SYBEX Sample Chapter

ASP, ADO, and XML Complete

Chapter 10: Sample Application

Copyright © 2001 SYBEX Inc., 1151 Marina Village Parkway, Alameda, CA 94501. World rights reserved. No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way, including but not limited to photocopy, photograph, magnetic or other record, without the prior agreement and written permission of the publisher.

ISBN: 0-7821-2971-4

SYBEX and the SYBEX logo are either registered trademarks or trademarks of SYBEX Inc. in the USA and other countries.

TRADEMARKS: Sybex has attempted throughout this book to distinguish proprietary trademarks from descriptive terms by following the capitalization style used by the manufacturer. Copyrights and trademarks of all products and services listed or described herein are property of their respective owners and companies. All rules and laws pertaining to said copyrights and trademarks are inferred.

This document may contain images, text, trademarks, logos, and/or other material owned by third parties. All rights reserved. Such material may not be copied, distributed, transmitted, or stored without the express, prior, written consent of the owner.

The author and publisher have made their best efforts to prepare this book, and the content is based upon final release software whenever possible. Portions of the manuscript may be based upon pre-release versions supplied by software manufacturers. The author and the publisher make no representation or warranties of any kind with regard to the completeness or accuracy of the contents herein and accept no liability of any kind including but not limited to performance, merchantability, fitness for any particular purpose, or any losses or damages of any kind caused or alleged to be caused directly or indirectly from this book.

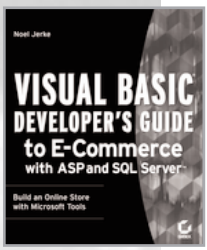
SYBEX Inc.
1151 Marina Village Pkwy.
Alameda, CA 94501
USA
Phone: 510-523-8233
www.sybex.com

Chapter 10

Sample Application

Before we go full bore into database design in Part III, we are going to build a very simple e-commerce application based on Active Server Pages and SQL Server. This will help to get our feet wet with the ASP development environment.

Our sample application will be a simple form to purchase a subscription to a publication. This form will take in name and address information and credit card data.



Adapted from *Visual Basic® Developer's Guide to E-Commerce with ASP and SQL Server™*, by Noel Jerke
ISBN 0-7821-2621-9 752 pages $49.99

Building the Data Table

The first thing we will need is a simple database table that we can insert our subscriptions into. The obvious fields are in the table for the subscriber's name, address, credit card information, etc., as shown in Listing 10.1.

CREATING A TABLE IN SQL SERVER

Database management will be covered in more depth in the next few chapters. But if you want to create the following table now, you have a couple of options. In SQL Server Enterprise Manager, you can open the database you want to use, right-click Tables, and choose New Table. Then manually create the fields with the names and vartypes shown in Listing 10.1. Another option is to open Query Analyzer and connect to your database. Then simply type the code exactly as you see it and execute the code by either pressing F5 or choosing Query and then Execute.

Listing 10.1: Subscription Database Table

```
CREATE TABLE dbo.Subscriptions (
  idSubscription int IDENTITY (1, 1) NOT NULL ,
  chrFirstName varchar (100) NULL ,
  chrLastName varchar (100) NULL ,
  chrAddress varchar (150) NULL ,
  chrCity varchar (100) NULL ,
  chrState varchar (10) NULL ,
  chrZipCode varchar (15) NULL ,
  chrPhone varchar (25) NULL ,
  chrEmail varchar (100) NULL ,
  chrCardName varchar (150) NULL ,
  chrCardType varchar (50) NULL ,
  chrCardNumber varchar (25) NULL ,
  chrExpDate varchar (50) NULL ,
  intProcessed tinyint NULL DEFAULT 0,
    /* Default to 0 */
  dtEntered datetime NULL
    DEFAULT GETDATE(),
    /* Default to current date */
```

```
    intLength tinyint NULL
)
```

A couple of status fields are included in the table. The `intProcessed` field would be used to flag the order as processed so an indication of what subscriptions have been retrieved can be easily tracked. This field should be defaulted to 0, to indicate "unprocessed." The next status field is the `dtEntered` field. This defines the date the subscription was entered into the database. It should be defaulted to the current date.

Building the HTML Form

To build the HTML page, we will need to create an HTML form and HTML elements on the page. Then we will build a script page to process the data entered by the user.

The first part of the page is straightforward, as shown in Listing 10.2. The standard HTML headers for the page are created. We also start out the form by setting it to post results to the `ProcessSub.asp` page, which will process the subscription.

Listing 10.2: `Subscription.asp` **Page**

```
<%@ Language=VBScript %>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft
    Visual Studio 6.0">
</HEAD>
<BODY>

<BR><BR>

<center>

<!--  Setup the Header --!>
<font size="4" color="blue"><b>
XYZ Publication
</b></font>

<!--  Start the form that will post to the
      ProcessSub.asp page. --!>
<form method="post" action="ProcessSub.asp">
```

The next section of the page is the table that contains the form for displaying the input fields of the subscription page (see Listing 10.3). There are several key actions on the page. First, if the user enters invalid data, we want to be able to send him back to this form and have the data he entered repopulated into the form.

