Introducción a *Active Server Pages*

Introducción a la Introducción

Active Server Pages (ASP), es una tecnología propietaria de Microsoft. Se trata básicamente de un lenguaje de tratamiento de textos (scripts), basado en Basic, y que se denomina VBScript (Visual Basic Script). Se utiliza casi exclusivamente en los servidores Web de Microsoft (*Internet Information Server* y *Personal Web Server*). Los *scripts* ASP se ejecutan, por lo tanto, en el servidor y puede utilizarse conjuntamente con HTML y Javascript para realizar tareas interactivas y en tiempo real con el cliente.

Con ASP se pueden realizar fácilmente páginas de consulta de bases de datos, funciones sencillas como obtener la fecha y la hora actual del sistema servidor, cálculos matemáticos simples, etc.

Predisposición

Desde que accedí por la red a una página con un tutorial *on-line*, pensé que afrontar el reto de aprender algo a través de aquel sistema era una tarea de titanes. Estaba en inglés, se le cansa a un@ la vista tanto rato delante de la pantalla, no había un archivo con los fuentes comprimidos que pudiera bajarme (para poder mejorar la dinamicidad y minimizar la cuantía de la factura de la compañía telefónica), tardaba mucho, tenía publicidad por todas partes,... un rollo. Lo que me hizo desistir en pocos minutos de mi intento por culminar aquella gesta.

Ahora que yo sé algo que algunos otros todavía no saben, me llaman personas que quieren saber lo que yo sé, y aunque me esfuerzo en advertirles que es muy poco, todos quieren saber, aunque sea, ese poco. Con ese afán surgió este trabajo.

Si no sabes nada de ASP y te gustaría aprender un poco, mi recomendación es que sigas leyendo estas páginas (para que este trabajo tenga algún sentido), sino, mejor te diriges al cuadrito esa de arriba que pondrá algo así como *dirección*, *address* o *url*, y escribes otro conjunto de símbolos, que a poder ser, sean caracteres ASCII y que especifiquen una localización de recurso válida en Internet.

Si estás leyendo este párrafo es que o bien no haces caso a las advertencias, o sufres un repentino ataque de interés inusitado, en un área cuasi mono-plataforma y mono-sistema, o tienes otras razones personales que no soy quien para juzgar. En cualquier caso estás aquí y ahora, y pretendes pasar algo de tiempo ampliando conocimientos o criticando este trabajo (tan loable la primera como la segunda).

Requisitos Previos

Para no hacer de este tutorial un trabajo largo y pesado, supondremos que el/la hábid@ lector/a tiene conocimientos las áreas más comunes relacionadas con las

tecnologías para la Web: HTML y Javascript; algo de lenguajes de programación, sobre todo Basic (MS Basic), y otro poco de SQL, para las consultas de base de datos.

Además es totalmente necesario tener algo de tiempo para leer el manual y entender los ejemplos. Para llevar a cabo pruebas de programas es necesario tener acceso a un servidor con soporte para ASP, como pueden ser los anteriormente mencionados *Internet Information Server* o *Personal Web Server*.

Para el manejo de bases de datos, dado que estamos en un entorno cuasi totalmente Microsoft, en estas páginas se explicarán las bases de datos Access, aunque la dinámica es muy similar en otras.

Interfaz

Para hacer que todo vaya rápido y fácil, en estas páginas no hay demasiadas imágenes, ni applets, ni videos,... lo que redunda en que si a alguien le interesa imprirlo, sólo tendrá que hacer pequeños retoques aquí y allá.

El código ASP que contienen los archivos que veremos está puesto en **azul**. Los comentarios dentro del código, para que aparezcan más significativos los he puesto en **verde**, y los enlaces en **rojo**. En algunos casos, las etiquetas de HTML no relevantes para el ejemplos se muestran en **gris**.

En todas las páginas hay, bien al principio, bien al final, o puede que en ambos lugares, unas *barras* de color gris con enlaces a las secciones inmediatamente anteriores y posteriores.

Herramientas

ASP, VBScript y Javascript son lenguajes de programación comunes, luego su sintaxis es implementada en código ASCII, por lo que para poder crear, editar y modificar dicho código, sólo es necesario utilizar un simple y común editor de textos, como puede ser el "edit" del *DOS*, el "Notepad" o el "Wordpad" de los entornos *Windows*, o cualquiera de los múltiples editores de texto que existen en los entornos **IX* (Emacs, vi, joe, jed,...), así como en los *Mac*.

