CURSO PRACTICO DE ASP

Por Víctor Valenzuela Ruz

Las páginas ASP cumplen una importante función en la red de redes, pues nos permiten obtener, de forma simple y variada, información específica a nuestros requerimientos. Ya no es necesario el estar creando nuevas páginas cada vez que deseamos subir nueva información, ni estar remodelando páginas publicadas, con la finalidad de lograr tener al día toda la información. Ahora, con las páginas ASP, podremos crear una plantilla con una diagramación inteligente y versátil, conectar dicha página a una Base de Datos y mostrar así, un contenido distinto para cada requerimiento, todo esto, con sólo 2 páginas, una en HTML, que invoca a la segunda página, la ASP que genera esta una presentación en formato HTML, cuyo código fuente se verá, como código HTML simple, sin la presencia de instrucciones extrañas.

Introducción

¿En que aplicaciones se puede obtener los mejores resultados con las páginas ASP?. Pues básicamente, son 2 los tipos de aplicaciones que se le pueden dar de manera importante. Una, son las llamadas **Listas de Correo** donde los usuarios ingresan sus **Emails** y se les envían información de forma periódica. Estas listas, suelen solicitar información adicional del usuario y todo esto conforma una Base de Datos realmente importante. En este caso, se manejan los conceptos de **Alta** y de **Baja**, es decir, de ingreso y eliminación de un registro. Además de esto, si en el formulario de inscripción, se solicita, por ejemplo, una dirección URL del usuario, la configuración de su PC, sus hobbies, etc., todo esto representa un cumulo de información que puede ser utilizada por todos, previa **Búsqueda** según criterios propios.

La segunda aplicación importante, es a mi modo de ver, la de diarios y revistas **OnLine**. La implicancia de la Base de Datos es por demás evidente. Desarrollar páginas que muestren información que se irá actualizando constantemente, sin tener ya que dirigirnos a un editor y generar nuevo texto para la nueva página HTML, sino que simplemente, los redactores terminan sus artículos y estos son ingresados a los campos respectivos de la Base de Datos e inmediatamente se genera una nueva página, con nuevo contenido, con nuevas direcciones. Todo esto, solamente con 2 páginas y 1base de datos.

Adicionalmente, y tomando como referencia el primer ejemplo, si aunamos a esto el que los usuarios publiquen artículos, podemos ver como estamos incrementando el uso de las páginas ASP.

Un poco de información técnica, no viene mal y también es necesaria..... así que no dejemos esto para el final final sino nos va a cansar más..... veamos.

El término ASP son las siglas de **Active Server Page,** método para crear programas que se ejecutan en un servidor de Web, disponible por primera vez con

Microsoft Internet Information Server 3.0. El IIS (Internet Information Server) Servidor de Internet de Microsoft de alto rendimiento, seguro y extensible basado en Windows NT Server. IIS es compatible con World Wide Web, FTP y Gopher.

En la actualidad, son ya varios los Webs que incluyen páginas ASP como inicio de sus Sites (**index.asp**) lo que demuestra la versatilidad de estas páginas para cumplir un rol por demás protagónico.

En este curso, aprenderemos a crear reportes simples, búsquedas, ingresos y eliminación de registros. Con un poco de imaginación de parte de Uds. podrán unir 2 o 3 páginas en una, con funciones más completas, como por ejemplo, realizar una búsqueda exacta, simple, que muestre coincidencias según el criterio, que permita seleccionar cual registro eliminar, e inclusive, páginas que permitan editar los datos de un determinado registro o previa selección de una lista de registros. También, con imaginación, podremos realizar la **consistencia de datos** mediante VBScript dentro de una página ASP, logrando así, una autentificación de datos.

¿Cuáles son las herramientas recomendadas para un diseño ASP?

En este sentido, solamente puedo opinar a titulo personal, y sin deseo de ser contundente ni engreído. Cada cual, sabe como lidiar con sus pulgas..... bueno, eso decía mi abuelo.

En mi caso, yo utilizo el **FrontPage** como un diagramador que me permite generar la estructura visual de mi pagina HTML (en un formato inicial). Si bien es cierto las prestaciones del **FrontPage 2000** son verdaderamente notables, sigo utilizándolo como un programa de inicio para el diseño propiamente dicho. Utilizo luego, para una programación más **detallista** un programa **editor** como el **Home Site** que me permite trabajar con algunas libertades que el **FP** no me ofrece, ya que, si se han dado cuenta los que usan dicho programa, como que formatea el código a sus requerimientos y a veces desarregla mucho de lo programado. Cuando entro a programar de lleno en ASP no he encontrado mejor herramienta que el **Visual Interdev** que me permite generar códigos limpios en ASP. Es importante disponer de una variedad de **browsers** que me aseguren la correcta visualización del trabajo. Para ello, utilizo el **Internet Explorer - Nestcape - Opera - Mosaic**, que me permite cubrir un gran espectro de los navegadores y sus respectivas bondades y limitaciones de visualización de páginas HTML o ASP.

Entonces, no esperemos más, y demos inicio al presente curso...

Conexión ODBC.

Veamos ahora, los inicios de un trabajo en ASP. Para ello necesitamos, evidentemente, una base de datos. Puede ser una base de datos con registros o sin ellos. Es decir, bien puede ser una base con datos ingresados o solamente su estructura. Para el caso, ambas opciones valen.

En este caso, vamos a considerar la creación de nuestra base de datos con **Microsoft Access**. Se pueden utilizar otro tipo de base de datos, dependiendo lo que soporte nuestro ODBC. Las más usadas, a parte del Access son las de Excel y las de FoxPro.

Empezaremos con la creación de nuestra base de datos. Estas son los pasos a seguir:

Cargar el Access

De la pantalla que nos aparece, seleccionar **Bases de datos en blanco**

Seleccionamos la carpeta para alojar nuestra base de datos, y el nombre de la misma. Para efectos de este curso, llamaremos a la base de datos **Correo**.

Ahora que esta creada, aparecerá una ventana con 6 fichas. Seleccionamos la que dice **Tablas** y que es la que se muestra por defecto. Pulsamos el botón **Nuevo** que es el único que esta activo.