The repopulation is done by reading session variables set in the `ProcessSub.asp` page when the data is in error. Our first challenge is the length of the subscription (set in `intLength`). If the user selected two-year or three-year subscriptions, then we will want to set the proper radio button. If not, then the one-year option will be set.

Listing 10.3: `Subscription.asp` **Continued**

```
<!--  Next the table starts that will
      layout the data entry form -->
<table border=1>

<!--  Subscription Length -->
<tr>
  <td align="right">Subscription Length:</td>
  <td>
<%
    '  Check to see if a length was set. If so
    '  then default the radio button selected.
    if session("intLength") = "1" then
      CheckOne = "Checked"
      Flag = 1
    end if

    if session("intLength") = "2" then
      CheckTwo = "Checked"
      Flag = 1
    end if

    if session("intLength") = "3" then
      CheckThree = "Checked"
      Flag = 1
    end if

    '  If this is the first time the form is
    '  displayed in the session then default to
    '  a length of one year.
```

```
    if Flag <> 1 then CheckOne = "Checked"

%>
  <!--  Radio buttons for selecting
        the length -->
  <input type="radio" value="1"
     name="intLength" <%=CheckOne%>>
     One Year
  <input type="radio" value="2"
     name="intLength" <%=CheckTwo%>>
     Two Year
  <input type="radio" value="3"
     name="intLength" <%=CheckThree%>>
     Three Year
  </td>
</tr>

<!--  First Name -->
<tr>
  <td align="right">First Name:</td>
  <!--  Input field for the first name -->
  <td><input type="text"
    value="<%=session("chrFirstName")%>"
    name="chrFirstName"></td>
</tr>

<!--  Last Name -->
<tr>
  <td align="right">Last Name:</td>
  <!--  Input field for the last name -->
  <td><input type="text"
    value="<%=session("chrLastName")%>"
    name="chrLastName"></td>
</tr>

<!--  Address -->
<tr>
  <td align="right">Address:</td>
  <!--  Input field for the address -->
  <td><input type="text"
    value="<%=session("chrAddress")%>"
    name="chrAddress"></td>
</tr>
```

```
<!--  City -->
<tr>
  <td align="right">City:</td>
  <!--  Input field for the city -->
  <td><input type="text"
    value="<%=session("chrCity")%>"
    name="chrCity"></td>
</tr>

<tr>
  <td align="right">State:</td>
  <td><input type="text"
    value="<%=session("chrState")%>"
    name="chrState" size=2></td>
</tr>

<!--  Zip Code -->
<tr>
  <td align="right">Zip Code:</td>
  <!--  Input field for the zip code -->
  <td><input type="text"
    value="<%=session("chrZipCode")%>"
    name="chrZipCode"></td>
</tr>

<!--  Phone Number -->
<tr>
  <td align="right">Phone:</td>
  <!--  Input field for the phone number -->
  <td><input type="text"
    value="<%=session("chrPhone")%>"
    name="chrPhone"></td>
</tr>

<!--  Email Address -->
<tr>
  <td align="right">Email Address:</td>
  <!--  Input field for the email address -->
  <td><input type="text"
    value="<%=session("chrEmail")%>"
    name="chrEmail"></td>
</tr>
```

```
<!--  Name on Card -->
<tr>
  <td align="right">Name on Card:</td>
  <!--  Input field for the email address -->
  <td><input type="text"
    value="<%=session("chrCardName")%>"
    name="chrCardName"></td>
</tr>
```

A process similar to the length of subscription logic needs to take place for the card type. If the user selected MasterCard or American Express, then we want to reselect those options when the user is returned to the form (see Listing 10.4).

Listing 10.4: `Subscription.asp` **Continued**

```
<!--  Input field for the credit card type -->
<tr>
  <td align="right">Card Type:</td>
  <td>

<%
    ' Check to see which card was selected
    ' previously if there was an error.
    if session("chrCardType") = "Visa" then
      SelVisa = "Selected"
    end if

    if session("chrCardType") = _
      "MasterCard" then
      SelMC = "Selected"
    end if

    if session("chrCardType") = "AmEx" then
      SelAmEx = "Selected"
    end if
%>

    <!--  Select box for the type of cards -->
    <select name="chrCardType">
      <option value="Visa"
        <%=SelVisa%> >Visa
      <option value="MasterCard"
        <%=SelMC%>>MasterCard
```

```
        <option value="AmEx"
          <%=SelAmEx%>>American Express
      </select>

    </td>
  </tr>

  <!--  Credit Card Number -->
  <tr>
    <td align="right">Card Number:</td>
    <!--  Input field for the credit
       card number -->
    <td><input type="text"
       value="<%=session("chrCardNumber")%>"
       name="chrCardNumber"></td>
  </tr>

  <!--  Credit card expiration date -->
  <tr>
    <td align="right">Expiration Date:</td>
    <!--  Input field for the expiration date -->
    <td><input type="text"
       value="<%=session("chrExpDate")%>"
       name="chrExpDate"></td>
  </tr>
```

The last section of our page is the HTML Submit button for sending the form data to the server (see Listing 10.5). Then the form and the page are closed out.

