Active Server Pages

Active Server Pages are Microsoft’s newest server-based technology for building dynamic and interactive web pages for use on the World Wide Web or an intranet.

Active Server Pages allows the programmer to add server side script embedded within web pages that can build or add on to static web pages. With a scripting ability, you can add data retrieved from a database, or build new pages based on the users responses to previous web pages. Intranet websites when combined with ASP programming can be used to deliver up-to-the minute information to everybody connected to the web server. This could mean customer information, bank statements, current news, or anything that can be updated on a per minute basis. Another nice thing about ASP is that it can work within existing web designs already in place. The only requirement is that the web server is Microsoft IIS 4 and have service pack 2 installed.

The dynamic nature of ASP pages come from the underlying scripting languages involved. ASP currently supports Vbscript and Javascript. You might be thinking "Javascript is already available to web pages and has been supported by browsers for quite some time", this is true but Javascript from within ASP pages is executed on the server and can do quite a bit more. Vbscript also gives you the facility to create instances of objects that are created in just about any other programming environment, such as Visual C++, Visual Basic.

ASP is also much easier to use than CGI using perl or C++. It already has objects associated with receiving information via Query String and through Form postings. It also facilitates using cookies easily with the same API used for Query Strings and Form data. With ODBC drivers installed on the web server, talking to a database is as simple as executing SQL commands and displaying the information on your web pages with whatever formatting you want to use. ASP pages work almost exactly the same as html pages except that the actual code is hidden from the person at the browser level. The only thing that they see are the results from the asp code, also browser side javascript works exactly the same way that it did in standard html. And if you use cookies, you can dynamically create web pages that are customized on a per user basis. For example, you could have every person viewing your web page to see a different background or font. This could be accomplished in one of two ways. You could store a cookie on the users machine or you could have a database which held the preferences of the users. You can also track information about where certain people actually go within your web site, and possibly catalogue that information into a database of some sort. The best way to
create ASP pages is through Visual Interdev a product from Microsoft. When using Interdev it actually creates the directories directly on the web server (if you have access) and sets all the permissions so that the asp pages are automatically executed instead of downloaded when a request for that page is issued.

How to connect to an ODBC database in ASP

ActiveX Data Objects (ADO) are an easy-to-use yet extensible technology for adding database access to your Web pages. You can use ADO to write compact and scalable scripts for connecting to Open Database Connectivity (ODBC) compliant databases and OLE DB compliant data sources. If you are a scripter with a modest understanding of database connectivity, you will find ADO's command statements uncomplicated and approachable. Likewise, if you are an experienced database programmer you will appreciate ADO's advanced language neutral connection and query manipulation features.

The first thing we need to do is to create an object to connect to the database with. This is done like this:

Set DataConn=Server.CreateObject("ADODB.Connection").

Note: this isn’t a database connection yet, its just the object. The next thing is to open the connection to the database and logging in at the same time.

(DataConn.Open "DSN=Database_Name;UID=login_id;DBALIAS=Database_Name;", "login_id", "password")

This is the basic connection string and it will work 99.9% of the time. Database_Name is the name of the database as it appears in the ODBC settings of the web server. Login_id is exactly what you’d expect it to be, it’s the user id of the database, and password is the password needed to login as Login_id.

What does it mean to be "connected" to a database?" Well it means you are now logged in and can execute arbitrary commands that can either retrieve information, modify existing information, or to insert new information. You do this by issuing commands via SQL, a query language that is widely accepted as the norm.

Executing SQL in ASP

Now that we’ve connected to the database and have a basic understanding as to what this means, we are ready to start issuing commands to the database to get it to do what we want it to do. The commands are made through a query language called SQL (Structured Query Language). They are really easy to execute once you have a connection. For instance we called our connection Dataconn earlier, we’ll stick to this convention. To execute an SQL command structure the statement in this way: Dataconn.execute(SQL).

If you don’t have a solid foundation of SQL, I recommend a good book on SQL. One that I’ve used is SQL for Dummies. It is structured from a beginners perspective and still has enough detail to be used
as a reference by most database programmers.

