Using Advanced HTML Editor to Create a Content Management System

In this tutorial I'm going to show you a practical example of how to make use of the Advanced HTML Editor extension. We're going to use it to create a simple news updating application, the type you might use if you were creating a community site. In our example, it's a Jazz Club (nice!), but it can be any site you care to create.

The advantage of the Advanced HTML Editor is that it allows users to input information in a way more familiar them. It's just like a word processor! This means you don't have to spend time teaching them HTML, or doing casual updates yourself. Our example will show how a few simple server behaviors can combine with the Editor to create an application that means less work for you.

What You'll need

We'll be using ASP in this example, and making use of a SQL sever database. So you'll need a webserver or test machine that can run ASP and has access to SQL server. This means that you'll need IIS (Internet Information Services). This is available on Windows 2000 and XP Pro, and you may have to install it from your OS disk if it wasn't installed originally. Windows 98 users can get a stripped down version called Personal Web Server, again on their Operating System CD. If you're using Windows ME, you *can* install PWS but it's not supported – here's some instructions; proceed at your own risk!

SQL Server is available from Microsoft. A slimmed-down version that includes Enterprise Manager for development only (useful for developing on a local machine) is available from Microsoft for $49.95, or there's a free version of SQL Server that doesn't include Enterprise Manager. (More about Enterprise Manager below).

You'll also need the Advanced HTML editor extension, which you'll need to install. Remember to restart Dreamweaver after loading the new extension. Everything else in this tutorial assumes Dreamweaver MX 2004, though it should also work in plain old DMX too. We're only using server behaviors, so you won't have to worry about any hand coding of ASP.

Since we're using server behaviors, it's not a huge jump to create this example in PHP. I'll give you some tips for this at the end of the article.

Starting Off

First thing to do is set up our site in Dreamweaver. We'll be using the ASP/VB Script server model. You'll need to set up the site details to match your own server setup, be it a locally running IIS, a server on your network or a web server. You'll need to have a site set up so you can use the server behaviors, and so the Advanced HTML Editor knows where to store its files. (If you need help setting up your site, Molly Holzschlag has written a premium tutorial "Setting Up your Site")

In the root of your site create the following files (empty for now, but we'll need them to link to as we go through). Since we've set up using the ASP/VBScript server model, just right clicking in the site window and selecting New File should create a **.asp** file:

- news.asp
- login.asp
- news_admin.asp

- news_add.asp
- news_edit.asp
- main.css
- news.css

news.asp will be our main page, and will show the news to users of the site. login.asp, news_admin.asp, news_add.asp and news_edit.asp will form part of our new administration area. This will be password protected so that only the right people can update the news. The two CSS files will be used to style the pages we produce.

Setting up the Database

Now we have our files, it's time to ready our database. We're going to use two database tables for this application: cms_news and cms_users.

cms_news contains each of our news items.

Column	Type	Features
id	bigint	Primary key, identifier
news_item	vchar(1000)	
news_date	datetime	Default of getdate()
news_live	vchar(1)	Default value of 'n'

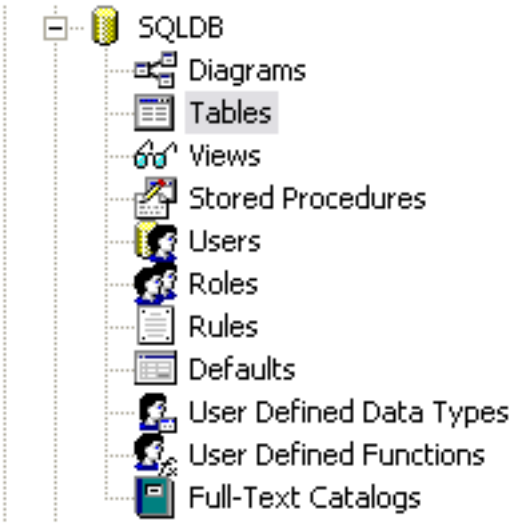
Each column stores a piece of information about our news item. **Id** is a unique number so we know which new item is which. **news_item** is the actual text of the item. **news_date** is the date it was added. **news_live** is a simple Y/N option to indicate if a news item is live to our site, or hidden from public view – maybe it's not fully updated yet.

cms_users is used to check if a user is valid when they log into the admin system

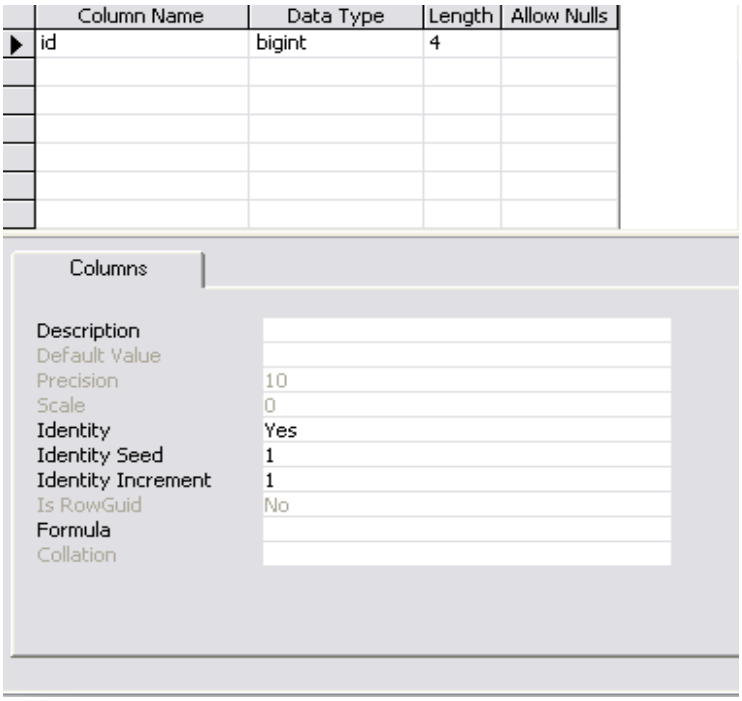
Column	*Type*	*Features*
id	bigint	Primary Key, identfier
cms_user	vchar(20)	
cms_pass	vchar(20)	

Create the tables using your favored tool (Some prefer the Enterprise Manager tool that comes with the paid-for editions of SQL Server, but many hosting packages come with an online tool you can use). If you're using Enterprise Manager, you should get a window like the one below:

If you highlight the SQL Server Group icon and right click, the option to add a New SQL Server Registration will be available. Using the wizard provided, fill in your SQL server details and it will appear underneath the SQL Server Group tree. Click on the databases folder and access your database (or create it if you haven't yet).



Right click on the tables icon, and select New Table. The new table dialog will open:



Fill the information in as provided in the tables above and then right click and choose the save option. Close the window.

I also added some test data; you can do this by right clicking on a table, choosing **Open Table > Return All Rows**, and then adding a new row. Alternatively you can use the Query Analyser tool to execute some SQL like that below:

```
INSERT INTO cms_news (news_item, news_live) VALUES ('A test item news item', 'y')
```

This inserts a new news item. Note that we only provided the **news_item** and **news_live** values, since the date is set automatically with the **getdate()** function. Similarly we can add a new user with the following SQL:

```
INSERT INTO cms_users (cms_user, cms_pass) VALUES ('admin', '6868')
```

Which inserts a new user called admin with a password of 6868.

We'll use this data to make sure everything is working testing the application.