Listing 10.5: The End of `Subscription.asp`

```
  <!--  Submit button -->
  <tr>
    <td colspan="2" align="center">
      <input type="submit" value="Subscribe!"
         name="submit">
    </td>
  </tr>

  </table>

  </center>
```

```
<!--  Closing tag for the end of the form -->
</form>

</BODY>
</HTML>
```

The input page is fairly straightforward. If you are new to ASP coding, then mixing script code and HTML tags in the same page might take some getting used to. But it is precisely this powerful integration that makes ASP such a rich development environment for building Web applications.

Programming the Script Code

Now the real programming fun begins on the processing of the subscription request. Our goal in this page is several-fold. First, we want to retrieve the data from the user and validate it. We want to ensure that she has entered in values for all required fields, and when possible we want to validate that the data is correct.

Second, we want to then give feedback to the user if there is an error. A message will be displayed telling the user certain fields are incorrect. We will provide a link back to the subscription page for the user. That is where the session variables and repopulating the subscription form come into play.

Third, if the data is valid, we want to thank the user. In this case, we are going to re-display the input data for good customer service feedback. And of course, we need to be sure to insert the subscription data into the database for later retrieval.

As with the `subscription.asp` page, the `processsub.asp` page opens up with basic HTML tagging. Listing 10.6 shows the page code.

Listing 10.6: `ProcessSub.asp` **Page**

```
<%@ Language=VBScript %>

<HTML>

<BODY BGCOLOR="WHITE">
```

Our first task is to retrieve the data from the form. We utilize the Request object to retrieve the data and reference the field names on the form (see Listing 10.6). The data is stored in variables for later use.

NOTE

Variables do not have to be used to store the form data. The Request object could be used throughout the page. But the variable use makes for easier manipulation of the data later.

Listing 10.7: ProcessSub.asp **Continued**

```
<%

'  Retrieve all of the data that the
'  user entered by using the request object.
intLength = Request("intLength")
chrFirstName = Request("chrFirstName")
chrLastName = Request("chrLastName")
chrAddress = Request("chrAddress")
chrCity = Request("chrCity")
chrState = Request("chrState")
chrZipCode = Request("chrZipCode")
chrPhone = Request("chrPhone")
chrEmail = Request("chrEmail")
chrCardName = Request("chrCardName")
chrCardType = Request("chrCardType")
chrCardNumber = Request("chrCardNumber")
chrExpDate = Request("chrExpDate")
```

The next step is to check each field and validate it (see Listing 10.8). For most of the fields, we are simply going to ensure that the field is not blank. For the state field, we do a little more validation to ensure that the length is not more than two characters. On the credit card expiration date, we can use the IsDate function to validate that it is a valid date.

Listing 10.8: ProcessSub.asp **Continued**

```
'  Check to see if the first name was entered.
if chrFirstName = "" then

  '  Give an error if not.
  strError = "You did not enter in your" & _
    " first name.<BR>"

end if
```

```
'  Check to see if a last name was entered.
if chrLastName = "" then

  strError = strError & "You did not enter" & _
    " in your last name.<BR>"

end if

'  Check to see if an address was entered
if chrAddress = "" then

  strError = strError & "You did not enter" & _
    " in your address.<BR>"

end if

'  Check to see if a city was entered.
if chrCity = "" then

  strError = strError & "You did not enter" & _
    " in your city.<BR>"

end if

'  Check to see if the state was entered
'  of if the length is more than two
'  characters.
if chrState = ""  or len(chrState) > 2 then

  strError = strError & "You did not enter" & _
    " in a valid state.<BR>"

end if

'  Check to see if a zip code was entered.
if chrZipCode = "" then

  strError = strError & "You did not" & _
    " enter in your zip code.<BR>"

end if

'  Check to see if the card name was entered.
```

```
if chrCardName = "" then

  strError = strError & "You did not enter" & _
    " in the name on your credit card.<BR>"

end if

'  Check to see if the card number was entered
if chrCardNumber = "" then

  strError = strError & "You did not enter" & _
    " in your credit card number.<BR>"

end if

'  Check to see if the card expiration
'  date was entered
if (chrExpDate = "") or _
  (isdate(chrExpDate) = false) then

  strError = strError & "You did not enter" & _
    " in a valid credit card" & _
    " expiration date.<BR>"

end if
```

Now that the data is validated, we are ready to take appropriate action. We can check the `strError` variable to see if it is set. If it is, then there was an error. If not, then there was no error.

```
'  Now we check to see if there are any errors.
if strError <> "" then


%>
```

If there is an error, we simply display the appropriate message and write out the error string. The key though is ensuring we have the data from the form stored so that it can be retrieved and displayed when the user returns to the form. The best way to do this is with session variables, which will stay *alive* while the user's session is still in progress (see Listing 10.9). Then on the subscription form we can retrieve those values and display them.

