Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2000 Proceedings

Americas Conference on Information Systems (AMCIS)

2000

Developing Internet E-Commerce Applications with Database Access using Active Server Pages

Grove N. Allen
University of Minnesota, allen088@umn.edu

Developing Internet E-Commerce Applications with Database Access using Active Server Pages

Gove N. Allen, Department of Information and Decision Sciences
University of Minnesota, allen088@umn.edu

Abstract

Connecting web sites to databases is foundational to electronic commerce on the Internet. Microsoft's Active Server Pages is one tool which allows web sites to dynamically interact with database management systems. This tutorial shows how to create an electronic commerce application using this technology.

Introduction

Business-to-business electronic commerce has been conducted through EDI (electronic data interchange) since the 1960s (Kosiur, 1997). Recently, the increased popularity of the Internet has allowed businesses to pursue business-to-consumer electronic commerce on an unprecedented scale. The foundational technology of virtually all Internet electronic commerce applications (whether business-to-business or business-to-consumer or consumer-to-consumer) involves connecting Internet sites to database management systems. This connection can be accomplished through a number of means including Perl and CGI (common gateway interface), Java applets and servelets, Sun's Java Server pages, Microsoft's Active Server Pages, PHP, Altaire's Cold Fusion, and many others (Ladd, et al., 1988). This tutorial shows how to create an electronic commerce web site using Microsoft's Active Server Pages. The example that is developed in this tutorial is a business-to-consumer web site using Microsoft SQL Server as the back-end database management system.

Business Example

The business that will be used as the foundation for the code examples of this tutorial is an award-winning restaurant in the northwest United States. Mama Vallone's Steak House and Inn has an informational web site (www.mamavallones.com) but would like to add a page that allows customers to enter take-out orders. Mama Vallone's menu is stored in a database management system (Microsoft SQL Server) and is updated several times a year. The take-out order page must show the current menu and current prices. Customers who visit the page need to be able to compose an order by selecting from the menu and submit payment information online. A set of web pages will be generated in this tutorial to meet these needs.

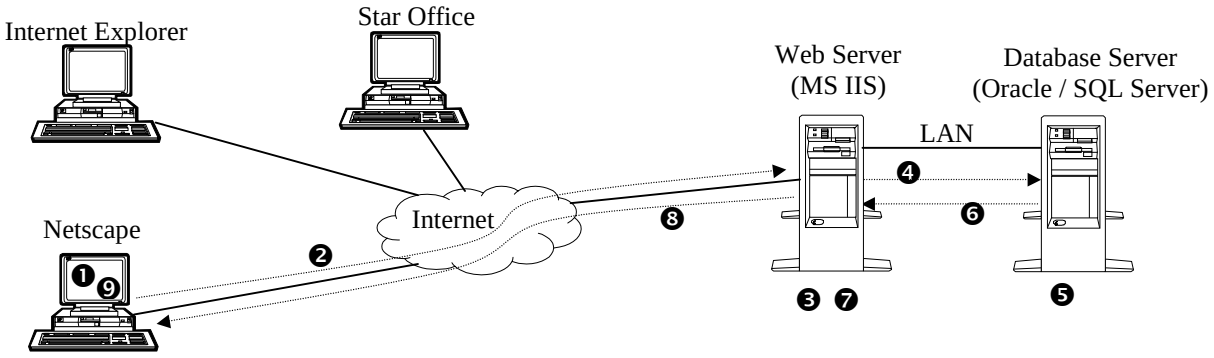
Prerequisites

This is not a tutorial about web page design or hypertext markup language (HTML). This is not a tutorial about database design or Structured Query Language (SQL). This is a tutorial about using Microsoft Active Server Pages (ASPs) scripts to dynamically create HTML documents including data stored in a SQL-92 compliant database management system. To appreciate this tutorial, readers should be familiar with HTML including frames and tables. The reader should also be familiar with the select statement of SQL including SQL-92 join syntax. The reader need not have any familiarity with Active Server Pages, but experience in at least one programming language (Visual Basic, C++, Pascal, etc) will be helpful.

The examples in this tutorial will be stored in a directory which has been configured in IIS to allow scripts to execute. While this configuration is beyond the scope of the tutorial, several example ASPs are installed in properly configured directories during standard installations of Windows NT 4 or Windows 2000 Advanced Server. Any of these directories would work well to test the techniques demonstrated in this turorial.

