Creating a Simple Blog Part 2

By: Tom Muck

One of the frequent questions in the forums is "How do I create a blog?" There are many commercial blog systems out there, and many full-featured blog providers, like Blogger, but you can also create a simple blog using the standard tools of Dreamweaver. This tutorial will be equally applicable to ColdFusion, ASP, PHP, ASP.NET, or JSP because we will be using standard Dreamweaver server behaviors to create the blog. Part 1 (http://www.communitymx.com/abstract.cfm?cid=7CC52) of the tutorial covered how to create a simple blog using point-and-click features of Dreamweaver. Part 2 will focus on adding more features to the blog, such as a system for adding and viewing comments, and more administration options.

This series of articles will assume that you know how to work with Dreamweaver to create a data-driven site. For example, you should know how to create databases and connections within your chosen server model.

When we left off, we had a basic blog that allowed you to post items and display them. Now you need a way to edit these items once they are posted. We'll create another page in the Admin area called **update.php** (or .cfm, .asp, or whatever file type you are using) that will allow this. The items were displayed on the **index.php** file in the admin folder. We'll turn this index page into a main administration page, where we'll be able to click on an item to update it or delete it. First, we should put a link at the top of the index page to the **insert.php** page that was already created, so we can insert or update items directly from the index page:

```
<p><a href="insert.php">Insert a blog item</a></p>
```

Next, within the repeat region already on the page that displays the blog items, we'll create a link to an update page that will be created next. The easiest way to do this is in design view. Follow these steps:

1. Create a new line below the **blog_item** field in the display and type the words "Update item". Select the text.
2. Open the Property Inspector and click the folder icon next to the Link field.
3. In the dialog box, click the **Data Sources** radio button and choose the **blog_item_id** field.
4. Type the page name and an **id** URL variable in front of the dynamic data, as shown below and in **Figure 1.** The code shown in the figure will be slightly different for the different server models, but the steps are the same:

```
update.php?id=
```

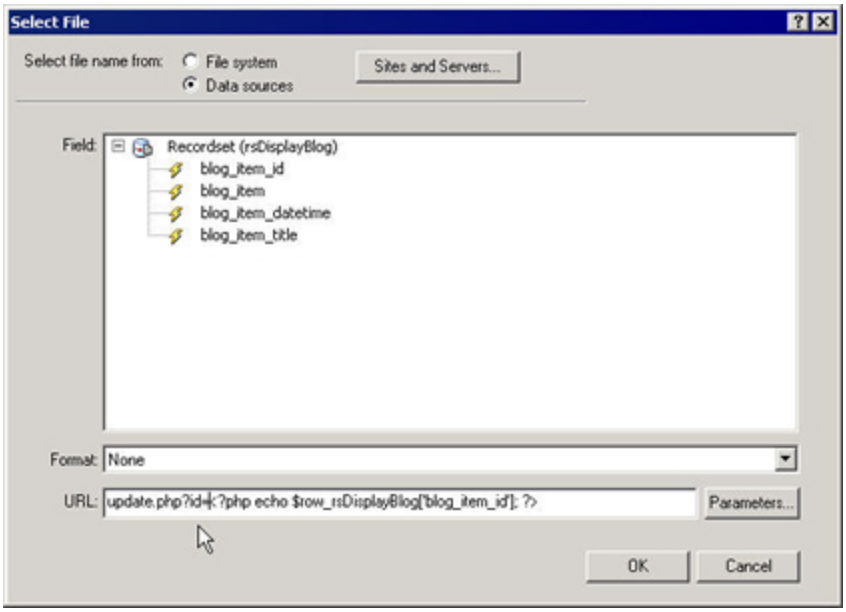

Figure 1: *The link dialog box to set the dynamic link*

This effectively chooses the current blog item and passes the **blog_item_id** field to the update page. Next, we create the update page.

