

Chapter 8

Creating Visitor Accounts Through Username Validation

Over the last few years, there has been a substantial increase in the number of sites that are allowing visitors to create and maintain their own user accounts. Most of



these sites allow their visitors to browse through the “public” sections of their site, but require the creation of an individual account when it comes time to order products, contribute content, or view pages that the organization just doesn’t want available to the general public. In most cases, the visitor is asked to provide the appropriate demographic information, select a username, and then choose a password. This information is stored in a database, and registered users can return to the site at any time, log in to their accounts, and surf the site in its entirety.

Registering visitors has the added benefit of providing organizations with the ability to gain a more detailed understanding of who is visiting the site and to develop custom content that focuses on the individual visitor. For instance, some sites currently track the search terms that are used to locate products. This information is then used to customize the site’s content to meet the perceived needs of the visitor during the next visit. Whether it be for purchasing products, searching for a job, or maybe just playing a few games online, many sites are customizing content for their visitors based upon their activities while logged in to an individual account.

From a Web developer’s point of view, adding this functionality to a site has the added benefit of providing you with the ability to collect valuable information about your visitors. Whether it be a shipping address so you can quickly deliver products or an email address so you can notify customers of upcoming promotions, the more you know about your visitors, the more likely you can develop content that suits their needs. For instance, if you track the expressions entered into a product search box, you might be able to better understand what items your customer wanted to see and be able to add products that you might not have previously offered.

To assist you in adding individual accounts to your site, UltraDev has several built-in server behaviors that speed up the development process. In particular, this chapter focuses on three things:

- Enabling visitors to create individual accounts
- Protecting pages from unauthenticated visitors
- Testing your new pages

Enabling Visitors to Create User Accounts

The first step in allowing users to access password-protected areas of your site is to develop a process that allows each user to create an account. The most common way to

do this with UltraDev is to create a signup page including a form that allows each user to enter the necessary information and choose a username and password. When the form is submitted, the contents are placed into a database where they can be referenced for validation when the visitor returns and attempts to log in again.

The site must also include several additional elements that help returning visitors access their accounts and the secured sections of the site. First, the site needs links that allow the user to log in and log out. Ideally, the site should be able to detect whether an individual has logged in or not and should then only display the Login link to visitors who haven't yet signed in. Likewise, authenticated users should only see the Logout link.

The last element a site needs for minimal functionality is a login page for returning visitors. This page usually includes a form that accepts a username and password and submits the information to the Web server. The user's credentials are then checked against the values stored in the database and the user is validated or denied access based on whether or not the values match.

Luckily, UltraDev makes the process of adding user accounts relatively easy. From the menu bar you can quickly add forms, text fields, and buttons to your pages. In addition, built-in server behaviors help you add new users to your database and validate credentials supplied by returning visitors.

Adding Dynamic Links for Creating an Account, Logging In, and Logging Out

The first step to adding user accounts and allowing returning visitors to log in and log out is to provide appropriate links. As I mentioned before, the goal is to create dynamic links that are only visible at the appropriate time. To accomplish this, we can use UltraDev's Show Region behavior to display only the Login and Create New Account links when the user has not already logged in. Likewise, the same behavior can be used to display the Logout link only when the user has successfully logged in.

Exercise 8.1 Adding Dynamic Links

1. Open UltraDev. Open the `nrfdefault.dwt` page located in the Templates folder of the InsideUD4 site.
2. Open the Server Behaviors panel by selecting Window/Server Behaviors from the menu bar.
3. Click the plus sign on the Server Behaviors panel and select Recordset (Query) from the drop-down menu.

4. In the Recordset dialog box, type **rsLogin** in the Name field. Select the `connSales_Database` connection from the Connection dropdown. If you do not see the `connSales_Database` connection, please refer to Exercise 7.1.
5. Click the Advanced button. The Advanced view of the Recordset dialog box, shown in Figure 8.1, provides you with the ability to create your own custom SQL queries. Notice that UltraDev has already begun a SQL query for us indicating that the query should select everything (indicated by the asterisk following SELECT in the SQL panel) from the `tbCustomers` table.

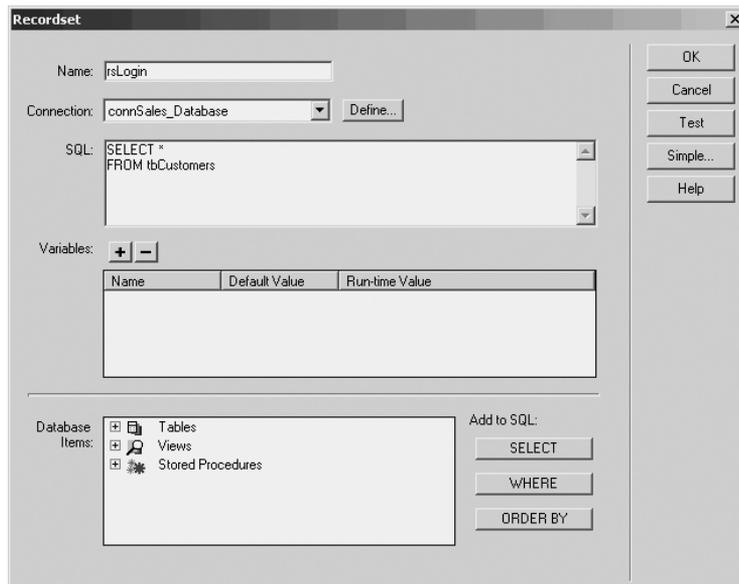


Figure 8.1 The advanced view of the Recordset dialog box allows you to define custom SQL queries.

6. Place the insertion point in the SQL panel after the word `tbCustomers`.
7. In the Database Items panel, click the plus symbol next to Tables. Click the plus symbol next to `tbCustomers`. You should now see all the columns (fields) available in your database.
8. Highlight the `CustomerID` column and click the WHERE button. Notice that UltraDev automatically adds a qualifier to your SQL query.
9. Highlight the `Password` column and click the WHERE button.
10. Fill in the rest of the SQL query as shown in Figure 8.2. To add the variables, simply click the plus sign and type the variable entries.

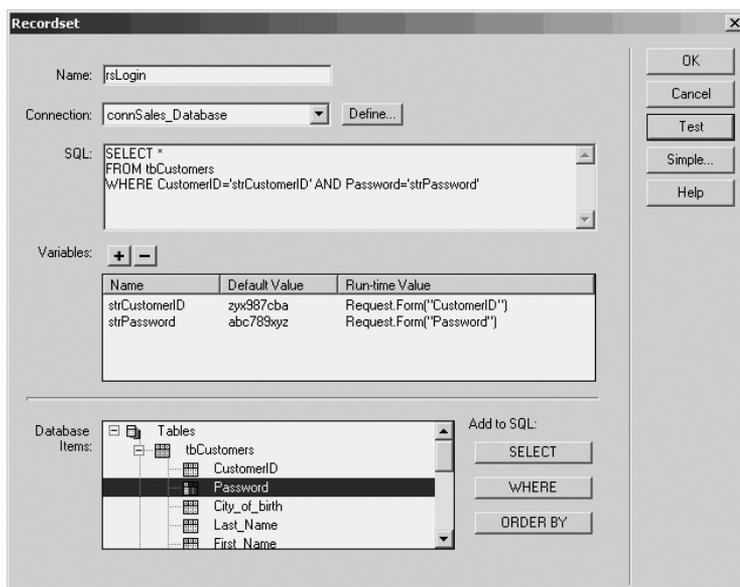


Figure 8.2 Your SQL query and variables should look like this.

11. Click the Test button. This query simply takes whatever values are stored in the CustomerID and Password text fields of a form and compares them to the CustomerID and Password fields in the database. Because there are no entries in the database that match the default customer ID or password, the test query should return no data.



Caution

Selecting Default Values for Your SQL Queries Whenever you create a custom SQL Query in UltraDev, you have the ability to test the query with default values to ensure that the query behaves properly. However, if you choose default values that are common, you could create a potential security problem within your site.

For instance, suppose you enter a default value of abc for the CustomerID and 123 for the Password. If a visitor comes along and creates a user account that uses abc and 123 for his username and password, everyone will now be able to access your account without supplying a username and password because of the default values that exist in the database.

Therefore, when selecting default values, be sure to enter values that are exotic and that no visitor is likely to choose. When you are done testing your pages, you can either leave the exotic default values or change the default values to input information from a session variable.

12. Click OK to close the Test SQL Statement dialog box and click OK in the Recordset dialog box. UltraDev now displays the rsLogin recordset in the Server Behaviors panel.