Active Server Pages Overview

ASPSs allows for the dynamic creation of HTML documents on Microsoft's web server, Internet Information Server (IIS). ASPs allow developers to build dynamic web sites by adding server-side scripts to static HTML pages. When a browser requests an ASP, these scripts execute and build an HTML file, which is then returned to the browser. These scripts have the ability to instantiate objects that can provide different functionality. Although such object can be created using tools such as Microsoft's Visual C++ or Visual Basic, several are installed by default as a part of IIS. These standard components allow developers to access any database that supports an ODBC connection while the HTML response is being generated. This allows a page to be delivered to a browser that is composed of data about a particular order, delivery, customer, etc. Figure 1 shows the process by which a user's request for an ASP which accesses a database is fulfilled.



❶ A user with a browser requests an Active Server Page.
❷ The browser sends an HTTP request (including parameters) to the web server.
❸ The web server begins building the HTML response to the HTTP request.
❹ The web server requests the database server to execute a query.
❺ The database server executes the query.
❻ The database server sends the result set of the query to the web server.
❼ The web server uses the query results to finish building the HTML response.
❽ The web server sends the HTML response to the browser that requested it.
❾ The browser renders and displays the HTML document based on the results of the query.

Figure 1. Generating an Active Server Page using Database Data

Tutorial Setup

To understand the steps in this tutorial, it helps to see the final destination. The result of this tutorial will appear as seen in Figure 2.

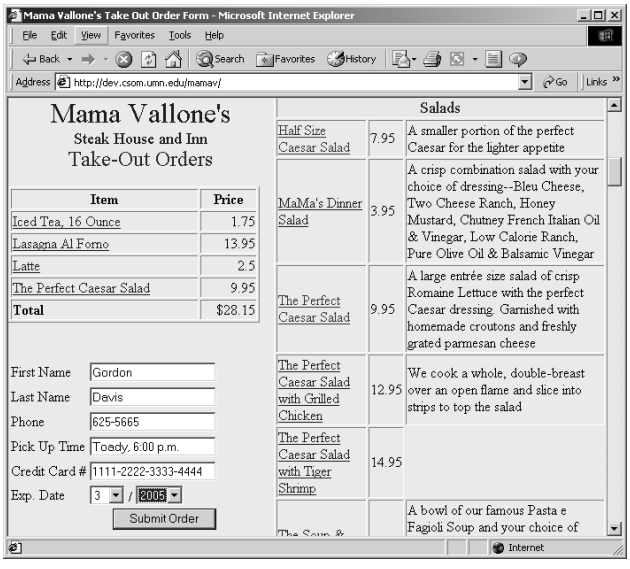


Figure 2. The Finished Product

This example uses two framesets (default.htm and OrderFrame.htm). A complete listing of these files and all files used in this tutorial can be examined in the Appendix.

The framesets organize the content documents as seen in Figure 3.

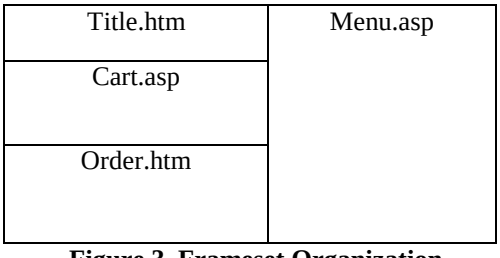


Figure 3. Frameset Organization

To build the finished product, we will need to learn eight techniques. These eight comprise the remainder of this tutorial and are as follows:

1. Accessing Database Data
2. Adding Parameters to Hypertext Links
3. Using Session Variables
4. Reading Parameters from a Hypertext Request
5. Inserting Data into a Database
6. Deleting Data from a Database
7. Building a Form to Collect User Input
8. Updating Data in a Database.

Accessing Database Data

An Active Server Page is simply an HTML document with an embedded script that is interpreted by the web server. The script is executed after the browser requests the page and before the server dispatches the page. Thus,

the script can be used to specify the content of the page. All ASPs must have the extension ".asp" as a part of the file name. This tells IIS to examine the page for script elements and to process those elements before dispatching the page.

In this example, we will build a page that displays Mama Vallone's menu using the information stored in MS SQL Server. We begin with the following HTML document:

```
<html>
<head>
<title>
Mama Vallone's Take-Out Menu
</title>
</head>
<body bgcolor=#f0f0b8>
</body>
</html>
```

This page has no text in the body of the document. We will generate the body entirely with the ASP script. To identify a script that is to be executed on the server, we use the server-side script tag: <% %>. This is not an HTML tag because it is never interpreted as part of the HTML document. Instead, the web server executes the script as the document is being built to respond to the browser's request. The server-side script tag and its contents will be removed from the source of the ASP document so the browser never receives the code of the script. IIS supports server-side scripts written in either VBScript or Jscript. This tutorial uses VBScript exclusively.

