

Homework 8  
1.264, Fall 2006  
Web site continuation  
Due: Tuesday, November 21

### **A. Web site continuation**

Requirements for homework 8 are:

- Build the order entry Web pages used by your customers to select products and place them in an order.
- Follow the basic approach of the tutorial in chapter 22 of [DW]. The tutorial adds, edits and deletes products in a table. In your Web site, you will add orders and order detail; the same approach can be used.
  - You do not need to be able to edit or delete either orders or order detail; even though a real system would allow it. It's not required for homework 8 even if your requirements state it.
- Use your requirements and design documents to determine the details of your approach. If you need to modify your requirements or design, change those documents (your homework solutions).

After you complete this homework, a user must be able to place an order. For example, he or she can select items from the catalog into an order, or enter items manually into the order, or use another process, as defined in your requirements.

### **B. Details**

The steps below simplify the development substantially to limit the time you spend on this homework. You may use a different approach if you wish. In the suggested approach below, you modify two existing pages (login.asp and product\_details.asp) and create two new pages (orderEntry.asp and orderSummary.asp). This is likely to be the quickest strategy, since you need to write the least number of new pages, and the steps that you must take parallel DW tutorial steps fairly closely.

Steps and hints:

1. **SQL Server:** You'll need to make a minor change to the ChemicalOrder table using the SQL Server client. Make order number an int, and make it an 'identity' column, which will automatically assign the order number and increment it each time a new order is created. The SQL Server client will ask if you want to make the same change in the ChemicalOrderDetail table; answer 'yes'.
2. **SQL Server:** Manually enter an order or two and some order details into your database using the SQL Server client. Your result sets will contain something when you start, which will help in troubleshooting any initial problems in building your order entry pages.

3. **login.asp**: Rename your login.html page to login.asp. Put a recordset with repeating region on this page to display about 10 customers from the customer table. Use the 'go to detail page' feature to allow a user to select one customer and have that customer's certificate number be available on the next page, orderEntry.asp. You will replace this approach with a true login page in homework 9; this is a temporary implementation to identify the customer at the start of order entry. Use the DW tutorial examples of 'go to detail page' as a guide.

4. **orderEntry.asp**: On the orderEntry.asp page:

a. Use the 'record insertion form wizard' to create a form on which the customer enters order date, purchase order or credit card number, carrier cert number and mode. Even if your requirements have the user entering the data in a different order, have the user enter it all here to save development steps. (If some data is entered later, you must build more pages and use the 'edit' feature to put that data into the order later; this is quicker.)

b. Create recordsets for carriers and modes so that the user can select only valid carriers and modes for the order. In the DW tutorial, vendor and category recordsets are created for validation; use the same approach for carriers and modes. To keep it simple, just show carriers with mode 'Air'; there are about 247 of them. (Create your recordset with carriers with an 'Air' mode.) Show all four modes (air, highway, rail, water) in a dropdown menu in the form, but in your users will always need to choose air. This keeps your dropdown list from taking forever to build, or requiring a lot of work to manage. (The list of carriers with an air mode will look strange; these are companies with in-house shipping that use air carriers.)

c. Put the value of the customer certificate number into the form. In a real system it would be a hidden field, but show it here so you can troubleshoot any problems. Use the syntax shown near the bottom of p. 797 in DW.

d. This page is built using the same steps as the DW tutorial, pp 788 ff, for the 'insert product page'. In this homework you are inserting a row in the ChemicalOrders table (instead of inserting a new product as in the tutorial). As in the DW example, don't put the primary key field in the table since SQL Server fills it in automatically.

e. You do not need to validate the credit card number; you don't need to have the customer enter an expiration date, etc. Handle it very simply. You can make the customer enter the date; you don't need to fill it in automatically.

5. **catalog.asp**: After the user fills in orderEntry.asp, he or she is taken to the catalog.asp page (after hitting the submit button) to choose a category of products to browse. There are no changes necessary in catalog.asp.

6. **products.asp**: The user then goes to products.asp to choose a specific product. There are no changes necessary in products.asp.

7. **product\_details.asp**: After choosing a product from products.asp the user is on the product\_details.asp page that shows the product UN number and price (and other details).