Nos muestra ahora, la ventana de **Nueva Tabla** y de ahí, recomiendo que seleccionemos la opción **Vista Diseño** y luego pulsamos **Aceptar**.

Empezamos a ingresar los nombres de los campos de nuestra tabla. Los campos que a continuación menciono, serán los usados para los ejemplos de este curso. **Nombre**, en tipo, dejamos por defecto **Texto**. Siguiente campo, **Apellido**, también del tipo texto y finalmente **Email** y en este caso, el campo será de tipo **Numérico**.

Cerramos la ventana y le decimos **Si** cuándo nos pregunte si guardamos la tabla. Nos pedirá el nombre para la tabla y le dejaremos por defecto el que Access nos sugiere..... **Tabla1**.

Cuándo nos pregunte sobre un campo clave, le decimos **No**.

Y listo!

Cerramos el Microsoft Access y damos por concluido la creación de nuestra base de datos.

Ahora procederemos a realizar la conexión ODBC correspondiente. Esta conexión **es indispensable** pues con ella, lograremos que las páginas ASP puedan encontrar a la base de datos en cuestión. Una recomendación. Sabemos bien que los servers

tienen una carpeta donde instalan todos los Web Sites que se alojan ahí. Esta carpeta, es la **raíz** del servidor, en lo que respecta a los servicios de alojamiento. Pero bien sabido es que la capacidad de sus HD es mayor. Por eso, es muy recomendable el alojar las bases de datos en carpetas **fuera de la raíz** para así lograr mayor seguridad y privacidad.

Seguimos.....

Estos son los pasos a seguir para una adecuada conexión ODBC.

Copiamos la base de datos **Correo.md**b a la carpeta donde va a estar alojada. Para efectos del curso, y evitar mayores confusiones, asumiremos que dentro de nuestro Site, hay una carpeta llamada **Base de Datos** y ahí alojaremos nuestra base de datos. No quiero con esto, entrar en contradicción con lo dicho en el párrafo anterior, por lo que reitero, es sólo para efectos didácticos.

Vamos a **Panel de Control** y seleccionamos el icono **ODBS 32 bits**.

La ventana que se nos presenta, contiene,normalmente, 6 fichas. La que debemos de seleccionar es **System DSN o DSN del** Sistema. En esta ventana, aparecerán las bases de datos **ya** instaladas.

Pulsamos **Agregar o Add** y se nos presenta una pantalla donde se mostrarán los controladores o drivers de las bases de datos existentes. La ventana se llama **Create New Data Source** y en ella seleccionamos el controlador que necesitamos, en este caso, **Microsoft Access Driver (*.mdb)**.

Le damos **Aceptar**.

Ahora nos muestra una ventana donde nos piden 3 datos datos, pero sólo 2 son indispensables, El nombre de la base de datos **(Data Source Name)** donde ingresamos el nombre de nuestra base de datos **sin** extensión. Luego debemos **Seleccionar** la base de datos y esto no es sino ir a la carpeta donde la copiamos, que hemos denominado **ASP** para efectos de este curso. Una vez ubicada la seleccionamos, y damos **OK** a todas las ventanas y cerramos el **Panel de Control**.

Hemos finalizado la conexión ODBC

Recapitulando:

Hemos creado nuestra base de datos **Correo.mdb** y la hemos alojado en la carpeta **ASP** de nuestro Web Site. Así mismo, hemos establecido la conexión **ODBC** correspondiente con nuestra base de datos.

Estamos listos para continuar con nuestro curso.

Listados.

Ahora vamos de lleno a la programación de páginas ASP. Para ello debemos de tener en cuenta algunas instrucciones que son **indispensables** en todas las páginas que vamos a crear. Estas son las que establecen las conexiones de la página con la base de datos, las que abre tanto la base de datos como la tabla y las que las cierran, evidentemente. Partimos de una página simple, en lenguaje HTML que nos coloca una cabecera que diga **Listado de la Base de Datos**. Este es el código:

```
<%@ Languaje="VBScript" %>
<html>
<head>
<title>Listado de la Base de Datos</title>
</head>
<body>

<p><big><big>Listado de Cuentas de Correo</big></big><br><br></p>
.....Código adicional
</html>
```

Un detalle que me olvidé de mencionar, es la etiqueta **<%@ LANGUAGE="VBSCRIPT" %>** dado que vamos a utilizar este lenguaje como parte de la estructura de las páginas ASP.

Ahora, adicionamos una tabla de una fila y 3 columnas, para colocar los títulos de los campos, que son Nombre, Apellido y Email. El código que genera esta tabla es el siguiente:

```
<div align="center"><center>
<table border="3" cellpadding="0" cellspacing="0" width="100%">
<tr>
<td width="33%"><big>Nombre</big></td>
<td width="33%"><big>Apellido</big></td>
<td width="33%"><big>Email</big></td>
</tr>
```

Como se habrán dado cuenta los que conocen de lenguaje HTML, no he cerrado la tabla ni la división, pues este sólo es un encabezado. Ahora, las filas que se generen a partir de ahora, serán generadas utilizando los datos que se encuentran en nuestra base de datos. **Esta** es la verdadera bondad de las páginas ASP.

La sintaxis de este lenguaje indica que todo aquello que es código VBScript, va entre **<% y %>**, así como las etiquetas en HTML son, por ejemplo y . Lo primero que vamos a hacer, es como dije al inicio, establecer las conexiones y abrir la base de datos y la tabla. No olvidar el cerrarlas, pero eso lo veremos más adelante, para no romper la ilación.

