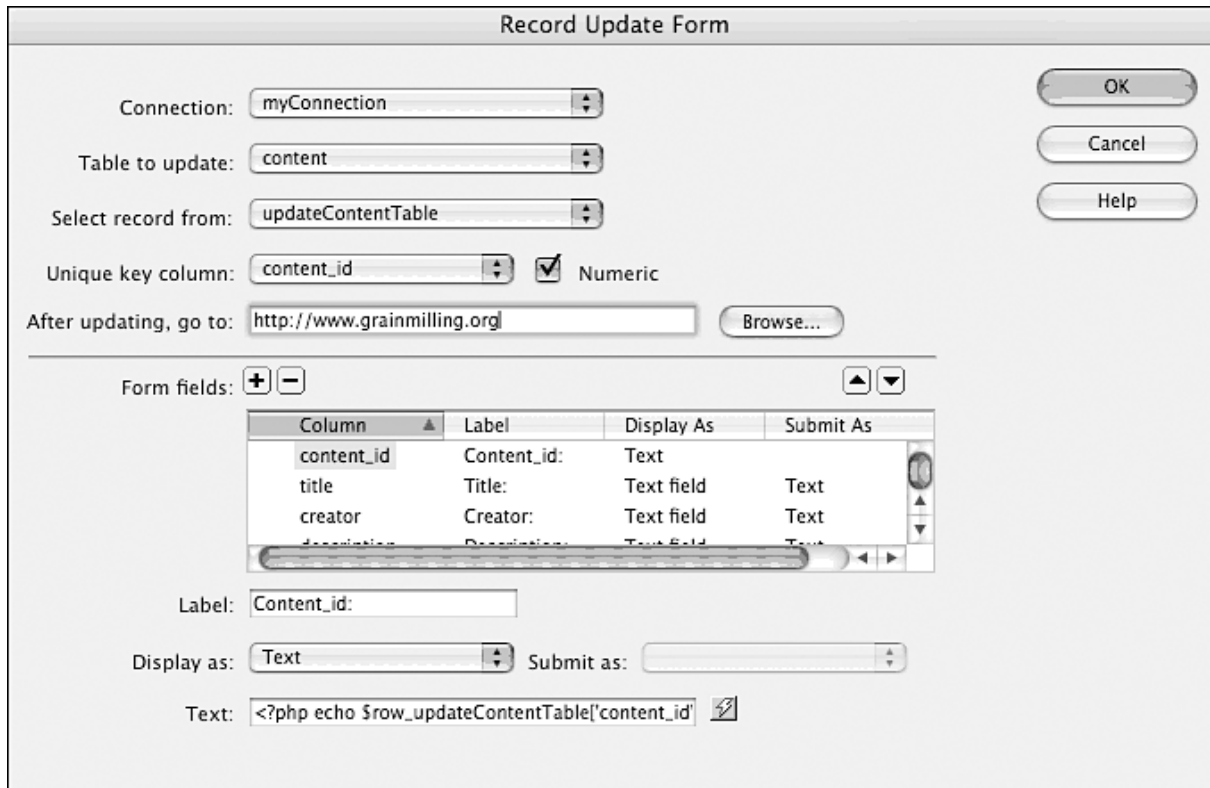


FIGURE 27 Create a Recordset for an Update Form.

Fig 28



The dialog box is titled "Record Update Form". It contains several configuration options:

- Connection: myConnection
- Table to update: content
- Select record from: updateContentTable
- Unique key column: content_id (with a checked "Numeric" checkbox)
- After updating, go to: http://www.grainmilling.org (with a "Browse..." button)

Below these options is a "Form fields:" section with a table:

Column	Label	Display As	Submit As
content_id	Content_id:	Text	
title	Title:	Text field	Text
creator	Creator:	Text field	Text
description	Description:	Text field	Text

Below the table, there are input fields for "Label" (Content_id:), "Display as" (Text), "Submit as" (empty), and "Text" (code: `<?php echo $row_updateContentTable['content_id']`).

Buttons for "OK", "Cancel", and "Help" are located in the top right corner.

FIGURE 28 Record Update Form Dialog.

Fig 29

Content id:	{updateContentTable.content_id}
Title:	{updateContentTable.title}
Creator:	{updateContentTable.creator}
Description:	{updateContentTable.description}
Publisher:	{updateContentTable.publisher}
Date:	{updateContentTable.date}
Resource_type_lookup:	{updateContentTable.resource_type_lookup}
Resource_identifier:	{updateContentTable.resource_identifier}
Language:	{updateContentTable.language}
Coverage:	{updateContentTable.coverage}
Format:	{updateContentTable.format}
Resource_size:	{updateContentTable.resource_size}
	<input type="button" value="Update record"/>

FIGURE 29 Update Form Preview.

Fig 30

```
<a href="updateContent.php?content_id=<?php echo $row_content_profiles['content_id']; ?>">  
  
</a>
```

FIGURE 30 Pass an ID to an Update Form.

Fig 31

```
<a href="updateContent.php">  
  
</a>
```

FIGURE 31 Add a Link to an Image.