Lab 7.1: Using ActiveX Data Objects

To complete the lab exercises in this chapter, you must have the required software. For detailed information about the labs and setup for the labs, see Labs in this course.

Objectives

In this lab, you will use ADO code in Active Server Pages to retrieve and update a database. You will also modify existing .asp files. Optionally, you will add code to handle possible database errors.
After completing this lab, you will be able to use ADO to:

- Create a connection to a database.
- Create a Recordset object and retrieve records.
- Add a new record to a table.

Prerequisites

Before beginning this lab, you should be able to:

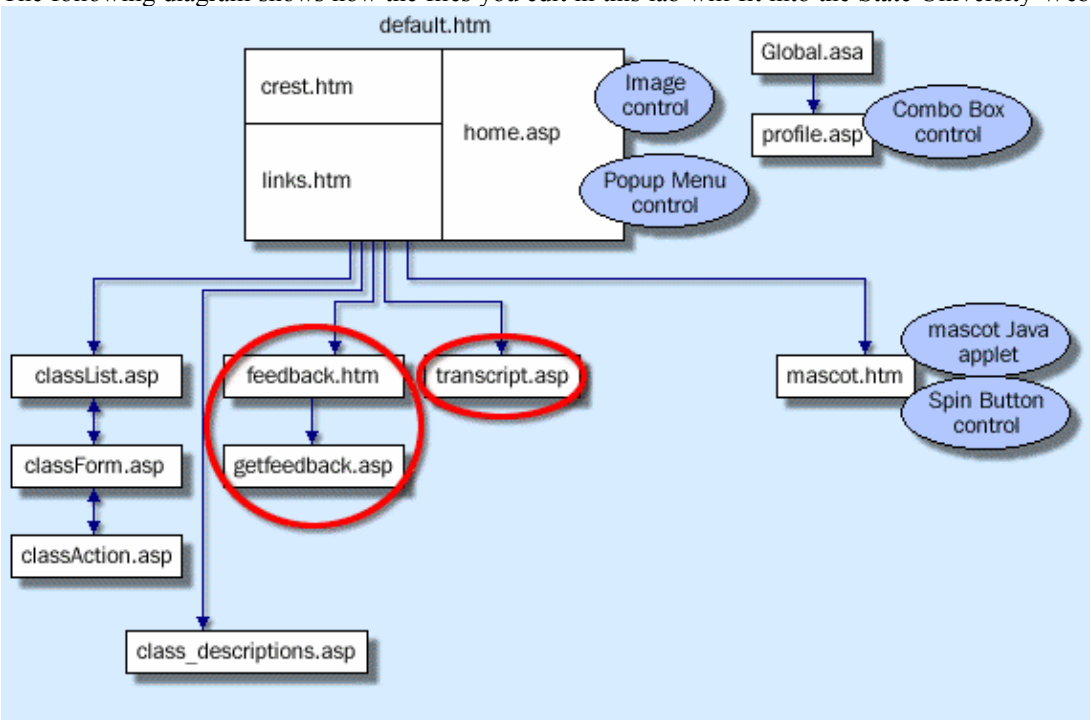
- Create Web pages, .asp files, and projects in Microsoft Visual InterDev.
- Write server-side script in an Active Server Page.
- Call an ActiveX server component from an .asp file.

Lab Setup

To complete this lab, you need the following:

- The Visual InterDev State University Project
- The State University Microsoft SQL or Microsoft Access Database

The following diagram shows how the files you edit in this lab will fit into the State University Web site.



Estimated time to complete this lab: **60 minutes**

Note There are project and solution files associated with this lab. If you installed the labs during Setup, these files are in the folder <*Install Folder*>\Labs\Lab07.1 on your hard disk. If you did not install the labs during Setup, you can find them in the \Labs\Lab07.1 folder of the *Mastering Web Site Development* CD-ROM.

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 7: Creating Database-Aware Web Pages.

Exercise 1: Retrieving Records

In this exercise, you will create an Active Server Page that uses ADO to retrieve the classes and grades for the student.

Exercise 2: Adding Records

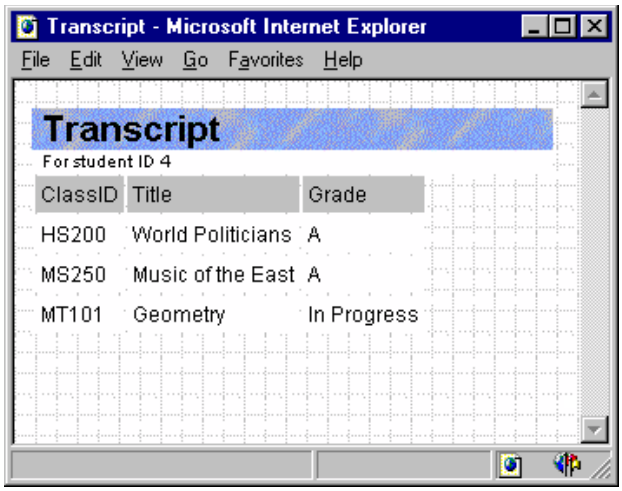
In this exercise, you will create an Active Server Page that obtains feedback information from a student and uses ADO to add the information to the Feedback table in the database.