Creating the Blog Item Update Page

The update page will be built similarly to the insert page. We'll use an Application object so that we can avoid having to create form elements manually. Everything is done for us. The one prerequisite to building the update form, however, is to create a recordset that pulls only one record -- the record that is passed in the URL string from the index page. Follow these steps to create the recordset. We'll do this in the Simple dialog box for the recordset, only because it will be easier to describe for all server models as the steps are the same. Create the recordset named **rsUpdateBlogItems**. Set it up using the **blog_items** table, showing all fields, and filtering the **blog_item_id** field by a URL parameter named **id**, as shown in **Figure 2**:

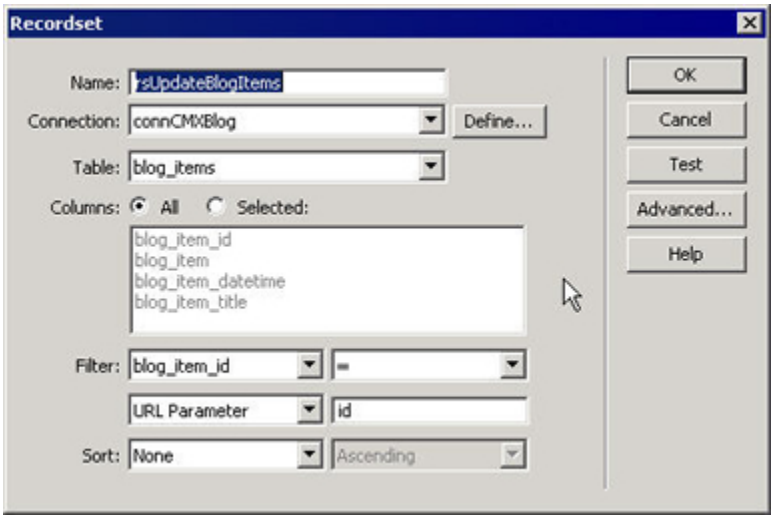

Figure 2: *Setting up the Simple Recordset Dialog Box*

Next, add the **Record Update Form Wizard**. To use this wizard, click **Insert > Application Objects > Update > Record Update Form Wizard**. There is also a button on the Application insert bar that performs the same function. That brings up the interface shown in **Figure 3** (the interface may vary

slightly in different server models) .

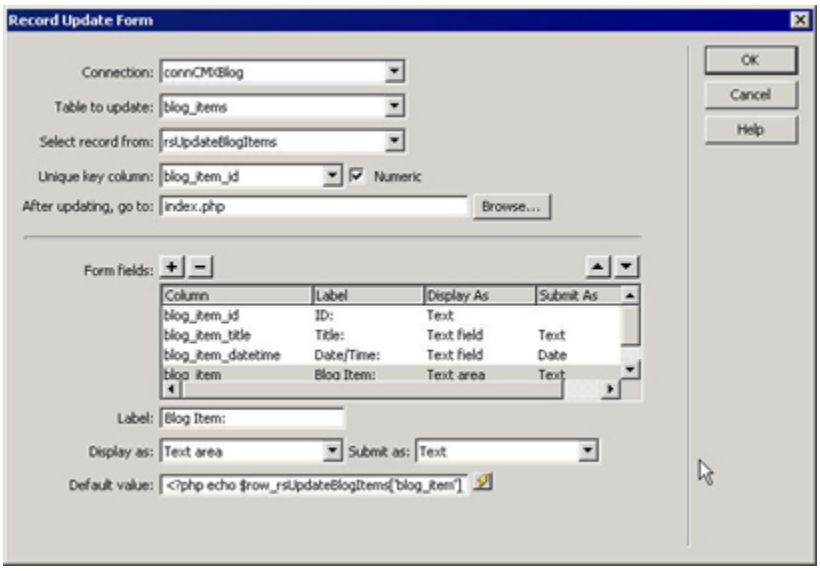

Figure 3: *The Record Update Form Wizard*

You should set it up using the following parameters:

- Data Source: your CMXBlog database
- Table to Update: blog_items
- Select record from: rsUpdateBlogItems
- Unique key column: blog_item_id
- After updating, go to: index.php
- Form Fields: Set the following fields in this order (use the up/down arrow keys to change the order):

Column	Label	Display As	Submit As
blog_item_id	ID:	Text	
blog_item_title	Title:	textfield	Text
blog_item_datetime	Date/Time:	textfield	Date*
blog_item	Blog Item:	textfield	Text

*depending on your database/server model, you may want to use Text instead of Date. Also, the format of the date entered may be different for various server model/database configurations (i.e. 03/03/2004, 2004/03/03, 2004-03-03). Part 3 of this series will show how insert the current date/time automatically with a little hand-coding.