Listing 10.9: ProcessSub.asp **Continued**

```
    <!-- Note the error -->
    <B><font color="red">
    There is an error in your
    subscription request:<BR><BR>
    </b></font>

  <%

    '  Write out the error messages
    Response.Write strError

  %>

    <!--  Link back to the subscription page -->
    <BR>
    Click <a href="subscription.asp">here</a>
    to update.

  <%

    '  Set session variables to the
    '  subscription form can be re-populated
    Session("intLength") = _
      Request("intLength")
    Session("chrFirstName") = _
      Request("chrFirstName")
    Session("chrLastName") = _
      Request("chrLastName")
    Session("chrAddress") = _
      Request("chrAddress")
    Session("chrCity") = _
      Request("chrCity")
    Session("chrState") = _
      Request("chrState")
    Session("chrZipCode") = _
      Request("chrZipCode")
    Session("chrPhone") = _
      Request("chrPhone")
    Session("chrEmail") = _
      Request("chrEmail")
    Session("chrCardName") = _
```

```
      Request("chrCardName")
    Session("chrCardType") = _
      Request("chrCardType")
    Session("chrCardNumber") = _
      Request("chrCardNumber")
    Session("chrExpDate") = _
      Request("chrExpDate")

else

%>
```

If all of the data was valid, then we are ready to process the subscription form. An appropriate thank you message is displayed and then a recap of the form data is displayed (see Listing 10.10).

Listing 10.10: ProcessSub.asp **Continued**

```
<!--  Thank the customer for the order -->
<font size="4" color="blue">
Thank you for your order!
It will be processed immediately.</font>

<!--  Redisplay the data entered into
the subscription -->
<BR><BR>
<Table>
  <tr><td align="right">
    <B>Name:</b></td>
    <td><i>
    <% = chrFirstName & " " & chrLastName %>
    </i></td></tr>

  <tr><td align="right">
    <B>Address:</b></td>
    <td><i>  <% = chrAddress %></i>
    </td></tr>

  <tr><td align="right">
    <B>City:</b></td>
    <td><i>  <% = chrCity %></i>
    </td></tr>

  <tr><td align="right">
```

```
  <B>State:</b></td>
  <td><i>  <% = chrState %></i>
  </td></tr>

<tr><td align="right">
  <B>Zip Code:</b></td>
  <td><i>  <% = chrZipCode %></i>
  </td></tr>

<tr><td align="right">
  <B>Phone:</b></td>
  <td><i>  <% = chrPhone %></i>
  </td></tr>

<tr><td align="right">
  <B>Email:</b></td>
  <td><i>  <% = chrEmail %></i>
  </td></tr>

<tr><td align="right">
  <B>Card Name:</b></td>
  <td><i>  <% = chrCardName %></i>
  </td></tr>

<tr><td align="right">
  <B>Card Type:</b></td>
  <td><i>  <% = chrCardType %></i>
  </td></tr>

<tr><td align="right">
  <B>Card Number:</b></td>
  <td><i>  <% = chrCardNumber %></i>
  </td></tr>

<tr><td align="right">
  <B>Expiration Date:</b></td>
 <td><i>  <% = chrExpDate %></i>
  </td></tr>
```

Now we are ready to do the important step of inserting the data into the database. The first step is to create an ADO connection object to the database. You will need an ODBC DSN to connect to the database. Be

sure to create the DSN. Note that in Listing 10.11 a file DSN is being utilized, but a system DSN could be created instead. Be aware that user DSNs operate only under the context of the user for which they were created, rendering them unsuitable for use within IIS.

Next we have to sanitize the data for insertion into the database. We have to ensure that any single quotes that may be entered are doubled up so they can be inserted and not confused as delimiters. Examples of this problem would include last names (e.g., O'Brien), cities, addresses, etc. Using the Replace command makes it easy to replace these single quotes with doubles. In this case we will check the First Name, Last Name, Address, Card Name, and City.

TIP

SQL server will interpret two single quotes together (") as only one single quote. We will need to double up all single quotes that are part of the data to be stored in a field. Our values that are being inserted should start with a single quote and end with one as well.

Once the data is ready, we can build a SQL statement for inserting the data into the database. And then we are ready to execute the SQL statement (see Listing 10.11).

Listing 10.11: `ProcessSub.asp` **Continued**

```
<%

'  Create an ADO database connection
set dbSubs = server.createobject _
  ("adodb.connection")

'  Open the connection using our
'  ODBC file DSN
dbSubs.open("filedsn=SubForm")

'  If any of our names have a single
'  quote, we will need to double it to
'  insert it into the database
chrFirstName = _
  replace(chrFirstName, "'", "''")
chrLastName = _
  replace(chrLastName, "'", "''")
chrAddress = _
```

```
    replace(chrAddress, "'", "''")
  chrCardName = _
    replace(chrCardName, "'", "''")
  chrCity = replace(chrCity, "'", "''")

  '  SQL insert statement to insert
  ' the subscription data into the database
  sql = "insert into subscriptions(" & _
          "chrFirstName, " & _
          "chrLastname, " & _
          "chrAddress, " & _
          "chrCity, " & _
          "chrState, " & _
          "chrZipCode, " & _
          "chrPhone, " & _
          "chrEmail, " & _
          "chrCardName, " & _
          "chrCardType, " & _
          "chrCardNumber, " & _
          "chrExpDate, " & _
          "intLength) " & _
          "values (" & "'" & _
           chrFirstName & "', '" & _
           chrLastName & "', '" & _
           chrAddress & "', '" & _
           chrCity & "', '" & _
           chrState & "', '" & _
           chrZipCode & "', '" & _
           chrPhone & "', '" & _
           chrEmail & "', '" & _
           chrCardName & "', '" & _
           chrCardType & "', '" & _
           chrCardNumber & "', '" & _
           chrExpDate & "', " & _
           intLength & ")"

  '  Execute the SQL statement
  dbSubs.execute(sql)

end if

%>
```

```
</body>
</html>
```

That's it for the user-side programming. In the next section, we will explore how we can use Web-based reporting to retrieve the subscriptions.