Making the CSS pages

We'll start with the simplest bits of the system, our CSS files. We're using two, for reasons that I'll explain later. The CSS will be used to style our news items as shown in the screenshot below:

Open main.css in Dreamweaver MX and add the following style declarations:

```
.highlight{
color: blue;
font-weight:bold;
}

.sideblock{
float: right;
width: 200px;
background: #eee;
}

h1{
font-size: 1.2em;
}

h2{
font-size: 1.1em;
}

h3{
font-size: 1em;
}

body{
font-size: 72%;
font-family:Arial, Helvetica, sans-serif;
}
```

Open news.css and add the following styles:

```
.news {
      background-color: #66CCFF;
      padding: 2px;
      height: auto;
      width: 400px;
      margin-top: 10px;
      border: 1px solid #333399;
}

.news p{
margin: 4px;
padding: 0;
}
```

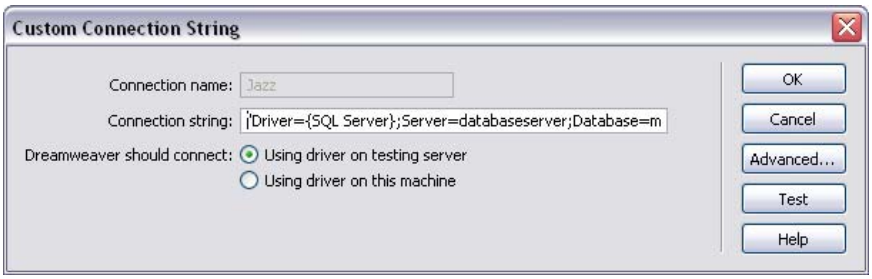
main.css contains the styles that will be applied to all pages (including the admin section), news.css is only going to be used on the news.asp page to break up the articles and ensure that paragraphs are correctly styled.

Making the login.asp Page

Open up login.asp and add a form containing two textareas and a submit button. Name the text areas username and password, and set the 2nd one to the password type so that it echoes asterisks rather than the actual password. Rename the form to **validate**. Set the page's title to News Admin Login. The page should look something like this:

For a real application may wish to add some text saying "you are not logged in, please log in now."

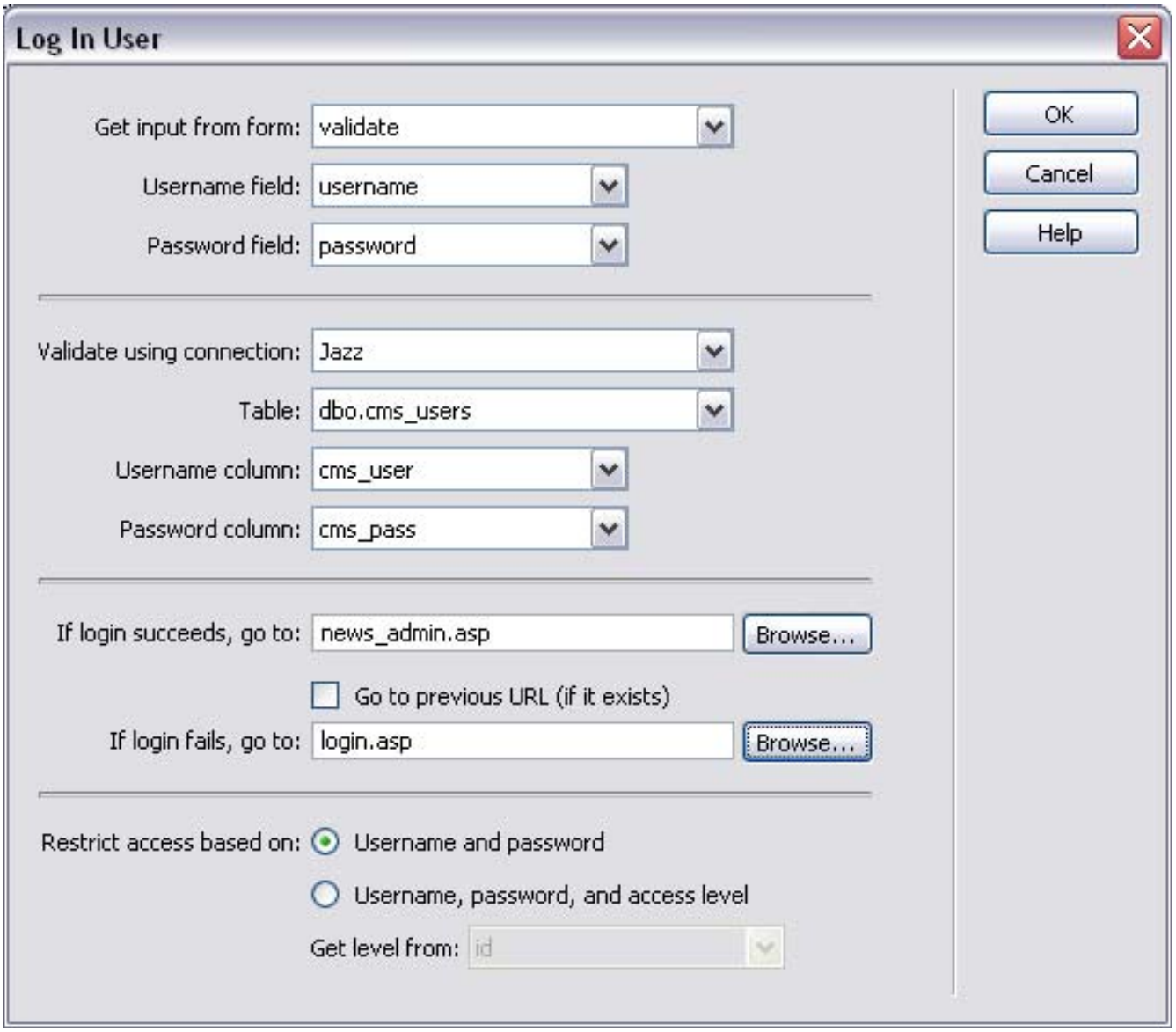
In the databases panel create a new connection by clicking on the **+** and adding your database details:



I called the connection Jazz. I used a custom connection string (the bits of information you'd need to fill with your own details are **databaseserver**, **mydatabase**, **username** and **mypassword**):

```
"Driver={SQL
Server};Server=databaseserver;Database=mydatabase;UID=username;Password=mypassword;"
```

Next we move to the Server Behaviors panel. Select **+** > **User Authentication** > **Login-In User**. A dialog box will pop up:

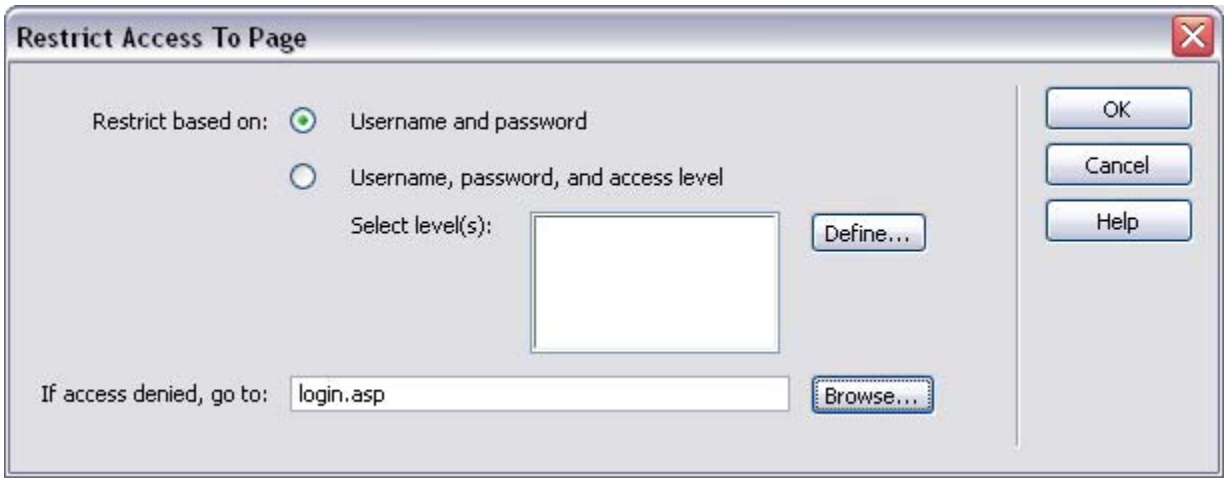


The behavior should automatically detect which form fields to grab the login information from, but if it doesn't happen like that, make sure the username and password are the right way round. We choose our database to validate the user against, the database table in which the user data is held and the columns for the username and password. Again, make sure the dropdowns are set to the correct column. We also set the pages to go to if we succeed or fail to login. Click OK to close the dialog.