**Storing "Select" results into a recordset**

First off, What is "Select?" Select, is the type of SQL statement that brings back data from a database. Since you don’t know how many records are coming back, or how big each record is, it is very difficult to just put them into an array or other standard datatype. How do we catch this incoming information then? Luckily almost all programming languages that offer database connectivity give you a data structure that will store this incoming information into an easy to use object. ASP gives us this object so that all we have to do is declare an object and set its value = to a Dataconn.execute(SQL) statement where the SQL is formatted as a select statement.

An example. SET RS_Info=DataConn.Execute("SELECT * FROM Contact_Info")

What does this do? Well first it orders the database to bring back EVERY column of EVERY record in the Contact_Info Table, and stores it into a variable called RS_Info. We could restrict this call some. Lets say we just wanted the first and last name of all the people in the Contact_Info table. How do we do just that? It’s easy:

SET RS_Info=DataConn.Execute("SELECT First_Name, Last_Name FROM Contact_Info")

Now What if we just wanted all the "Smith's" from that table, can we just do that? Of course we can. SET RS_INFO=DataConn.Execute("SELECT * FROM Contact_Info WHERE Last_Name = 'Smith'")

This will bring back all the information on all the Smith’s in the Contact_Info table.

**Navigating through a recordset**

If you just wanted to use the first name from each record in the recordset, you’d need a way to pull the name from the whole record. This is accomplished like this: NAME = RS_Info.Fields.Item("First_Name"). This will take the first name (provided its one of the columns you brought back in the select statement used to build the recordset) and store it into a variable called NAME. This will only pull the name out of the Current record the recordset is pointing to.

Looping through records in a record set: Lets say we’d need to display all the records in the Contact_Info table on a web page. Here’s a little snippet of Asp code that will get that done.

Set DataConn=Server.CreateObject("ADODB.Connection").
(DataConn.Open "DSN=Database_Name;UID=login_id; DBALIAS=Database_Name;", "login_id", "password")

Set RS_Info=DataConn.Execute("Select First_Name, Last_Name FROM Contact_Info")

DO WHILE NOT RS_EMPS.EOF
Response.Write RS_Info.Fields.Item("First_Name") + " " +
RS_Info.Fields.Item("Last_Name") + "<br>"
What this snippet does is writes one name per line and uses the HTML `<BR>` Tag to go to the next line. It loops through the entire recordset until EOF comes (similar to working through a file).

**Storing Information into a database**

There are 2 ways to store information in a database. One is to update existing information, the other is to insert new information into the database. First we’ll look at the update statement. It typically looks something like this:

```
Update Contact_Info Set First_Name = ‘Bruce’, Last_Name = ‘Moore’
WHERE ID = ‘1234’
```

This will update the person with id of 1234 to have the First_Name of Bruce and Last_Name of Moore. You can update all of the fields except the one that makes the record unique, in this case the id.

The insert statement is slightly different. It typically looks something like this:

```
Insert Into Contact_Info values( ‘Bruce’, ‘Moore’, ‘1234’).
```

This assumes that there are only 3 fields in the table. If there are more fields then you have to add all of them in the exact order that they are recognized in the database, you can’t switch Last and First Names.

**Closing connections and recordsets**

Closing a recordset and connection is easy. `RS_Info.close` will close the recordset. And `DataConn.close` will close the connection to the database. Closing them is very easy and also very important. If you don’t close the connections to the database then it thinks more sql is coming and is always listening to that connection. If too many connections happen and none of them close, eventually you’ll exceed the ability of the database to accept new connections and programs will start to fail. The errors won’t be obvious and hard to track down. So make absolutely sure that every time you open a connection to a database or recordset on an ASP page that you close it within that SAME page otherwise the connection might persist until the database is stopped and restarted, which could be a disaster in a real time environment such as a bank.

**Conclusion**

Asp is a growing develop environment in web programming because of its ease of doing things like handling form data, and queryString data. It even implements cookies in a very easy to use way. One of the advantages of ASP is that it lends very easily to database connectivity through the website.
Asp is really easy to design within Microsoft Visual Interdev. You can drag and drop all the controls you need and just write the code seamlessly behind it. The only thing you have to do is to add `<%@ Language=VBScript %>` at the very top and to include any asp code between `<% %>` tags. It embeds itself very nicely into standard html. The server blocks the user from ever seeing this extra code so unlike javascript it is completely hidden. The IIS server will parse and execute all asp code and modify the html code so that the results from the asp are incorporated into the final page layout.