Microsoft ha tenido la deferencia de crear, dentro de su *suite* de desarrollo *DevStudio*, una aplicación específica para administración y creación de *proyectos Web*, denominada *Visual InterDev*, que en el momento de realizarse este documento estaba en su versión 6.0.

Sin embargo, en este documento no nos basaremos en dicho software, ya que nos limitaremos a dar unas pinceladas sobre los conceptos básicos del lenguaje, sin afán de introducirnos en áreas más profundas, como la administración de proyectos, y demás.

Introducción a *Active Server Pages*

1. Conceptos iniciales

1.1. Declaración del lenguaje

Como ocurre en otros lenguajes de programación, en ASP existe una sentencia de declaración opcional del lenguaje a utilizar.

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

Esta declaración se pone al principio del archivo, antes de cualquier otra expresión.

1.2. Bloques de código y Comentarios

En páginas ASP, para introducir bloques de sentencias hay que escribir los símbolos reservados:

```
<% {sentencias} %>
```

donde *sentencias* pueden ser una o varias expresiones del lenguaje, como se muestra en el siguiente ejemplo:

```
 ...
 <%
 Request("param")
 Response.Write(Now)
 while not cond do
       rem do nothing
 loop
 %>
...
```

En este punto queremos llamar la atención del lector sobre el hecho de que las sentencias en VBScript no se separan por punto y coma (;).

Los comentarios de código VBScript se especifican mediante la palabra reservada `rem` o con el carácter comilla simple (`'`) y tienen un ámbito de una línea. Por ejemplo:

```
 <%
 rem Esto es un comentario
 '   que ocupa varias
```

```
 rem líneas
%>
```

Y este es un comentario mal construído:

```
 <%
 rem Esto es un comentario
     pero es ya no lo es así que el procesador de ASP
     lo interpretará como código, y dará error
%>
```

1.3. Carácterísticas del lenguaje

Aquí voy a hablar de la declaración de variables, los tipos de las variables, de las sentencias... muy someramente, porque tengo que preparar otra página con información precisa sobre todo esto. Será la última...

Para más información, véase el capítulo **5. Breve referencia del lenguaje**, o la ayuda sobre ASP que ofrece *Microsoft* (*ASP Roadmap*).

1.4. Forma de una página ASP

Para ir abriendo boca, vamos a ver a continuación, que aspecto tiene un archivo de texto que contiene código ASP y que genera como salida, un documento HTML, es decir, una página Web.

```
<%@ LANGUAGE="VBScript" %>
<%
  rem Declaración de variables y funciones a realizar antes de
visualizar el documento
  rem como por ejemplo, inicializar drivers de bases de datos, o
redireccionar a
  rem otros documentos
%>
<HTML>
<HEAD>
<TITLE>T&iacute;tulo...</TITLE>
</HEAD>
<BODY>
<%
  rem Este texto se ve en el documento cuando lo abrimos
  Response.Write("Esto es texto simple<BR>")
  Response.Write("<B>En el que tambi&eacute;n puedo introducir
etiquetas HTML</B><BR>")
%>
<I>Adem&acute;s es posible mezclar bloques ASP con etiquetas de
HTML</I><BR>
<%
  Response.Write("Aunque este es todav&iacute;a un ejemplo muy
sencillo<BR>")
  Response.Write("y con ninguna interactividad...")
%>
</BODY>
```

```
</HTML>
```

Que se vería de la siguiente manera:

Esto es texto simple
En el que también puedo introducir etiquetas HTML
Además es posible mezclar bloques ASP con etiquetas de HTML
Aunque este es todavía un ejemplo muy sencillo
y con ninguna interactividad...

Introducción a *Active Server Pages*

2. Entrada y Salida

2.1. *Response*

Como su nombre indica, la sentencia *Response* sirve para enviar respuestas (información de salida) al documento HTML que se visualizará en el navegador, para redireccionar a otros recursos, etc.

2.1.1. *Response.Write*

Como sabemos para escribir texto en el documento que se pretende visualizar hay que escribir:

```
 <%
   Response.Write({cadena})
%>
```

Una cadena es cualquier combinación de caracteres ASCII, quitando la *comilla doble*. Si queremos que aparezca este símbolo debemos introducirla dos veces (""). Veamos algunos ejemplos:

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

<%

  Response.Write("<HTML>")

  Response.Write("<HEAD>")

  Response.Write("<TITLE></TITLE>")

  Response.Write("</HEAD>")

  Response.Write("<BODY>")

  Response.Write("Esta p&aacute;gina genera todas las etiquetas
de un documento<BR>")

  Response.Write("HTML normal y corriente...")