The login page is now finished, so you can close that as well. Pretty easy, so far, huh? It doesn't get any harder, either!

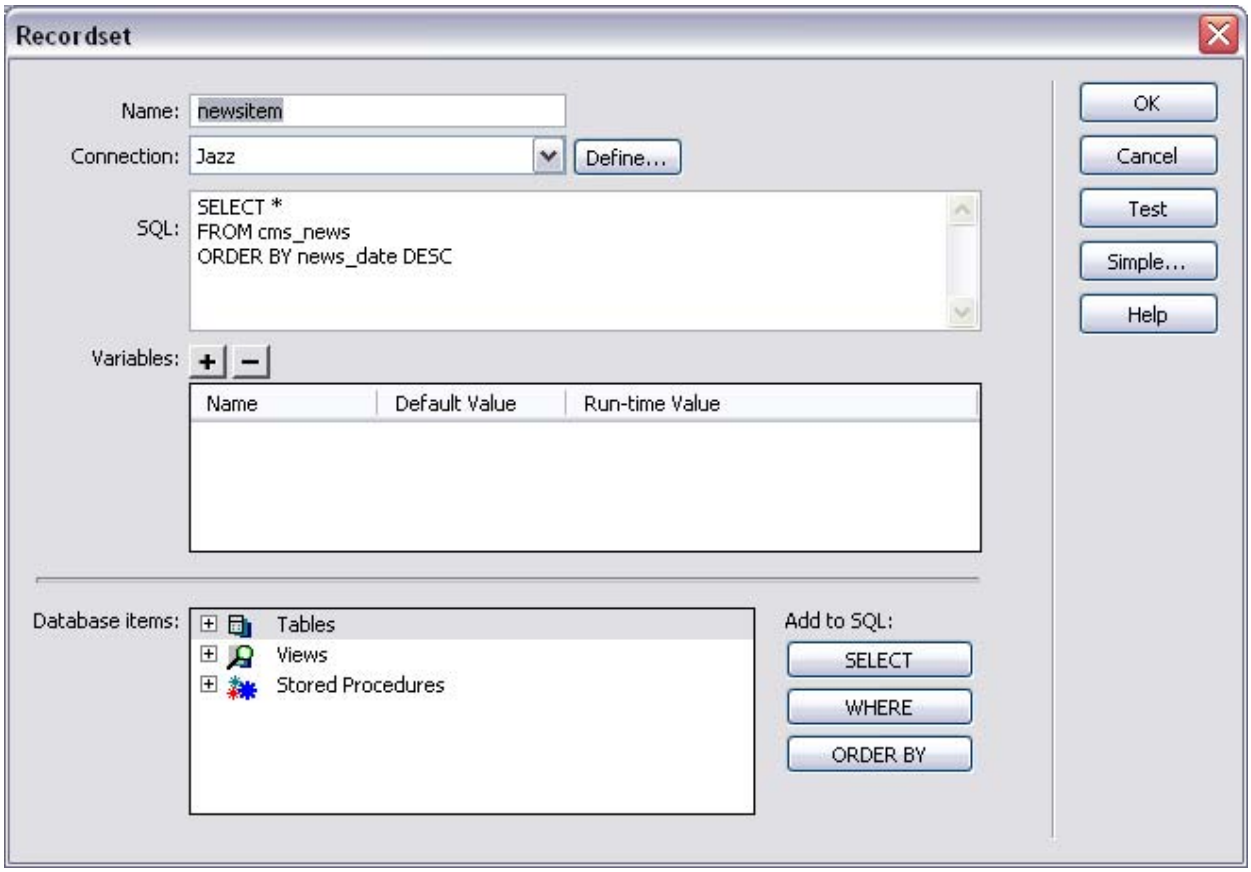
Creating the news_admin.asp page

Open up **news_admin.asp** and set the page title to News Admin and add a heading so people know where they are. First we'll restrict access to the page. In the server Behaviors panel got to **+** > **User Authentication** > **Restrict Access to Page**. The dialog is pretty simple:



We simply restrict access based on username and password, and set the page to boot unwanted users out to login.asp.

Now we're going to create a recordset to display all our news items. In the server behavior panel hit **+** > **Recordset**. Click the **advanced** button and fill in the dialog as below:



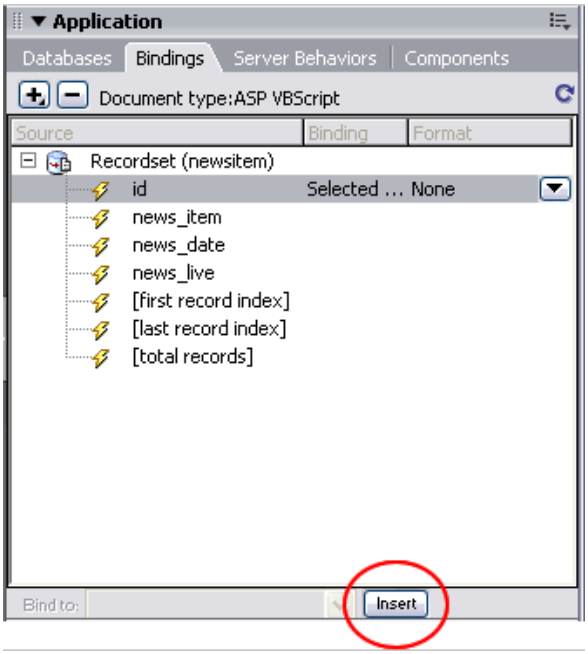
The SQL query is:

```
SELECT *
FROM cms_news
ORDER BY news_date DESC
```

This query grabs all the data from the **cms_news** table and orders it by the date it was added, newest first. Hit the **test** button to check that the query works. If you added the test data, you should get one row containing our test news item.

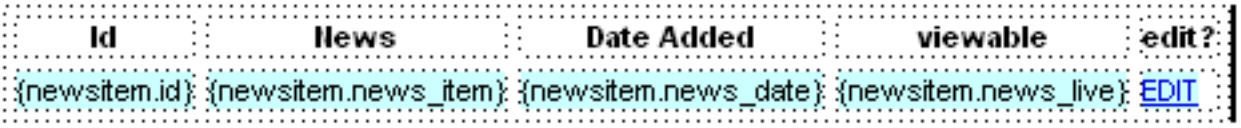
Attach our **main.css** stylesheet to the page. Next, insert a new table into the page. It'll need 5 columns and 2 rows. In the table headers put **id**, **news**, **date added**, **viewable** and **edit?**

Move to the 2nd row and make sure the cursor is inside the table cell. Then open up the **Bindings** area of the **Application panel**. Highlight the id column of the newsitem recordset and hit the insert button (at the base of the panel, see screenshot).



