Programación en ASP

Manual por: DesarrolloWeb.com [http://www.desarrolloweb.com/]
"Tu mejor ayuda para aprender a hacer webs"

Versión on-line
http://www.desarrolloweb.com/manuales/8
[http://www.desarrolloweb.com/manuales/8

Introducción a la programación en ASP

Tal como hemos explicado, **ASP (Active Server Pages) es la tecnología para la creación de páginas dinámicas del lado del servidor desarrollada por Microsoft**.

El tipo de servidores que emplean este lenguaje son aquellos que funcionan con sistema operativo de la familia de Windows NT. Afortunadamente, también podemos visualizar páginas ASP sobre Windows 95/98, pero esto lo veremos más adelante.

Para escribir páginas ASP utilizamos un lenguaje de scripts, que se colocan en la misma página web junto con el código HTML. Comúnmente este lenguaje de scripts es **Visual Basic Script**, que deriva del conocido Visual Basic, aunque también se pueden escribir los scripts ASP en otro lenguaje: JScript, que deriva a su vez del conocido Javascript.

Existe una versión de Visual Basic Script en el lado cliente y otra en el lado del servidor. En los dos casos, como su nombre indica, el lenguaje de base es Visual Basic por lo que su aprendizaje puede ser perfectamente coordinado, ya que las sentencias y las sintaxis son prácticamente las mismas. En ASP, al estar programando páginas del lado del servidor, utlizaremos Visual Basic Script del lado del servidor y en este manual nos centraremos en este punto. El lector interesado por la sintaxis de Visual Basic Script y su programación del lado del cliente puede encontrar en este mismo sitio otro manual para tratar exclusivamente Visual Basic Script [http://www.desarrolloweb.com/manuales/1].

Este manual va destinado a aquellos que quieren comenzar de cero el aprendizaje de este lenguaje y que buscan en él la aplicación directa a su proyecto de sitio o a la mejora de su sitio HTML. Los capítulos son extremadamente simples, sino simplistas, buscando ser accesibles a la mayoría. Ellos serán complementados posteriormente con otros artículos de mayor nivel destinados a gente más experimentada.

Antes de comenzar a leer este manual es altamente aconsejable, sino imprescindible, haber leído previamente el manual sobre páginas dinámicas [http://www.desarrolloweb.com/manuales/7/] en el cual se explica a grandes rasgos qué es el ASP, algunos conceptos útiles sobre el modo de trabajar con páginas dinámicas al mismo tiempo que nos introduce algunos elementos básicos de la programación como pueden ser las variables y las funciones.

Del mismo modo, puede resultar extremadamente útil el haber leído o leer inmediatamente después el manual de Visual Basic Script [http://www.desarrolloweb.com/manuales/1/] en el cual se explica más en profundidad el lenguaje Visual Basic que resulta ser utilizado en la mayoría de scripts ASP.

Otra referencia a la cual haremos alusión es el tutorial de SQL [http://www.desarrolloweb.com/manuales/9/] que nos será de gran ayuda para el tratamiento de bases de datos.

Nuestra intención es la de ir publicando paulatinamente diferentes capítulos de modo que rogamos un poco de paciencia a aquellos que estén esperando la continuación. Todo irá llegando.

Esperamos que este manual resulte de vuestro agrado y que corresponda a nuestras expectativas: El poder acercar este lenguaje a todos aquellos amantes del desarrollo de webs que quieren dar el paso hacia las webs "profesionales".

Los scripts que usamos en estos primeros ejemplos pueden ser descargados aquí [http://www.desarrolloweb.com/articulos/reportajes/capitulos/asp/pack1.zip].

Si te interesa trabajar con un editor específico de ASP te recomendamos el MS Visual Interdev [http://msdn.microsoft.com/vinterdev]. Otra posibilidad es el Drumbeat [http://www.drumbeat.com/] de Macromedia aunque para empezar ninguno de los dos resulta absolutamente indispensable. También podemos elegir Homesite [http://www.desarrolloweb.com/articulos/331.php], un editor que no es específico para las ASP, pero que se comporta bastante bien y ofrece ayudas interesantes.

Pasos previos I: Instalación del PWS

En capítulos anteriores hemos explicado que, dada la naturaleza de los lenguajes de lado servidor, nos es imposible trabajar offline como hacíamos para el caso de las páginas HTML que almacenábamos en nuestro disco duro. También dijimos que esto no era completamente cierto ya que podíamos resolver este eventual problema instalándonos en nuestro PC un servidor propio. Este servidor distribuido por Microsoft tiene dos versiones diferentes que son utilizadas dependiendo del equipo que estemos utilizando. Para los usuarios de W95 o W98, la versión disponible se llama Personal Web Server (PWS).

Si trabajamos bajo sistema Windows NT, o las versiones Profesional y Server de Windows 2000 y XP, el servidor a instalar es el Internet Information Server (IIS). En este caso os referimos a nuestro manual sobre la Instalación de IIS en Windows XP profesional [http://www.desarrolloweb.com/manuales/36/].

Existe también la posibilidad de trabajar en plataformas UNIX empleando en este caso el ChilisoftASP [http://www.sun.com/software/chilisoft/].

Los usuarios de W95 tienen varias posibilidades para hacerse con el PWS: Descargarlo del sitio Microsoft [http://www.microsoft.com/ntserver/nts/downloads/recommended/NT4OptPk/default.asp] , a partir de una antigua versión de Frontpage 98, instalándolo desde la opción pack de W-NT 4.0 o desde el CD de instalación de W98 (directorio add-ons/pws).

Los usuarios de Windows ME o XP Home edition, no tienen soporte para PWS, aunque se pueden probar una serie de pasos para lograr utilizarlo en el sistema. Este documento de Microsoft explica mejor este asunto [http://support.microsoft.com/default.aspx?scid=kb;EN-US;q266456].