We begin our discussion of database access by introducing Server.CreateObject. Server is one of ASP's five built-in objects. It has a method called CreateObject which is used to instantiate objects which are not built in. In this case, we will use Server.CreateObject to instantiate two Active Data Objects (ADO) components, a connection and a recordset. The code to accomplish this is as follows:

```
Set conn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.RecordSet")
```

The set statement is used to assign the reference of an object to a variable so that object can be used in the script. In this example "conn" and "rs" are variables which now reference the newly created connection and recordset objects. In Active Server Pages, variables are untyped, thus they do not need to be declared except to control scope. Once these statements are inserted into the server-side script tag and into the HTML document, we have created and ASP document, which looks as follows:

```
<html>
<head>
<title>
Mama Vallone's Take-Out Menu
</title>
</head>
<body bgcolor=#f0f0b8>
<%
 Set Conn = Server.CreateObject("ADODB.Connection")
 Set rs = Server.CreateObject("ADODB.RecordSet")
%>

</body>
</html>
```

Since this is now an ASP document, its file name needs the ".asp" extension such as "menu.asp."

So far, the script has only created objects that can be used to access data that resides in a database. To access those data, we need to open the connection and then open the recordset. Each of these objects has an open method. To open the connection, we first need an ODBC connection to the database. This can be established as a System DSN (data source name), as a File DSN, or as a Connection String. In this example, we will use a File DSN. Although creating a File DSN is beyond the scope of this tutorial, it is fairly simple and can be accomplished using the ODBC 32 tool in the Control Panel.

To open the connection object, we supply the File DSN, the user name, and the password ass seen here:

```
Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"
```

To open the recordset object, we supply an open connection object and query to specify the records which will comprise the recordset.

```
query="select * from menuitem order by menugroup desc, itemName"
rs.open query, conn
```

At this point, we have written the code necessary to access a set of records from a database, but we have not specified any way to include those records in the html response that will be sent to the browser. To use VBScript to include a value in the response document, we use the write method of the built-in object called response. To open a table tag from inside a server-side VBScript script, the following syntax would be used:

```
response.write "<table border=1>"
```

To access a field of the current record of a recordset, we use the variable which references the recordset followed by either a number that identifies the column (beginning with zero) or a string that evaluates to the field name. If a record set has at least three fields and the third field is named "Price" then the following statements would both include the value from the third field of the current record of the recordset named "rs" in the html response:

```
response.write rs("price")
response.write rs(2)
```

Since we can only access fields from the current record of a record set, we need some way to iterate through the recordset. The recordset object has a "moveNext" method which allows us to move to the next record and an "EOF" method which indicates if the record has moved past the last record. Combining these with VBScript

Do..Loop loop, we can effectively iterate through the recordset as follows:

```
do until rs.eof
  'code to execute for each record goes here
  rs.movenext
loop
```

At this point, we have seen everything we need to print the contents of menu table in the menu.asp web page. The code to do this is a follows:

```
response.write "<table border=1>"
do until rs.eof
  response.write "<tr>"
  response.write   "<td>"
  response.write     rs("itemName")
  response.write   "</td>"
  response.write   "<td>"
  response.write     rs("Price")
  response.write   "</td>"
  response.write   "<td>"
  response.write     rs("description")
  response.write   "</td>"
  response.write "</tr>"
  rs.movenext
loop
response.write "</table>"
```

This code creates an HTML table with the item name, price and description of each item in the menuItem table.

The last step is to close the recordset and connection objects. Each object has a close method. The recordset should be closed first as follows:

```
rs.close
conn.close
```

The menu shown in the right frame of Figure 1 has one additional quality, a row is added to the table each time the recordset moves to a record with a different value in the "menuGroup" field. The code for this is included in Appendix with the listing for menu.asp.

Adding Parameters to Hypertext Links

Menu.asp from the previous sections allows a user with a web browser to see the current menu offered by Mama Vallone's. Since it is dynamically created using data from the database, it will always be as current as the menu information in the database. The next step in creating this electronic commerce application is to give the user the ability to select an item from the menu to be included in the order. We will create another ASP to show the currently selected items called cart.asp; however, first we will modify menu.asp to allow the user to click on the name of a menu item and add it to their order.

In order to accomplish this, we will make the menu item name a hyperlink using the anchor tag. The link will reference cart.asp and will include necessary instructions to add a particular item to the order. This information will be included in the anchor tag's hypertext reference as parameters.

The normal form of the anchor tag is as follows:

```
<a href=http://www.gove.net>
  Gove's home Page
</a>
```

The "href" parameter holds the URL of the page that should be displayed when the user follows the link. To add a parameter to the URL, one simply appends a question mark (?) followed by the name of the parameter followed by the equals sign (=) followed by the parameter's value. Hyperlink parameters can only include certain characters. Characters which are not allowed can be coded. For example, the space character is not allowed. If a parameter's value needs to include a space, one can replace the space with the plus sign (+) which will be interpreted as a space. Although it will not be used in the example of this tutorial, the built-in server object has a method called "urlencode" which will properly encode hyperlink parameters and their values.