Move to the next cell and repeat for each of id, news_item, news_date, and news_live. For the date, set the format dropdown to the **Date/Time** > **17 January 2000** option to format it.

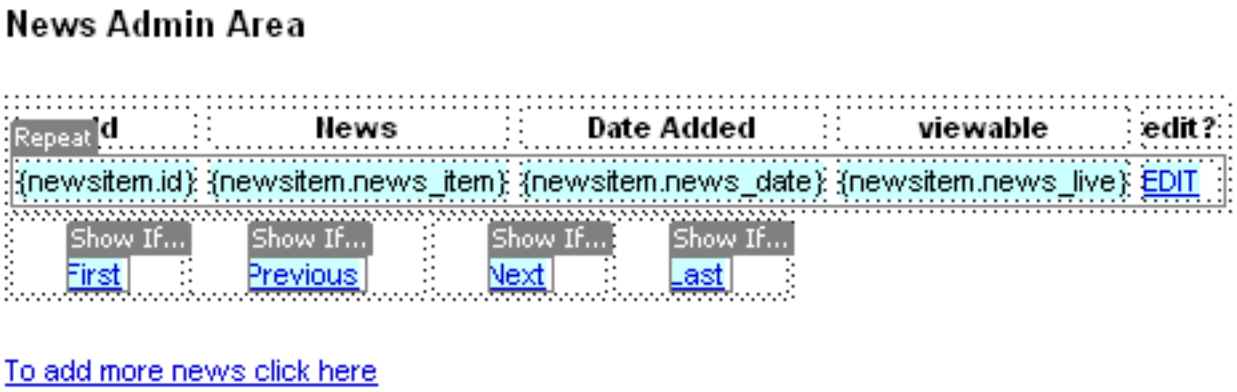
In the last cell add the text EDIT. Highlight this text and click on the browse for file link option in the parameters panel. Select the file news_edit.asp. Then hit the parameters button. Name the parameter id, and set it to take its value from the id column of the newsitem recordset (hit the lightning bolt icon to access the recordset). We'll use this link to tell the edit page which record we want to edit. The table should now look something like this:



Next, we want to make sure we can display all the items in the news. Highlight the row containing the bound data, and then add a **Repeat Region** server behavior. Choose the **newsitem** recordset and **5 records at a time** and hit OK.

Move the cursor below the table and, in the **insert panel**, select the **application tab** and hit the **recordset paging** button. Choose the **newitem** recordset and the **text** option.

Finally, add a link at the bottom of the page with the text "To add a new item click here" and link it to news_add.asp. The finished page should look something like this:



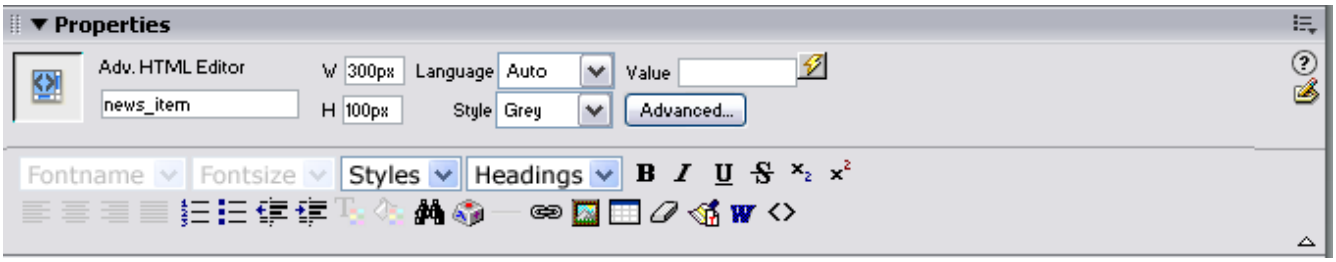
Making the news_add.asp Page

Now we're ready to play with our Advanced HTML editor extension. Open up the **news_add.asp** page , set the page title to "Add News", add a suitable heading and attach the **main.css** stylesheet using the link method.

In the Insert panel click the form elements tab. Add a form to the page and call it addnews. Now, at the very end of the forms section of the insert panel is our Advanced HTML Editor. Click the button and the editor will appear on the page. You'll get a warning notifying you that several folders have been added to your site, and not to forget to upload them. If you forget to upload them, then the news application won't work!

These folders contain the scripts used to control the HTML editor. If you look in the **Files Panel** you should see new folders called **AdvHTML_Images**, **AdvHTML_Popups** and **ScriptLibrary**.

If you click on the editor and have the Properties panel open, you'll see all the options available for the editor:

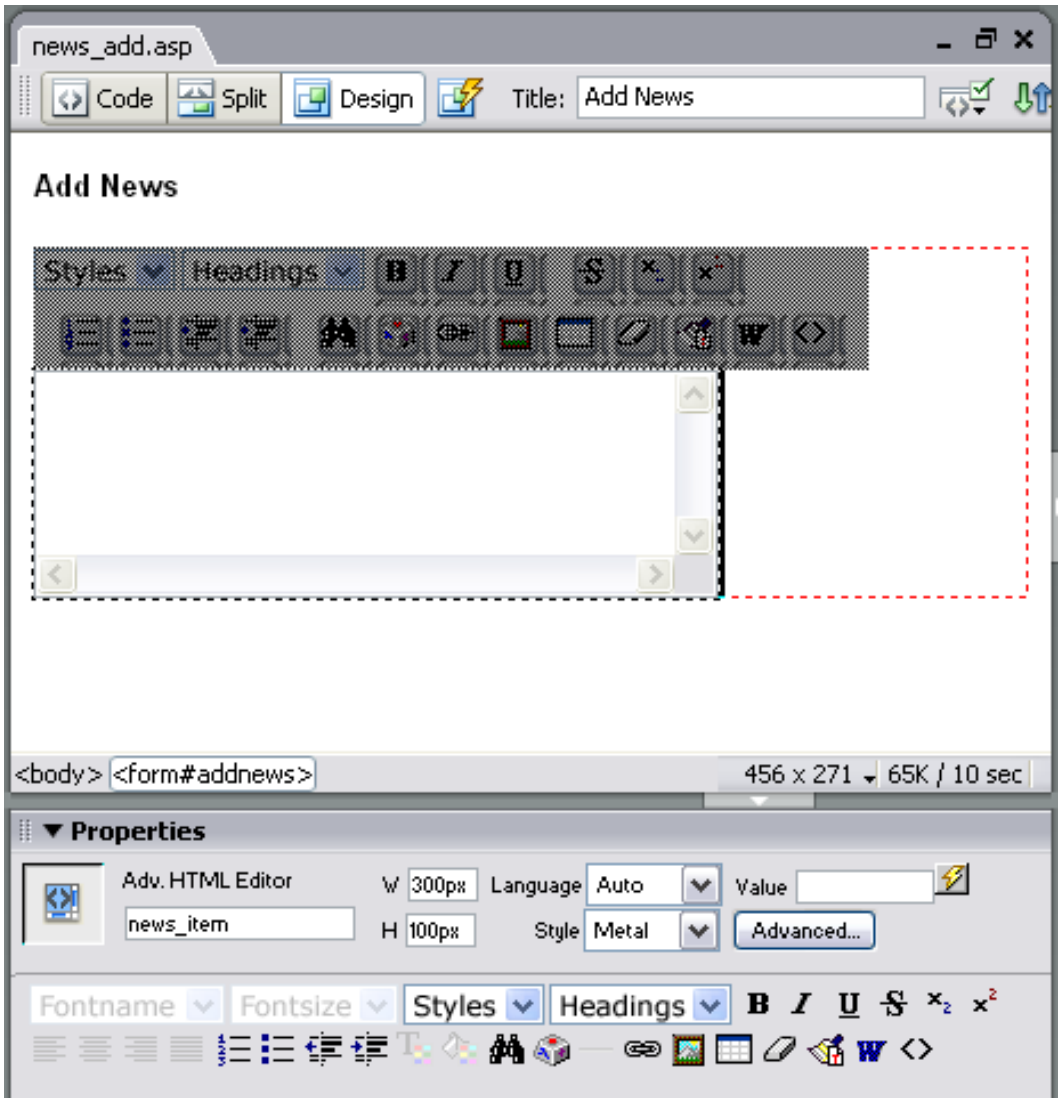


The top of the panel are the items form element name, width, height, language, style and initial value. These can be used to set the visual appearance of the editor to fit in with your site. There's also an advanced button, but we'll ignore this for now.