With that in place, click **OK** and you have your blog item update page. You can dress it up with titles, CSS, etc, but the page is functional at this point. The completed page is shown in **Figure 4:**

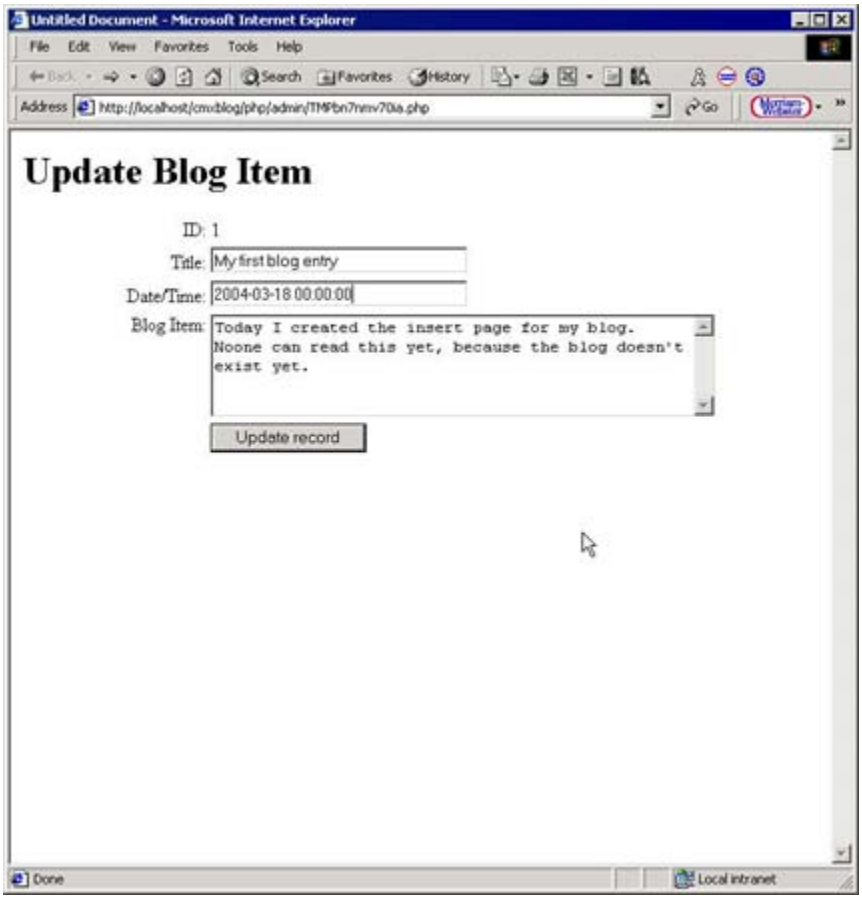

Figure 4: *The completed update page*

The update page is now completed. It's time to move on to the comment sections.

Adding the Insert and View Comment pages

As you may recall, the blog has two index pages: the index page in the admin area, which we just worked on, and the index page in the main site. Moving back to the index page in the main site, we'll add a couple of links to the **comment.php** and **viewcomment.php** pages (again, using .cfm, .asp, or whatever server model you are using.) Open the index page in the main site and add two links inside the repeat region, as you did for the index page in the admin section. Follow the same steps that you used to add the link for the update page -- pass the **blog_item_id** variable in the URL. You should create links on text such as Add Comment and View Comments, as shown in **Figure 5**:

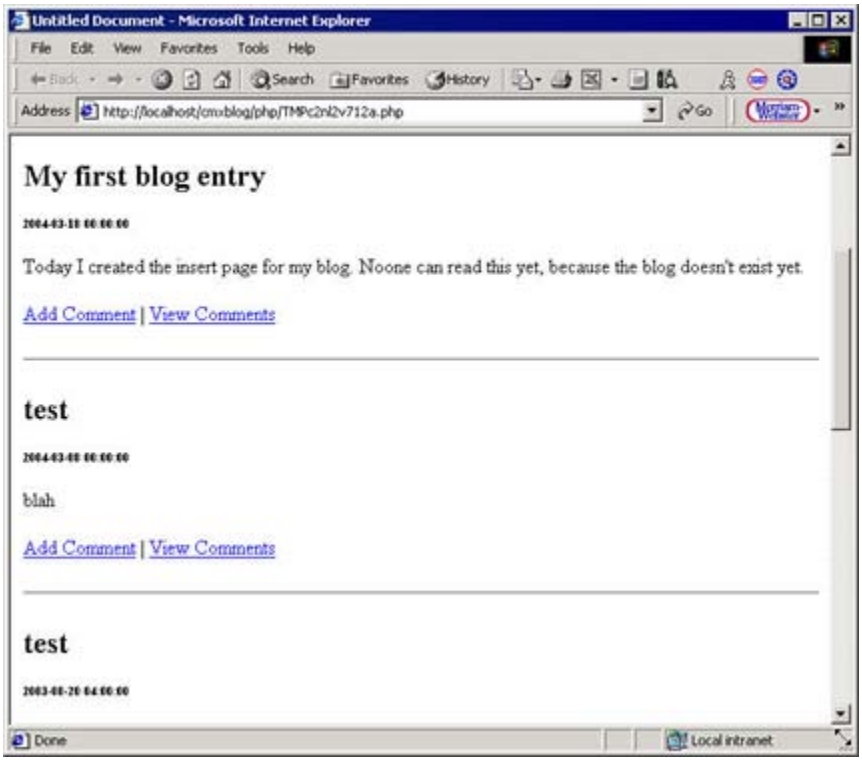

Figure 5: *After adding the Add Comment and View Comments links*

Next, create the comment.php page, which will allow people to add comments. Before adding the insert to the page, however, we'll add a URL variable to the Bindings panel. This will make it easier to pass the **blog_item_id** to the database insert, which you'll see in a moment.

1. Click the plus sign (+) on the Bindings panel.
2. Choose URL Variable
3. Type the name **id**
4. Click OK.

The panel should now show the URL variable, **id**. Nothing has been added to the page at this point, but the variable will be useful for the insert.

Next, we'll use the Record Insertion Form Wizard, as shown in Figure 6:

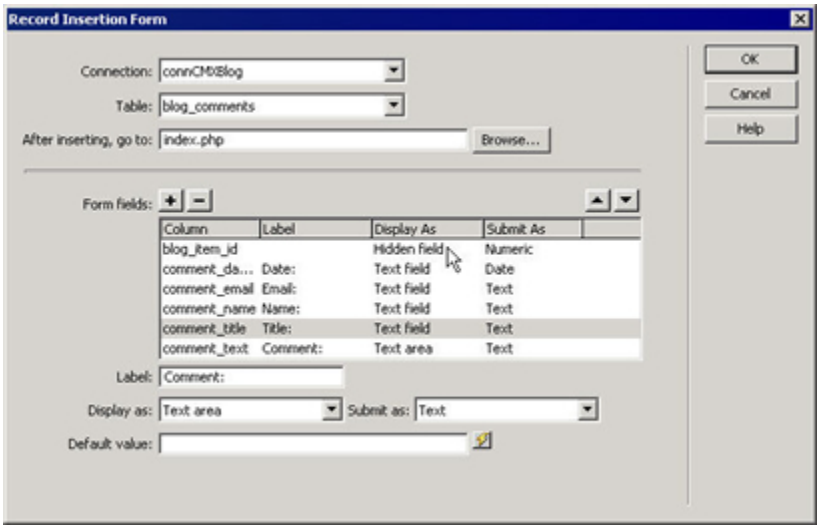

Figure 6: *The Record Insertion Wizard*

You should set it up using the following parameters:

- Data Source: your CMXBlog database
- Table: blog_comments
- After inserting, go to: index.php
- Form Fields: Use the minus button to remove the **blog_comment_id** field, and set the following fields in this order (use the up/down arrow keys to change the order):