El código que se necesita es el siguiente:

```
<%
Dim Conexion,Tabla
Set Conexion=Server.CreateObject("adodb.connection")
Set Tabla=Server.CreateObject("adodb.recordset")
Conexion.Open "Correo"
Tabla.Open "Tabla1",Conexion
%>
```

Veamos..... la primera instrucción **Dim** nos permite declarar las variables que vamos a utilizar. Luego, definimos dichas variables con las instrucciones mostradas. La primera de ellas,
Set Conexion=Server.CreateObject("adodb.connection") nos permite establecer un tipo de conexión con el server, conexión que luego la usaremos para abrir nuestra tabla. La siguiente instrucción,
Set Tabla=Server.CreateObject("adodb.recordset") ,nos permite definir la variable como la encargada de manipular la base de datos (**objeto Recorset**). No vamos a entrar en muchos detalles, porque esto más bien corresponde al lenguaje VBScript, y este no es un curso de dicho lenguaje, sino de páginas ASP.

Luego, establecemos los vínculos de las variables, indicando con **Conexion** la operación de **abrir** nuestra base de datos, mediante el método **Open**. Luego, definimos que tabla vamos a abrir con la variable **Tabla** y también debemos indicar con que medio (conexión) estamos accediendo a dicha tabla (**Tabla.Open "Tabla1",Conexion**).

Ahora, vamos a construir la tabla en si, con la información que tengamos en ella. Como les dije en el capitulo **Conexión ODBC**, la base de datos puede estar con registros o vacía. Asumamos que hemos ingresado manualmente algunos registros. Entonces, vamos a crear la tabla "prototipo" para cada registro con el siguiente código HTML:

```
<tr>
<td width="33%">Nombre</td>
<td width="33%">Apellido</td>
<td width="33%">Email</td>
</tr>
</table>
</center></div>
```

Como ven, ahora si se cerró la tabla con las instrucciones **</tr>** y **</table>** pero esto sólo nos genera una fila y por consiguiente un registro. Pero ahora, lo que vamos a hacer es sustituir **Nombre, Apellido** y **Email** por el contenido de los registros respectivos. Y eso lo hacemos con la siguiente sentencia:

<%=Tabla.Fields("Nombre del Registro")%>

donde en vez de **Nombre del Registro** efectivamente colocamos el nombre del registro en cuestión que deseemos mostrar. Percatarse de que se inicia la instrucción con = y que usamos la instrucción **Fields** para invocar el contenido de un registro, cuyo nombre siempre ira entre paréntesis y entre comillas. Luego, dicho código, quedaría de la siguiente forma:

```
<tr>
<td width="33%"><%=Tabla.Fields("Nombre")%></td>
<td width="33%"><%=Tabla.Fields("Apellido")%></td>
<td                                                    width="33%"><a
href="mailto:<%=Tabla.Fields("Email")%>"><%=Tabla.Fields("Email")%></a></td>
</tr>
</table>
</center></div>
```

Ahora hemos incrementado nuestro código con lo explicado. Sin embargo, nos falta un detalle. Esto **sólo** nos muestra el contenido de un registro y nosotros deseamos que liste **todo** el contenido de la tabla. Para ello, debemos utilizar la instrucción **While ... Wend** que nos permitirá hacer una serie de operaciones mientras se cumpla un requisito pre-establecido por nosotros. La sintaxis de esta instrucción es:

```
While condición
  Acciones a ejecutar
Wend
```

Evidentemente hay varios tipos de condiciones, aunadas a una gran variedad de funciones, pero la que nos interesa es una que nos permita ir desde el primer registro hasta el final, es decir, hasta que lleguemos al final de la tabla. En otras

palabras, necesitamos que se ejecuten nuestras acciones mientras **no** sea el final de la tabla. Finalmente quedaría así nuestro lazo o bucle:

```
While Not Tabla.EOF
Acciones a ejecutar
Tabla.MoveNext
Wend
```

Con el operador lógico **Not** aunado a **EOF** le estamos diciendo a nuestra página ASP precisamente lo que antes habíamos comentado: realiza las acciones a ejecutar mientras no sea el final de la tabla. Pero a esta instrucción, le falta el elemento que ayude a moverse de registro en registro, pues de por si, la instrucción **While...Wend** no lo hace y esta función esta a cargo de **Tabla.MoveNext**.

Ahora una acotación. Tal vez ya se hayan dado cuenta de que no estoy usando el nombre de la tabla (**Tabla1**), sino de la variable que ha abierto dicha tabla, **Tabla**.

Este es un método aconsejable.

Ahora si tenemos terminada nuestra primera página ASP que realiza un listado completo de la base de datos seleccionada. El código final, con las cláusulas de cierre del HTML, sería el siguiente:

Archivo LISTADO.ASP

```
<%@ Language="VBScript" %>
<html>
<head>
<title>Listado de la Base de Datos</title>
</head>
<body>
<big><big><p>Listado de Cuentas de Correo</big></big> <br><br></p>
<div align="center"><center>
<table border="3" cellpadding="0" cellspacing="0" width="100%">
<tr>
<td width="33%"><big>Nombre</big></td>
<td width="33%"><big>Apellido</big></td>
<td width="33%"><big>Email</big></td>
</tr>
<%
Dim Conexion,Tabla
Set Conexion=Server.CreateObject("adodb.connection")
Set Tabla=Server.CreateObject("adodb.recordset")
Conexion.Open "Correo"
```

```
Tabla.Open "Tabla1",Conexion
While Not Tabla.EOF
%>
<tr>
<td width="33%"><%=Tabla.Fields("Nombre")%></td>
<td width="33%"><%=Tabla.Fields("Apellido")%></td>
<td
width="33%"><ahref="mailto:<%=Tabla.Fields("Email")%>"><%=Tabla.Fields("E
mail")%></a></td>
</tr>
<%
Tabla.MoveNext
Wend
%>
</table>
</center></div>
</body>
</html>
```

Un bono adicional.....jejeje.....bueno, se habrán dado cuenta de que en la línea donde mostramos el **Email** hay dos sentencias VBScript. Esto tiene un sentido. Si recordamos el lenguaje HTML, la línea que establece un link a una dirección de correo electrónico es:

Un texto

Pues bien.....hemos sustituido el contenido de **sucorreo@dominio.com** por la instrucción VBScript correspondiente al campo **Email** y para que coincida con el texto que sirve de link, hemos reemplazado **Un texto** por la misma expresión VBScript. Esto mismo, por supuesto, es aplicable para los links a URL's.