> **Nota:** También existe otra posibilidad para ejecutar páginas ASP, aparte de los mencionados PWS e IIS, que funciona muy bien con versiones de Windows como ME o XP Home edition. Se trata de un servidor gratuito y muy simple, llamado Baby ASP Web Server [http://www.desarrolloweb.com/articulos/2229.php] y que hemos comentado en un artículo de DesarrolloWeb.com.

Por otro lado, los usuarios de Windows 2000 y XP Profesional deben utilizar IIS 5.0, que se encuentra en la instalación. Es recomendable que leáis también las notas de los visitantes al pie de página, porque encontraréis muchos más datos sobre problemas en distintas versiones y compatibilidades con otros sistemas que van apareciendo.

Algunas versiones del PWS anteriores a la 4.0 requieren un archivo adicional asp.exe [http://download.microsoft.com/download/fp98/Utility/1.00/WIN98/EN-US/Asp.exe] para poder reconocer páginas ASP.

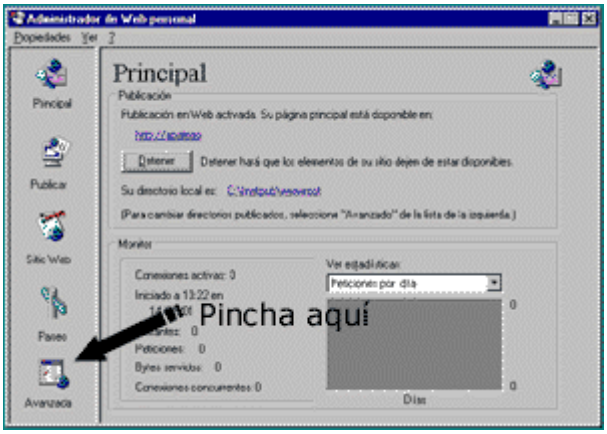
PWS podría ser considerado como una versión "light" del IIS4. En realidad en PWS no es suficientemente versátil para ejercer de servidor de un sitio de un tamaño mediano aunque si que podría en un momento dado hacerse cargo de un sitio de tamaño reducido y no muy concurrido. De todas formas, la utilidad del PWS radica sobre todo en que nos permite realizar las pruebas del sitio que vayamos a desarrollar en "local" sin necesidad de colgar nuestros archivos en el servidor que alberga nuestro sitio cada vez que queramos hacer una prueba sobre una pequeña modificación introducida. Esto resulta a todas luces práctico sobre todo para

principiantes que necesitan hacer pruebas con una relativa frecuencia permitiendo el ahorro de mucho tiempo.

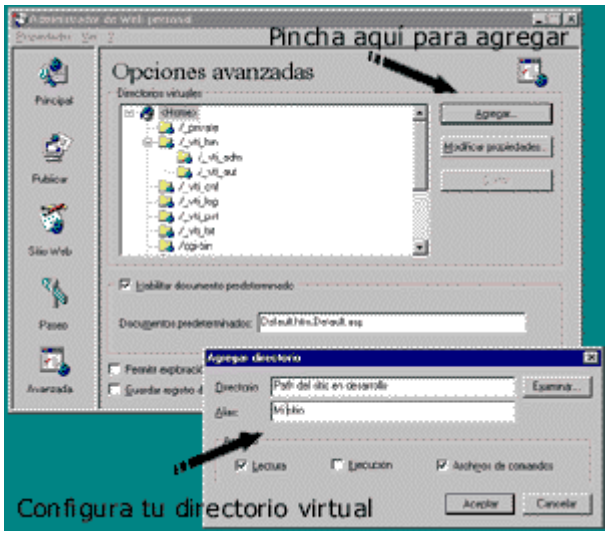
Dado que la mayoría de los posibles lectores de este manual trabajan en entorno W95 y 98, en este capítulo nos limitaremos a la descripción del PWS dejando el IIS4 para futuros capítulos. Sin embargo, las explicaciones que damos pueden ser igualmente útiles para quienes tengan que utilizar este último el cual presenta un funcionamiento básico análogo. El uso del PWS es extremadamente fácil. Una vez instalado, podemos observar la introducción de un nuevo icono en la barra de tareas así que en el menú de inicio correspondientes a la aplicación. A partir de cualquiera de ellos podemos tener acceso a la página principal o gestionario.

El siguiente paso es crear un directorio virtual dentro del cual alojaremos nuestra página. Hablamos de directorio virtual debido a que nuestra página puede estar alojada en cualquier parte de nuestro disco duro, donde a nosotros nos plazca y con un nombre de directorio que tampoco tiene por qué parecerse al que incluiremos en la URL cuando queramos ejecutar la página. De hecho, la URL que debemos introducir en el navegador para visualizar nuestra página ASP es del tipo: http://localhost/nombre_del_directorio virtual/archivo.asp

Como se puede observar, en este tipo de dirección no se especifica el camino en el disco duro donde se encuentran nuestros archivos.



Volviendo a la creación de nuestro directorio virtual, para hacerlo debemos pinchar sobre el icono "Avanzado" el cual nos da acceso a las opciones avanzadas del PWS. Una vez ahí, el siguiente paso es "Agregar" un directorio virtual. Una ventana en la que tendremos que introducir el nombre de dicho directorio virtual y especificar en qué carpeta del disco duro tenemos guardados los archivos y carpetas de la página aparecerá.



Como puede verse, la cosa es fácil. Ahora no queda más que introducir en el navegador el tipo de URL mencionada anteriormente para ejecutar los scripts creados.

Una opción interesante en el menú avanzado es la selección del tipo de archivo que será ejecutado por defecto. Aquí podríamos poner archivos con nombre index.html o index.asp o bien con el nombre default o home...