Column	Label	Display As	Submit As
blog_item_id		Hidden Field	Numeric
comment_datetime	Date:	textfield	Date*
comment_email	Email:	textfield	Text
comment_name	Name:	textfield	Text
comment_title	Title:	textfield	Text
comment_text	Comment:	Text area	Text

The **blog_item_id** will relate the comment back to the original blog item that is being commented on. This should be passed as a hidden form field to prevent users from editing it. One extra thing needs to be done to pass the **blog_item_id** to the database: you need to set up a default value. This can be done easily using the URL variable that we previously set up. Click the dynamic data icon (lightening bolt) next to the Default Value field. Choose the URL variable that you just set up. After doing this, the value of the URL variable will automatically populate the hidden field, and also add the **blog_item_id** to the database insert.

Now that there is a way to enter comments, you need a way to view comments. Open the viewcomments page and add a recordset to the page, using the Simple dialog box once again. Create the recordset named **rsViewComments**. Set it up using the **blog_comments** table, showing all fields except **comment_id** and **blog_item_id**. It should be filtered on **blog_item_id** and ordered by **comment_datetime** in ascending order, as shown in Figure 7:

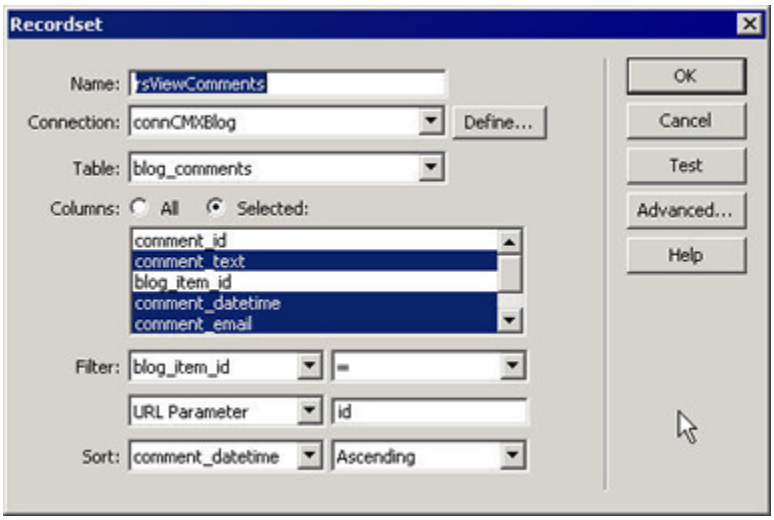

Figure 7: *The rsViewComments recordset*

Next, do the same as you did in part 1 of this series for the index page, only this time you will be using the fields from the **blog_comments** table. After adding the recordset to the page, give your page a title and a heading (like "Comments ") and then drag these recordset fields from the Bindings panel to the page one by one: **comment_title**, **comment_datetime**, **comment_name, comment_email, comment_text** . You can make **comment_title** an **<h2>** tag and **comment_datetime** an **<h3>** tag. You can also add an **<hr>** tag beneath the comment.

With that in place, drag your cursor over the dynamic items and the **<hr>** tag, effectively selecting the entire comment. Next, apply a Repeat Region to the area, showing all records, by clicking the plus sign (+) in the Server Behaviors panel and clicking Repeat Region (or by choosing it from the Application item in the Insert bar or Insert menu). If you browse the page now, you will see any comment that you inserted previously.

Now you should be able to browse your blog items using the index page, comment on them using the comment page, and view all comments on a particular item using the viewcomments page.

Conclusion

Blogs are simple content management systems that can be put together easily using some built-in Dreamweaver functionality. In this part, we've added functionality to the blog to allow updating items, and posting and viewing comments to the items. The next part of the series will move away from the automated features of Dreamweaver and show some of the more advanced aspects of blogs, such as creating an RSS feed. Also, because dates can be entered incorrectly, we'll show a couple of different techniques for inserting accurate dates into the database.

Approximate download size: 455k

Keywords
blog,coldfusion,asp,php,asp.net,jsp,rss