Y esto es todo por ahora amigos..... en nuestra próxima reunión, veremos como ingresar datos mediante las páginas ASP.

¡Hasta la próxima entrega!

Altas.

Esta bien, hemos aprendido como ver el contenido, pero ahora empezamos con lo bueno. El ingreso de datos. Este procedimiento puede ser enriquecido notablemente con una adecuada **consistencia** de datos, es decir, el de evaluar si están bien escritos (por ejemplo los Emails), o si algún campo considerado indispensable ha sido omitido, o tal vez, verificar que el usuario no haya escrito con acentos para evitar

así algún tipo de conflicto al momento de procesar la información..... o lo que su imaginación quiera por último.

Empecemos entonces..... Primero, como lo dije en el **capítulo anterior <3.asp>** lo primero que debemos hacer es declarar nuestras variables y definir las conexiones. Para seguir con el ejemplo anterior, vamos a definir **Conexión** y **Tabla**.

```
<%
Dim Conexion,Tabla
Set Conexion=Server.CreateObject("adodb.connection")
Set Tabla=Server.CreateObject("adodb.recordset")
Conexion.Open "Correo"
Tabla.Open "Tabla1",Conexion
%>
```

Hasta aquí, todo normal. Nada ha variado. Ahora, si bien es cierto que la página de listados sólo es invocada por un simple link desde un menú en una página HTML, para ingresar, debemos de tener un **Formulario** para poder ingresar dichos datos. Para el ejemplo, nos basta uno como el siguiente:

	Nombre del Campo
Nombre	**Dato1**
Apellido	**Dato2**
Email	**Dato3**

En este formulario, debemos destacar un punto muy importante. El método de envió debe ser **Post** y la acción debe ser la página ASP que realiza el ingreso.

Para nuestro ejemplo, la llamaremos **ingreso.asp**. El código HTML final de este formulario es el siguiente:

```
<form method="Post" action="ingreso.asp">
<input type="text" name="Dato1" size="20">
<input type="text" name="Dato2" size="20">
<input type="text" name="Dato3" size="20">
<input type="submit" value="Ingresar" name="B1"><input type="reset" value="Restablecer" name="B2">
</form>
```

Para efectos didácticos, se ha omitido todo aquel código HTML que usamos para una presentación más estética, como son tablas, bordes, tipo de fuente, etc. Lo verdaderamente importante es lo que se esta mostrando arriba. El nombre de los

campos, el método de envió de datos, la acción correspondiente y el tipo de botón que se usa para enviar dicha información. Ya dejo a Uds. el trabajo de crear un formulario presentable y bonito. Pero estas pautas, son **insoslayables**.

Lo que vamos a hacer es ingresar un registro a nuestra Base de Datos. Evidentemente, queremos evitar la duplicidad de registros. Esto es, que no permitiremos que un usuario ingrese 2 veces **la misma** información. Esto lo vamos a conseguir de la siguiente forma:

```
Temp="Select * From Tabla1 Where UCase(Nombre)='" & UCase(Request("Dato1")) & "' And UCase(Apellido)='" & UCase(Request("Dato2")) & "' And UCase(Email)='" & UCase(Request("Dato3")) & "'"
```

¿Y esto?.....bueno, no se preocupen, **parece** complicado, pero no lo es en realidad, simplemente hemos **reorganizado** los datos desde el punto de vista de la sintaxis para que puedan ser correctamente interpretados. Vamos paso a paso.....
En primer lugar, vemos la presencia de una nueva variable, **Temp**. Esta variable, no tiene definición alguna, sino que cumple el rol de almacenar, para hacer más simple la corrección y depuración, una sentencia larga. Podemos declarar la variable en la misma línea que declaramos Conexión y Tabla, o declararla luego, pero siempre antes de la equivalencia que hemos establecido. Veamos ahora, las nuevas instrucciones que hemos agregado. Aquellos que recuerden la programación en Clipper, dBase y FoxPro, van a entenderlo mejor, y los demás.....bueno.....también.

Select Seleccina
***** Todos los registros
From De
Tabla1 Nuestro archivo..... aquí pudimos utilizar la variable Tabla, pero como luego la volveriamos a usar, es mejor evitar algún tipo de conflicto posible
Where Donde
Ucase Convertir a mayúsculas lo que este entre parentesis ()... Ejm. Ucase(Nombre). En caso de ser un dato en particular, este iría entre comillas, así Ucase("Nombre")
Nombre En este ejemplo, es el nombre del campo
& Operador de unión, equivalente al + de otros lenguajes
Request Solicita datos. En este caso, de una variables que han sido enviadas a esta página en particular
Dato1 Nombre de la variable, en el ejemplo, perteneciente al formulario mostrado al inicio de esta página.
And Operador lógico..... que se cumpla esta condición **Y** esta otra condición

Si deseamos entenderlo de forma corrida, como una expresión, entonces, **Temp** contendría la siguiente instrucción:

Suena un poco complicado, pero sería la traducción más cercana a la realidad de la expresión antes mostrada. Ahora bien, pero nosotros estamos viendo una serie de comillas dobles y simples. Eso tiene su explicación. Para entenderlo mejor, mostraremos como se escribe **normalmente** esa sentencia.

```
Temp="Select * From Tabla1 Where
 Ucase(Nombre)=Ucase(Request("Dato1")) And
 Ucase(Apellido))=Ucase(Request("Dato2")) And
 Ucase(Email))=Ucase(Request("Dato3"))"
```

Ahora analicemos esta línea. Vemos varias comillas, ¿no es cierto?. Bueno, cuando declaramos una variable, como **Temp** lo hacemos mediante una igualdad y el contenido va **entre comillas**. Si nos ceñimos a esta definición, entonces, el verdadero valor de **Temp** sería **"Select * From Tabla1 Where Ucase(Nombre)=Ucase("**.¿Se dan cuenta del error, al usar tantas comillas dobles?. Bueno, para eso es que usamos **&** y las comillas simples ('). Y ahora, debemos proceder a separar y volver a unir la expresión ya conocida. La vamos a separar en los siguientes elementos:

- **Select * From Tabla1 Where Ucase(Nombre)=**
- **Ucase(Request("Dato1"))**
- **And Ucase(Apellido)=**
- **Ucase(Request("Dato2"))**
- **And Ucase(Email)=**
- **Ucase(Request("Dato3"))**

Empecemos a unir las partes..... recordemos primero, que la variable **Temp** debemos declararla en función a esta expresión dividida. Luego, empezamos con **Temp="Select * From Tabla1 Where Ucase(Nombre)="** pero debemos reemplazar estas comillas dobles finales por una simple para poder hacer bien la concatenación (unión), con lo que quedaría así **Temp="Select * From Tabla1 Where Ucase(Nombre)='"**. Notese que no hay espacios entre la comilla simple y la doble. Ahora, unimos con el operador **&** la siguiente expresión, tal cual esta, porque nos interesa que las comillas dobles de **Dato1** se queden tal cual están. Y esta la unimos con otro operador **&** con la tercera parte de la expresión, pero esta, **entre comillas simples**, y por lo tanto la expresión, hasta el momento, queda de la siguiente forma: **Temp="Select * From Tabla1 Where Ucase(Nombre)='" & Ucase(Request("Dato1")) & "'And Ucase(Apellido)='"**. Seguimos de igual forma, armando la expresión, pues son idénticos los criterios. El detalle será al final de la misma, pues hemos empezado con una comilla al declarar la variable **Temp="** y debemos de terminar con comilla. Pero como no podemos colocar una comilla doble, la sustituimos por una simple así **""** (se que no se nota, así que la amplio

pero no dejen espacio entre ellas **" ' "**). Entonces, **¡por fin!**, tenemos la expresión terminada y es tal cual la mostramos al inicio.

Ahora un descanso.....¡uf!

Sigamos.....Recapitulando. Hemos declarado las variables de conexión y el algoritmo que nos permitirá verificar que no se esta ingresando el mismo registro dos veces. Así mismo, hemos visto las características del formulario que permite el ingreso de los datos. Ahora, veamos un poco de lógica de programación. La idea es que los datos que nos vienen del formulario, no hayan sido ingresados anteriormente (cosa que hemos solucionado **afortunadamente**). Entonces, estamos ante dos posibilidades. Si se han ingresado, entonces no permitimos que se vuelvan a ingresar y le indicamos al usuario que los datos ya están ingresados.

Si no se han ingresado, entonces, simplemente, los ingresamos y le informamos al usuario que el proceso ha tenido éxito. Manos a la obra.

Hasta el momento, este es el código que hemos generado:

```
<%@ LANGUAGE="VBSCRIPT" %>
<html>
<head>
<title>Ingreso de Datos</title>
</head>
<body>
<%
Dim Conexion,Tabla
Set Conexion = Server.CreateObject("ADODB.Connection")
Set Tabla = Server.CreateObject("ADODB.Recordset")
Dim Temp
Conexion.Open "Correo"
Tabla.Open "Tabla1",Conexion
Temp="Select * From Tabla1 Where UCase(Nombre)='" & UCase(Request("Dato1")) & "' And UCase(Apellido)='" & UCase(Request("Dato2")) & "' And UCase(Email)='" & UCase(Request("Dato3")) & "'"
```

Ahora nos toca entonces, abrir la Base de Datos con el criterio ya explicado. Notése que hemos **ya** abierto Tabla1, pero como Tabla1. Ahora vamos a seleccionar de **Tabla1** aquellos registros que reúnan las condiciones de la expresión almacenada en la variable **Temp**. Si no existe dicho registro, entonces **Tabla** deberá de estar **vacía**. Ese es el criterio que ejecuta las siguientes líneas:

```
Tabla.Open Temp,Conexion,2,3,1

If Tabla.BOF And Tabla.EOF Then
```

```
Tabla.AddNew
Tabla("Nombre") = Request("Dato1")
Tabla("Apellido") = Request("Dato2")
Tabla("Email") = Request("Dato3")
Tabla.Update
%>
<p>Ingreso de datos completado</p>
<%
Else
Response.Write "El registro ya existe"
End If
Tabla.Close
Conexion.Close
%>
```

Veamos las novedades en este código. Primero, tenemos la presencia de **AddNew** que lo que hace es crear un registro en blando al final de la tabla. Las siguientes 3 líneas, simplemente trasladan la información de las variables **Dato1,2 y 3** a sus respectivos campos dentro del registro, mediante la instrucción **Request**. Luego, le indicamos que actualice la información mediante **Update**.
Notese que tanto en AddNew como en Update, se ha usado la variable **Tabla** seguida **de un punto .**

La sentencia **If Tabla.BOF And Tabla.EOF Then** traducido quiere decir ".... si el final y el inicio de la tabla coinciden" (coinciden porque ambas son positivas, no como en el caso del listado cuando colocamos el operador lógico **Not**). Esto se entiende si partimos de la idea de que empezamos desde el inicio de la tabla y recorremos todos los registros y al llegar al final, no hay nada.....entonces al no haber registros, es que esta **vacía**, pero vacía desde el punto de vista de que **no se encontró los datos del registro que se intenta ingresar**. Pero claro, si estuvieran ahí, es decir, si se encuentran, entonces no se cumple esta condición, por lo tanto, ahí interviene **Else** que quiere decir "entonces, si no se cumple lo primero, hacemos lo que sigue". Dentro de la sintaxis de **If** notamos que dicha línea termina con **Then**. La traducción literal sería " si el final y el inicio de la tabla son iguales entonces has". Mediante VBScript podemos escribir código HTML mediante las instrucciones **Response.Write** y el criterio de concatenación es el mismo ya explicado, arriba, por la mitad.....¿recuerdan el laberinto de las comillas?, bueno.....a ese me refiero. Es decir, si quieren, pueden incluir mayor información, pero para el caso, lo dejamos ahí.