Exercise 3 (Optional): Handling Database Errors

In this exercise, you will modify your .asp file to handle errors.

Exercise 1: Retrieving Records

State University students will be able to retrieve their transcripts online. Once they have entered their student ID through the profile.asp file, they can see a list of classes they have completed.
In this exercise, you will create the Active Server Page that uses the student ID from the session object to retrieve the classes and grades for that student. Using ADO, you will establish a connection to the State University database, retrieve the transcript for the student into a recordset, and then print the records into an HTML table.
The following illustration shows how the completed Active Server Page will look when returned to the student.



➢ **Add files to project**

1. Using Microsoft Visual InterDev, open the StateU project.
2. In the \MWD\Labs\Lab07.1 folder, add the file getfeedback.asp to the root of the StateU project. This file replaces the one you created in Lab 2.

➢ **Create the data connection**

1. In the file transcript.asp, just before the <HTML> tag, add server script that retrieves the student ID from the session object and places it in a variable named frmStudentID.
2. On the line after the </TABLE> tag, type **<%** to begin entering server script.
3. Create a **Connection** object named conn, by using the **CreateObject** function.
4. Using the session values stored in the global.asa file for the State University connection, set the **ConnectionTimeout** and **CommandTimeout** properties for the object.
5. Invoke the **Open** method to establish the connection. Use the session values stored in the global.asa to set the connection string, user name, and password.

➢ **Retrieve the transcript records**

1. Use the **CreateObject** method to create a **Recordset** object variable named rsTranscript.
2. Set the **ActiveConnection** property of rsTranscript to the conn **Connection** object.
3. To retrieve the transcript records for the student ID, invoke the **Open** method of rsTranscript.

   The query should retrieve the classid, title, and grade fields from the Enrollment table and Classes table, where StudentID equals frmStudentID.

➢ **Write the transcript records to HTML**

1. In the Active Server Page, after the server script from the previous procedure, use <TABLE> tags to create an HTML table that has three columns. In the first row, enter the column headers ClassID, Title, and Grade.
2. After the first row of the table, enter server script that defines a **Do Until...Loop** statement to loop through the recordset. Use the **EOF** property of the recordset to determine when to exit the loop.
3. In the loop, write script that prints the ClassID and Title fields from the rsTranscript recordset. Use the beginning and ending <TD> tags around the fields to print the fields in the table columns.
4. To convert the Grade field from a number to a letter before printing, write a **Select Case** statement that converts 4, 3, 2, 1, 0 to A, B, C, D, and F, respectively, and then print the letter grade in the third column. Include a Case Else clause to check for a no grade.
5. In the last step of the loop, print HTML tags to start a new table row, and add server script to move to the next record in the rsTranscript recordset.
6. Add a link to links.htm that links the text Get Transcript to transcript.asp.

➢ **Test your solution**

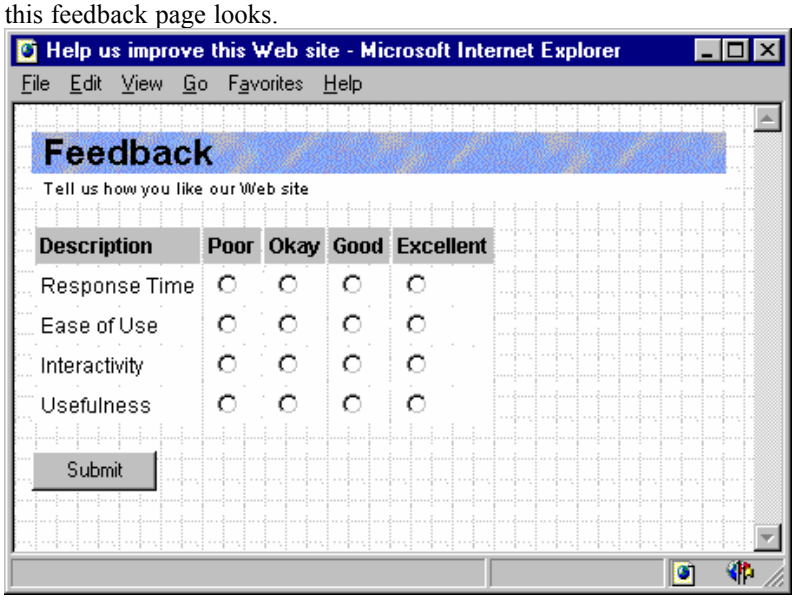
- Save the .asp file and test it by previewing the transcript.asp file.
  Note: The transcript.asp file uses a student ID provided on the login form. If you preview the transcript.asp before logging in with a valid student ID, you will receive an error message.