Pasos previos II: Conexión a BD

El siguiente paso, una vez instalado el servidor que nos permite trabajar en local, es crear los vínculos con las bases de datos que explotaremos en nuestros scripts. En efecto, la utilización de páginas dinámicas está muy frecuentemente asociada con el empleo de bases de datos.

Una base de datos es sencillamente un conjunto de tablas en las que almacenamos distintos registros (artículos de una tienda virtual, proveedores o clientes de una empresa, películas en cartelera en el cine...). Estos registros son catalogados en función de distintos parámetros que los caracterizan y que presentan una utilidad a la hora de clasificarlos. Así, por ejemplo, los artículos de una tienda virtual podrían catalogarse a partir de distintos campos como puede ser un número de referencia, nombre del artículo, descripción, precio, proveedor...

Las bases de datos son construidas sirviéndose de aplicaciones tales como el Microsoft Access o el MySQL las cuales resultan bastante sencillas de utilizar con unos conceptos mínimos.

Nuestro objeto aquí no es explicar la forma de explotarlas sino cómo establecer una conexión entre la base de datos, almacenada en cualquier lugar del disco duro y nuestra página web alojada también en cualquier parte y reconocida por nuestro servidor personal a partir del directorio virtual.

Para crear este vínculo, nos servimos de los conectores ODBC (Open DataBase Connectivity) los cuales establecen el enlace con la base de datos.

El primer paso para crear esta conexión es ir al panel de configuración y abrir el icono ODBC 32bits. Dentro de él, deberemos crear un DSN (Data Source Name) de tipo sistema o usuario. Para ello nos colocamos en la solapa correspondiente (DSN sistema o DSN usuario) y seleccionamos "Añadir". A continuación se nos pedirá seleccionar los controladores de la aplicación que hemos utilizado para crear la base de datos, el nombre que le queremos asignar (aquel que empleemos en nuestros scripts) y el camino para encontrarla en el disco duro.

Esta DSN permite en realidad definir la base de datos que será interrogada sin necesidad de pasar por la aplicación que hayamos utilizado para construirla, es decir, con simples llamadas y órdenes desde nuestros archivos ASP podremos obtener los datos que buscamos sin necesidad de ejecutar el Access o el MySQL los cuales, evidentemente, no tendrán por qué encontrarse en el servidor donde trabajemos.

Inicio a la programación en ASP

A lo largo de los capítulos precedentes nos ha quedado claro que el ASP es un lenguaje orientado a las aplicaciones en red creado por Microsoft que funciona del lado servidor. Es en

efecto el servidor quien se ocupa de ejecutarlo, interpretarlo y enviarlo al cliente (navegador) en forma de código HTML.

ASP es principalmente utilizado sirviéndose del lenguaje Visual Basic Script que no es más que una versión light del Visual Basic. Sin embargo, es posible programar páginas ASP en Java Script. Lo único que hay que hacer es especificar en la propia página qué tipo de lenguaje estamos utilizando.

Dado que el lenguaje ASP está muy frecuentemente embebido dentro del código HTML, es importante poder marcar al servidor qué partes están escritas en un lenguaje y cuáles en otro. Es por ello que todas las partes del archivo que están escritas en ASP estarán siempre delimitadas por los símbolos: <% y %>.

De este modo, cuando realicemos nuestros scripts, lo primero que debemos definir es el tipo de lenguaje utilizado, lo cual se hace del siguiente modo:

<% @ LANGUAGE="VBSCRIPT" %> Para el caso en el que programemos en Visual Basic Script

<% @ LANGUAGE="JSCRIPT" %> Si nos servimos del Java Script en servidor para programar en ASP

Los scripts que serán presentados en este manual estarán basados en el VBS, el cual presenta toda una serie de prestaciones que lo hacen sin duda más accesible y apto para ASP. No es por nada que es el propio Microsoft quien ha creado ambos.

Con los elementos que hemos presentado hasta ahora, ya estamos en situación de poder escribir nuestro primer programa en ASP. Vamos a crear un programa que calcule el 20% de impuestos que habría que añadir a una serie de artículos. Para plasmar el concepto de función, explicado en el manual de páginas dinámicas, vamos a definir una función "impuesto" que emplearemos sucesivas veces. El programa podría resultar algo así:

```
<% @ LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD>
<TITLE>Funcion impuesto</TITLE>
</HEAD>
<BODY>
<%Function impuesto(precio_articulo)
precio_final=precio_articulo+precio_articulo*20/100
Response.Write precio_final
End Function%>
Un libro de 3500 ptas. se quedará en un precio de <% impuesto(3500) %>
<br>
Una camisa de 6000 ptas. tendrá un precio final de <% impuesto(6000) %>
<br>
Un CD de música de 2000 ptas. costaría <% impuesto(2000) %> ptas.
</BODY>
</HTML>
```

Si quieres ver el efecto que produce pincha aquí [http://www.guiartemultimedia.com/desarrolloweb/cursoasp/funcion.asp]

Como puede verse, el script contiene dos partes fundamentales: Una primera en la que definimos la función que llamamos impuesto que depende únicamente de una variable (precio_articulo). Impuesto permite añadir un 20% al precio del artículo e imprimir el resultado en pantalla (Response.Write). En la segunda parte nos servimos de la función para realizar los cálculos necesarios y mostrarlos en pantalla acompañados de texto.

Resulta muy interesante, una vez ejecutado el script, ver el código fuente. Como puede verse, el código HTML que muestra el browser no coincide con el que nosotros hemos escrito. Algo que no debe sorprendernos ya que, como ya hemos explicado, el servidor se encarga de procesarlo y hacerlo comprensible al navegador.

Bucles y condiciones I

La programación exige en muchas ocasiones la repetición de acciones sucesivas o la elección de una determinada secuencia y no de otra dependiendo de las condiciones específicas de la ejecución.