Testing the Application

Now we are ready to begin testing. Calling the `subscription.asp` page from your Web server accesses the Web page shown in Figure 10.1.

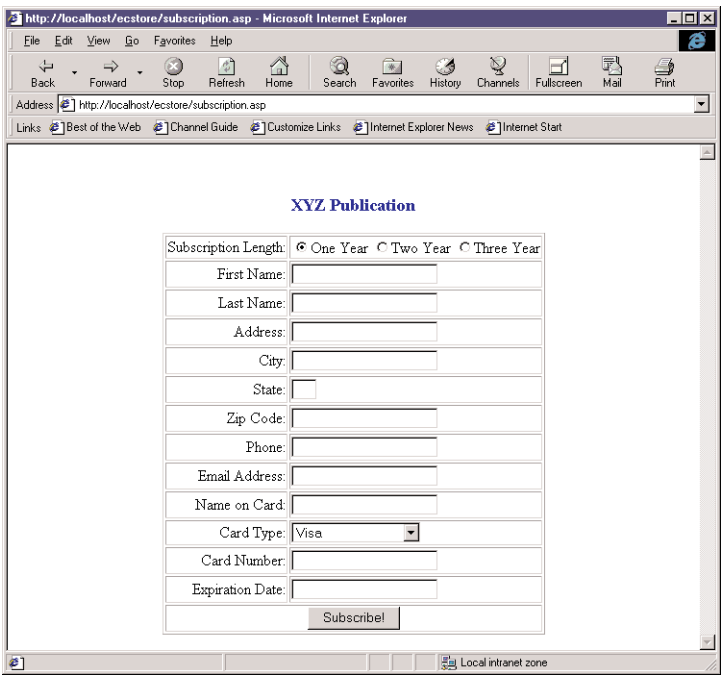


FIGURE 10.1: The `subscription.asp` page

Now we need to go ahead and enter data into the form. We will want to enter in some invalid data so that we can test the error handling. Figure 10.2 shows the form filled out with sample data. Note that the expiration date is invalid. When done, we need to submit the form to the `ProcessSub .asp` page.

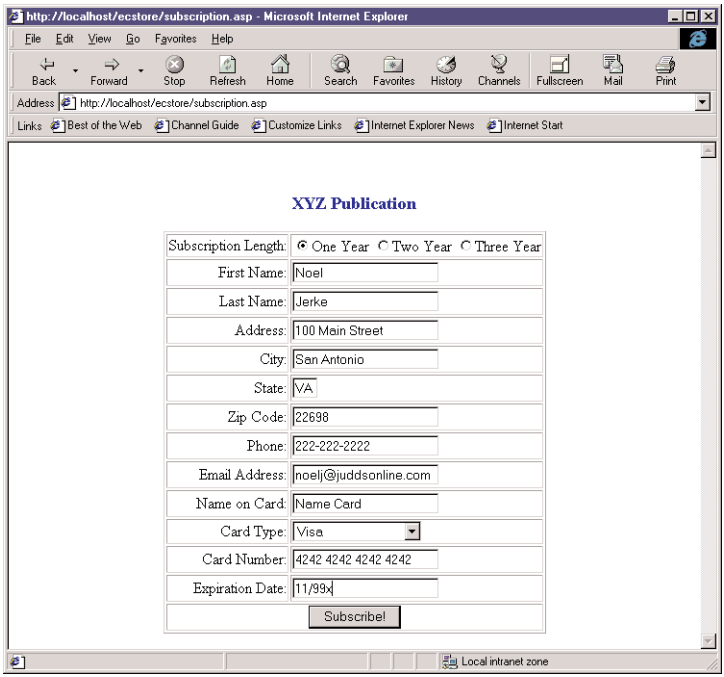


FIGURE 10.2: Entering invalid data into the subscription page

The `ProcessSub.asp` page will process the data. And, in fact, if all is working properly, we should see an error message indicating that the expiration date is invalid. Figure 10.3 shows the error message.