13. Switch to the Code view by clicking the Show Code View button.

14. Scroll to the top of the code and find the line that reads

```
rsLogin__strCustomerID = "Session("MM_Username")"
```

Remove the outer quotes from around the session information and change this line to read as follows:

```
rsLogin__strCustomerID = Session("MM_Username")
```

15. A few lines down, find the line that reads

```
rsLogin__strPassword = "Session("MM_Password")"
```

Remove the quotes from around this session information as well. The line should now read

```
rsLogin__strPassword = Session("MM_Password")
```

The default variables that were initially entered in the rsLogin recordset tell the recordset to check whether the user has previously logged in and created a session variable. If he has logged in, the rsLogin recordset contains the appropriate record from the tbCustomers table.

The only problem with this arises from the fact that UltraDev places quotes around the session code, which results in an empty recordset. To resolve the issue, you can simply remove the quotes in the Code view.

Keep in mind, however, that if you open the recordset on any particular page, UltraDev will “fix” the problem. To undo UltraDev’s fix, just view the code again and retype the session information.

16. Switch back to the Design view. Notice that the rsLogin recordset now has an exclamation point next to it because you removed the quotation marks from the session variable information. Don’t worry about this; the recordset still functions properly.



Note

Adding Recordsets and Server Behaviors to a Template When you add a recordset or server behavior to a template, all pages that are built from the template will also inherit the recordset and server behavior that are associated with the template at the time. Adding these elements to your templates is a quick and easy way to add a dynamic behavior to all the pages in your site.

Keep in mind, however, that there are times when you won’t want to apply a server behavior to all your pages. For instance, suppose you want to limit access to a certain area of your site to users who have entered a valid username and password. Applying these limitations to a template could result in protecting pages that you wanted to remain available to all visitors. In this case, you might consider creating a second template for use with those pages that should be password protected.

17. Place the insertion point in the empty cell below the View Cart button on the left side of the template. In the cell, type **Create Account**.
18. Highlight the Create Account text and type **http://localhost/insideud4/newuser.asp** in the Link field of the Property inspector. Press Enter.
19. With the Create Account text highlighted, click the plus sign on the Server Behaviors panel and select Show Region/Show Region If Recordset Is Empty. From the dialog box shown in Figure 8.3, select the rsLogin recordset and click OK. This behavior has the effect of only showing the Create Account link if the user has not logged on using a valid username and password.



Figure 8.3 Select the rsLogin recordset.

20. Place the insertion point after the Create Account text and press the Tab key on your keyboard to create a new cell. With the insertion point in the new cell, click the Align Center button on the Property inspector.
21. Type **Login** in the cell.
22. Highlight the Login text and type **http://localhost/insideud4/login.asp** in the Link field of the Property inspector. Press Enter.
23. With the Login text highlighted, click the plus sign on the Server Behaviors panel and select Show Region/Show Region If Recordset Is Empty. From the dialog box, select the rsLogin recordset and click OK. Again, this behavior has the effect of showing the Login link only if the user has not entered a valid username and password.
24. Place the insertion point after the Login text and press the Tab key on your keyboard to create a new cell. With the insertion point in the new cell, press the Align Center button on the Property inspector.
25. Type **Logout** in the cell.
26. Highlight the Logout text and type **http://localhost/insideud4/logout.asp** in the Link field of the Property inspector. Press Enter.
27. With the Logout text highlighted, click the plus sign on the Server Behaviors panel and choose Show Region/Show Region If Recordset Is Not Empty.

28. From the dialog box, select the rsLogin recordset and click OK. This behavior displays the Logout text only after a user has entered a valid username and password. As shown in Figure 8.4, you should now have login and logout links that have server behaviors.

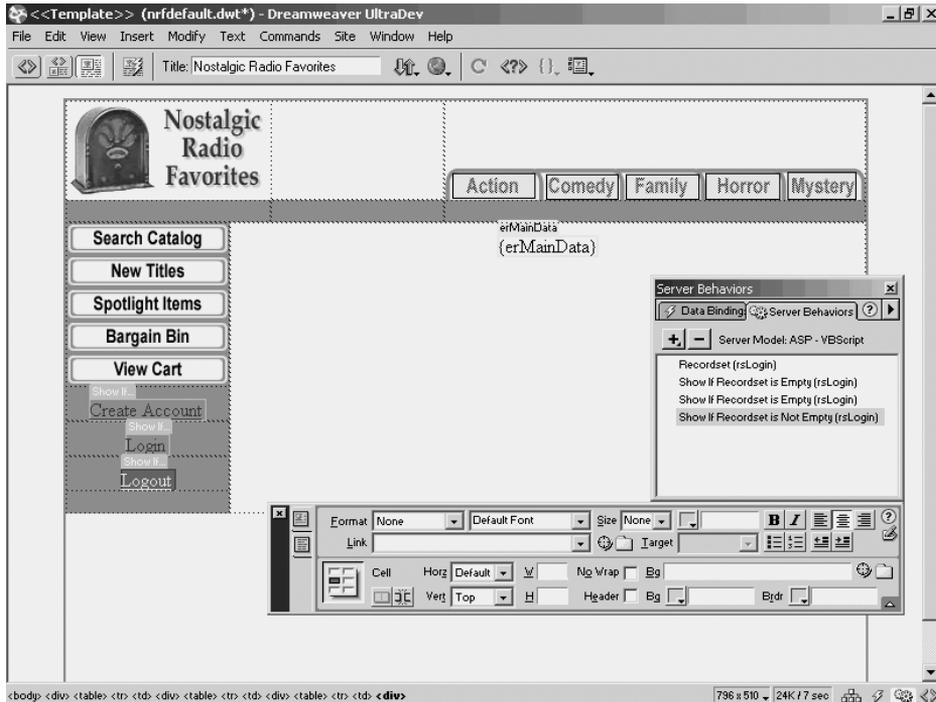


Figure 8.4 Your template with login and logout links.

29. Save your template. When asked whether dependent pages should be updated, respond Yes.
30. Close the Update Pages dialog box and close the nrfdefault.dwt template.
31. Do not close UltraDev.



Note

Testing the Additions to Your Pages With the links added to your pages, you have taken the first step toward allowing users to create their own user accounts. At this point, however, the links are pointing to pages that don't exist so don't worry about testing the new additions to your page just yet. When you get to the end of the chapter, you'll have the opportunity to see everything in action and test the site's added functionality.

Creating a New User Signup Form

A couple of years ago, the university where I work began developing a Web-based employment database. When we considered what information we needed to collect from our job seekers, we came up with an extensive list of items ranging from their personal information (such as name and address) and demographic information (such as gender, race, and age) to their employment history (such as degrees earned and relevant experience). Although a lot of the data seemed irrelevant to the job search process, we decided to collect as much information as possible and then filter the information that was visible to employers to include only data that was relevant to a job search.

A few months after implementing the database, however, we discovered that a lot of the information that was unrelated to the individual's job search was helpful to us in understanding who our job seekers were and what types of employment opportunities were being sought. With this information in mind, we were able to market our database to employers who were more likely to offer positions that met the needs of our job seekers.

The point here is that one of the biggest difficulties in developing a new user signup form is to determine what information you should collect from your visitors. Not only is it imperative that you think about information that is absolutely necessary, but you must also break out your crystal ball and predict what information could conceivably be needed in the future.



Caution

Balancing What You Want with What You Need When developing a new user form, you should always keep in mind that no one likes to fill out a form that requests too much information and takes too long to complete. If the form takes each visitor more than a couple of minutes to fill out, you might reconsider some of the data you are requiring. The last thing you want to do is drive visitors away from your site because they don't want to take the time to create an account.

Starting with the Template

As with the other pages in your site, you can develop your new user account page from scratch or save time by using a previously created template. Pages that include simple forms that interact with your database should cause no problems when built from a template. Keep in mind, however, that using a template restricts access to the Head section of your code. Because of this, any forms that use JavaScript behaviors (such as checking for required fields) will have to be disconnected from the template before the behaviors can be applied.

**Note**

Learning More About Templates If you are interested in learning more about the potential limitations of templates, check out this TechNote on the Dreamweaver Support Pages:

http://www.macromedia.com/support/dreamweaver/ts/documents/behavior_templates.htm

Exercise 8.2 Creating a New User Account Page

1. Open UltraDev, if it isn't already open, and select File/New from Template from the menu bar.
2. From the Select Template dialog box, choose the InsideUD4 site and select nrfddefault from the available templates. Click Select. As shown in Figure 8.5, UltraDev creates a new page based on the previously developed template.