Como ejemplo, podríamos hacer alusión a un script que ejecute una secuencia diferente en función del día de la semana en el que nos encontramos.

Este tipo de acciones pueden ser llevadas a cabo gracias a una paleta de instrucciones presentes en la mayoría de los lenguajes. En este capítulo describiremos someramente algunas de ellas propuestas por el VBS y que resultan de evidente utilidad para el desarrollo de páginas ASP.

Para evitar el complicar el texto, nos limitaremos a introducir las más importantes dejando de lado otras cuantas que podrán ser fácilmente asimilables a partir de ejemplos prácticos. Para una mayor información sobre este tipo de elementos así como sobre los tipos de variables, referirse al manual de VBScript [http://www.desarrolloweb.com/manuales/1/] que trata más detenidamente estos aspectos.

Las condiciones: IF

Cuando queremos que el programa, llegado a un cierto punto, tome un camino determinado en determinados casos y otro diferente si las condiciones de ejecución difieren, nos servimos del conjunto de instrucciones If, Then y Else. La estructura de base de este tipo de instrucciones es la siguiente:

```
IF condición THEN
  Instrucción 1
  Instrucción 2
  ...
ELSE
  Instrucción A
  Instrucción B
  ...
END IF
```

Llegados a este punto, el programa verificará el cumplimiento o no de la condición. Si la condición es cierta las instrucciones 1 y 2 serán ejecutadas. De lo contrario (Else), las instrucciones A y B serán llevadas a cabo.

Una vez finalizada la estructura, deberemos cerrar con un End If.

Esta estructura de base puede complicarse un poco más si tenemos cuenta que no necesariamente todo es blanco o negro y que muchas posibilidades pueden darse. Es por ello que otras condiciones pueden plantearse dentro de la condición principal. Hablamos por lo tanto de condiciones anidadas que tendrían una estructura del siguiente tipo:

```
IF condición THEN
  Instrucción 1
  Instrucción 2
  ...
ELSE
  IF condición2 THEN
    Instrucción A
    Instrucción B
    ...
```

```
  ELSE
    Instrucción X
    ...
  END IF
END IF
```

De este modo podríamos introducir tantas condiciones como queramos dentro de una condición principal. En este tipo de estructuras es importante cerrar correctamente cada uno de los IF con sus END IF correspondientes. De gran ayuda es la instrucción ELSE IF que permite en una sola línea y sin necesidad de añadir un END IF introducir una condición anidada.

El uso de esta herramienta resultará claro con un poco de práctica. Pongamos un ejemplo sencillo de utilización de condiciones. El siguiente programa permitiría detectar la lengua empleada por el navegador y visualizar un mensaje en dicha lengua.

```
<% @ LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD>
<TITLE>Detector de Lengua</TITLE>
</HEAD>
<BODY>
<%
'Antes de nada introducimos mensajes en forma de variables
  espanol="Hola"
  ingles="Hello"
  frances="Bonjour"

'Ahora leemos del navegador cuál es su lengua oficial
  idioma=Left(Request.ServerVariables("HTTP_ACCEPT_LANGUAGE"),2)

'Formulamos las posibilidades que se pueden dar
If idioma="es" Then
  Response.Write espanol
ElseIf idioma="fr" Then
  Response.Write frances
Else
  Response.Write ingles
End If %>
</BODY>
</HTML>
```

Para poder ver el funcionamiento de este script es necesario cambiar el idioma preferido lo cual puede ser realizado a partir del menú de opciones del navegador.

Si quieres ver el efecto que produce pincha aquí [http://www.guiartemultimedia.com/desarrolloweb/cursoasp/detectorlengua.asp]

Como puede verse, las variables que contienen texto son almacenadas entre comillas.

Para leer la lengua aceptada por el navegador lo que hacemos es definir una variable (idioma) que recoge las dos primeras letras empezando por la izquierda del idioma aceptado por el navegador ("HTTP_ACCEPT_LANGUAGE"). Este idioma aceptado puede ser requerido como una variable del objeto ServerVariables. Por el momento dejaremos esto tal cual, ya nos encargaremos de verlo más detenidamente en otros capítulos.

La tercera parte de script se encarga de ver si el navegador está en español (es), francés (fr) o en cualquier otro idioma que no sea ninguno de estos dos y de imprimir cada uno de los mensajes que proceda en cada caso.

Otro punto a comentar es el hecho de poder comentar los programas. Como puede observarse, dentro del script hemos introducido unos mensajes que nos sirven para leerlo más fácilmente. Estos mensajes no ejercen ninguna influencia en el desarrollo del mismo. Para introducirlos es necesario escribirlos detrás de un apóstrofe: '

Los comentarios son de gran utilidad cuando tratamos con programas muy extensos y complicados los cuales. En estos casos, resultan de gran ayuda a la hora de depurar fallos o introducir modificaciones. Es altamente aconsejable el acostumbrarse a utilizarlos.

Bucles y condiciones II

Los bucles FOR

En muchas ocasiones resulta necesario el ejecutar un conjunto de instrucciones un número definido de veces. Esto puede ser llevado a cabo a partir de la instrucción FOR/NEXT.

La estructura clásica:

```
FOR contador=número inicial to número final STEP incremento
  Instrucción 1
  Instrucción 2
  ...
NEXT
```

A partir de este tipo de estructuras ejecutamos las instrucciones contenidas entre el FOR y el NEXT un cierto número de veces definido por el número inicial, final y el incremento. El incremento resulta de 1 por defecto.