The question mark (?) only works to identify the first parameter. All subsequent parameters are identified by the ampersand character (&). The link to cart.asp that has a parameter named "mode" with a value of "add" and a parameter named "menuItemID" with a value of "64" is as follows:

```
<a href=cart.asp?mode=add&menuItemID=64>
Cappuccino
</a>
```

Our menu.asp already writes the name of each menu item as seen here:

```
response.write   "<td>"
response.write     rs("itemName")
response.write   "</td>"
```

We have only to add the code to for the anchor tag with the specific menuItemID. This is accomplished with the following code:

```
response.write   "<td>"
response.write    "<a target=cartPage href=cart.asp?mode=add&menuItemID="
response.write    rs("menuitemID")
response.write    ">"
response.write     rs("itemName")
response.write    "</a>"
response.write   "</td>"
```

The "target" parameter directs the browser to display the page returned from the web server as a result of following the hyperlink to the frame named "cartPage."

With hyperlinks that have parameters, an ASP can read the values of the parameters and act differently based on those values. In this case, cart.asp will add to the order

the item that corresponds to the value passed with the "menuItemID" parameter.

It would be useless to pass parameters to an ASP if there were not some way to interpret those parameters' values. ASPs has a built-in object called "Request" which allows a developer to access the parameters passed as a part of an HTTP request. Request has several methods; however, this tutorial will deal only with an abbreviated syntax which accesses parameter values from the HTTP request whether those values were embedded in a link or submitted as a part of a form. The syntax to reference these values is as follows:

```
request("parameterName")
```

To simply write the value of the a parameter named "mode" to the response document, the code is as follows:

```
response.write request("mode")
```

In this tutorial we will use the "mode" parameter to allow cart.asp to perform many different database tasks. Cart.asp will be responsible to create the order, to add items to the order, to remove items from the order, and to display the items currently on the order. While these tasks could be accomplished by several different pages, placing them in one ASP is very convenient for both development and maintenance.

To accomplish this, we will first read the value of the "Mode" parameter and convert it to upper case (to allows case insensitive comparisons). We will then assign a default value if the parameter was not passed and use a select . . . case statement to decide what to do based on the value of the "mode" parameter. The code which does this follows:

```
varMode = ucase(Request("Mode"))
if not varMode > "" then varMode = "DISPLAY"
select case varMode
  case "DISPLAY" :  DISPLAY
  case "ADD" :  ADD
  case "DELETE" :  DELETE
  case Else
    response.write "This page is only displayed when "
    response.write "there is a system logic error."
    response.write "<BR><BR>"
    response.write "Contact Mama Vallone"
end select
```

This code fragment reads the value of "mode" parameter and assigns its uppercase equivalent to a variable named "varMode." If nothing was passed to the "mode" parameter, then varMode is not greater than an empty string and is set to "DISPLAY," which is the default mode. The "select . . . case" statement then calls one of three sub procedures to accomplish a different task depending on the value of varMode. If an unanticipated value is passed to the mode parameter, an error message is displayed. With this "mode processor" in place, a separate sub procedure can be called for each value of the mode parameter. The sub procedures can be completely independent resulting in the same level of control that results in separate ASPs for each taks; however, in this approach, the functionality which is logically connected is located in the same place.

Using Session Variables

The web server which accepts the requests for Mama Vallone's will need to simultaneously accept input from many customers and keep track of who is ordering what. Fortunately, ASPs allow for session variables which hold their value even though a user is changing from one page to another. No matter which page in a site sets the value of a session variable, other pages in that site can read that value for the current session. A session is created when a user first accesses a page in a site and exists until the server has not received any requests from that browser instance for a specified period of time (default is 20 minutes). If a user closes his or her browser completely, and then reopens it and returns to the site, the server would create a separate session and the variables the former session would be inaccessible.

In this example, a session variable is convenient to hold the order identifier (saleID) that a particular customer is composing. By placing the saleID in a session variable, each time cart.asp needs to modify an order, it can simply refer to the session's saleID variable to identify the proper order to manipulate. Also, when a customer first comes to the site, saleID session variable will be unassigned. We can use this condition as a trigger to insert a record into the sale table and get the saleID that pertains to this customer's order.

Session variables are identified by the built-in object called "Session." The syntax for creating or accessing a session variable called "saleID" is as follows:

```
Session("saleID")
```

Using this notation, we can check to see if the saleID session variable has a value less than one (below the valid range for saleID) to determine if a new saleID should be generated. The code to do this follows:

```
if session("saleID") < 1 then
  session("saleid") = createOrder
end if
```

The createOrder function simply inserts a new record into the Sale table, then retrieves it to see what the automatically generated saleID is and returns that value. The createOrder function will be discussed in the next section.