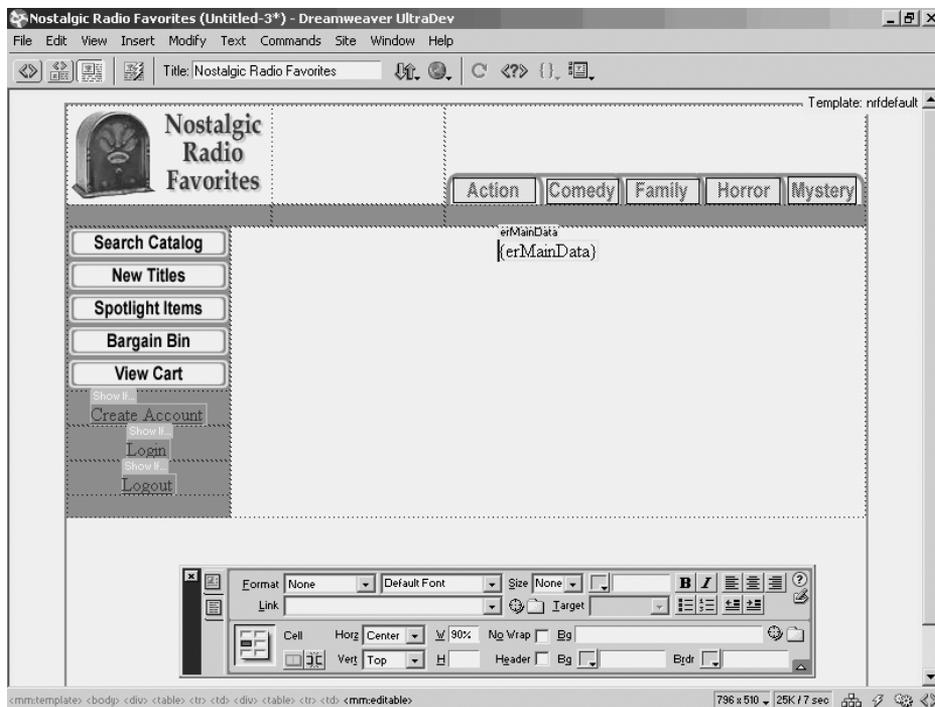


Figure 8.5 A new page has been created from the template.

3. In the erMainData editable region, highlight the {erMainData} text and delete it. In the editable region, type **Create New User Account**.

4. Highlight the text and select Heading 1 from the Format dropdown on the Property inspector. In addition, select Arial, Helvetica, sans serif as the font style and 3 for the font size. Click the Bold button.
5. Place the insertion point at the end of the line and press Enter. From the menu bar, select Modify/Templates/Detach from Template. As shown in Figure 8.6, the editable regions are removed from the page.

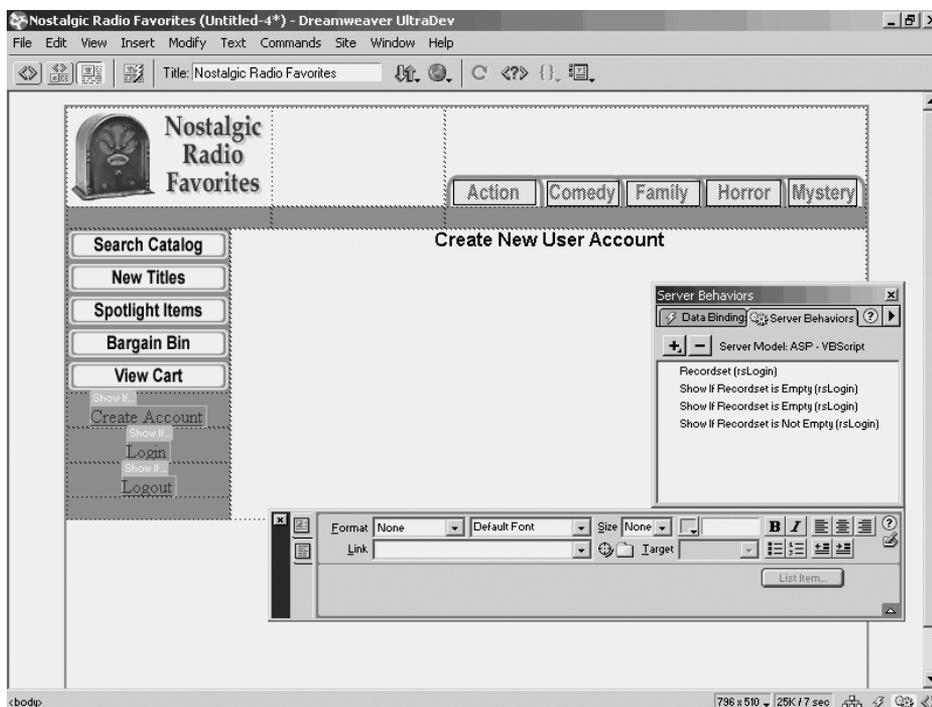


Figure 8.6 The page has been detached from the template.

6. Save your page in the root directory of your InsideUD4 site. The filename should be newuser.asp.



Tip

Tracking Pages that Don't Rely on Templates Creating a page from a template is a quick way to maintain uniformity across your site. Keep in mind, however, that any pages you detach from the template won't be updated when you change the template. For this reason, it's a good idea to make a note on your site map when you detach a page. This serves as a reminder to manually make the updates to the detached pages whenever you update your templates.

Adding the Input Form

The next step to creating your new user account page is to add a signup form that includes the appropriate text labels, text fields, and buttons. Whereas a typical new user account might ask a user to enter personal information, demographic information, and account information (such as username and password), a new user account form can ask for as little or as much information as your organization needs. The only requirement is that your database have a table with a single field established where the data entered into the form can be stored. Once you have created the appropriate fields in your database, you can use UltraDev menus to develop your signup form with a few clicks of your mouse.

Exercise 8.3 Adding an Input Form to the New User Account Page

1. In the `newuser.asp` page, place the insertion point on the line below your Header text.
2. From the menu bar, select Insert/Form. As shown in Figure 8.7, UltraDev places a blank form in the editable region indicated by a red, dashed border.

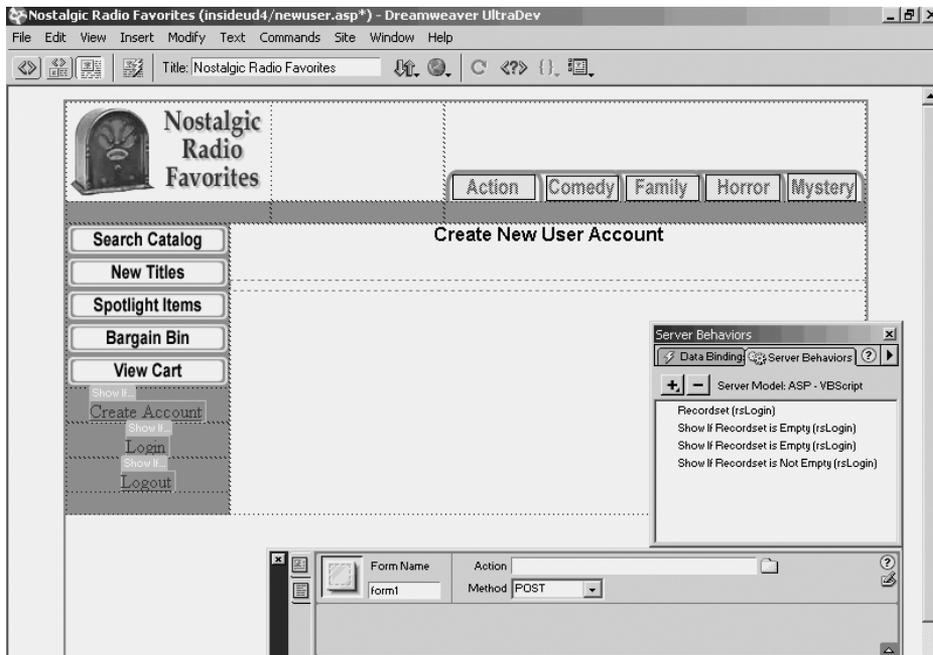


Figure 8.7 A form has been added to the page.

3. In the Property inspector, type **fmNewuser** into the Form Name field. If you do not see the Form Name field, the form is not selected. To select the form, click anywhere on the form's red border.
4. With the form selected, choose Insert/Table from the menu bar. Add a table with **11** rows, **2** columns, and a width of **100%**. Set the border at **0**. Set the cell padding and cell spacing to **1**. Click OK.
5. Highlight all the cells by clicking in the top-left cell and dragging your cursor to the bottom-right cell. Click the Align Right button on the Property inspector. Set the cell width to **50%** by typing the value in the W field of the Property inspector.
6. Place the insertion point in the top-left cell and type **Choose Username:**. Select Insert/Form Objects/Text Field from the menu bar. As shown in Figure 8.8, UltraDev adds a text field to your form.