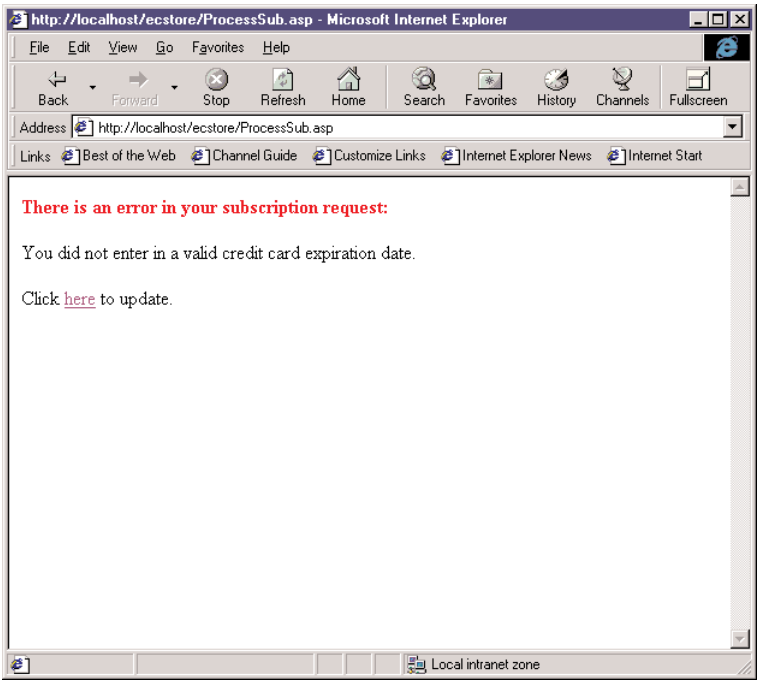


FIGURE 10.3: Error page with invalid expiration date

Now we can click on the error link and go back to the subscription page. When we do so, the data form should be re-populated with our subscription data, error messages, and all. Figure 10.4 shows a correctly entered subscription.

TIP

You may want to have the field name highlighted in red to help indicate on the subscription form which field is invalid.

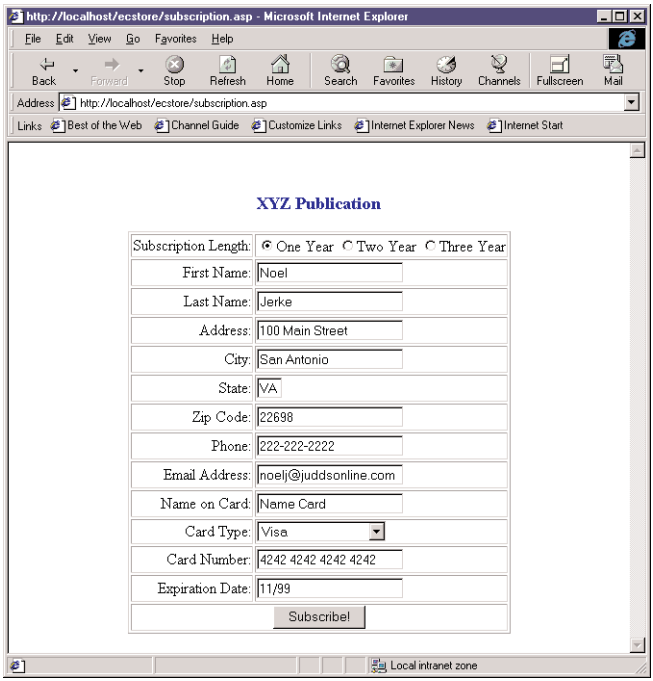


FIGURE 10.4: Entering valid data into the subscription page

Now we can correct the data and then resubmit the subscription data. When we do, the thank you response is displayed with a recap of the data. Figure 10.5 shows the thank you page. We should also be able to verify that the data went into the database.

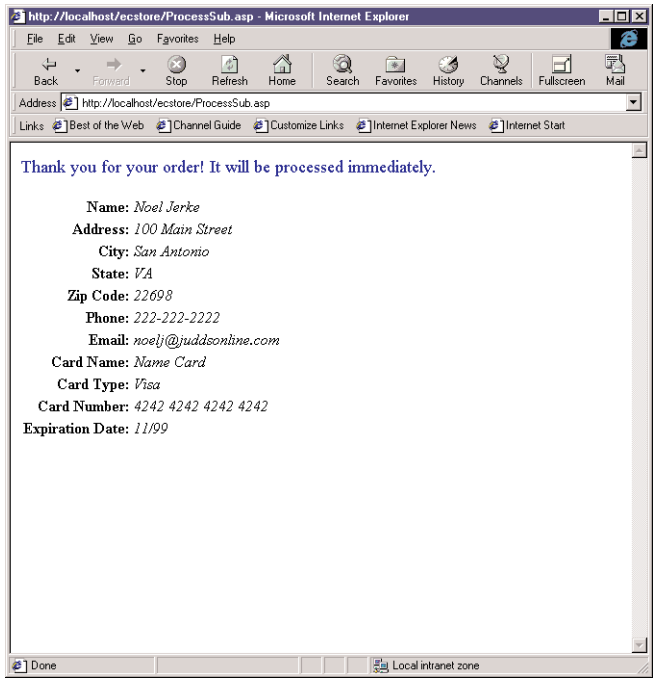


FIGURE 10.5: Thank you message after a successful subscription

Now that we have completed the user experience, we need to worry about the back-end management of the subscription data. We will need a way to retrieve the subscriptions.

Managing the Application

The last piece of our e-commerce sample application is the reporting form. The purpose of the form is to report out the subscription data entered since the last subscriptions were processed. It will also give an option for the user to mark the current listing of subscriptions as processed. Listing 10.12 shows the code for the `SubReport.asp` page.