Inserting Data into a Database

The createOrder function interacts with the database. Just like menu.asp, it must create a connection object to connect to the database and since it reads data from the

database, it also needs a recordset object. The code for this function is as follows:

```
function createOrder

    Set Conn = Server.CreateObject("ADODB.Connection")
    Set rs=Server.CreateObject("ADODB.RecordSet")

    Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"

      ip=request.servervariables("remote_addr")
      query="insert into sale (IPaddress) values('"&ip&"')"
      conn.execute(query)

      query = "select saleid from sale where ipaddress='"&ip&"'"
      query = query & " order by orderCreated desc"
      rs.open query, conn
      createOrder=rs("saleID")
    rs.close
    conn.close

end function
```

This function instantiates the objects necessary to access the database, opens the connection, obtains the IP address of the requesting browser, uses that IP address as the value for a field in the Sale table. Then it inserts a record into the Sale Table. When a record is inserted into the Sale table, it automatically gets a new saleID and the orderCreated field is assigned to the current date and time. This is done so when the function queries the database for orders created from a specific IP address and requests them in descending order by orderCreated, the one which was just inserted will appear at the top of the result set. The value of saleID is then passed back as the return value of the function.

This example introduces the "execute" method of the connection object. This method is used for all queries which do not retrun a results set, such as insert, update and delete queries.

Cart.asp has three more sub procedures: add, display, and delete. The add sub procedure is executed when the user clicks on the name of one of the menu items from menu.asp. It reads the value passed to the menuItemID parameter and uses it with the value of the saleID session variable to insert records into the table called "sale_item," which lists the items on a particular order. It then calls the "display" sub procedure, which writes the list of items currently on the order along with their prices and totals the order. As seen below, this sub procedure also uses the execute method of the connection object:

```
Sub Add
    Set Conn = Server.CreateObject("ADODB.Connection")
    Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"
      SaleID = session("SaleID")
      itemID = request("MenuItemID")
      query = "insert into Sale_Item(menuItemID, SaleID)"
      query = query & " values("&ItemID&","&SaleID&")"
      conn.execute(query)
    conn.close
    set conn=nothing
    display
end sub
```

The "display" sub procedure simply reads the list of menu items associated with a specific sale and writes them to the response document in an HTML table. This procedure introduces the native function "FormatCurrency" which accepts a numeric argument and returns a formatted string. The code for the "display" sub procedure is as follows:

```
Sub Display
    Set Conn = Server.CreateObject("ADODB.Connection")
    Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"

    set rs=Server.CreateObject("ADODB.RecordSet")

    query = "select * from menuitem inner Join sale_Item on"
    query = query & " menuitem.menuItemID=sale_item.menuItemID"
    query = query & " where sale_item.saleid=" & session("saleid")
    query = query & " Order by itemname"

    rs.open query, conn

    response.write "<table width=100% Border=1>"
    response.write "<tr><th>Item</th><th>Price</th></tr>"
    total=0
    do until rs.eof
      response.write "<tr>"
      response.write   "<td>"
      response.write    "<a target=cartPage href=cart.asp?"
      response.write    "mode=delete & lineItemID="
      response.write    rs("lineItemID")
      response.write    ">"
      response.write     rs("itemName")
      response.write    "</a>"
      response.write   "</td>"
      response.write   "<td align=right valign=top>"
      response.write    rs("price")
      response.write   "</td>"
      response.write "</tr>"
      total=total + cdbl(rs("price"))
      rs.movenext
    loop
    response.write "<tr><td><b>Total</b></td><td align=right>"
    response.write formatCurrency(total)
    response.write "</td></tr>"
    response.write "</table>"

    rs.close
    conn.close

end sub
```

Deleting Data from a Database

The process for deleting records from a table in a database is virtually identical to the process for inserting records into a table. The difference is found in the query which is passed to the execute method of the connection object. The code for the "delete" sub procedure of cart.asp follows:

```
Sub Delete
    Set Conn = Server.CreateObject("ADODB.Connection")
    Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"
      SaleID = session("SaleID")
      itemID = request("lineItemID")
      query="delete from Sale_Item where lineItemID = "&ItemID
      conn.execute(query)
    conn.close
    set conn=nothing
    display
end sub
```

This procedure is called when the user clicks on the name of a menu item that is in the list of selected items. This completes the discussion of the code for cart.asp. The reader is encouraged to refer to the appendix to review the code as it is assembled in the file.