  Response.Write("</BODY>")
```

```
    Response.Write("<HTML>")
%>
```

```
<%@ LANGUAGE="VBScript" %>
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<%
  Response.Write("Esta p&aacute;gina genera todas las etiquetas
de un documento<BR>")
  Response.Write("HTML normal y corriente...")
%>
</BODY>
<HTML>
```

Los dos ejemplos anteriores son equivalentes. Si además queremos escribir el valor de alguna variable:

```
<%@ LANGUAGE="VBScript" %>
<%
  hoy = Date
%>
```

```
  Response.Write("Hoy es:" & hoy & ")
%>
</BODY>
<HTML>
```

Aunque como es de suponer existe una manera algo más ágil y abreviada para indicar lo anterior, con menos caracteres:

```
<%@ LANGUAGE="VBScript" %>
<HTML>
```

```
  </BODY>

 <HTML>
```

2.1.2. *Response.Redirect*

En ocasiones puede ser útil tener una página que tras un determinado tratamiento de algún dato obtenido del cliente, *llame* a otra página, o simplemente como método de actualización de enlaces antiguos. En cualquiera de estos casos se utiliza la sentencia ***`Response.Redirect`***:

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

  <%

    rem Este enlace ha quedado obsoleto, redireccionar a...

    Response.Redirect("http://www.w3.org/Style/")

    rem Todo lo que hay por debajo de este punto: etiquetas HTML,
  código ASP

    rem no llega a ser interpretado por el procesador de ASP jamás

 %>
```

La utilidad del código queda patente si tenemos en cuenta que con la dinamicidad de la red, frecuentemente se dan modificaciones en las localizaciones de los recursos. Veamos ahora ejemplo de redireccionamiento para tratamiento de datos, y de paso anticipamos algo de lo que veremos en el siguiente punto:

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

  <%

    opcion = Request("param_opcion")

    Select Case opcion

      Case 1: Response.Redirect("pag1.html")

      Case 2: Response.Redirect("pag2.html")

      Case 3: Response.Redirect("pag3.html")
```

```
<%
  Select Case Request("param_opcion")
    Case 1: Response.Redirect("pag1.html")
    Case 2: Response.Redirect("pag2.html")
    Case 3: Response.Redirect("pag3.html")
  End Select
%>
```

2.2. *Request*

La sentencia Request tiene como misión obtener los valores de los parámetros que pueden pasárseles a las páginas ASP. La forma de pasar pares atributo-valor (parámetro-valor) es la siguiente:

Donde podemos apreciar que con los datos introducidos por el cliente (en un formulario, o por otros medios), llamamos a una página de tratamiento de esos datos. Los parámetros que se pasan son:

dni	12345678

Los carácteres `%20` que muestra el ejemplo entre los apellidos hacen referencia al caracter espacio en codificación UTP.

También es posible pasar parámetros a otra página a partir de campos de un formulario. En este caso, los nombres de los parámetros vienen dados por los nombres que hay que asignar a dichos campos, y la página que los trata o recoge se especifica con el atributo `ACTION` de la etiqueta `FORM`.

```
<%@ LANGUAGE="VBScript" %>
<HTML>
<HEAD>
<TITLE><TITLE>
</HEAD>
<BODY>

<FORM METHOD="POST" ACTION="guardar.asp">
  <INPUT TYPE="TEXT" NAME="nombre">
  <INPUT TYPE="TEXT" NAME="apellidos">
  <INPUT TYPE="TEXT" NAME="dni">
  <INPUT TYPE="SUBMIT">
</FORM>