I'm going to use the metal style, which gives a clean metallic look. I'll also set the size of the editor to 300px by 100px, since we want our news items to be relatively small. I've set the name of the editor to **news_item**.

At the bottom of the properties panel are the various HTML options you can allow users to set in the editor. Initially they can add anything, but you want to be careful about this. Consider how each of the options could be used to break the look of you site. For example, if you've used Verdana all over and users have the option to change the font to Times, then things are going to start looking messy.

To disable an option in the editor, just click the relevant area of the properties panel. I'm going to disable the font-name, font-size, alignment, horizontal rules, text color and background color options (see screenshot below). If the users want to change things they'll have to make use of the styles I've defined in **main.css**.



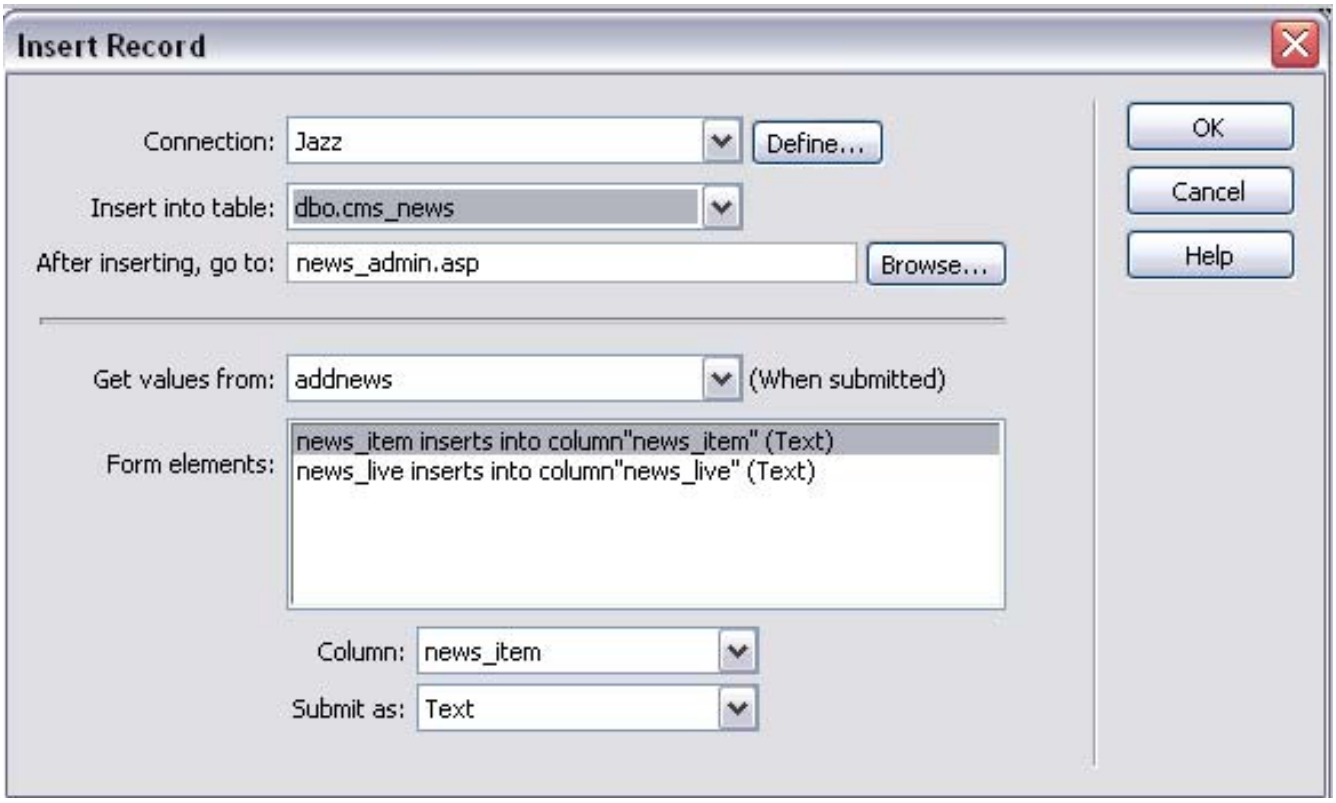
If I wanted I could also disable lists, indenting, searching, the character map, hyperlinks, images, tables, subscript, superscript, underline, strikethrough, bold, italic, the option to cleanup formatting , cleanup the font tags, word cleanup and switch to HTML view. There are a lot of options, so I'd recommend having a play with them to see which are most suitable for your site.

Standards compliance-minded among you will be happy to learn that the editor uses strong and em tags for its bold and italic. It also degrades to a normal textarea in browsers that it's not compatible with (it only works in Internet Explorer 5.5 and up at the moment)

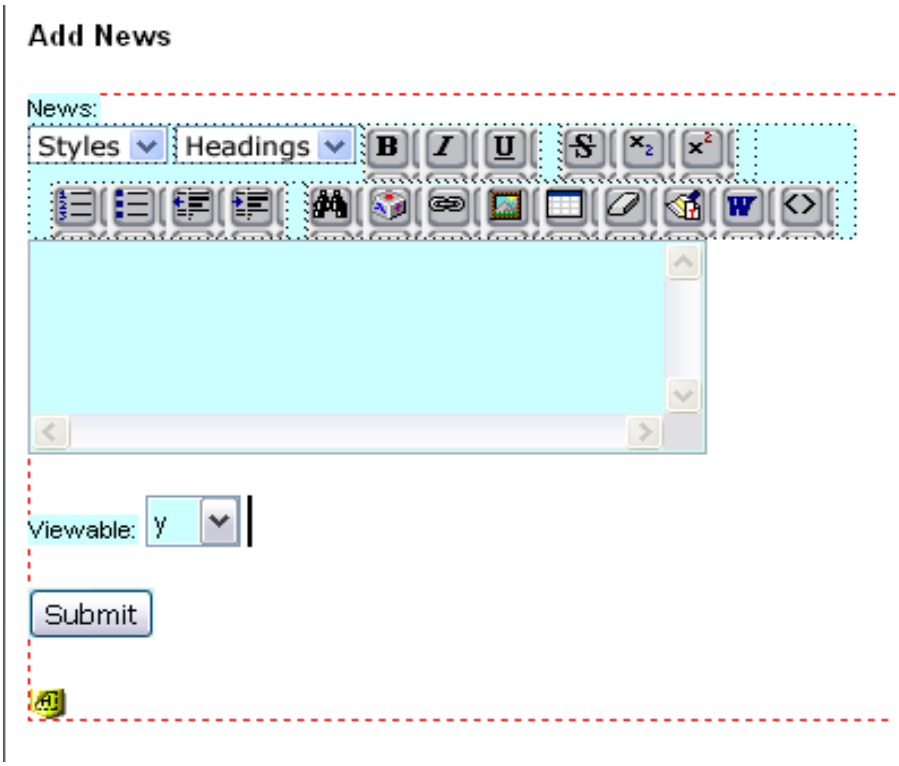
Now that we have the editor sorted, we'll add a dropdown box with the label viewable and the options **y** and **n**, and a **submit** button.