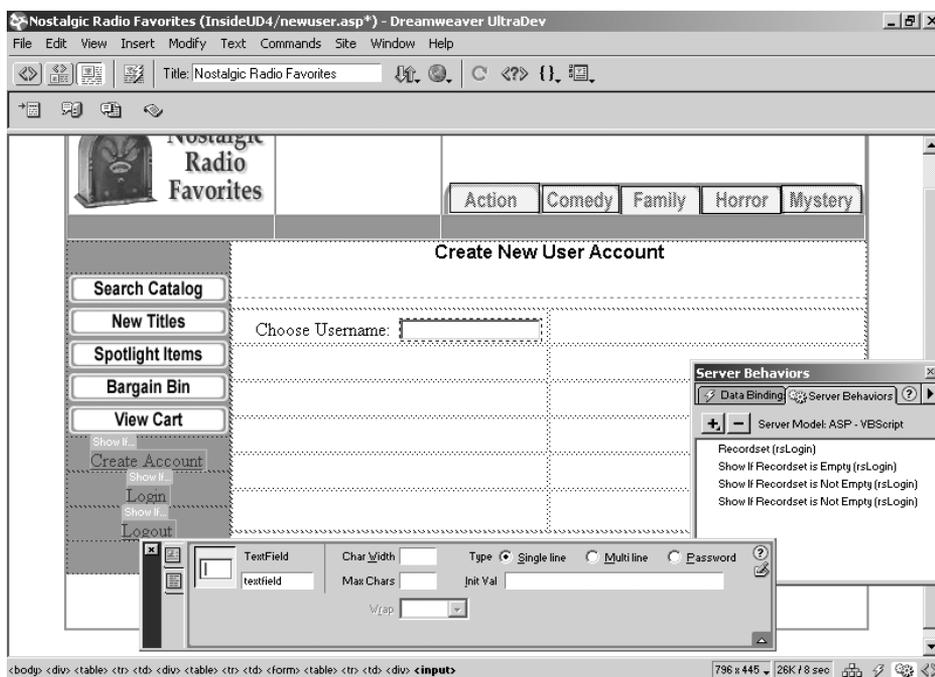


Figure 8.8 The text field has been added.

7. With the text field selected, type **tfUsername** in the TextField field of the Property inspector. Press Enter.
8. Place the insertion point in the top-right cell and type **Choose Password:**. Select Insert/Form Objects/Text Field from the menu bar. Name this text field **tfPassword**.

9. With the tfPassword text field selected, choose the text field type Password from the Property inspector. Selecting the Password type will cause the browser to display an asterisk each time a character is typed rather than the actual character.



Tip

Verifying Passwords Although you aren't going to do it in this exercise, you may want to add a Re-enter Password field where the user is required to enter his new password a second time. You can then compare the first password field and the second to ensure that they are identical. Doing this verifies that the user entered what he intended and saves him from the hassle of having to retrieve his password due to a mistyped character.

If you want to add a little script to your page that verifies that the entered passwords match, just create two password fields and name them tfPassword and tfPassword2. Switch to the Code view. Add the following code to the Head section of your page:

```
<SCRIPT LANGUAGE = "JavaScript">

function verifyPassword(){
paswd1 = document.fmNewuser.tfPassword.value;
paswd2 = document.fmNewuser.tfPassword2.value;

if (paswd1 != paswd2){
alert("Passwords do not match!");
document.fmNewuser.tfPassword.focus();
document.fmNewuser.tfPassword2.select();
}
else{
alert("Your password has been accepted");
}
}
</SCRIPT>
```

In addition, you will need to highlight the form and switch to the Code view. Find the line of code that reads

```
<form name="fmNewUser" method="post" action="">
```

Edit the line to read as follows:

```
<form name="fmNewUser" onSubmit="verifyPassword()" method="post"
  ↪action="">
```

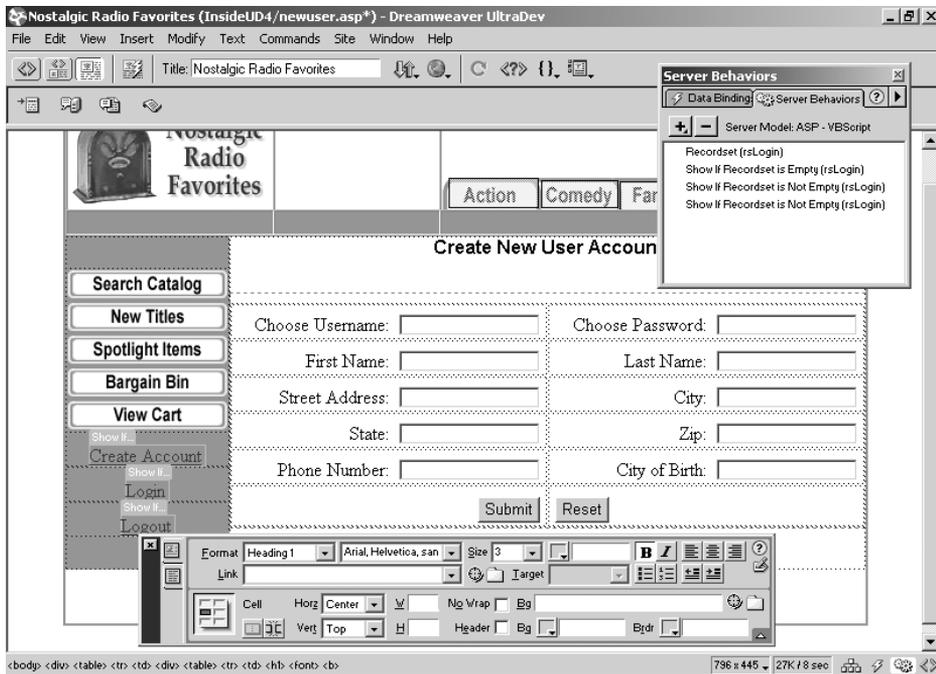
When the form is submitted, the script is called and checks the contents of the two password fields and then displays an alert as to whether or not the passwords match. Keep in mind that this function relies on the proper naming of your form and password text fields. If your form is not named fmNewuser or your field names are not tfPassword and tfPassword2, the function will fail.

10. Continue filling in the cells using the information shown in Table 8.1. Remember to name each text field and set all text field types to single line.

Table 8.1 Label and Text Field Criteria

Text Field Label	Text Field Name
First Name:	tfFirstname
Last Name:	tfLastname
Street Address:	tfAddress
City:	tfCity
State:	tfState
Zip:	tfZip
Phone Number:	tfPhone
City of Birth:	tfCityofbirth

11. Place the insertion point in the bottom-left cell and choose Insert/Form Objects/Button. In the Property inspector, name the button btSubmit. Leave the label as Submit and the Action as Submit form.
12. Place the insertion point in the bottom-right cell and click the Align Left button on the Property inspector. Choose Insert/Form Objects/Button from the menu bar. Rename the second button btReset and change the label to Reset. Change the the button Type to Reset form. Your page should now look like Figure 8.9.

**Figure 8.9** The form, table, and form elements have been added.

13. Save your page.

**Note**

How Secure Is That Form? Web site security should always be on the mind of every Web developer. When your users transmit their personal data to your database, they are relying on you to provide them with as much privacy as is appropriate. Unfortunately, some Web developers fail to realize that information submitted via a standard form is available to prying eyes because no encryption is used when the data is transmitted.

Therefore, when the need arises to transfer confidential information such as credit card information or social security numbers, or other highly personal information, Web developers should consider the use of a Secure Sockets Layer (SSL) connection. SSL is a protocol developed by Netscape that provides a secure connection between the visitor's Web browser and the Web server by encrypting any information transmitted between the two.

If you will be collecting information beyond traditional "directory" information (such as name, address, and phone number), you might consider looking into adding SSL functionality to your site. For more information on SSL, check out

<http://home.netscape.com/security/techbriefs/ssl.html>

Verifying That Required Fields Are Filled

When you are considering what data you would like to collect from your visitors, you might also want to decide what information should be required to create a new user account. For instance, if you are shipping a product to the user, his street address, city, state, and zip would obviously be necessary. In other instances, you may need his phone number, credit card number, or some other information. In any case, it's likely that information such as the new username and password will always be required.

To verify that the visitor has entered data into a specific field, you can use the UltraDev form validation behavior. With this behavior, JavaScript is used to verify whether or not a value has been entered into a text field and whether the entered data is a number, an email address, or falls within a range of numbers. Because the validation process takes place on the client side, prior to submitting the data to the database, the visitor will be notified if a required field has been left empty. The information can be changed and resubmitted.