No, no me he olvidado, sino que dejé para el final, esos numeritos que están en la línea Tabla.Open Temp,Conexion,2,3,1. Tienen un significado medio obscuro, tipo

Microsoft..... (Uds. saben..... Microsoft y sus exquisiteces). Esos númeritos representan los **punteros** en VBScript. No vamos a entrar en mayores detalles, así que solamente les diré que esa combinación en particular, permite **crear y grabar** datos en un nuevo registro. Si desean mayores datos, pues bueno. a buscar un buen manual de VBScript o consíganse el **Visual Studio** y tendrán tooooodas las respuestas a sus inquietudes.

Finalmente (que alivio, ¿no?), tenemos el siguiente código que nos permite ingresar datos (Ud. lo ponen más bonito, ¿ok?):

Archivo INGRESO.ASP

```
<%@ LANGUAGE="VBSCRIPT" %>
<html>
<head>
<title>Ingreso de Datos</title>
</head>
<body>
<%
Dim Conexion,Tabla
Set Conexion = Server.CreateObject("ADODB.Connection")
Set Tabla = Server.CreateObject("ADODB.Recordset")
Dim Temp
Conexion.Open "Correo"
Tabla.Open "Tabla1",Conexion
Temp="Select * From Tabla1 Where UCase(Nombre)='" &
UCase(Request("Dato1")) & "' And UCase(Apellido)='" &
UCase(Request("Dato2")) & "' And UCase(Email)='" &
UCase(Request("Dato3")) & "'"
Tabla.Open Temp,Conexion,2,3,1

If Tabla.BOF And Tabla.EOF Then
Tabla.AddNew
Tabla("Nombre") = Request("Dato1")
Tabla("Apellido") = Request("Dato2")
Tabla("Email") = Request("Dato3")
Tabla.Update
%>
<p>Ingreso de datos completado</p>
<%
Else
Response.Write "El registro ya existe"
End If
```

```
Tabla.Close
Conexion.Close
%>
</body>
</html>
```

Buscar.

Bueno, hasta ahora, hemos visto como conectar una Base de Datos utilizando el ODBC, como generar un listado y como ingresar datos. Ahora, veremos como **buscar** datos. Y vamos a considerar dos opciones posibles para buscar. Una búsqueda **exacta** y una búsqueda **simple**.

Para ello, retomaremos parte de lo mostrado tanto en **ingresos.asp** como en **listado.asp**. Vamos a tomar del primero, las rutinas correspondientes a la captura de datos, más no así a la consistencia utilizada para ver si los datos existen. Tomaremos del segundo archivo, las rutinas para mostrar los resultados **coincidentes** según el criterio ingresado. Empezaremos, indicando que se creará un formulario simple, que permita el ingreso de un criterio y que el usuario pueda optar por una búsqueda **exacta** o una búsqueda **simple**. Veamos.

Nombre

Búsqueda exacta **Búsqueda simple**

En este caso, el campo que contendrá el dato que será buscado en el campo **Nombre** de la Base de Datos se llama **Criterio**. Los **Botones de Opción** se llaman **Tipo**. Estos son todos los datos que necesitamos. Por supuesto, el archivo ASP se llamará **buscar.asp**. Veamos ahora, el código HTML que gobierna este formulario, sin los arreglos que doy por hecho Uds. van a incluir:

```
<form method="Post" action="buscar.asp">
<input type="text" name="Criterio" size="20">
<input type="radio" value="V1" checked name="Tipo">
<input type="radio" name="Tipo" value="V2">
<input type="submit" value="Buscar" name="boton1">
<input type="reset" value="Restablecer" name="boton">
</form>
```

Vamos a considerar los siguientes puntos. Están en **negrita** aquellos datos que queremos resaltar en esta explicación. Siempre el método **Post** para el envío de

datos del formulario. Nuestra variable principal se llamará **Criterio**; a los **Botones de Opción** les hemos asignado el nombre de **Tipo** como grupo y los valores **V1**para una búsqueda **exacta** y **V2** para una búsqueda **simple**. Recordemos estos nombres, pues los vamos a utilizar en la página ASP que realice la búsqueda.

Veamos un poco de lógica de programación. Primero, vamos a abrir la Base de Datos y establecer la conexión respectiva. Luego, vamos a determinar que tipo de búsqueda hay que realizar. Si es **Tipo = V1** sabremos que es una búsqueda exacta; luego, si no es **V1** tendrá que ser, obviamente, una búsqueda simple. ¿Cómo definiremos el criterio de búsqueda exacta y simple?. Bueno, definimos como **búsqueda exacta** a la coincidencia **total** del criterio ingresado. Podríamos ir más aún, y realizar una búsqueda **precisa y exacta** considerando mayúsculas y minúsculas. Pero eso es simple de inferir como hacerlo. Aquí, usaremos nuestra función **UCase**. Una **búsqueda simple** será cuando basta que el criterio ingresado **este dentro del campo nombre no importanto su ubicación**. Así por ejemplo, si ingresamos un criterio **car** y seleccionamos búsqueda simple, nos mostrará, por nombres, tanto a **Carlos** como a **Ricardo**. Este será nuestro criterio de búsqueda. Entonces, vamos a necesitar la instrucción **If..... Then..... Else..... End If** para lograr esto. Pero empezamos como siempre, con la declaración de variables y las conexiones.

```
<%
Dim Conexion,Tabla,Modo
Set Conexion=Server.CreateObject("adodb.connection")
Set Tabla=Server.CreateObject("adodb.recordset")
Rango=Request("Tipo")
Conexion.Open "Correo"
```