He or she can now fill out a short form that you will add at the bottom of this page to order this item and to indicate the quantity desired. The ChemicalOrderDetail table needs UN number, price, quantity and order number.

a. Create a form at the bottom of orderEntry.asp with the four fields needed to insert a row into ChemicalOrderDetail table:

- Get the UN number and price by dragging them from the rsProductDetails binding into the fields in the form. Dreamweaver can be flaky here, and you may need to use the Code view to make sure it's correct—it should look essentially like the example in item 7 on p. 797 in DW, with `<%=rs.... .Value)%>` being in the value= field on the form.

- UN number, price and order number would normally be hidden text on the form, but display them here to help in troubleshooting.

- We will do something flaky to get the order number into the form. Create a recordset rsOrders that contains just one row:

```
SELECT TOP 1 OrderNbr FROM ChemicalOrder ORDER BY OrderNbr  
DESC
```

This will get the highest order number, which will be the latest order created. In real life, the order number would be contained within an SQL transaction, and other checks would be implemented as well; just getting the highest order number would not work. We'll do it simply here, so you don't have to implement a SQL transaction. Put the orderNbr value from the recordset into the order number field on the form.

b. You do not need to check whether the quantity is in stock, or that it can be carried on the chosen mode. The quantity in stock is shown on the page and you could show the shipping quantity limits on the page as part of the product details. You could also get and show the valid modes for the customer through a recordset. Thus, all these elements are (or can be) present and software logic could be implemented on this page to check them, or to display the information, so the user can adjust the order as needed. Again, we simplify here to limit your effort, so you don't need to implement any of the checks.

c. You do not need to check that the customer can use the chosen mode. You do not need to handle backorders, split shipments, or any other complications, even if your requirements stated that you must.

d. After the form is completed by the user hitting the submit button (which should be titled 'Add to order'), the user will be taken to a page called orderSummary.asp, which you will build next.

**8. orderSummary.asp:** This is a new page. Create three recordsets:

- a. rsChemicalOrders, that has just the single row for the highest order number from ChemicalOrder, joined with the customer and carrier tables so that customer name and carrier name can be displayed. Write the SQL in advanced mode when defining the recordset; it should be

```
SELECT ... FROM (joins...) WHERE OrderNbr IN (SELECT TOP 1 ... as above)
```

b. rsChemicalOrderDetails, that has just the rows of details for this order. Join ChemicalOrderDetail with ChemName to display the product names; use the same approach as rsChemicalOrders to use the highest order number to get just this order

c. rsOrderSummary, that has aggregate queries using the SQL SUM keyword. This recordset (query) sums the items' extended prices (price times quantity) to get the subtotal, then applies a 5% sales tax rate (hard code the 5% or 0.05 in the query), and then adds the subtotal and tax to get the order total.

d. Lay out the page to display the order header info at the top, based on rsChemicalOrders, the order detail in the middle, based on rsChemicalOrderDetails, and the order subtotal, tax and total on the bottom, based on rsOrderSummary

e. Format the currency fields with currency format using the format property in the bindings window, as in the DW tutorial for prices (p.747).

f. At the bottom of the page, put two hyperlinks. One can be labeled 'Add more items', and it takes the user back to categories.asp to select more items to add to the order. The other hyperlink can be labeled 'Done'; this completes the order and takes the user back to the orderEntry.asp page, where he or she can place another order or navigate to another part of the site. You do not need to build an order confirmation screen; the orderSummary.asp page is sufficient.

g. As the user adds more items to the order (selecting 'add more items' each time), the middle part of this page shows all the items (order details) as they grow. In the middle section show the item, its name, quantity, price and extended price (price \* quantity), which you can compute in the SQL statement in the rsChemicalOrderDetails recordset.

You're done!

You do not need to have any functions to display, add, modify or delete rows in any other table (carriers, modes, orders, etc.), even if your requirements called for them.

Navigation among your pages must be reasonable. A user should not be able to get to the order summary page from anyplace other than the product detail page, for example. Make sure your hyperlinks are arranged correctly.

If you find any problems with JOINS in Dreamweaver, remember you can create views in SQL Server and then use those views in Dreamweaver to avoid having JOINS. (I had no problems with JOINS in this homework.)

Save and test your work frequently.

Your Web site must be complete and operational on the Class Server on the homework due date. The TAs will grade what is on the Web site.