**Tip**

Advanced Form Validation If you want a more advanced form validation option, check out the JavaScript Integration Kit extension found on the Dreamweaver Exchange. Although this extension says it is for Flash 5, installing it in UltraDev gives you added password validation options.

Exercise 8.4 Verifying Form Fields

1. In the newuser.asp page, select the fmNewuser form by clicking on the red border. Open the Behaviors panel by choosing Window/Behaviors from the menu bar.
2. Click the plus sign on the Behaviors panel and choose Validate Form.
3. From the Validate Form dialog box, shown in Figure 8.10, highlight each entry in the list one at a time and click the Required checkbox. This results in requiring the visitor to fill in every text box before the form can be submitted.

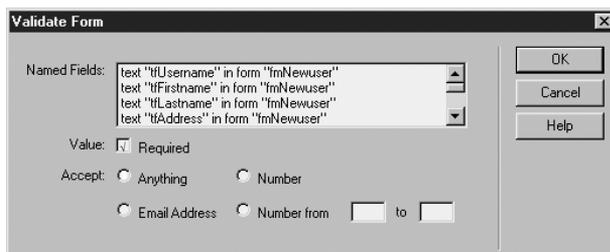


Figure 8.10 The Validate Form dialog box ensures that required fields are filled.

4. Click OK to close the Validate Form dialog box.
5. Close the Behaviors panel.



Note

Using JavaScript for Form Validation Although UltraDev makes it very easy to validate a form using the built-in behavior, there is a downside. Since the form validation behavior uses JavaScript, it can be disabled on the browser side. Those who have disabled JavaScript in the browser may be able to slip in entries that are not complete. Your alternative is to employ server-side scripting, such as a CGI Script. This, however, places a larger load on the Web server and may slow down your site's performance.

My suggestion is to give the JavaScript behavior a try. If you find numerous incomplete forms, consider switching to a CGI form.

Submitting the Data to the Database

After the visitor fills in the form and clicks the Submit button, you want the data to be placed into the appropriate database fields. To accomplish this, UltraDev provides the Insert Record server behavior that does all the work for you. All you have to do is tell UltraDev which database it should connect to and then specify which form field should be linked to which database field and UltraDev does the rest.

Exercise 8.5 Linking Form Fields to the Appropriate Database Fields

1. If it is not already visible, open the Server Behaviors panel.
2. Click the plus sign on the Server Behaviors panel and choose Insert Record from the drop-down menu.
3. In the Insert Record dialog box, shown in Figure 8.11, select the connSales_Database connection.

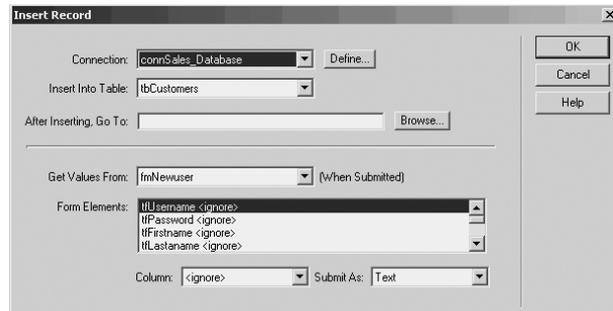


Figure 8.11 The Insert Record dialog box helps you add your form's contents to the database.

4. From the Insert Into Table dropdown, choose the tbCustomers table.
5. In the After Inserting/Go To field, type **newuser_confirmation.asp**.
6. Select the fmNewuser form from the Get Values From drop-down menu.
7. In the Form Elements panel, highlight the tfUsername element. In the Column field, choose the CustomerID column. As shown in Figure 8.12, the Element now shows that the information stored in tfUsername will be inserted into the CustomerID column.

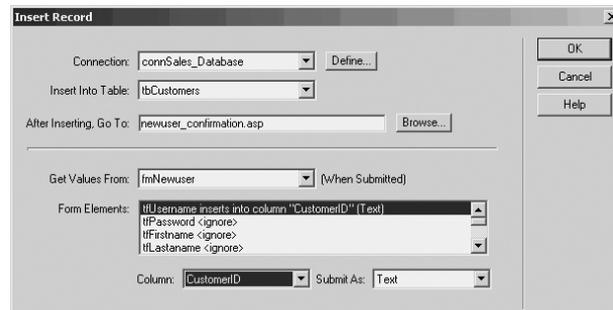


Figure 8.12 The tfUsername field is now linked to the CustomerID column of the database.

8. Using the values in Table 8.2, assign the appropriate form element to the column. Each value should be submitted as text to the database.

Table 8.2 Form Elements and Their Corresponding Columns

Form Element	Database Column
TfPassword	Password
tfFirstname	First_Name
tfLastname	Last_Name
tfAddress	Address
tfCity	City
tfState	State
tfZip	Zip
tfPhone	Phone
tfCityofbirth	City_of_birth

**Caution**

Duplicate Insert Entries Be very careful when assigning your form elements to their corresponding columns. If you assign multiple form elements to update the same column, the insert action will fail and you will receive a nasty error message notifying you that updating a single column from multiple form elements is not allowed.

9. Click OK to close the Insert Record dialog box.
10. Save your page.

Avoiding Duplicate Usernames

Now that you have your form tied to your database fields, you need to make sure that the username the visitor has chosen is unique. To do this, use the UltraDev Check New Username server behavior. When the behavior is applied to a form field, UltraDev adds a function to your page that searches the specified username field for an entry that matches the one entered into the form. If the username already exists, the visitor is redirected to a new page where they are notified of the problem.

Exercise 8.6 Checking for Duplicate Names

1. In the Server Behaviors panel, click the plus sign and choose User Authentication/Check New Username from the drop-down menu.
2. In the Check New Username dialog box, shown in Figure 8.13, select tfUsername from the Username Field dropdown.
3. In the If Already Exists, Go To field, type **username_taken.asp**. Click OK.
4. Save your newuser.asp page.
5. From the menu bar, select File/New From Template. In the Select Template dialog box, choose the InsideUD4 site and the nrfdefault template. Click Select.

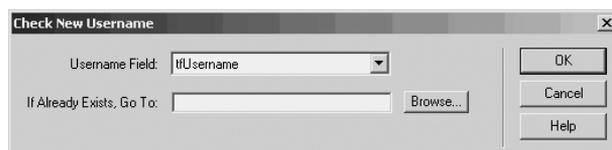


Figure 8.13 The Check New Username dialog box allows you to avoid duplicate usernames.

6. With the new page displayed, select **Modify/Templates/Detach From Template**. You are detaching this page from the template because you are going to use a JavaScript behavior that needs to be placed in the Head section of the document.
7. Highlight the `{erMainData}` text and delete it.
8. With the insertion point placed where the `erMainData` text was, choose **Insert/Table** from the menu bar. Add a table that has **2 rows, 1 column**, and a width of **80%**. Set the border, cell spacing, and cell padding to **0**. Click **OK**.
9. Highlight both cells by clicking in the top cell and dragging your cursor to the bottom cell. Click the **Align Center** button on the Property inspector.
10. In the top cell, type the following text block:

We're sorry, but the username you have selected has already been taken. Please click the Back button on your browser's button bar or click the link below to return to the form and choose a different username.
11. In the bottom cell, type **Return to the form**. Highlight this text and type a pound sign (**#**) in the Link field of the Property inspector. This converts the text to a link that has no destination. Press **Enter**. Your page should now look like Figure 8.14.
12. To create a link that performs the same function as the browser's **Back** button, we call a simple JavaScript function. Open the Behaviors panel by choosing **Window/Behaviors**.
13. With the text still highlighted, click the plus sign on the Behaviors panel and choose **Call JavaScript**.
14. In the **Call JavaScript** dialog box, shown in Figure 8.15, type **history.back()**; Click the **OK** button.
15. Save the page as **username_taken.asp**.

Adding a Confirmation Page

The final step in developing the new user registration process is to create a confirmation page that lets the new user know that his account was created successfully. A well-designed confirmation page not only informs the user that no errors occurred in the process, but also provides the user with his first opportunity to log in.