Aquí hemos visto una nueva variable, que de momento no esta participando: **Modo**. Esta variable será la que se encargue de determinar el **Tipo** de búsqueda y la definimos así utilizando la cláusula **Request**. Téngase en cuenta de que aún no hemos abierto **Tabla1**. Ahora que conocemos los "secretos de las comillas", será mucho más sencillo entender las siguientes instrucciones:

```
If Rango="1" Then
Temp="Select * From Tabla1 Where UCase([Nombre])='" & UCase(Request("Criterio")) & "'"
Else
Temp="Select * From Tabla1 Where InStr(UCase([Nombre]),'" & UCase(Request("Criterio")) & "')>0"
End If

Tabla.Open Temp, Conexion
```

```
If Tabla.BOF And Tabla.EOF Then
%>

<p><font face="Arial">No se ha encontrado ningún registro que reúna las condiciones del criterio <strong><%=Request("Criterio")%></strong> en la Base de Datos</font></p>

<%
Else
%>
<div align="center"><center>
<table border="1" cellpadding="0" cellspacing="0" width="100%">
<tr>
<td width="33%"><strong><font face="Arial">Nombres</font></strong></td>
<td width="33%"><strong><font face="Arial">Apellidos</font></strong></td>
<td width="33%"><strong><font face="Arial">Email</font></strong></td>
</tr>
<% While Not Tabla.EOF %>
<tr>
<td width="33%"><%=Tabla.Fields("Nombre")%></td>
<td width="33%"><%=Tabla.Fields("Apellido")%></td>
<td width="33%"><%=Tabla.Fields("Email")%></td>
</tr>
<%
Tabla.MoveNext
Wend
Tabla.Close
Conexion.Close
End If
%>
</table>
</center></div>

</body>
</html>
```

Prácticamente hemos terminado, pues casi todas las rutinas las hemos visto en capítulos anteriores. Sin embargo, debo hacer dos acotaciones. Hubiéramos podido generar el link para el correo, pero no se ha hecho. Esto no quiere decir que no se pueda hacer, sino que mostramos otra forma, la más simple, de mostrar los datos. El otro punto, es la aparición de una nueva función **InStr** que tiene la siguientes sintaxis:

InStr(Cadena donde buscar, Cadena a buscar)

Entonces, nos daremos cuenta de que hemos considerado el contenido de los campos de los registros de la Base de Datos, como las **cadenas donde buscar** y el campo **Criterio** ingresado mediante el formulario, como la **cadena a buscar** dentro de la primera. Esto para el caso de una búsqueda simple. Esta función arroja un número, que si es **cero** indica que no ha habido coincidencia alguna.

Caso contrario, retornará el número de carácter donde **empieza** la coincidencia de la cadena a buscar. Por ejemplo, y sólo como una ampliación de la explicación, pues para nuestro ejemplo, no altera en nada el contenido. Si la **cadena a buscar** fuera **car** y la **cadena donde buscar** fuera **Ricardo**, entonces el valor que arrojaría sería **3**; si fuera **Carlos** la **cadena donde buscar** el número sería **1**. Creo que con estos ejemplos aclaramos el panorama.

Veamos, pues, el código final de nuestra rutina de búsqueda:

Archivo BUSCAR.ASP

```
<%@ Languaje="VBScript" %>
<html>

<head>
<title>Buscar datos</title>
</head>

<body>
<p><font face="Arial">Resultados obtenidos con <strong><%=Request("Criterio")
%></strong></font></p>

<%
Dim Conexion,Tabla

Set Conexion=Server.CreateObject("adodb.connection")
Set Tabla=Server.CreateObject("adodb.recordset")

Dim Temp,Rango
Rango=Request("Tipo")

Conexion.Open "Correo"

If Rango="1" Then
Temp="Select * From Tabla1 Where UCase([Nombre])='" &
UCase(Request("Criterio")) & "'"
Else
Temp="Select * From Tabla1 Where InStr(UCase([Nombre]),'" &
```

```
UCase(Request("Criterio")) & "')>0"
End If

Tabla.Open Temp, Conexion

If Tabla.BOF And Tabla.EOF Then
%>

<p><font face="Arial">No se ha encontrado ningún registro que reúna las condiciones del criterio <strong><%=Request("Criterio")%></strong> en la Base de Datos</font></p>

<%
Else
%>
<div align="center"><center>
<table border="1" cellpadding="0" cellspacing="0" width="100%">
<tr>
<td width="33%"><strong><font face="Arial">Nombres</font></strong></td>
<td width="33%"><strong><font face="Arial">Apellidos</font></strong></td>
<td width="33%"><strong><font face="Arial">Email</font></strong></td>
</tr>
<% While Not Tabla.EOF%>
<tr>
<td width="33%"><%=Tabla.Fields("Nombre")%></td>
<td width="33%"><%=Tabla.Fields("Apellido")%></td>
<td width="33%"><a href="mailto:<%=Tabla.Fields("Email")%>><%=Tabla.Fields("Email")%></a></td>
</tr>
<%
Tabla.MoveNext
Wend
Tabla.Close
Conexion.Close
End If
%>
</table>
</center></div>
</body>
</html>
```

¿Ven?. No es tan complicado como parece. Y las posibilidades se pueden ampliar si consideramos una búsqueda por varios campos. Es más largo el código, pero básicamente son las mismas rutinas.

Bajas.

La eliminación de datos. Si han visto al cargar que esta página demora mucho menos que las anteriores, simplemente es porque hay poco que decir que no se haya dicho antes. Y eso es lo bueno de desarrollar páginas como si fueran funciones, pues podemos tomar una y otra y lograr cosas bien interesantes. Por ejemplo. Para eliminar un registro, primero debemos encontrarlo. Ergo, usamos nuestra página **buscar.asp** para hacer eso. Esto implica que debemos usar un formulario que permita manejar bien esta pagina y será nuestro **buscar.htm**. En fin, el primer paso es usar simplemente, la búsqueda, sea simple o exacta.