Next, go to the **Server Behaviors panel** and add another **Restrict Access to Page** behavior, exactly the same as before.

Now, we're going to add an insert Record Behavior, so that we add our news item to the database when we submit the form. Click + > Insert Record, and fill the dialog in with the details shown below:



Our page should now look something like this:



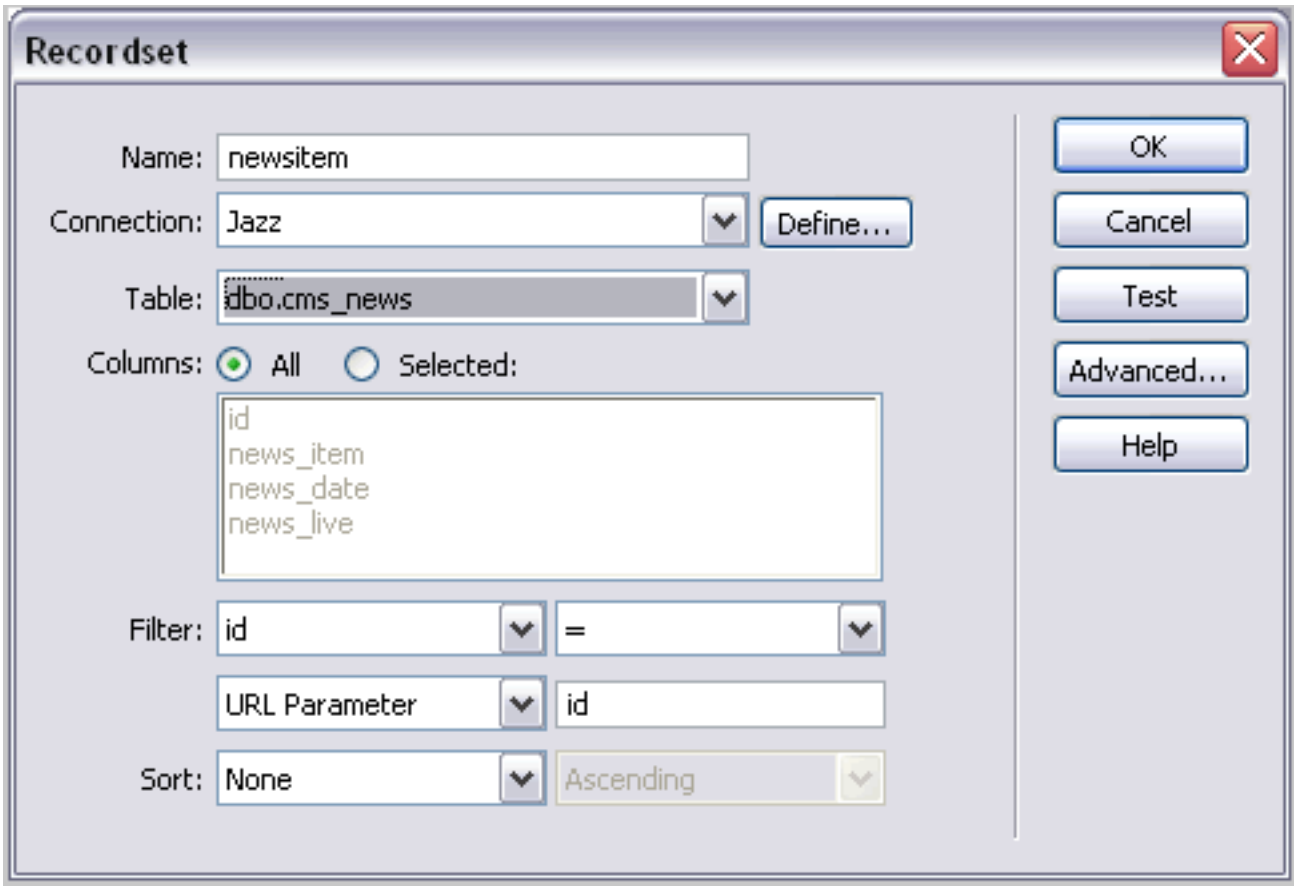
If everything has gone according to plan, when you test the page you should be able to add new items to the news, and they will appear on the list of our main, news_admin.asp page. Note that we don't have to add the date, since the table's default will set it automatically when we insert the record.

Creating the news_edit.asp Page

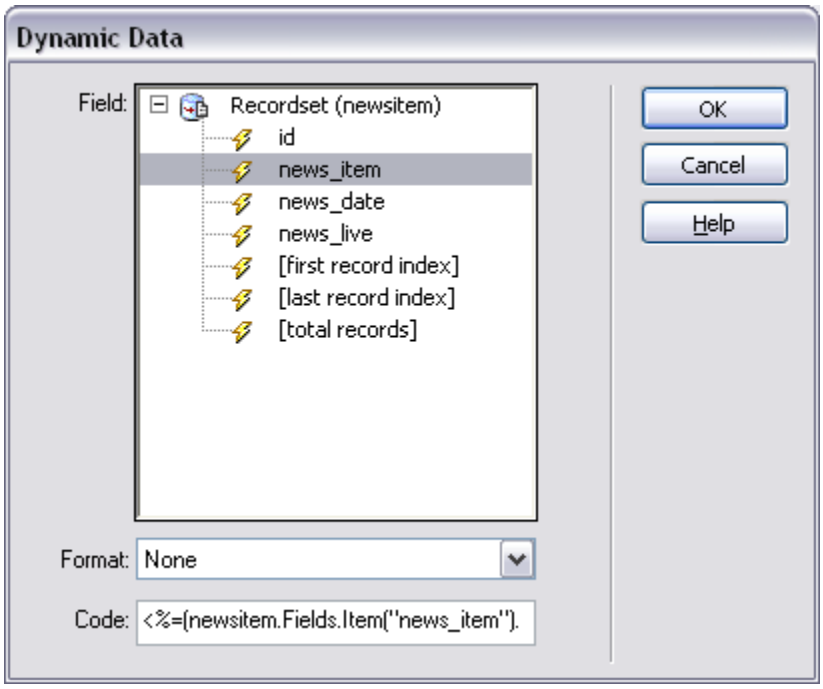
Okay, so we have the ability to add the news to our database, now we need to be able to edit it in case things go wrong (like that typo you missed, or some additional information).

