APPENDICE B

Le Active Server Page

B.1 Introduzione ad ASP

La programmazione web è nata con la Common Gateway Interface. L'interfaccia CGI tuttavia presenta dei limiti: ad esempio anche per semplici elaborazioni vengono richieste molte risorse al server, e in certi casi ciò può portare a ritardi sulla risposta o ad un eccessivo carico sul server stesso. Si pensi al caso di invio di dati in qualche modo errati (ad esempio, una data non valida) che attivano un'istanza di applicazione CGI con l'unico effetto di notificare l'erroneo inserimento di dati e con la conseguenza di consumare risorse del server e della banda di trasmissione.

Per superare questo genere di problemi, la Microsoft, rilasciando la versione 3.0 di *Internet Information Server (IIS)*, ha introdotto sul mercato degli scripting server-side, la tecnologia *Active Server Pages (ASP)*, con l'idea di sfruttare non solo le potenzialità degli strumenti server dedicati alla connettività, ma anche attraverso la tecnologia *COM (Common Object Model)*, sfruttare tutte le risorse che il server Microsoft ha a disposizione e coinvolgendo anche i linguaggi di scripting come Jscript e Vbscript.

B.2 A cosa serve e come funziona

Le pagine ASP sono completamente integrate con i file HTML, non necessitano di compilazione, sono orientate agli oggetti e usano componenti server ActiveX.

La figura B1 sintetizza lo schema di funzionamento di un'applicazione ASP:

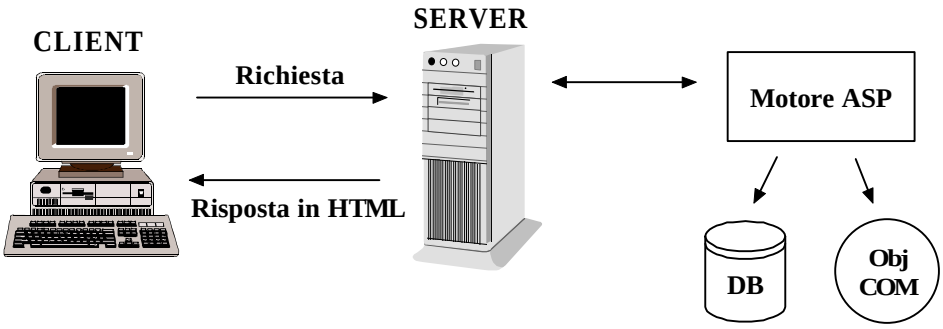

Figura B1:Funzionamento di un'applicazione ASP

1. Il browser richiede una pagina ASP;
2. il web server preleva la pagina ASP ed attiva l'interprete ASP per la sua pre-elaborazione, coinvolgendo eventualmente altri componenti presenti sul server;
3. il risultato dell'elaborazione viene inviato al browser tramite il protocollo HTTP.

B.3 Principale utilizzo della tecnologia ASP

L'uso più comune di questa tecnologia è quello di mediatore tra web server e una base di dati. Grazie ad *ActiveX Data Object* (ADO) è possibile accedere a qualsiasi base di dati ODBC o OLEDB compatibile, usufruendo dei vantaggi che questa tecnologia offre, tra i quali la semplicità di accesso ai dati e l'indipendenza dell'applicazione dal tipo di base di dati.

ASP supporta un'efficiente gestione delle connessioni sfruttando la funzione di *connection pooling* di ODBC 3.5. Questa funzione consente la condivisione di una connessione tra richieste provenienti da utenti diversi, evitando di creare nuove connessioni e riducendo il numero di quelle inattive.