Pongamos un ejemplo:

```
<% @ LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD>
<TITLE>Bucle for/next</TITLE>
</HEAD>
<BODY>

<%For i=1 to 5%>
<font size=<%Response.Write i%>>Vuelta número <%Response.Write i%></font><br>
<%Next

For i=5 to 1 Step -1%>
<font size=<%Response.Write i%>>Contamos atrás: <%Response.Write i%></font><br>
<%Next%>

</BODY>
</HTML>
```

Este script compuesto de dos bucles cuenta primero de 1 a 5. La variable i toma por lo tanto todos los valores enteros comprendidos entre estos dos números y puede ser utilizada dentro del bucle como lo hacemos en este caso para aumentar el tamaño de la letra. El segundo bucle realiza el proceso inverso (el incremento es negativo) produciendo una disminución del tamaño de la letra.

Si quieres ver el efecto que produce pincha aquí [http://www.guiartemultimedia.com/desarrolloweb/cursoasp/buclefornext.asp]

Lo que puede resultar interesante para ver hasta qué punto el programar páginas dinámicas puede ahorrarnos texto con respecto a la misma página programada en código HTML, es mirar el código fuente de la página a partir del navegador.

Bucles y condiciones III

Los bucles DO WHILE/LOOP

Otra forma de realizar este tipo de secuencias bucle es a partir de la instrucción DO WHILE. En este caso lo que especificamos para fijar la extensión del bucle no es el número de vueltas sino el que se cumpla o no una condición. La estructura de este tipo de bucles es análoga a la de los bucles FOR/NEXT:

```
DO WHILE condición
  Instrucción 1
   Instrucción 2
   ...
LOOP
```

El bucle se dará mientras la condición propuesta siga siendo válida.

Como se verá en ejemplos posteriores, este tipo de bucles resulta muy práctico para la lectura de bases de datos.

Todo este tipo de controladores de flujo (condiciones y bucles) pueden ser matizados y optimizados a partir del uso de operadores lógicos. Así, podemos exigir que sean dos las condiciones que se den para llevar a cabo un conjunto de instrucciones:

IF condición 1 AND condición 2 THEN ...

También podemos requerir que sea una de las dos:

IF condición 1 OR condición 2 THEN...

Del mismo modo, es posible exigir que la condición de un bucle DO sea la inversa a la enunciada:

DO WHILE NOT condición

He aquí, en conclusión, un conjunto de recursos básicos para controlar el desarrollo de programas. Su utilidad resultará más que patente y su uso irá haciéndose intuitivo a medida que nos familiaricemos con el lenguaje.

Los objetos ASP

El ASP es un lenguaje diseñado para la creación de aplicaciones en internet. Esto quiere decir que existen toda una serie de tareas bastante corrientes a las cuales debe dar un tratamiento fácil y eficaz. Nos referimos por ejemplo al envío de e-mails, acceso a archivos, gestión de variables del cliente o servidor como pueden ser su IP o la lengua aceptada...

El lenguaje VB propiamente dicho no da una solución fácil y directa a estas tareas sino que invoca a los denominados objetos que no son más que unos módulos incorporados al lenguaje que permiten el desarrollo de tareas específicas. Estos objetos realizan de una manera sencilla toda una serie de acciones de una complejidad relevante. A partir de una llamada al objeto este realizará la tarea requerida. En cierta forma, estos objetos nos ahorran el tener que hacer largos programas para operaciones sencillas y habituales.

Algunos de estos objetos están incorporados en el propio ASP, otros deben de ser incorporados como si se tratase de componentes accesorios. Por supuesto, no podríamos ejecutar correctamente un script en el cual tuviésemos que llamar a un objeto que no estuviese integrado en el servidor. Este tipo de "plug-in" son generalmente comprados por el servidor a empresas que los desarrollan.

Como todo objeto del mundo real, los objetos del mundo informático tienen sus propiedades que los definen, realizan un cierto numero de funciones o métodos y son capaces de responder de una forma definible ante ciertos eventos.

Dado el nivel de esta obra, la descripción de la totalidad de objetos con sus métodos y propiedades resulta ciertamente fuera de lugar. Nos contentaremos pues con ir describiendo los más frecuentemente utilizados y ejemplificarlos de la manera más práctica dejando la enumeración exhaustiva en forma de apéndice.

Objeto Request I

Bucles y condiciones son muy útiles para procesar los datos dentro de un mismo script. Sin embargo, en un sitio internet, las páginas vistas y los scripts utilizados son numerosos. Muy a menudo necesitamos que nuestros distintos scripts estén conectados unos con otros y que se sirvan de variables comunes. Por otro lado, el usuario interacciona por medio de formularios cuyos campos han de ser procesados para poder dar una respuesta. Todo este tipo de factores dinámicos han de ser eficazmente regulados por un lenguaje como el ASP.

Como veremos, todo este tipo de aspectos interactivos pueden ser gestionados a partir del objeto Request.

El objeto Request nos devuelve informaciones del usuario que han sido enviadas por medio de formularios, por URL o a partir de cookies (veremos de qué se tratan seguidamente). También nos informa sobre el estado de ciertas variables del sistema como pueden ser la lengua utilizada por el navegador, el número IP del cliente...

Transferir variables por URL

Para pasar las variables de una página a otra lo podemos hacer introduciendo dicha variable en la dirección URL de la página destino dentro del enlace hipertexto. La sintaxis sería la siguiente:

Para recoger la variable en la página destino lo hacemos por medio del objeto Request con el método Querystring:

Request.querystring("variable1")
Request.querystring("variable2")

Las dos páginas serían así:

```
<HTML>
<HEAD>
<TITLE>Página origen.asp</TITLE>
</HEAD>
<BODY>
<a href="destino.asp?saludo=hola&texto=Esto es una variable texto">Paso variables saludo y texto a la página
destino.asp</a>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>destino.asp</TITLE>
</HEAD>
<BODY>
Variable saludo: <%Response.Write Request.Querystring("saludo")%><br>
Variable texto: <%Response.Write Request.Querystring("texto")%><br>
</BODY>
```

```
</HTML>
```

Comentario: El archivo origen podría ser con extensión htm, ya que solo tiene código html.
Por otro lado, al principio del archivo destino tendriamos que poner: <% @ LANGUAGE="VBScript" %>
para que funcionara.