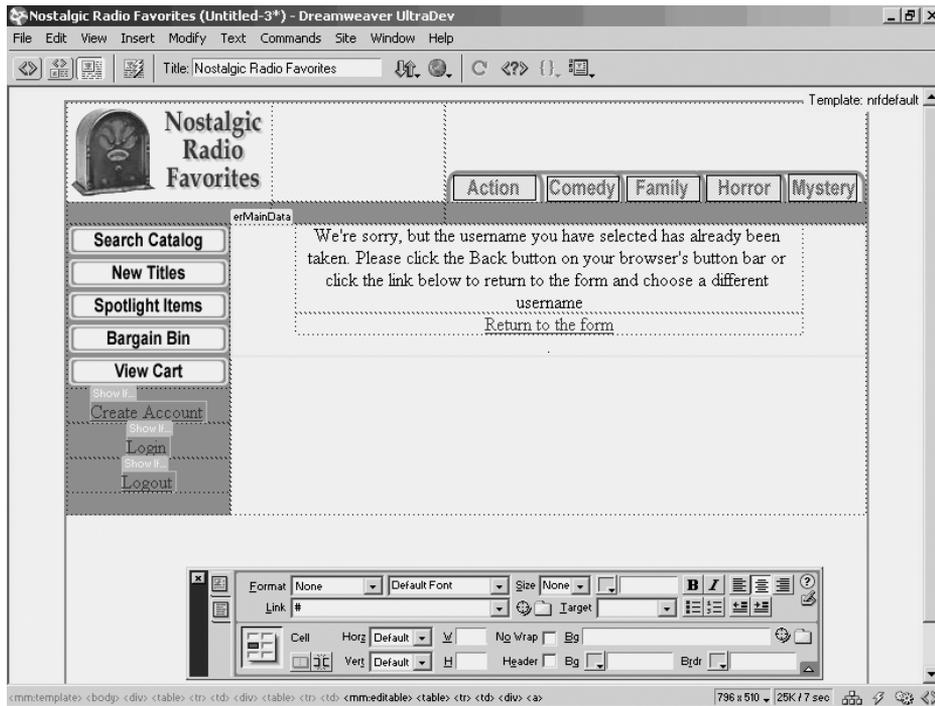


Figure 8.14 Your page with the appropriate text and link.



Figure 8.15 The Call JavaScript dialog box allows you to add custom scripts to your pages.

Exercise 8.7 Creating a Confirmation Page

1. From the menu bar, select File/New From Template. In the Select Template dialog box, choose the InsideUD4 site and the nrfdefault template. Click Select.
2. Highlight the {erMainData} text and delete it.
3. With the insertion point placed where the erMainData text was, choose Insert/Table from the menu bar. Add a table that has 2 rows, 1 column, and a width of 80%. Set the border, cell spacing, and cell padding to 0. Click OK.
4. Highlight both cells by clicking in the top cell and dragging your cursor to the bottom cell. Click the Align Center button on the Property inspector.

5. In the top cell, type the following text block:
Congratulations! Your account has been created successfully and your information has been added to our database.
6. Place the insertion point in the bottom cell and type **To log in for the first time, click here.**
7. Highlight the words “click here.” In the Link field of the Property inspector, type **login.asp**. Press Enter.
8. Save the page as **newuser_confirmation.asp**.

Allowing Returning Visitors to Log In and Out

Now that visitors have the opportunity to create their own user accounts, it's time to give them the means to use them. Although a login form may seem like just a simple form that checks to see if the username and password match the values stored in the database, there is actually a lot more to it than that.

When a visitor logs in to your site, he should then be able to view all of the pages that were previously restricted to the general public. This means that you somehow have to let all the pages in the site know that this user has successfully entered his username and password and has been admitted to the site.

To accomplish this, insert a small piece of code into your login page that stores a small text file, also known as a *cookie*, on the visitor's hard drive. This cookie is updated as the user actively navigates the site. By default, the session expires once the visitor leaves the site, closes his Web browser, or is idle for 20 minutes.

The only downside to using cookies to maintain a session is the fact that visitors can set their browsers to stop cookies from being placed on their computer. If your site relies upon cookies to maintain a session, a visitor who has cookies disabled won't be able to effectively navigate your site.

Exercise 8.8 Creating a Login Form

1. Create a new page from the nrfdefault template. Close any other open pages. Close the Behaviors panel.
2. Highlight the {erMainData} text and type the following text:
Welcome Back! Please use the following form to log in to our site.
3. Press Enter.
4. From the menu bar, select Insert/Form. Name the Form **fmLogin**.

**Tip**

Allowing Access to Your Pages by Passing Variables Another way to give visitors access to your pages after they are authenticated is to pass a variable from page to page. Each page would then check whether that variable exists in the database and allow access to the page when appropriate. A variable is passed from one page to another by appending the variable to the URL for the page.

For instance, suppose you clicked on a link to `http://localhost/insideud4/newpage.asp` and you are an authenticated user. The original page could pass your username to the next page by changing the link to `http://localhost/insideud4/newpage.asp?username=yourusername`. The `newpage.asp` document would then understand that you are allowed access to the page and would enable you to view its content.

This method, however, has its drawbacks. While it may be useful for passing small variables from page to page, it quickly becomes cumbersome when passing a large number of variables from page to page. For instance, suppose you wanted to give your visitors the ability to maintain a shopping cart that lasted throughout their sessions. As they moved through the site adding items to their carts, the list of variables being passed from page to page could become huge. The more effective approach would be to create a session using a cookie, and to store the shopping cart information in the cookie. Because some servers place a limit on the maximum number of characters that can be placed in a URL, this effectively restricts the viability of this method for passing large amounts of data.

Another disadvantage of passing URL parameters is that they could pass sensitive information such as passwords or credit card information in a visible manner. In cases where you are passing sensitive information, you are much better off using session variables or cookies. This way, the sensitive information is stored on the user's computer in a much more secure manner.

5. With the form selected, select Insert/Table from the menu bar. Create a table that has 3 rows, 2 columns, and a width of 50%. Set the border, cell padding and cell spacing to 1. Click OK.
6. Select the cells in the left column by moving your cursor to the top of the left column. When your cursor becomes a downward-pointing arrow, click the mouse button.
7. On the Property inspector, click the Align Right button.
8. Select the cells in the right column and click the Align Left button on the Property inspector.
9. In the top-left cell, type **Username:**. In the middle-left cell, type **Password:**.
10. Place the insertion point in the top-right cell and select Insert/Form Objects/Text Field from the menu bar. Name the new text field **tfCustomerID**.
11. In the middle-right cell, add an additional text field and name it **tfPassword**. In the Property inspector, set the text field type to Password.
12. In the bottom-left cell, add a submit button by choosing Insert/Form Objects/Button. Name the button **btSubmit**. Press Enter.

13. Add a Reset Form button to the bottom-right cell. Name the button **btReset**. Your page should now look like Figure 8.16. Press Enter.

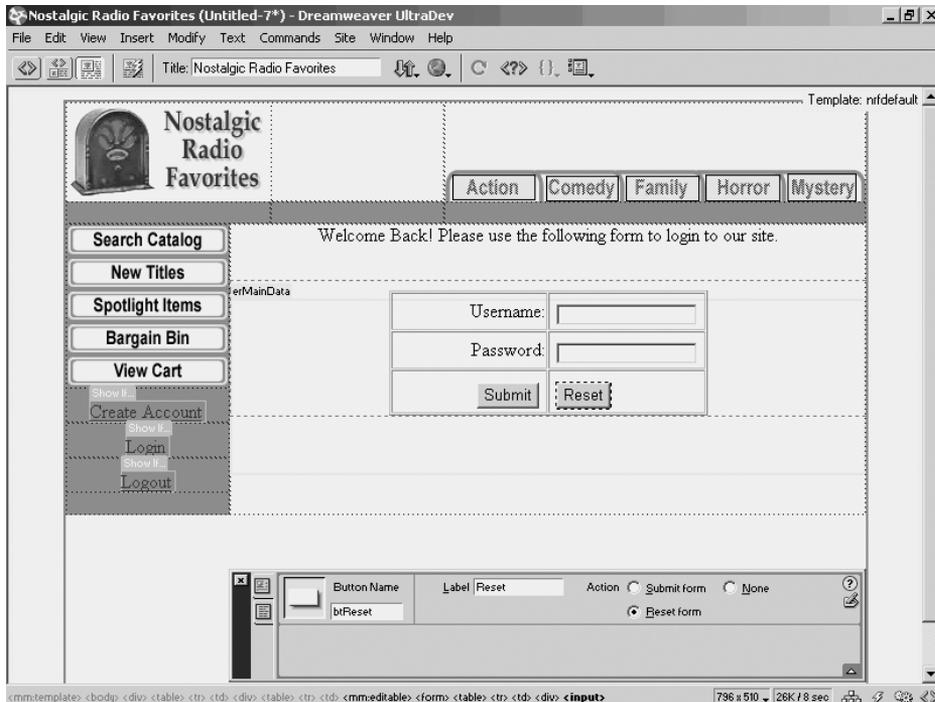


Figure 8.16 Your page with the login form.