B.4 Supporto dei linguaggi di scripting

ASP supporta in modo nativo due linguaggi di scripting, *VBScript* e *JScript*. Se non diversamente specificato, ASP interpreta il codice script che trova all'interno dei marcatori <% e %> come codice VBScript, il linguaggio predefinito. E' possibile

comunque modificare il linguaggio predefinito sia a livello di web server, cioè per tutte le applicazioni ASP gestite da IIS, che a livello di singola applicazione, cioè per tutte le pagine che compongono un'applicazione. E' comunque da tener presente che impostare un linguaggio predefinito può portare problemi di compatibilità con altri web server non adeguatamente configurati: infatti, portando un'applicazione ASP da un web server con JScript come linguaggio predefinito su un web server con VBScript predefinito è evidente che la sua esecuzione causa problemi di interpretazione sul server di destinazione. Tramite la direttiva *@ LANGUAGE* è possibile specificare il linguaggio da utilizzare all'interno di una determinata pagina ASP. Così, ad esempio, se una pagina ASP contiene come prima linea la direttiva

```
<% @ LANGUAGE = "JScript" %>
```

il motore ASP interpreterà il codice contenuto all'interno di quella pagina come codice JScript. Inoltre, all'interno di una pagina possono essere utilizzati linguaggi di scripting diversi, potendo sfruttare al massimo le caratteristiche proprie di ciascuno di essi. Utilizzando una versione arricchita del tag <SCRIPT> è possibile specificare che un blocco di codice deve essere interpretato secondo un determinato linguaggio:

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server" >
Codice script
</SCRIPT>
```

L'attributo RUNAT specifica a che livello deve essere interpretato lo script; se non viene impostato, lo script è destinato all'interpretazione da parte del browser.

Oltre a VBScript e JScript, è possibile utilizzare un qualsiasi linguaggio di scripting il cui *scripting engine* sia stato installato sul web server. Uno *scripting engine* è un interprete di un determinato linguaggio realizzato come oggetto COM secondo lo standard *ActiveX Scripting*.

B.5 La sintassi

B.5.1 Cenni preliminari

Dopo aver visto gli aspetti principali di ASP, cominciamo a studiarne la sintassi. Possiamo dividere una pagina ASP in tre parti:

1) Testo

2) Marcatori HTML

3) Comandi script

In un documento con estensione ".asp" è consentito utilizzare variabili, cicli e istruzioni di controllo, grazie alla possibilità di richiamare la sintassi di un linguaggio di scripting, come ad esempio il VBscript e il Jscript, ma anche perl e altri. La scelta del linguaggio dipende in primo luogo dalle necessità del programmatore e dal tipo di esecuzione che si vuole avere: se si vogliono eseguire gli script dal lato server è preferibile utilizzare il VBscript, mentre se ci si vuole affidare alla potenza degli "scripting engine" (motore che interpreta i comandi dei linguaggi di scripting e li esegue) dei singoli navigatori è sicuramente meglio utilizzare il Jscript, semplice ed efficace.

Il codice ASP è sempre delimitato da i due marcatori <% e %>. Ad esempio la seguente riga:

<% x="ciao" %>

assegna alla variabile x la stringa "ciao". Una pagina può essere costituita solamente da codice ASP, oppure avere ASP immerso nel codice HTML: in entrambi i casi l'estensione deve essere ".asp".

Come accennato, all'interno della pagina è possibile sempre inserire un codice in un linguaggio diverso da ASP: in questo caso deve però essere dichiarato dove tale codice va eseguito. Supponendo allora di voler usare uno script in Jscript, la giusta sintassi è la seguente:

<SCRIPT LANGUAGE = Jscript RUNAT= [server oppure client]>

Il valore di default per il parametro RUNAT è “server”. Una volta definita una funzione in un qualsiasi linguaggio di script, per richiamarla basta usare l’istruzione ***call***. Esiste un’altra istruzione, ***include***, che permette di inserire in un file asp, il contenuto di un file esterno che può essere di testo, html, asp, grafica o qualsiasi altro file presente sul server. La sintassi è la seguente:

<!--INCLUDE FILE="nomefile.est" -->

e va usata sempre al di fuori dei tag <%,%>, che delimitano gli script ASP.

B.5.2 ASP e le basi di dati

In questa sezione ci occuperemo di come ASP permette l’interfacciamento con le basi di dati ODBC compatibili.

Affinché gli utenti potessero interagire con i DBMS tramite un browser, ovvero permettere la consultazione on-line di basi di dati la Microsoft ha associato ai suoi web Server, oltre alla possibilità di utilizzare l’interfaccia ODBC, la tecnologia IDC (Internet Database Connector).

Internet Database Connector è uno strumento che può essere utilizzato per spedire query ad una base di dati e formattare i dati che ritornano, in una pagina HTML. In figura B2 è illustrato come IIS e Personal Web Server consentono l'accesso ad una base di dati.

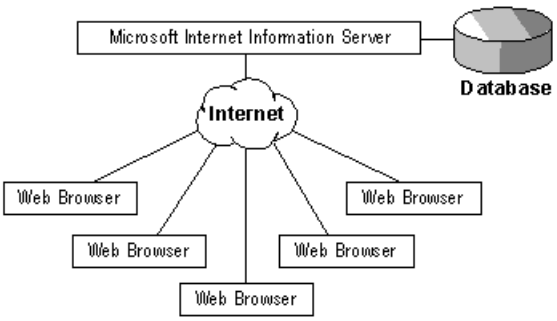

Figura B2: Accesso ad una base di dati con PWS

I web browser inviano richieste ai server Internet usando il protocollo HTTP; i server web rispondono alle interrogazioni con documenti HTML grazie proprio a IDC.

IDC usa due tipi di file per controllare l'accesso alla base di dati e la generazione del documento HTML di risposta:

1. file .*idc* Internet Database Connector;
2. file .*htx* HTML extension

I file **.idc** contengono le informazioni necessarie per connettersi all'appropriato ODBC data source e consentire l'esecuzione di istruzioni SQL. Inoltre questi file contengono informazioni sul nome e la locazione del file HTML.

I file **.htx** contengono i template per il documento HTML che viene restituito al web browser dopo che i dati spediti hanno interagito con la base di dati attraverso IDC. Per usare IDC lo strumento è ASP, il quale nasconde sia all'utente che al programmatore la struttura e la costruzione dei file .idc e .htx.

B.5.2.1 ASP e ADO

Per utilizzare i driver ODBC, e quindi poter leggere una base di dati, ASP si serve della tecnologia ADO (ActiveX Data Objects) sviluppata da Microsoft per il suo linguaggio ActiveX. Tramite ADO è comunque possibile connettere una applicazione anche a basi di dati non ODBC compatibile quali ad esempio OLE DB.

Per identificare la base di dati su cui lavorare, gli script ADO hanno bisogno che sia specificato un DSN (data source name) che univocamente specifichi il nome e il "luogo fisico" dove si trova la base di dati. A sua volta il DSN contiene le informazioni che riguardano la configurazione della base di dati, le specifiche sulla sicurezza, dove e come sono allocati i dati e può registrare le modifiche del file di log.

ODBC mette a disposizione tre tipi di DSN: *User*, *System o File*. Eccone una breve descrizione:

- *Il System DSN*, permette agli utenti di avere un login;
- *Lo User DSN* controlla l'accesso di determinati utenti;
- *Il File DSN*, che mantiene il form del text file, permette l'accesso a diversi utenti e

facilita il passaggio di dati da un server ad un altro;

Gli oggetti di ADO, sono sette, ma tutti sono in qualche modo collegati con l'oggetto Connection, quello che permette di connettere l'applicazione ASP con una base di dati ODBC compatibile. Quindi una volta dichiarato un oggetto ***Connection***, è possibile gestire gli errori di connessione attraverso l'oggetto ***Error***. Oppure possiamo compiere operazioni sulla base di dati usando l'oggetto ***Command***, grazie al quale è possibile specificare stringhe che possono essere query o comandi per interagire con la base di dati. A sua volta command, permette di definire parametri all'interno delle stringhe di comando utilizzando l'oggetto ***Parameter***. In alternativa all'uso di Command, ADO da la possibilità di accedere ai record di una base di dati utilizzando l'oggetto ***Recordset*** che a sua volta sfrutta i metodi dell'oggetto ***Field*** attraverso il quale è facile spostarsi tra i campi di una tabella di una base di dati.

B.5.2.2 Connessione ad una base di dati

Vediamo in che modo si stabilisce la connessione tra il file ASP e la base di dati. I metodi sono diversi e possono essere utilizzati tutti indifferentemente. Per quel che riguarda la *visualizzazione*, la connessione avviene usando l'oggetto ADODB.RecordSet:

Set rst = Server.CreateObject("ADODB.recordset")
rst.Open strQuery, strProvider

Con il metodo **rst.open** apriamo materialmente la comunicazione con la base di dati che si trova sul server, nel percorso individuato attraverso la stringa **strProvider** e definiamo anche l'interrogazione (in questo caso statica) che si vuole effettuare sulla base di dati e che viene letta attraverso la stringa **strQuery**.

strProvider=*"DRIVER=Microsoft Access Driver (*.mdb); DBQ=" &*
Server.MapPath("/") & " Tesi\Sigmod\Sigmod.mdb;"
strQuery=*"SELECT .. FROM … WHERE ..*

Su strQuery non c'è molto altro da dire, mentre è interessante porre l'attenzione su strProvider. Notiamo infatti che per definire il path dove si trova il file *Access*, utilizziamo il metodo dell'oggetto *Server*, ***MapPath***: in questo modo rendiamo il percorso "relativo" e non assoluto. Ciò però potrebbe comportare problemi di carico sul server. Se allora il programma viene sviluppato solo per una macchina e non si prevede di dover in futuro, esportare le pagine realizzate su altri server, è preferibile usare la seguente modalità per definire il path:

```
strProvider="DRIVER=Microsoft Access Driver (*.mdb); DBQ="&
    "c:\inetpub\wwwroot\Tesi\Sigmod\Sigmod.mdb;"
```

Infatti in questo caso specifichiamo il path assoluto sulla macchina che stiamo utilizzando.

Il secondo metodo, che può essere utilizzato per l'inserimento e la cancellazione di un record, sfrutta la seguente sintassi:

```
Set cn = Server.CreateObject("ADODB.Connection")
 cn.Open strProvider
```

In questo caso ci limitiamo a connettere la base di dati con la nostra pagina. Quindi non abbiamo ancora definito nessuno strumento per lavorare sui record. Per farlo dobbiamo definire il seguente oggetto:

```
Set cm= Server.CreateObject("ADODB.Command")
 Set cm.ActiveConnection = cn
```

Ora cm, ci permetterà di svolgere update, delete e insert, sulla base di dati.

Il terzo metodo per la connessione sfrutta invece le proprietà del driver. Per ora analizziamo il metodo con cui avviene tale connessione. In questo caso occorrerà interagire con il sistema operativo. All'interno del pannello di controllo occorre

selezionare la voce **ORIGINE DATI ODBC** e selezionare poi il foglio DSN System in modo da scegliere il DSN da associare e il percorso dove si trova la nostra base di dati. Ultimate queste operazioni, la nostra base di dati sarà localizzata da ADO, con il nome che gli abbiamo assegnato.

B.5.2.3 L'oggetto RecordSet

Abbiamo visto in precedenza alcune proprietà dell'oggetto RecordSet ma vale la pena approfondirne alcuni aspetti, dato che questo oggetto risulta essere un mezzo versatile e potente, per la gestione dei dati in una base di dati creando un set di record e le operazioni per operare su di esso.

All'interno di pagine ASP è possibile utilizzare l'SQL standard per svolgere operazioni di insert, delete e update.. Un altra via, è utilizzare i metodi che RecordSet mette a disposizione. Supponiamo allora di voler inserire un record in una tabella della nostra base di dati. Dopo essersi connessi utilizzando ***Server.CreateObject("ADODB.recordset")*** e ***rst.Open* strProvider**, per inserire un nuovo record basterà scrivere:

```
rs. AddNew
rs("Nome_campo1")=valore1
rs("Nome_campo2")=valore2
….
Rs.update
```

Analogamente utilizzando il metodo:

rs.delete

elimineremo il record che si sta puntando (anche se è possibile specificare quale record cancellare). Un altro importante metodo è ***rs.*close** i che chiude la connessione stabilita con la base di dati su cui si sta operando.