Building a Form to Collect User Input

We've discussed one way to assign the values to parameters in an HTTP request—specifying them as a part of a hyperlink. The other way is to allow users to enter them on an HTML form. While a discussion of the HTML tags used in conjunction with the <form> tag is beyond the scope of this tutorial, a simple form is used in order.htm as seen below:

```
<form action=order.asp method=post target=_top>
 <table border=0>
  <tr><td>First Name</td>
   <td><input type=text size=20 name=fname></td></tr>
  <tr><td>Last Name</td>
   <td><input type=text size=20 name=lname></td></tr>
  <tr><td>Phone</td>
   <td><input type=text size=20 name=phone></td></tr>
  <tr><td>Pick Up Time</td>
   <td><input type=text size=20 name=pickup></td></tr>
  <tr><td>Credit Card #</td>
   <td><input type=text size=20 name=CCNumber></td></tr>
  <tr><td>Exp. Date</td>
   <td><select name=ccExpMonth>
       <option value=1>1
       <option value=2>2
       <option value=3>3
       <option value=4>4
       <option value=5>5
       <option value=6>6
       <option value=7>7
       <option value=8>8
       <option value=9>9
       <option value=10>10
       <option value=11>11
       <option value=12>12
      </select>
      /
      <select name=ccExpYear>
       <option value=2000>2000
       <option value=2001>2001
       <option value=2002>2002
       <option value=2003>2003
       <option value=2004>2004
       <option value=2005>2005
       <option value=2006>2006
       <option value=2007>2007
       <option value=2008>2008
       <option value=2009>2009
       <option value=2010>2010
      </select>
   </td></tr><tr><td></td>
   <td align=right>
   <input type=submit value="Submit Order">
   </td></tr></table>
</form>
```

The "target" parameter of the <form> tag on the first line indicates in which frame the resulting document should be displayed. Its value (_top) is a reserved value which indicates that the response should replace the topmost frameset in which the current page is embedded. The "action" parameter of the <form> tag indicates that order.asp is the file that will process the values entered in the form. The <input> tag and the <select> tag each have a name parameter. The value of the name parameter along with the value entered by the user for the input or select elements are passed to the web server. The ASP which is specified in the action parameter of the <form> tag accesses these values in the same manner as discussed in the section "Reading Parameters from an HTTP Request."

Updating Data in a Database

Updating database data is very similar to inserting and deleting—they all use the execute method of the connection object. This can be seen in the source of order.asp as follows:

```
<%
  if request("Fname")="" or _
    request("Lname")="" or _
    request("phone")="" or _
    request("Pickup")="" or _
    request("ccnumber")="" or _
    request("ccExpMonth")="" or _
    request("ccExpYear")=""       Then

    response.write "All infomration is required. Use your"
    response.write " browser's back button to try again."

  else
    Set Conn = Server.CreateObject("ADODB.Connection")
    Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"
     SaleID=session("SaleID")
     itemID=request("lineItemID")
     query = "update sale set fname='"&request("fname")&"',"
     query = query & " lname='"&request("lname")&"',"
     query = query & " phone='"&request("phone")&"',"
     query = query & " pickUpTime='"&request("Pickup")&"',"
     query = query & " CreditCardNumber='"&request("ccnumber")&"',"
     query = query & " ExpDate='"&request("ccExpMonth")
     query = query & "/"&request("ccExpYear")&"'"
     query = query & " where Saleid="&session("saleid")
     conn.execute(query)
    conn.close
    set conn=nothing

    response.write "<br><center><b>Thank you for your "
    response.write "order.</b></center><br><br>Feel free to "
    response.write "<a href=default.htm>create another order.</a><br>"
    session.abandon

  end if
%>
```

Notice that this code begins by checking to see if any of the values passed from the form are empty. If any are, then the response document simply includes an error message to be rendered by the browser. More sophisticated data validity measures can be implemented. For example, email addresses can be checked to see if they contain the "@" symbol, telephone numbers can be checked to make sure the are a certain length and contain no letter and so on. Also notice that after an order is successfully updated with the order information, the session is abandoned (third line from the end). This

destroys all session variables, so if the user places a new order, cart.asp will create a new order.

This concludes the discussion of the elements of this electronic commerce application. The application can be examined at http://dev.csom.umn.edu/mamav. The application and the source code will be available through May of 2001.

References

Kosiur, D. *Understanding Electronic Commerce*. Microsoft Press: Redmond WA, 1997.

Ladd, E., Ablan, J., Banick, S., Cassady-Dorion, L., Chandak, R., Doherty, D., Ellsworth, M., Santa Maria, P., Morgan, M., Morrison, M., Niles, M., Sloan, B., Sutter, R. *Using HTML4, Java 1.1, and JavaScript 1.2.* Que: Indianapolis IN, 1998.