Sin embargo..... no todo es gloria, no. Hay que hacer algunas modificaciones, y para corolario, vamos a agregar una bien simpática y vistosa. Queda en Uds. el mejorarla, tanto estéticamente como en programación. Como les dije en la **entrega anterior <5.asp>** aunque usemos una búsqueda exacta, esto implica la posiblidad de varias coincidencia en el **nombre**. Por ello, vamos a agregar a nuestro listado, una columna adicional con un **Botón de Opción** para que el usuario pueda elegir cual registro eliminar. La apariencia será algo así:

Nombre	Apellido	Email
Top of Form 1 Bottom of Form 1	Jorge	Lozano
jorloz@server.com.pe		
Top of Form 2 Bottom of Form 2	Germán	Hidalgo
germ@newserver.net		
Top of Form 3 Bottom of Form 3	Angelica	Aragón
angel@surnet.com.ar		
Top of Form 4 Bottom of Form 4	Jorge Luis	Perez
jlp@server.com.net		
Top of Form 5		
Bottom of Form 5		

Esta bien, se pudo haber mejorado, pero este no es un curso de diagramación ni de diseño sino de programación. En fin.....sigamos. Entonces, marcamos el registro que deseamos eliminar y lo eliminamos. ¿Por que estoy usando un **Botón de Opción**?, pues implemente porque puedo determinar con él si quiero que se marquen varios para eliminar o sólo uno. En este caso, tratamos con la segunda opción, la de eliminar solamente uno.

Sigamos.

Veamos entonces el código del nuevo archivo de búsqueda, **preliminar.asp**. Se darán cuenta de que es exactamente el mismo **buscar.asp** pero con una variación en la rutina de listado. Por supuesto, el formulario HTML sigue siendo el mismo, pero con la siguiente modificación (se que la van notar al acto):

```
<form method="Post" action="preliminar.asp">
<input type="text" name="Criterio" size="20">
<input type="radio" value="V1" checked name="Tipo">
<input type="radio" name="Tipo" value="V2">
<input type="submit" value="Buscar" name="boton1">
<input type="reset" value="Restablecer" name="boton">
</form>
```

y el código fuente de **preliminar.asp** es el siguiente:

```
<%@ Languaje="VBScript" %>
<html>
<head>
<title>Buscar datos</title>
</head>
<body>
<p><font face="Arial">Resultados obtenidos con <strong><%=Request("Criterio")
%></strong></font></p>

<%
Dim Conexion,Tabla

Set Conexion=Server.CreateObject("adodb.connection")
Set Tabla=Server.CreateObject("adodb.recordset")

Dim Temp,Rango
Rango=Request("Tipo")

Conexion.Open "Correo"

If Rango="1" Then
Temp="Select * From Tabla1 Where UCase([Nombre])='" &
UCase(Request("Criterio")) & "'"
Else
Temp="Select * From Tabla1 Where InStr(UCase([Nombre]),'" &
UCase(Request("Criterio")) & "')>0"
End If
```

```
Tabla.Open Temp, Conexion

If Tabla.BOF And Tabla.EOF Then
%>

<p><font face="Arial">No se ha encontrado ningún registro que reúna las condiciones del criterio <strong><%=Request("Criterio")%></strong> en la Base de Datos</font></p>

<%
Else
%>
<div align="center"><center>
<table border="1" cellpadding="0" cellspacing="0" width="100%">
<tr>
<td width="25%"><strong><font face="Arial">Nombres</font></strong></td>
<td width="25%"><strong><font face="Arial">Apellidos</font></strong></td>
<td width="25%"><strong><font face="Arial">Email</font></strong></td>
</tr>
<form method="Post" action="eliminar.asp">
<% While Not Tabla.EOF%>
<tr>
<td width="25%"><input type="radio" name="Registro" value="<%=Tabla.Fields("codigo")%>"></td>
<td width="25%"><%=Tabla.Fields("Nombre")%></td>
<td width="25%"><%=Tabla.Fields("Apellido")%></td>
<td width="25%"><a href="mailto:<%=Tabla.Fields("Email")%>><%=Tabla.Fields("Email")%></a></td>
</tr>
<%
Tabla.MoveNext
Wend
Tabla.Close
Conexion.Close
End If
%>
<p align="center"><input type="submit" value="Proceder" name="B3"></p>
</form>
</table>
</center>
</div>
</body>
</html>
```

Ahora bien. Con este código hemos generado un listado que nos permitirá seleccionar el dato a eliminar, simplemente marcando el **Botón de Opción** incluido en el listado. Este botón corresponde a la variable **Tipo**, y este dato nos permitirá determinar cual registro es el que deseamos eliminar.

Para poder determinar cual es el dato a eliminar, podemos seguir dos procedimientos. El primero, sería el de almacenar en una variable pública, un dato que es único para ese registro dentro de la tabla. Este dato puede ser el código o un identificador determinado. Como pudiera ser que aún con este dato, existiera la posibilidad de alguna duplicidad de registros, entonces, podemos aplicar el segundo procedimiento alternativo. En ambos casos, el criterio es el mismo. Este segundo procedimiento implica el incluir en la estructura de la tabla, un campo que podríamos llamar **Marca** y que solo almacenaría un carácter que diferencie el registro de los demás. Por ejemplo, podemos incluir una **X** para lograr tal diferenciación, dejando los demás en blanco preferentemente.

De esta forma, esta página ASP invocaría a otra que se encargaría de filtrar y preparar una tabla temporal justo para eliminar el registro seleccionado. Para ello, incluimos un botón que se encargue de llamar a dicha página ASP que llamaremos **Eliminar.asp**. El código resultante sería el siguiente:

```
<%@ LANGUAGE="VBSCRIPT" %>
<html>
<head>
<title>Eliminar.asp</title>
</head>
<body>
<%
Dim Conexion,Tabla
Set Conexion=Server.CreateObject("adodb.connection")
Set Tabla=Server.CreateObject("adodb.recordset")
Conexion.Open "CORREO"
Tabla.Open "Select * From Tabla1 Where [Nombre]='" & Request("Registro")
& "'",Conexion,2,3,1
If Tabla.BOF And Tabla.EOF Then
%>
<p><font face="Arial">El registro <strong>NO</strong> ha sido encontrado
<%
Else
Tabla.Delete
%>
</p>
<p><font face="Arial">El registro ha sido <strong>eliminado</strong>
satisfactoriamente
```

```
<%
End If
Tabla.Close
Conexion.Close
%>
</font></p>
<p><a href="../menu.htm"><strong><font face="Arial">Menú</font></strong></a></p>
</body>
</html>
```

Y esto es todo por ahora amigos.....

¡Hasta la próxima entrega!

Víctor A. Valenzuela Ruz