Si quieres ver el efecto que produce pincha aquí
[http://www.guiartemultimedia.com/desarrolloweb/cursoasp/origen.asp]

Objeto Request II

Transferir variables por formulario

El proceso es similar al explicado para las URLs. Primeramente, presentamos una primera página con el formulario a rellenar y las variables son recogidas en una segunda página que las procesa:

```
<HTML>
<HEAD>
<TITLE>formulario.asp</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="destino2.asp">
Nombre<br>
<INPUT TYPE="TEXT" NAME="nombre"><br>
Apellidos<br>
<INPUT TYPE="TEXT" NAME="apellidos"><br>
<INPUT TYPE="SUBMIT">
</FORM>
</BODY>
</HTML>


<HTML>
<HEAD>
<TITLE>destino2.asp</TITLE>
</HEAD>
<BODY>
Variable nombre: <%=Request.Form("nombre")%><br>
Variable apellidos: <%=Request.Form("apellidos")%>
</BODY>
</HTML>
```

Si quieres ver el efecto que produce pincha aquí
[http://www.guiartemultimedia.com/desarrolloweb/cursoasp/formulario.asp]

Otras utilidades de Request: las ServerVariables

El objeto Request nos da acceso a otras informaciones relativas al cliente y el servidor las cuales pueden resultar de una gran utilidad. Estas informaciones son almacenadas como variables las cuales son agrupadas en una colección llamada ServerVariables.

Dentro de esta colección tenemos variables tan interesantes como:

HTTP_ACCEPT_LANGUAGE	Nos informa de la lengua preferida por el navegador
HTTP_USER_AGENT	Indica cuál es el navegador utilizado.
PATH_TRANSLATED	Nos devuelve el path físico del disco duro del servidor en el que se encuentra nuestro script
SERVER_SOFTWARE	Nos dice qué tipo de software utiliza el servidor

Para visualizar en pantalla alguna de estas variables, debemos escribir algo como:

Response.write request.servervariables("nombre de la variable")

Una forma rápida de visualizar todas estas variables es a partir de un script con esta secuencia:

```
<%
For Each elemento in Request.ServerVariables
   Response.Write elemento&" : "&Request.ServerVariables(elemento)& "<br>"
Next
%>
```

Esto nos daría por un lado el nombre de la variable y del otro su valor. Este tipo de bucle For Each/Next se parece a otros ya vistos. En este caso, el bucle se realiza tantas veces como elementos tiene la colección (ServerVariables) que no es más que el conjunto de elementos comprendidos en la extensión del objeto (Request). Este tipo de bucle es aplicable a otras colecciones de éste y otros objetos como por ejemplo los Request.Form o Request.Querystring o las cookies. De esta forma seríamos capaces de visualizar el nombre y contenido de tales colecciones sin necesidad de enunciarlas una por una.

Objeto Response

Tal y como se ha visto, el objeto Request gestiona todo lo relativo a entrada de datos al script por parte del usuario (formularios), provenientes de otra URL, del propio servidor o del browser.

Nos queda pues por explicar cómo el script puede "responder" a estos estímulos, es decir, cómo, después de procesar los debidamente los datos, podemos imprimir estos en pantalla, inscribirlos en las cookies o enviar al internauta a una u otra pagina. En definitiva, queda por definir la forma en la que ASP regula el contenido que es enviado al navegador.

Esta función es tomada en cargo por el objeto Response el cual ha sido ligeramente mencionado en capítulos precedentes concretamente en asociación al método Write.

En efecto, sentencias del tipo:

<%Response.Write "Una cadena de texto"%>

o bien,

<%Response.Write variable%>

Tienen como cometido imprimir en el documento HTML generado un mensaje o valor de variable. Este método es tan comúnmente utilizado que existe una abreviación del mismo de manera a facilitar su escritura:

<% = variable %> es análogo a <%response.write variable%>

Es importante precisar el hecho que imprimir en el documento HTML no significa necesariamente visualizar en pantalla ya que podríamos servirnos de estas etiquetas para crear determinadas etiquetas HTML. He aquí un ejemplo de lo que se pretende decir:

<% path="http://www.misitio.com/graficos/imagen.gif" %>

<img src="<% =path %>">

Este fragmento de script nos generaría un código HTML que sería recibido en el navegador del siguiente modo:

```
<img src="http://www.misitio.com/graficos/imagen.gif">
```

Otro elemento interesante de este objeto Response es el método Redirect. Este método nos permite el enviar automáticamente al internauta a una página que nosotros decidamos.

Esta función, a todas luces práctica, puede ser empleada en scripts en los que enviamos al visitante de nuestro sitio a una u otra página en función del navegador que utiliza o la lengua que prefiere. También es muy útil para realizar páginas "escondidas" que realizan una determinada función sin mostrar ni texto ni imágenes y nos envían a continuación a otra página que es en realidad la que nosotros recibimos en el navegador.

> **Referencia:** La función redirect se puede estudiar en un artículo más extensamente: Redirigir al navegador a una URL con ASP al detalle [http://www.desarrolloweb.com/articulos/1208.php].