</BODY>
</HTML>
```

Con este código el procesador de ASP se encarga de construir internamente una llamada igual a la vista en el ejemplo anterior. La manera general de especificar la llamada a una página con parámetros, por lo tanto, es:

{nombre_pagina}**.asp?**{nom_var1}**=**{valor1}[**&**{nom_var2}={valor2}...]

Lo que está encerrado entre llaves puede ser prácticamente, cualquier cadena *válida* de caracteres y lo que está escrito en negrita debe aparecer tal como está.

Introducción a *Active Server Pages*

3. Algunas funciones básicas

3.1. Fecha y hora

3.1.1. Fecha (Date, Day, WeekDay, WeekDayName, Month, MonthName, Year)

Función	Descripción
Date	Devuelve la fecha del sistema, generalmente en formato anglosajón.
Day(fecha)	Devuelve el día del mes.
Weekday(date, [firstdayofweek])	
WeekDayName(weekday, abbreviate, firstdayofweek)	
Month	
MonthName	
Year	

3.1.2. Hora (Now, Time, Hour, Minute, Second)

Los datos devueltos por estas funciones se extraen de la fecha y hora del sistema (servidor) donde se ejecutan las mismas.

Función	Descripción
Now	
Time	
Hour	
Minute	
Second	

3.2. Tratamiento de cadenas

InStr([start,]string1, string2[, compare])

InstrRev(string1, string2[, start[, compare]])

Mid(string, start[, length])

Trim, LTrim, RTrim

Replace(expression, find, replacewith[, start[, count[, compare]]]) - Returns a string in which a specified substring has been replaced with another substring a specified number of times.

Right(string, length) - Returns a specified number of characters from the right side of a string.

Space(number) - Returns a string consisting of the specified number of spaces.

StrComp(string1, string2[, compare]) - Returns a value indicating the result of a string comparison.

StrReverse(string1) - Returns a string in which the character order of a specified string is reversed.

UCase(string)

LCase(string)

3.3. Numéricas

Abs - Valor absoluto - Abs(numero)

Atn - Arcotangente - Atn(numero)->radianes

Cos(numero)->radianes - coseno

Exp(numero) - e elevado a numero

Fix(numero)->numero - devuelve la parte entera de un real

Hex(numero)->string - Valor hexadecimal del numero

Log(numero)->numero - logaritmo

result=number1 **Mod** number2 - Used to divide two numbers and return only the remainder.

Oct(numero)->string - Valor octal del numero

Randomize

Rnd[(number)] - Returns a random number.

Round(expression[, numdecimalplaces]) - Returns a number rounded to a specified

number of decimal places.

Sgn(number) - Returns an integer indicating the sign of a number.

Sin(number) - Returns the sine of an angle.

Sqr(number) - Returns the square root of a number.

Tan(number) - Returns the tangent of an angle.

Introducción a *Active Server Pages*

4. Bases de datos

4.1. Declarar el *driver* de base de datos

Sin duda alguna, lo más importante que hay que saber respecto al manejo de bases de datos en ASP (presupuestos unos ligeros conocimientos en SQL y Access), es la inicialización del *driver* de base de datos. Existen dos maneras de hacerlo, a saber:

La primera consiste en declarar un **DSN de Sistema** a través de ODBC. Para ello iremos al botón Inicio de nuestro sistema Windows, y desde allí a la opción Configuración y después a Panel de Control. En la ventana que aparece debemos dirigirnos a ODBC (o a ODBC de 32 bits, según el sistema), y se nos abrirá una nueva ventana con una serie de solapas. Nos vamos a DSN de sistema.

En este punto vamos a añadir nuestro nuevo DSN para la base de datos que queremos tratar. Para ello seleccionamos en botón Agregar. Se abre una ventana que lleva por título Crear un nuevo origen de datos en el que se nos muestran los *drivers* de base de datos disponibles en nuestro sistema. Seleccionamos el deseado, en nuestro caso Microsoft Access Driver y pulsamos Finalizar.

Hecho esto se abre una nueva ventana, de nombre ODBC Microsoft Access 97 Setup. En el campo Data Source Name debemos escribir el identificador que emplearemos para la base de datos (si por ejemplo se trata de una base de datos de libros cuyo archivo se llama *biblio.mdb*, podríamos llamarla *libros*). Luego presionamos el botón Select para seleccionar el archivo de base de datos dentro de la jerarquía de directorios del sistema, y tras esto pulsamos Ok. Y ya podremos hacer referencia a ese origen de datos desde nuestras páginas.

Esta primera opción es muy rápida de configurara, sin embargo, es muy frecuente desarrollar las páginas en una máquina y ponerlas en otra (un servidor propiamente dicho), por lo que resulta lioso tener un DSN para cada base de datos del sistema.

La segunda es un poco más pesada, por el hecho de que hay que incluir una serie de **líneas de código** en cada página que haga uso de la base de datos, pero es mucho más flexible, puesto que si cambiamos de sistema, no debemos crear un nuevo DSN.

La declaración del driver debe hacerse antes de que se escriba algo en el documento

HTML de salida, y es tan simple como esto:

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

<%

  ' Declaramos el objeto de conexión a la base de datos

  Set ConexionBD =
Server.CreateObject("ADOdb.Connection")

  ' Abrimos el objeto con el driver específico

  ConexionBD.Open "DRIVER={Microsoft Access Driver
(*.mdb)}; " & "DBQ=" & Server.Mappath("/ruta/bd.mdb")

%>

<HTML>

...
```