14. Highlight the form by clicking on the red border. In the Property inspector, type **validation.asp** in the Action field. Press Enter.
15. Place the insertion point below the bottom border of the form and type **Click here to create an account**. Select the text and type **newuser.asp** in the Link field of the Property inspector. Press Enter.
16. Save the page as **login.asp**.

Once the form is successfully submitted and the username/password combination is validated or denied, the visitor is automatically redirected to a page that lets him know whether he was successfully logged on. This page also has the added functionality of creating the user's session and storing his session information on his local hard drive if he is successfully logged on.

**Note**

UltraDev's Log In User Server Behavior It's probably worth mentioning that UltraDev does include a Log In User server behavior that is easy to apply to a form. This behavior checks whether the username and password match those stored in the database and then redirects the visitor based on whether he was authenticated.

The behavior, however, has two major drawbacks. First, it does not allow Web developers to pass request variables in any way. This means that the behavior is really only good for protecting one page at a time. Second, UltraDev has identified a bug in the behavior that can compromise the security of your site. For more information on these problems, check out the following articles:

Passing form data from a login page to successive pages is unsuccessful

<http://www.macromedia.com/support/ultrudev/ts/documents/loginformdata.htm>

and

Log in user server behavior security issue

http://www.macromedia.com/support/ultrudev/ts/documents/login_sb_security.htm

Exercise 8.9 Adding a Validation Page

1. From the nrfdefault template, create a new page. Close the Login.asp page.
2. Delete the {erMainData} text and insert a table that consists of 2 rows, 1 column, and a width of 80%. Set the border, cell padding, and cell spacing at 0. Click OK.
3. Highlight both cells and click the Align Center button on the Property inspector.
4. In the top cell of the table, type the following text block:
You were successfully logged in. Please click here to continue viewing our site as an authenticated user.
5. Highlight the “click here” text and type **default.asp** in the Link field of the Property inspector. Press Enter.
6. In the bottom cell, type the following text:
We're sorry, but we were unable to validate your username and password. Please click the Back button on your browser's button bar to return to the login form.
7. Just as you did earlier with your dynamic login and logout links, the goal here is to display only the text that is appropriate depending on whether the login information submitted in the form was correct. Highlight the text in the top cell and click the plus sign on the Server Behaviors panel. From the menu, select Show Region/Show Region If Recordset Is Not Empty.

8. From the following dialog box, select the rsLogin recordset. Click OK.
9. Highlight the text in the bottom cell and click the plus sign on the Server Behaviors panel. From the menu, select Show Region/Show Region If Recordset Is Empty.
10. From the following dialog box, select the rsLogin recordset and click OK.
11. Highlight the text in the top cell and switch to the Source Code view by clicking the Show Code View button on the button bar. This code, shown in Figure 8.17, displays the text only if the user has been properly authenticated.

```

149 <% If Not rsLogin.EOF Or Not rsLogin.BOF Then %>
150     You were successfully logged in. Please <a href="default.asp">click
151     here</a> to continue viewing our site as an authenticated
152     user.
153 <% End If ' end Not rsLogin.EOF Or NOT rsLogin.BOF %>

```

Figure 8.17 The code in your page that displays the successful login message if the user is properly authenticated.

12. After the first line of this code, add the following lines:

```

<%session("MM_Username")=rsLogin.Fields.Item("tfCustomerID").Value%
>
<%session("MM_Password")=rsLogin.Fields.Item("Password").Value%>

```

Press Enter. Your code block should now look like Figure 8.18.

```

<% If Not rsLogin.EOF Or Not rsLogin.BOF Then %>
|<%session("MM_Username")=Request.Form("tfUsername")%>
  You were successfully logged in. Please <a href="default.asp">click
  here</a> to continue viewing our site as an authenticated
  user.
<% End If ' end Not rsLogin.EOF Or NOT rsLogin.BOF %>

```

Figure 8.18 After adding the code to create a session, your code should look like this.

Because this session will be used in conjunction with one of UltraDev's built-in server behaviors to restrict access to certain pages, it is important that you give the session the MM_Username name. Selecting a different name would result in the server behavior failing to restrict access.

13. Switch back to the Show Design View and notice that UltraDev has added a yellow ASP marker to your page. Save the page as **validation.asp**.

The final step in the login process is to provide visitors with the ability to log out. Although their sessions will close after an extended length of inactivity or whenever they close their browser windows, it's always a good idea to give your visitors the ability to manually close their sessions. This specifically protects visitors who may be surfing your site from public computers at a library or a student computer lab.

Since you added the logout link earlier, the only thing left to do is to add the logout page. This page simply thanks the visitor for logging out and uses one of UltraDev's server behaviors to close the user's session.

Exercise 8.10 Creating a Logout Page

1. Using the nrfdefault template, create a new page. Close the Validation.asp page.
2. In the new page, replace the {erMainData} text with the following text block:
Your session has ended. Thanks for visiting our site. Feel free to log back in by clicking the Login link.
3. On the Server Behaviors panel, click the plus symbol and choose User Authentication/Log Out User from the menu.
4. In the Log Out User dialog box, shown in Figure 8.19, select Log Out When: Page Loads. Leave the When Done, Go To field blank and click OK.

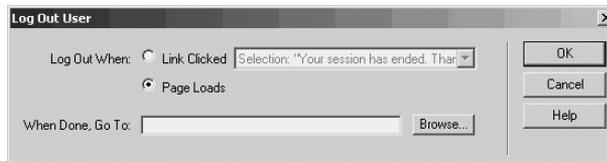


Figure 8.19 The Log Out User dialog box lets you dictate when the user is logged out.

5. Save the page as **logout.asp**.

Protecting Pages from Unauthenticated Visitors

The final piece of adding user authentication to your site is to protect those pages that should not be available to unauthenticated visitors. Since you created a session using the visitor's username as the session variable that is passed from page to page, you can add one of UltraDev's server behaviors to the protected pages that verifies that the user has entered a valid username and password prior to requesting the protected page.

Exercise 8.11 Restricting Access to a Page

1. Using the nrfdefault template, create a new page. Close the Logout.asp page.
2. Remove the {erMainData} text and replace it with the following text block:
Welcome to your shopping cart. Because you were properly authenticated, you are able to see this page.

3. On the Server Behaviors panel, click the plus sign and select User Authentication/Restrict Access To Page.
4. In the Restrict Access To Page dialog box, shown in Figure 8.20, choose to restrict the page based on Username and Password. In the If Access Denied, Go To field, type **http://localhost/insideud4/login.asp**.

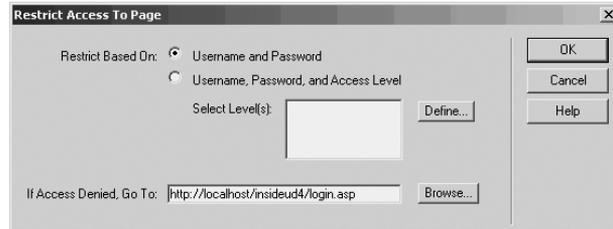


Figure 8.20 The Restrict Access To Page dialog box keeps unauthenticated visitors from viewing content.

5. Click OK. Save the page in the root of your site as **view_cart.asp**.

Testing Your New Pages

Are you ready to view the fruits of your labor? Now that you have added all the required recordsets, forms, and behaviors to your pages, you are ready to test them. Although this section takes a brief look at the results of the chapter exercises, I highly recommend that you take some time to play around with the new pages you've developed. Understanding what your visitors will see when they attempt to log on can help you streamline your pages and make them as user-friendly as possible.

Exercise 8.12 Testing the User Account Pages

1. Close all of your pages and close UltraDev.
2. Open your Web browser and enter **http://localhost/insideud4/default.asp** in the address field.
3. As shown in Figure 8.21, you should now see your home page with the Create Account and Login links showing. Because you have not logged in, the Logout link is hidden.
4. Click the View Cart button. Because you are not an authenticated user, you are automatically redirected to the login.asp page.
5. Click the Create Account link on the left side of your page. The Create New User Account page now allows you to enter your information and create a user account.

**Note**

Taking Care of Orphaned Pages As I mentioned earlier, when it comes to updating pages built using templates, UltraDev doesn't do a very good job of transferring recordsets to older pages. Because of this, it's probably a good idea to do a little cleanup before you test your pages.

Prior to this chapter, you have built six pages that rely on your `nrfdefault` template:

- `new_titles.asp`
- `new_titles_details.asp`
- `spotlight_items.asp`
- `spotlight_items_details.asp`
- `bargain_bin.asp`
- `bargain_bin_details.asp`

When you try to access each of these pages in your browser, you may receive an error. If you do, open your `nrfdefault.dwt` page and open the Server Behaviors panel. Right-click on the `rsLogin` recordset and choose Copy from the pop-up menu.