Appendix

These are the code listings from the complete project.

default.htm

```
<HTML>
<HEAD>
<TITLE>Mama Vallone's Take Out Order Form</TITLE>
</HEAD>
<FRAMESET cols="300,*" border="0" frameborder="no" framespacing="0">
  <FRAME name="orderFrame" src="orderFrame.htm" marginwidth="0" marginheight="0"
      scrolling="No" noresize framespacing="0" frameborder="NO">

  <FRAME name="menu" src="menu.asp" marginheight="0" scrolling="Auto" frameborder="NO"
       noresize framespacing="0" marginwidth="0">
</FRAMESET>
</HTML>
```

orderFrame.htm

```
<HTML>
<HEAD>
<TITLE>Mama Vallone's Take Out Order Form</TITLE>
</HEAD>
<FRAMESET rows="100,*,200" border="0" frameborder="no" framespacing="0">
  <FRAME name="Title" src="title.htm" marginheight="0" scrolling="No" frameborder="NO"
         noresize framespacing="0" marginwidth="0">
  <FRAME name="cartPage" src="cart.asp" marginheight="0" scrolling="Auto" frameborder="NO"
         noresize framespacing="0" marginwidth="0">
  <FRAME name="OrderPage" src="order.htm" marginwidth="0" marginheight="0" scrolling="No"
         noresize framespacing="0" frameborder="NO">
</FRAMESET>
</HTML>
```

menu.asp

```
<html>
<head>
<title>Mama Vallone's Take-Out Menu</title>
</head>
<body bgcolor=#f0f0b8>

<%    Set Conn = Server.CreateObject("ADODB.Connection")
      set rs = Server.CreateObject("ADODB.RecordSet")

      Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"
      query="select * from menuitem order by menugroup desc, itemName"
      rs.open query, conn

      currentGroup=""
      response.write "<table border=1>"
      do until rs.eof
        if currentGroup<>rs("menuGroup") then
          response.write "<tr><td colspan=3 align=center bgcolor=#f0e0f8>"
          response.write      "<font color=#800000 size=4>"
          response.write      rs("menuGroup")
          response.write      "</font</td></tr>"
        end if
        currentGroup=rs("menuGroup")
        response.write "<tr>"
        response.write   "<td>"
        response.write     "<a target=cartPage href=cart.asp?mode=add&menuItemID="
        response.write     rs("menuitemID")
        response.write     ">"
        response.write      rs("itemName")
        response.write     "</a>"
        response.write   "</td>"
        response.write   "<td>"
        response.write      rs("Price")
        response.write   "</td>"
        response.write   "<td>"
        response.write      rs("description")
        response.write   "</td>"
        response.write "</tr>"
        rs.movenext
      loop
      response.write "</table>"

      rs.close
      conn.close
%>
</body>
</html>
```

cart.asp

```
<html>
<head>
<title>
Mama Vallone's Take-Out Menu
</title>
</head>
<body bgcolor=#f0f0b8>

<%
if session("saleID") < 1 then
  session("saleid")=createOrder
end if

' -------------------- Mode Processor --------------------
varMode = ucase(Request("Mode"))
if not varMode > "" then varMode = "DISPLAY"
select case varMode
   case "DISPLAY"      :  DISPLAY
   case "ADD"          :  ADD
   case "DELETE"       :  DELETE

   case Else
      response.write "This page is only displayed when "
      response.write "there is a system logic error."
      response.write "<BR><BR>"
      response.write "Contact Mama Vallone"
end select

' -------------------- Sub Procedures --------------------
Sub Display
      Set Conn = Server.CreateObject("ADODB.Connection")
      Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"

      set rs=Server.CreateObject("ADODB.RecordSet")

      query = "select * from menuitem inner Join sale_Item on"
      query = query & " menuitem.menuItemID=sale_item.menuItemID"
      query = query & " where sale_item.saleid="&session("saleid")
      query = query & " Order by itemname"

      rs.open query, conn

      response.write "<table width=100% Border=1>"
      response.write "<tr><th>Item</th><th>Price</th></tr>"
      total=0
      do until rs.eof
        response.write "<tr>"
        response.write   "<td>"
        response.write     "<a target=cartPage href=cart.asp?"
        response.write     "mode=delete&lineItemID="
        response.write     rs("lineItemID")
        response.write     ">"
        response.write      rs("itemName")
        response.write     "</a>"
        response.write   "</td>"
        response.write   "<td align=right valign=top>"
        response.write     rs("price")
        response.write   "</td>"
        response.write "</tr>"
        total=total + cdbl(rs("price"))
        rs.movenext
      loop
      response.write "<tr><td><b>Total</b></td><td align=right>"
      response.write formatCurrency(total)
      response.write "</td></tr>"
      response.write "</table>"

      rs.close
      conn.close

end sub
```