Aquí se puede ver una secuencia de script que nos permitiría usar este método para el caso en el que tengamos páginas diferentes para distintas versiones de navegadores. En ella emplearemos un objeto que nos debe parecer familiar ya que lo acabamos de ver (Request.ServerVariables):

```
<%
tipo_navegador = Request.ServerVariables("HTTP_USER_AGENT")
If Instr(1, tipo_navegador, "MSIE 3", 1) <> 0 then
Response.Redirect "MSIE3.asp"
ElseIf Instr(1, tipo_navegador, "MSIE 4", 1) <> 0 then
Response.Redirect "MSIE4.asp"
ElseIf Instr(1, tipo_navegador, "MSIE 5", 1) <> 0 then
Response.Redirect "MSIE5.asp"
ElseIf Instr(1, tipo_navegador, "Mozilla/4", 1) <> 0 then
Response.Redirect "Netscape4.asp"
Else
Response.Redirect "cualquiera.asp"
%>
```

Por supuesto, acompañando este script, debemos tener los correspondientes archivos MSIE3.asp, MSIE4.asp, MSIE5.asp... Cada uno de ellos con las particularidades necesarias para la buena visualización de nuestra página en tales navegadores.

Ejemplo sencillo de uso de ASP

Vamos a ver un sencillo ejemplo realizado en ASP que sirva para ilustrar el trabajo desarrollado hasta el momento en el manual de la tecnología [http://www.desarrolloweb.com/manuales/8]. Este ejemplo es muy básico, aunque experimenta con varias de las utilidades vistas hasta ahora, como el trabajo con bucles y los objetos request y response, que sirven para recibir datos e imprimirlos en la página.

El ejemplo en concreto se trata de un generador de tablas de multiplicar. En principio, cuando se accede al archivo, se muestra la tabla del cero y un formulario donde podemos seleccionar otro número y ver su tabla de multiplicar. Al enviar el formulario se accede a la misma página, aunque ahora aparecería la tabla de multiplicar del número seleccionado en el formulario.

Formulario para seleccionar un número

Veamos ahora el formulario que muestra un campo de selección con los números del 1 al 10. Este formulario servirá para que el visitante pueda seleccionar la tabla que desea ver.

```
<form name=tb action=tb.asp method=post>
<P align=center>Seleccione una opción
<SELECT align=center name=tab style="WIDTH: 40px">
   <OPTION selected>1</OPTION>
   <OPTION >2</OPTION>
   <OPTION >3</OPTION>
   <OPTION >4</OPTION>
   <OPTION >5</OPTION>
   <OPTION >6</OPTION>
   <OPTION >7</OPTION>
   <OPTION >8</OPTION>
   <OPTION >9</OPTION>
   <OPTION >10</OPTION>
</SELECT>
<br>
<INPUT type=submit value="Ver tabla" >
</P>
</form>
```

Hay que fijarse que la página que va a recibir el formulario se llama tb.asp, según se indica en el atributo action. El único campo del formulario que se envía es llamado "tab", y guarda el número que se haya seleccionado.

Código para mostrar la tabla de multiplicar correspondiente

Empezamos recibiendo el dato del formulario que nos indica la tabla que el usuario quiere visualizar. En un principio no se recibe ningún dato del formulario (hasta que no se envíe el formulario no se sabe que tabla se desea ver y por tanto, habíamos dicho que se iba a mostrar la tabla del cero). Así pues, si no recibo nada, inicializo a cero la variable i, que guarda el número de la tabla de multiplicar a mostrar. En caso de que sí se reciba algo del formulario, se inicializa la variable i al valor recibido en el campo "tab".

```
'si no se está recibiendo datos del formulario
if request.form("tab")="" then
   'inicializo la tabla a mostrar a cero
   i=0
else
   'inicializo la tabla a mostrar al dato recibido en el formulario
   i=Request.Form ("tab")
end if
```

Ahora veremos un bucle que muestra la tabla de multiplicar del valor recibido por formulario. Este bucle hace una repetición desde 1 al 10 y se van realizando las multiplicaciones y mostrando los resultados.

```
'muestro la tabla del número que recibo del formulario
Response.Write "Tabla del " & i%><br><br><%
'realizo un bucle del 1 al 10 para mostrar la tabla correspondiente
for a=1 to 10
   Response.Write i &" x " & a & " = " & i*a%>
   <br>
   <%
next
%>
```

Código completo

El código completo del ejemplo se puede ver a continuación. Espero que sirva de ayuda para las personas que empiezan a dar sus primeros pasos con ASP.

```
<%@ Language=VBScript %>
<HTML>
<HEAD><title>Tablas de Multiplicar....</title>
</HEAD>
<BODY bgColor=skyblue>

<div align="center">
```

```
<form name=tb action=tb.asp method=post>
<P align=center>Seleccione una opción
<SELECT align=center name=tab style="WIDTH: 40px">
   <OPTION selected>1</OPTION>
   <OPTION >2</OPTION>
   <OPTION >3</OPTION>
   <OPTION >4</OPTION>
   <OPTION >5</OPTION>
   <OPTION >6</OPTION>
   <OPTION >7</OPTION>
   <OPTION >8</OPTION>
   <OPTION >9</OPTION>
   <OPTION >10</OPTION>
</SELECT>
<br>
<INPUT type=submit value="Ver tabla" name=submit1 >
</P>
</form>
<%

'si no se está recibiendo datos del formulario
if request.form("tab")="" then
   'inicializo la tabla a mostrar a cero
   i=0
else
   'inicializo la tabla a mostrar al dato recibido en el formulario
   i=Request.Form ("tab")
end if

'muestro la tabla del número que recibo del formulario
Response.Write "Tabla del " & i%><br><br><%
'realizo un bucle del 1 al 10 para mostrar la tabla correspondiente
for a=1 to 10
   Response.Write i &" x " & a & " = " & i*a%>
   <br>
   <%
next
%>
</div>

</BODY>
</HTML>
```

Las famosas cookies

Sin duda este término resultara familiar para muchos. Algunos lo habrán leído u oído pero no saben de qué se trata. Otros sin embargo sabrán que las cookies son unas informaciones almacenadas por un sitio web en el disco duro del usuario. Esta información es almacenada en un archivo tipo texto que se guarda cuando el navegador accede al sitio web.