Vemos que en la sentencia `ConexionBD.Open`, en la parte `DBQ="` tenemos lo siguiente `Server.Mappath()`, que es una variable que devuelve la ruta local del directorio raíz del servidor Web, y el parámetro que le pasamos hace referencia a la situación de la base de datos dentro de la jerarquía del servidor. Veamos el siguiente ejemplo.

Si tenemos nuestro servidor Web (*http://127.0.0.1* ó *localhost*) en un directorio del sistema denominado *C:\Inetpub*, y nuestra base de datos estará en *C:\Inetpub\biblioteca\libros.mdb*, en *Server.Mappath* deberemos indicar lo siguiente:

```
...

  ConexionBD.Open "DRIVER={Microsoft Access Driver
(*.mdb)}; " & "DBQ=" &
Server.Mappath("/biblioteca/libros.mdb")

...
```

Atención, cuando nos referimos al sistema de directorios local utilizamos la barra (\) para separar los directorios, pero cuando hacemos referencia al servidor, se separan con el otro tipo de barra (/).

Con esto hemos cumplido con el primer paso, definir el driver para utilizar la base de datos, pero todavía no podemos realizar ninguna consulta ni modificación. Para ello tenemos que definir lo que en Basic se conoce como *RecordSet*, que no es más que una agrupación lógica de registros (algo parecido a una variable de tabla lógica).

4.2. Realizar operaciones con la base de datos

Para ver qué es un *RecordSet* y para que sirve Volvamos otra vez al ejemplo:

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

<%

  ' Declaramos el objeto de conexión a la base de datos

  Set ConexionBD =
Server.CreateObject("ADOdb.Connection")

  ' Abrimos el objeto con el driver específico

  ConexionBD.Open "DRIVER={Microsoft Access Driver
(*.mdb)}; " & "DBQ=" &
Server.Mappath("/biblioteca/libros.mdb")

  Set RS = ConexionBD.Execute("select * from libros")

%>

<HTML>

...
```

Con esto conseguimos que se el objeto RS (RecordSet) esté *enlazado* con el resultado de una consulta de la tabla libros de la base de datos *libros.mdb*, es decir, que *contenga* los valores, de alguna manera.

Pero todavía no tenemos resultados visibles de que la operación se haya completado con éxito. Si hemos fallado en algo, en cambio, es muy probable que tengamos una poco prometedora página en nuestro visualizador con algún tipo de error. Mas seamos optimistas, si ponemos el código de arriba con algunos aditamentos podremos observar los datos de esa base de datos, en el navegador y en tiempo real.

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

<%

  ' Declaramos el objeto de conexión a la base de datos

  Set ConexionBD =
Server.CreateObject("ADOdb.Connection")

  ' Abrimos el objeto con el driver específico

  ConexionBD.Open "DRIVER={Microsoft Access Driver
(*.mdb)}; " & "DBQ=" &
Server.Mappath("/biblioteca/libros.mdb")

  Set RS = ConexionBD.Execute("select * from libros")

%>

<HTML>
```

```
<HEAD>

<TITLE>Consulta de Libros</TITLE>

</HEAD>

<BODY>

<%

  ' Como la bd no está vacía hacemos un tratamiento
hasta que no queden registros...

  Do while not RS.EOF

    ' Escribimos en la salida los datos que nos interesa

    Response.Write("<P>T&iacute;tulo: " & RS("titulo") &
"<BR>")

    Response.Write("Autor: " & RS("autor") & "</P>")

    ' nos movemos al siguiente registro

    RS.MoveNext

  Loop

%>

</BODY>

</HTML>
```

Quien sepa algo de HTML, SQL y Basic sabrá interpretar este código sin verlo ejecutado por el procesador ASP, y verá que