cart.asp (continued)

```
    Sub Add
        Set Conn = Server.CreateObject("ADODB.Connection")
        Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"
          SaleID = session("SaleID")
          itemID = request("MenuItemID")
          query = "insert into Sale_Item(menuItemID, SaleID)"
          query = query & " values("&ItemID&","&SaleID&")"
          conn.execute(query)
        conn.close
        set conn=nothing
        display
    end sub


    Sub Delete
        Set Conn = Server.CreateObject("ADODB.Connection")
        Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"
          SaleID = session("SaleID")
          itemID = request("lineItemID")
          query="delete from Sale_Item where lineItemID = "&ItemID
          conn.execute(query)
        conn.close
        set conn=nothing
        display
    end sub


    function createOrder
        Set Conn = Server.CreateObject("ADODB.Connection")
        Set rs=Server.CreateObject("ADODB.RecordSet")
        Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"
           ip=request.servervariables("remote_addr")
           query="insert into sale (IPaddress) values('"&ip&"')"
           conn.execute(query)
           query = "select saleid from sale where ipaddress='"&ip&"'"
           query = query & " order by orderCreated desc"
           rs.open query, conn
           createOrder=rs("saleID")
        rs.close
        conn.close
    end function

    %>

    </body>
    </html>
```

title.htm

```
    <html>
    <head>
    <title>
    Mama Vallone's Take-Out Menu
    </title>
    </head>
    <body bgcolor=#f0f0b8>
    <center>
    <font size=6 color=blue>Mama Vallone's</font><br>
    <font size=4 color=blue>Steak House and Inn</font><br>
    <font size=5 color=#800000>Take-Out Orders</font><br>
    </center>

    </body>
</html>
```

Order.htm

```
<html>
<head>
<title>
Mama Vallones Take-Out Menu
</title>
</head>
<body bgcolor=#f0f0b8>
<form action=order.asp method=post target=_top>
  <table border=0>
    <tr><td>First Name</td>
      <td><input type=text size=20 name=fname></td></tr>
    <tr><td>Last Name</td>
      <td><input type=text size=20 name=lname></td></tr>
    <tr><td>Phone</td>
      <td><input type=text size=20 name=phone></td></tr>
    <tr><td>Pick Up Time</td>
      <td><input type=text size=20 name=pickup></td></tr>
    <tr><td>Credit Card #</td>
      <td><input type=text size=20 name=CCNumber></td></tr>
    <tr><td>Exp. Date</td>
      <td><select name=ccExpMonth>
            <option value=1>1
            <option value=2>2
            <option value=3>3
            <option value=4>4
            <option value=5>5
            <option value=6>6
            <option value=7>7
            <option value=8>8
            <option value=9>9
            <option value=10>10
            <option value=11>11
            <option value=12>12
         </select>
         /
         <select name=ccExpYear>
            <option value=2000>2000
            <option value=2001>2001
            <option value=2002>2002
            <option value=2003>2003
            <option value=2004>2004
            <option value=2005>2005
            <option value=2006>2006
            <option value=2007>2007
            <option value=2008>2008
            <option value=2009>2009
            <option value=2010>2010
         </select>
      </td></tr><tr><td></td>
      <td align=right>
      <input type=submit value="Submit Order">
      </td></tr></table>
  </form>
</body>
</html>
```

Order.asp

```
<html>
<head>
<title>
Mama Vallones Take-Out Menu
</title>
</head>
<body bgcolor=#f0f0b8>
<%
    if request("Fname")="" or _
       request("Lname")="" or _
       request("phone")="" or _
       request("Pickup")="" or _
       request("ccnumber")="" or _
       request("ccExpMonth")="" or _
       request("ccExpYear")=""         Then

       response.write "All infomration is required. Use your"
       response.write " browser's back button to try again."

    else
       Set Conn = Server.CreateObject("ADODB.Connection")
       Conn.Open "filedsn=d:\virtualweb\mamav\mamav.dsn", "mamav", "sql"
         SaleID=session("SaleID")
         itemID=request("lineItemID")
         query = "update sale set fname='"&request("fname")&"',"
         query = query & " lname='"&request("lname")&"',"
         query = query & " phone='"&request("phone")&"',"
         query = query & " pickUpTime='"&request("Pickup")&"',"
         query = query & " CreditCardNumber='"&request("ccnumber")&"',"
         query = query & " ExpDate='"&request("ccExpMonth")
         query = query & "/"&request("ccExpYear")&"'"
         query = query & " where Saleid="&session("saleid")
         conn.execute(query)
       conn.close
       set conn=nothing

       response.write "<br><center><b>Thank you for your "
       response.write "order.</b></center><br><br>Feel free to "
       response.write "<a href=default.htm>create another order.</a><br>"
       session.abandon

    end if
%>

</body>
</html>
```