Exercise 2: Adding Records

The State University Web site includes a Web page that students can use to submit comments about how well the StateU Web site is filling their needs.

In this exercise, you will create an Active Server Page that uses ADO to obtain feedback submitted by students, and add it to the Feedback table in the StateU database. You will use the design-time **Include** control to add the file adovbs.inc, which will enable you to use the ADO constants in the server script.

The feedback page (feedback.htm) is provided for you in this exercise. The following illustration shows how this feedback page looks.



When a student submits the file feedback.htm, the file getfeedback.asp will run.

A starting point for the getfeedback.asp file has been provided for you. You will modify this file to retrieve the information from the form, update the database, and provide a confirmation message to the student.

➢ **Include the adovbs.inc file**

1. In Visual InterDev, open the StateU project.
2. Add the adovbs.inc file to the root of the StateU project.

Note The file adovbs.inc is installed in the \InetPub\ASPSamp by the Setup program of IIS 3.0 Active Server Pages. If you did not install the .asp sample files with IIS, you can copy the file adovbs.inc from the folder \MWD\Labs\Lab07.1.

---

3. Open the file getfeedback.asp to edit it in Visual InterDev.
4. On a blank line just above the <HTML> tag, insert the design-time **Include** control.
5. Set the source file of the **Include** control to adovbs.inc, and close the Properties Page and Object Editor.

   The information needed to include the control will be added to your file.
6. After the information for the **Include** control, add server script to obtain the form parameters from the **Request** object.
7. Store the four parameters that are passed, Response, Ease, Interactive, and Useful, in the variables **frmResponse**, **frmEase**, **frmInteractive**, and **frmUseful**, respectively.

➢ **Create the data connection**

1. Create a **Connection** object named conn by using the **CreateObject** method.

2. Using the session values stored in the global.asa for the State University connection, set the **ConnectionTimeout** and **CommandTimeout** properties for the object.
3. Establish the connection, by invoking the **Open** method. Use the session values stored in the global.asa for the connection string, user name, and password.

➢ Add a new record

1. To add a new record, call the Execute method on the conn object to run an SQL query that inserts the form parameters into the database. The following sample code shows how your query should look.

```
strSQL = "INSERT INTO feedback " & _
   "(Response, Useful, Interactive, Ease)" & _
   " VALUES (" & frmResponse & "," & frmUseful & _
   "," & frmInteractive & "," & frmEase & ")"
```

➢ Test your solution

1. Save the file getfeedback.asp, and test it by previewing the feedback.htm file and submitting the form.

   A response page should be returned to you.
2. In Data View, open the Feedback table and verify that the new record has been added.

Exercise 3 (Optional): Handling Database Errors

Error handling is an important part of writing robust Active Server Pages. In this exercise, you will modify the file getfeedback.asp to handle an error in adding a new record.

You will add a new file to the StateU project named error.asp. When an error is detected, getfeedback.asp will store the error information in session variables, and then redirect control to error.asp. The file error.asp will retrieve the error information from the session variables, display a message, and print the error title and error description in the HTML page.

This exercise demonstrates a simple error-handling approach for State University. However, in a real world application, you may want to use more powerful techniques, such as logging errors to a Web server log or by taking different actions based on the error.

- **Add files to the project**

1. In Visual InterDev, open the StateU project.
2. Add the file error.asp from the folder \MWD\Labs\Lab07.1 to the root of the StateU project.

- **Check for errors when adding a new record**

1. Open the file getfeedback.asp to edit it in Visual InterDev.
2. To change the default error-handling behavior, add the statement "On Error Resume Next" just before the statement that establishes a database connection.
3. Find the server script that adds the new record with the SQL insert statement.
4. After the **Execute** method is called to run the SQL insert, add an **If** statement to check if an error has occurred.
5. If an error has occurred:
   a. Set the ErrorTitle session variable to Feedback Form.
   b. Set the ErrorText session variable to the **Description** property of the **Err** object.
   c. Invoke the **Response.Clear** method to clear the HTML Response.
   d. Invoke the **Response.Redirect** method to redirect the .asp file to error.asp.

- **Test the error-checking code**

1. Save the file getfeedback.asp.
2. Preview the file feedback.htm in the browser and submit it without clicking any radio buttons.

   This will force the parameters to be empty which will cause the SQL insert to fail. Verify that the error.asp page displays properly.