Next, open each of the six pages, right-click in the Server Behaviors panel and click Paste. Save each page and the problem should now be resolved.

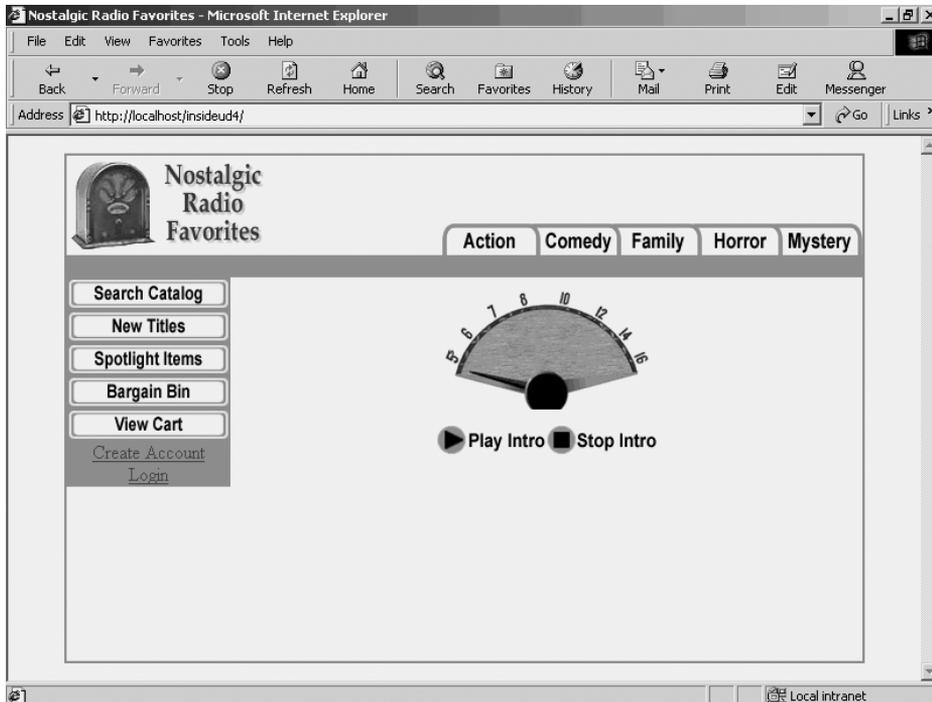


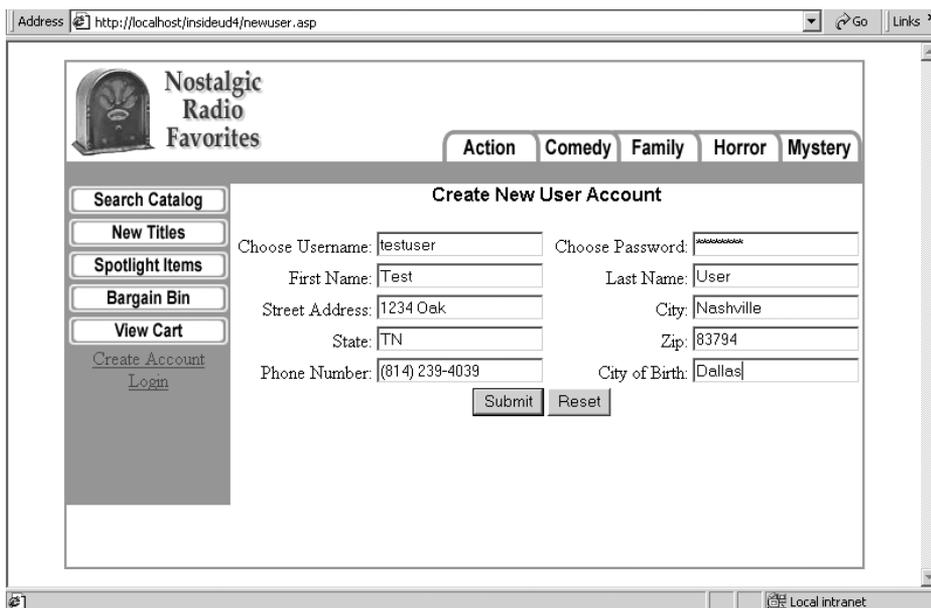
Figure 8.21 Your home page with dynamic links in place.

6. In the Choose Username field, type **testuser**.
7. Click the Submit button. Because all the required fields have not been completed, a pop-up message, shown in Figure 8.22, lets you know that you need to finish filling out the form.



Figure 8.22 An alert informing you that the required fields have not been filled in.

8. Click OK to close the pop-up box.
9. In the Choose Password field, type **testuser**. Because the field was designated as a password field, the actual text is replaced by asterisks.
10. Fill in the rest of the form as shown in Figure 8.23.
11. Click the Submit button. Having submitted a form that did not contain a username already in the database and with every field filled, you should now see the new user confirmation page shown in Figure 8.24.
12. Click the text that reads “click here” to login using the account you just created.
13. In the login form, type **testuser** for the username and **testuser** for the password. Click the Submit button. You should now see the validation.asp page letting you know that you successfully logged in.
14. Click the Logout button that is now visible on the left side of the page. As shown in Figure 8.25, UltraDev ends your session.



Address http://localhost/insideud4/newuser.asp

Nostalgic Radio Favorites

Action Comedy Family Horror Mystery

Create New User Account

Choose Username: Choose Password:

First Name: Last Name:

Street Address: City:

State: Zip:

Phone Number: City of Birth:

Search Catalog
New Titles
Spotlight Items
Bargain Bin
View Cart
[Create Account](#)
[Login](#)

Local intranet

Figure 8.23 Complete the new user account form using this information.



Address http://localhost/insideud4/newuser_confirmation.asp

Nostalgic Radio Favorites

Action Comedy Family Horror Mystery

Congratulations! Your account has been created successfully and your information has been added to our database.
To log in for the first time, [click here](#)

Search Catalog
New Titles
Spotlight Items
Bargain Bin
View Cart
[Create Account](#)
[Login](#)

Done Local intranet

Figure 8.24 Congratulations! You've added a new user to the database.

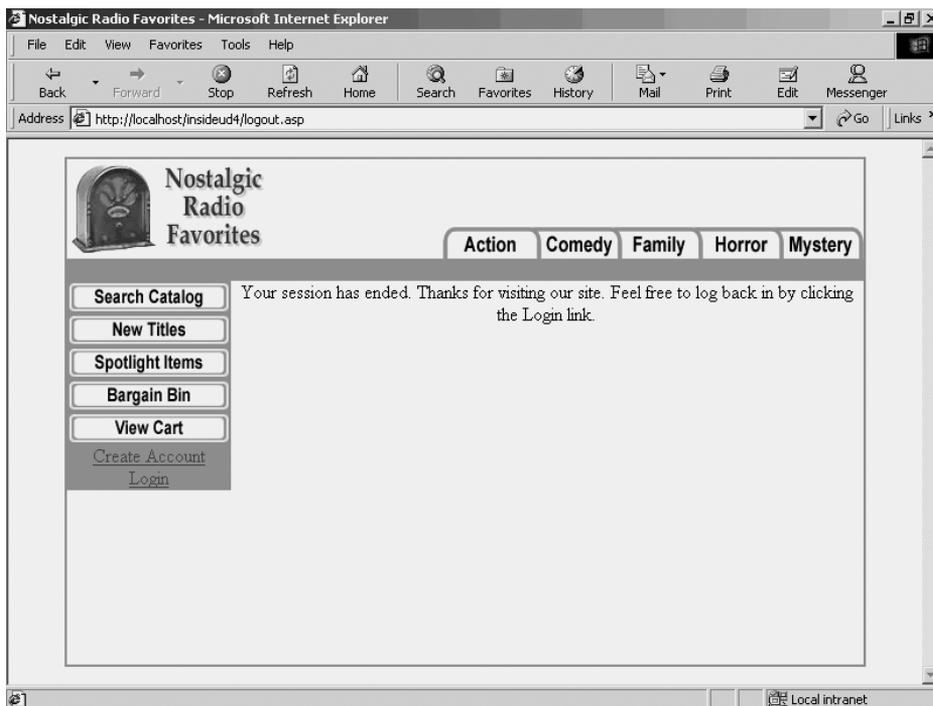


Figure 8.25 The logout page confirms that your session has ended.

Summary

This chapter demonstrated how to add pages to your site that allow visitors to create their own user accounts. In addition, it discussed adding dynamic links to your site, developing session variables, and protecting pages from unauthorized visitors.

The next chapter takes a look at adding search capabilities to your site and shows you how to allow your visitors to search your site using input forms.