Listing 10.12: `SubReport.asp`

```
<%@ Language=VBScript %>
<HTML>
<HEAD>
```

```
<META NAME="GENERATOR"
  Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>
```

The first step is to create our database connection. Then we need to check and see if we are to mark subscriptions as processed. If so, then there will be an `idSubscription` parameter on the URL. This is set later in the code when the clear option is selected. If the parameter is set, then all of the subscriptions that have an ID less than or equal to the subscription ID will be cleared. Anything above that will remain unprocessed and will be displayed (see Listing 10.13).

Listing 10.13: `SubReport.asp` **Continued**

```
<%

  '  Create an ADO database connection
  set dbSubs = server.createobject _
    ("adodb.connection")
  set rsSubs = server.CreateObject _
    ("adodb.recordset")

  '  Open the connection using our
  '  ODBC file DSN
  dbSubs.open("filedsn=SubForm")

  '  Retrieve any subscription IDs on the URL
  idSubscription = Request("idSubscription")

  '  Check to see if there is a value.
  if idSubscription <> "" then

    '  Built an SQL update statement
    '  to process the subs.
    sql = "update subscriptions set" & _
      " intProcessed = 1 where" & _
      "idSubscription <= " & _
       idSubscription

    '  Execute the SQL statement
    dbSubs.execute sql

  end if
```

Next we are ready to retrieve all of the subscriptions in the system that have not been processed. A SQL statement is built with the appropriate *where* clause, and then the SQL statement is executed with a record set returned.

```
'  Create a SQL statement to retrieve
'  any unprocessed subscriptions
sql = "select * from subscriptions" & _
  " where intProcessed = 0"

'  Execute the statement and retrieve
'  the record set
set rsSubs = dbSubs.Execute(sql)

%>
```

Next we are ready to begin the structure of the table that will be used to display the unprocessed subscriptions. The formatting is fairly simple, with field names on the left and the data on the right (see Listing 10.14).

TIP

You might want to put some logic in place to have the subscriptions listed in several columns instead of just one. If you are processing many subscriptions, that will reduce the number of pages that will be displayed.

Listing 10.14: SubReport.asp **Continued**

```
<!--  Start the table to display the subs. -->
<Table border="1">

<%

  '  Check to see if no subs are returned
  if rsSubs.EOF then

    '  If so, then write
    Response.Write _
      "No subscriptions to report."

  else
```

```
    '  Loop through the subs
    do until rsSubs.eof

%>

    <!--  Display the subscription data -->
    <TR>
    <TD align="right">First Name:</TD>
    <TD>    <%=rsSubs("chrFirstName")%></TD>
    </TR>

    <TR>
    <TD align="right">Last Name:</TD>
    <TD>    <%=rsSubs("chrLastName")%></TD>
    </TR>

    <TR>
    <TD align="right">Address:</TD>
    <TD>    <%=rsSubs("chrAddress")%></TD>
    </TR>

    <TR>
    <TD align="right">City:</TD>
    <TD>    <%=rsSubs("chrCity")%></TD>
    </TR>

    <TR>
    <TD align="right">State:</TD>
    <TD>    <%=rsSubs("chrState")%></TD>
    </TR>

    <TR>
    <TD align="right">Zip Code:</TD>
    <TD>    <%=rsSubs("chrZipCode")%></TD>
    </TR>

    <TR>
    <TD align="right">Phone:</TD>
    <TD>    <%=rsSubs("chrPhone")%></TD>
    </TR>

    <TR>
    <TD align="right">Email:</TD>
```

```
<TD>    <%=rsSubs("chrEmail")%></TD>
</TR>

<TR>
<TD align="right">Card Name:</TD>
<TD>    <%=rsSubs("chrCardName")%></TD>
</TR>

<TR>
<TD align="right">Card Number:</TD>
<TD>    <%=rsSubs("chrCardNumber")%></TD>
</TR>

<TR>
<TD align="right">Expiration Date:</TD>
<TD>    <%=rsSubs("chrExpDate")%></TD>
</TR>

<TR>
<TD align="right">Date Entered:</TD>
<TD>    <%=rsSubs("dtEntered")%></TD>
</TR>

<TR>
<TD align="right">Subscription Length:</TD>
<TD>    <%=rsSubs("intLength")%></TD>
</TR>

<TR>
<TD> </TD>
<TD> </TD>
</TR>
```

In order to be able to clear the listed subscriptions, we need to save the ID of the last subscription displayed so that it can be passed back to this page. The ID of the subscription is stored in the `idSubscription` variable. Then we advance the record set (see Listing 10.15).

Listing 10.15: SubReport.asp **Continued**

```
<%

    '  Store the last subscription id
    idSubscription = rsSubs("idSubscription")

    '  Move to the next sub
    rsSubs.MoveNext

    loop

  end if

%>

</table>

<BR><BR>
```

Finally we build a link back to this page with the ID of the last subscription so this report can be cleared. Note that the ID of the subscription is stored on the URL with the `idSubscription` parameter (see Listing 10.15).

Listing 10.16: SubReport.asp **Continued**

```
<!-- Link to this page with the last
     subscription ID -->
Click <a href="SubReport.asp?idSubscription=
  <%=idSubscription%>">here</a>
  to clear this report.

</BODY>
</HTML>
```

The page is then ready to be run. Make sure the database is seeded with some sample subscriptions. Figure 10.6 shows the report page with the sample data. Note the link to clear the report.