First we have to grab the record we are editing from the database. To do this we create a recordset using **Server Behaviors** > **+** > **Recordset(Query)**

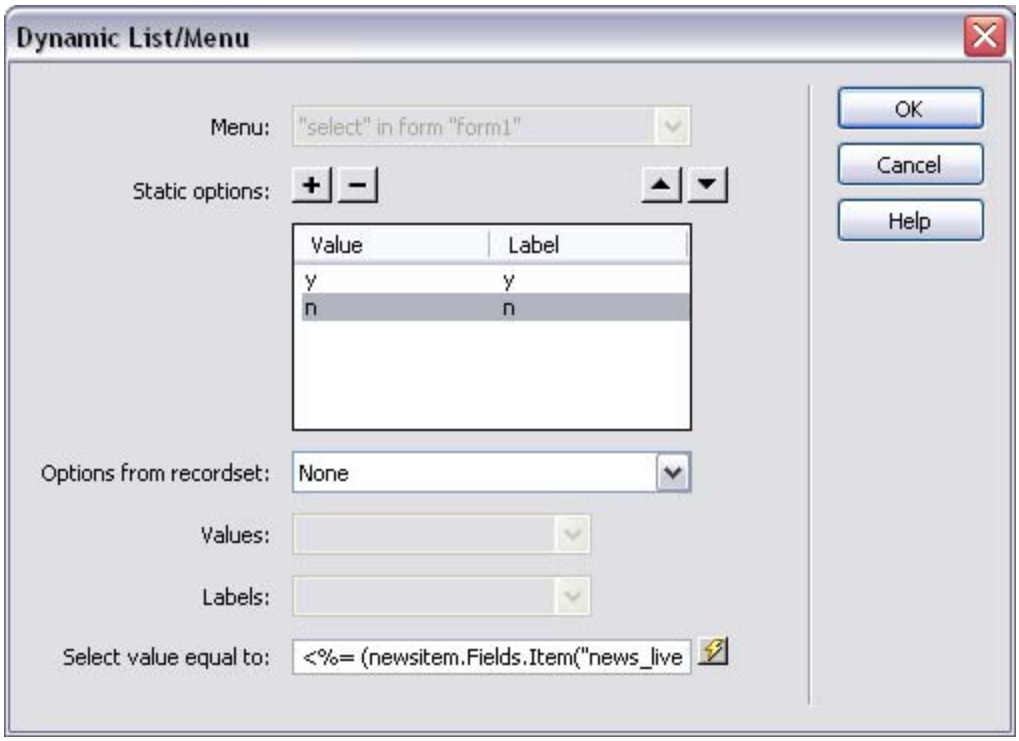
We're just going to grab every field from cms_news, but filter it by the **id** sent in the querystring from news_admin.asp. That way we only get the unique record we need. We'll call our recordset **newsitem**. The dialog should look like this:



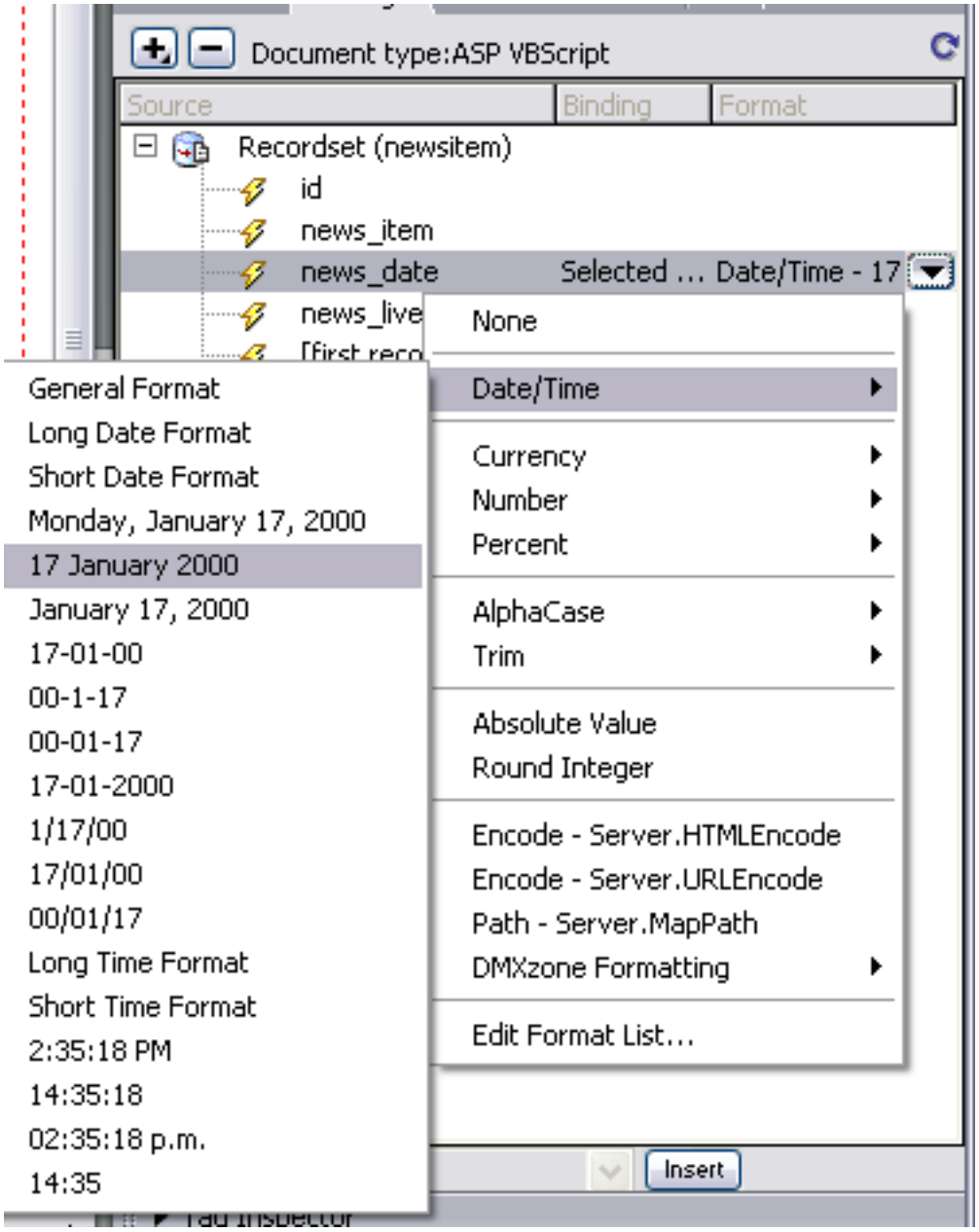
The HTML of the edit page is created much like the add page. The form contains an identical **Advanced HTML Editor** and other form elements. Unlike the **add** form, we need to set the form elements up to have default values. To add the default value to the Editor, click on the dynamic lightning flash by the value field. Select the **newitem** recordset and the **news_item** column, like this:



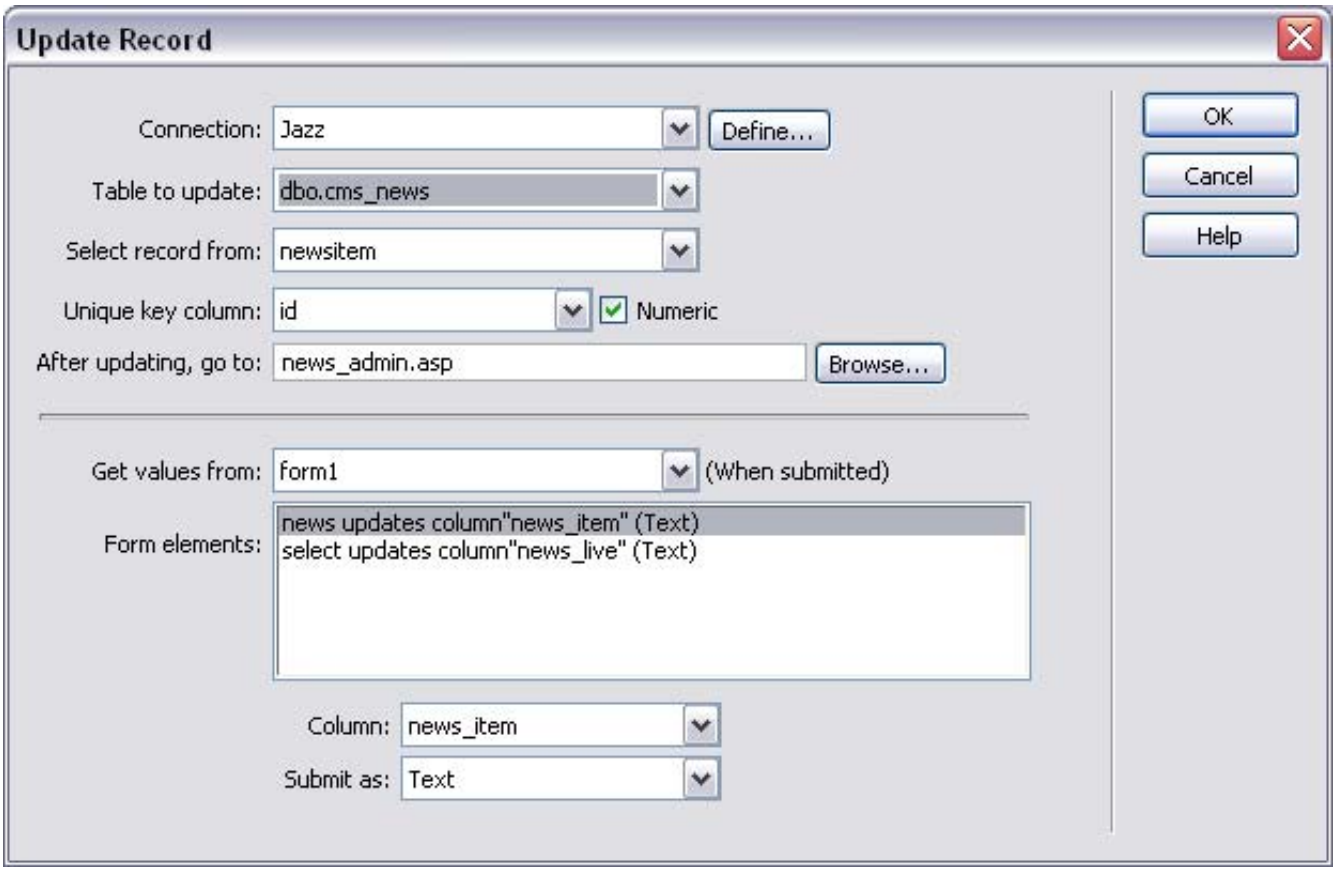
For the dropdown list, click the **Dynamic** button. Add the options **y** and **n**, and set **Select Value Equal** to be the **news_live** column of our recordset.



I've also added a brief "posted on: " text, and inserted the added the **news_date** column from our recordset after it. I've formatted the date in a more human recognizable way too, using the formatting option available from the bindings dialog (see below).



To get the record to change when we submit, we add an **Update Record** server behavior. Click **Server Behaviors** > **+** > **Update Record**.



Fill in the form with the details as shown above. Select the correct database table and recordset to be used, and make sure the id is set as the unique key column. Be careful that your editor and select box are married up with the correct database columns (otherwise the update might try to insert the whole text into the space for a y/n answer!).

Finally, we again add the Restrict Access server behavior to keep out unwanted guests.

Creating the news.asp page

Last, but not least, we need to create the news page that will be visible to the general public. Open up **new.asp** and attach both stylesheets to it. Go to **Server Behaviors** > **+** > **Recordset** and go to the advanced option. Call the recordset news, and use the **Jazz** connection. In the window add the following SQL:

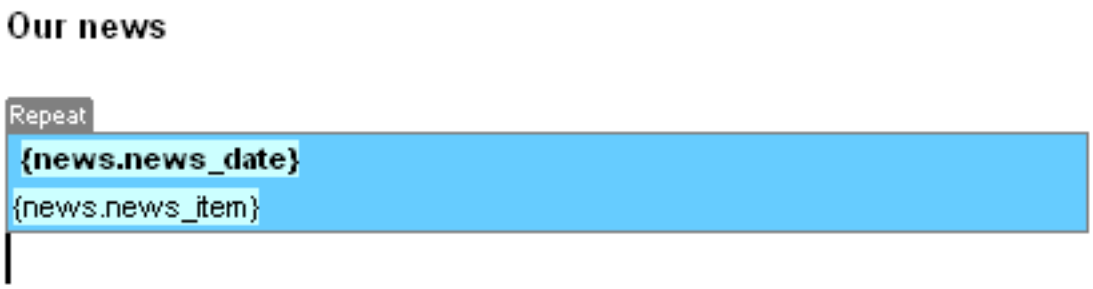
```
SELECT TOP 10 news_item, news_date
FROM c31.cms_news
WHERE news_live='y'
ORDER BY news_date DESC
```

This SQL grabs the **news_item** and **news_date** columns for the first 10 records in the database, but only if they are set to be live. It then orders them by the date they were added, highest first.

Next, we need to lay out our page. Fortunately we've kept things quite elegant. First add a main heading for the page (I used "Our News"). Underneath this we add a ** ** tag. This we set to have the class news (if your stylesheets are attached properly this area should now go blue).

Inside the div, put a paragraph with the text date in bold and below that a piece of text with no Format option with the text news. It's important this text is not formatted, since all the formatting for our news is inside the database. Highlight the date text and go to the **Bindings** panel. Select **news_date** from the news recordset and click the insert button. The text **{news.news_date}** should replace your dummy text in design view. I also changed the format to something more human-readable using the **Bindings Panel**'s **Format** dropdown. Highlight the news text and insert the **news_item** column.

Finally, highlight the entire ** ** and add a **Repeat Region Server Behavior**. In the dialog that appears, select show all records (since we've limited the number by our SQL anyway). The final page should look like this:



Finishing Off

The basic news system is now finished. You can add and update news items using a WYSIWYG tool. Upload your files and you should be able to test the pages. Try styling a paragraph in the **sideblock** class, and you'll see some of the more advanced formatting that can be achieved by combining CSS styles and the Advanced HTML Editor.

Obviously there are a few tweaks you could make to the system. At the moment new users are added directly into the database. You'd need to add a set of pages for creating new users. Unsurprisingly, this is very similar to the news insertion page, except with username and password fields.

Doing it all in PHP/MySQL

Since Advanced HTML Editor works with any type of form, you can also integrate it into any PHP application you are creating. To convert the example application into PHP, the main difference will be the database tables (since everything else is done via server behaviors). MySQL has a slightly different implementation of SQL to SQL Server, so we would create our tables differently.

The database **cms_news** would be created with the following SQL (you can use your favourite admin tool, I prefer PHPMyAdmin):

```
CREATE TABLE cms_news (
id BIGINT NOT NULL AUTO_INCREMENT ,
news_item TEXT NOT NULL ,
news_date DATE NOT NULL DEFAULT CURDATE(),
news_live CHAR( 1 ) DEFAULT 'n' NOT NULL ,
PRIMARY KEY ( id )
);
```

and cms_users would be created with the following SQL:

```
CREATE TABLE cms_users (
 id BIGINT NOT NULL AUTO_INCREMENT,
 cms_user VARCHAR(20) NOT NULL,
 cms_pass VARCHAR(20) NOT NULL,
 PRIMARY KEY (id)
);
```

Also **MySQL** uses a **LIMIT** clause after a query to limit the number of rows returned rather than **TOP**. Other than that – using Server Behaviors and Advanced HTML Editor, we've made a working password-restricted content management system, with an intuitive word processor-like interface in record time – and, once you own the Advanced HTML Editor, you can include it in any number of forms on any number of sites.

As Jazz fans say: nice!