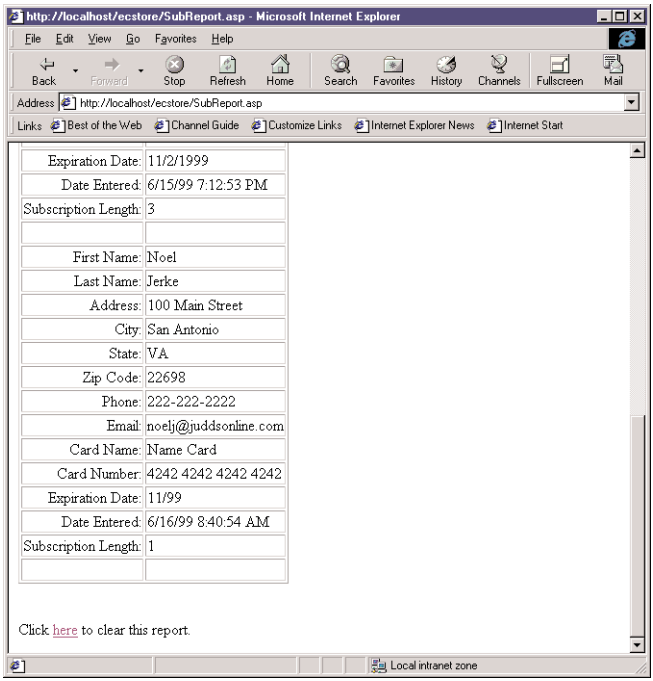


FIGURE 10.6: Subscriptions report page

Go ahead and click on the link to clear the subscriptions. When you do so, the page is re-called with the ID of the last subscription on the URL. Then the section of code is run to mark these subscriptions as processed. You should be able to check the processed fields in the database to verify they are set to 1. Any new subscriptions will be displayed, or a message is displayed indicating that no more subscriptions are available to be displayed. Figure 10.7 shows the processed page.

We might want to provide a richer interface for searching for subscriptions, processed and unprocessed. Date entered, length of subscription, etc., may be offered as options to search by.

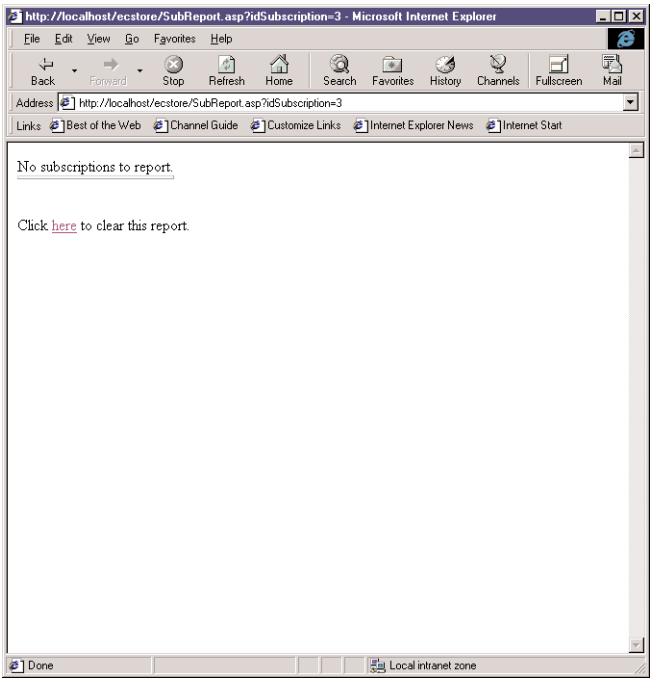


FIGURE 10.7: Cleared subscriptions report page

Final Considerations

Our sample application hits on the key tools we will be using for development—ASP, SQL Server, HTML, and a browser. For a site that needs a simple way to request subscriptions, memberships, or other data, this type of form will be more than adequate.

A few things should be considered when implementing this type of form. The first is security. Certainly the form should be encrypted with Secure Sockets Layer (SSL) to ensure the data cannot be easily *sniffed* on the Internet. You should also make sure that your usernames and passwords to access the database are not readily guessed or easily found out. The manager page should not be readily accessible to just anyone. You will want to secure it either with a password-protected form using SQL, or else by using Windows NT Authentication and an Access Control List (ACL) on the directory where the manager page exists.

Second, you may want to provide an order number back to the person who has just ordered so that he or she can make any queries referencing that number. The best way to implement that would be to build a stored procedure that inserts the subscription data and returns a parameter that is the ID of the identity column in the table. That can then be displayed in the thank you message to the user.

Finally, if you want to provide immediate processing of the credit card data, you might want to consider using tools such as CyberCash or HP/Veriphone. Then, if the user's order is cleared, you can immediately give him or her online access to content, etc.

What's Next?

This covers basic HTML and ASP programming. With that under your belt, you're ready to dive a little deeper into the database aspect of site design. In Part III, you'll learn more about the mechanics of database development, especially database objects, SQL syntax, and most importantly, how to tie it all into a Web application.