Es posible, por supuesto, ver estos archivos. Para abrirlos hay que ir al directorio C:\Windows\Cookies para los usuarios de IE 4+ o a C:\...\Netscape\Users\defaultuser para usuarios de Netscape. Como podréis comprobar, en la mayoría de los casos la información que se puede obtener es indescifrable.

La utilidad principal de las cookies es la de poder identificar al navegador una vez éste visita el sitio por segunda vez y así, en función del perfil del cliente dado en su primera visita, el sitio puede adaptarse dinámicamente a sus preferencias (lengua utilizada, colores de pantalla, formularios rellenados total o parcialmente, redirección a determinadas páginas...).

Para crear un archivo cookies, modificar o generar una nueva cookie lo podemos hacer a partir del objeto Response con una sintaxis como la siguiente:

Response.Cookies("nombre de la cookie") = valor de la cookie

El valor de la cookie puede ser una variable, un numero o una cadena delimitada por comillas.

Es importante saber que las cookies tienen una duración igual a la sesión, es decir, a menos que lo especifiquemos, el archivo texto generado se borrará desde el momento en que el usuario haya abandonado el sitio por un tiempo prolongado (generalmente unos 20 minutos) o que el navegador haya sido cerrado. Podemos arreglar este supuesto inconveniente mediante la propiedad Expires:

```
Response.Cookies("nombre de la cookie").expires = #01/01/2002#
```

Esto nos permite decidir cual es la fecha de caducidad de la cookie. Hay que tener en cuenta que esto no es más que hipotético ya que el usuario es libre de borrar el archivo texto cuando le plazca.

Por otra parte, es interesante señalar que el hecho de que definir una cookie ya existente implica el borrado de la antigua. Del mismo modo, el crear una primera cookie conlleva la generación automática del archivo texto.

Para leer las cookies, nada más fácil que usando el objeto Request de la siguiente forma:

```
variable = Request.Cookies("nombre de la cookie")
```

Si por ejemplo quisiéramos recuperar del archivo txt la cookie correspondiente a la lengua del cliente y almacenarlo en nuestro script para futuros usos, podríamos escribir:

```
lengua=Request.Cookies("lengua")
```

Las cookies son una herramienta fantástica para personalizar nuestra página pero hay que ser cautos ya que, por una parte, no todos los navegadores las aceptan y por otra, se puede deliberadamente impedir al navegador la creación de cookies. Es por ello que resultan un complemento y no una fuente de variables infalible para nuestro sitio.

Objeto Session

En los programas que hemos visto hasta ahora, hemos utilizado variables que solo existían en el archivo que era ejecutado. Cuando cargábamos otra página distinta, los valores de estas variables se perdían a menos que nos tomásemos la molestia de pasarlos por la URL o inscribirlos en las cookies o en un formulario para su posterior explotación. Estos métodos, aunque útiles, no son todo lo prácticos que podrían en determinados casos en los que la variable que queremos conservar ha de ser utilizada en varios scripts diferentes y distantes los unos de los otros.

Podríamos pensar que ese problema puede quedar resuelto con las cookies ya que se trata de variables que pueden ser invocadas en cualquier momento. El problema, ya lo hemos dicho, es que las cookies no son aceptadas ni por la totalidad de los usuarios ni por la totalidad de los navegadores lo cual implica que una aplicación que se sirviera de las cookies para pasar variables de un archivo a otro no sería 100% infalible.

Nos resulta pues necesario el poder declarar ciertas variables que puedan ser reutilizadas tantas veces como queramos dentro de una misma sesión. Imaginemos un sitio multilingüe en el que cada vez que queremos imprimir un mensaje en cualquier página necesitamos saber en qué idioma debe hacerse. Podríamos introducir un script identificador de la lengua del navegador en cada uno de los archivos o bien declarar una variable que fuese valida para toda la sesión y que tuviese como valor el idioma reconocido en un primer momento.

Estas variables que son válidas durante una sesión y que luego son "olvidadas" son definidas con el objeto Session de la siguiente forma:

Session("nombre de la variable") = valor de la variable

Una vez definida, la variable Session, será almacenada en memoria y podrá ser empleada en cualquier script del sitio web.

La duración de una sesión viene definida por defecto en 20 minutos. Esto quiere decir que si en 20 minutos no realizamos ninguna acción, el servidor dará por finalizada la sesión y todas las variables Session serán abandonadas. Esta duración puede ser modificada con la propiedad Timeout:

Session.Timeout = n° de minutos que queramos que dure

Una forma de borrar las variables Session sin necesidad de esperara que pase este plazo es a partir del método Abandon:

Session.Abandon

De este modo todas las variables Session serán borradas y la sesión se dará por finalizada. Este método puede resultar practico cuando estemos haciendo pruebas con el script y necesitemos reinicializar las variables.

Lo que se suele hacer es crear un archivo en el que se borran las cookies y se abandona la sesión. Este archivo será ejecutado cuando queramos hacer borrón y cuenta nueva:

```
<% @ LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD>
<TITLE>Puesta a cero</TITLE>
</HEAD>
<BODY>
<%
For Each galleta in Response.Cookies
  Galleta=""
Next
Session.Abandon
%>
Borrón y cuenta nueva!!<br>
<a href="url de la página de inicio">Volver al principio</a>
</BODY>
</HTML>
```

Trabajar con bases de datos en ASP

Una de las principales ventajas que presenta el trabajar con páginas dinámicas es el poder almacenar los contenidos en bases de datos. De esta forma, podemos organizarlos, actualizarlos y buscarlos de una manera mucho más simple.

ASP nos ofrece una forma muy eficaz de interaccionar con estas bases de datos gracias al uso del componente ADO (ActiveX Data Objects) el cual permite acceder a dichas bases de una forma sencilla.

Este ADO no es más que un conjunto de objetos que, utilizados en conjunto, nos permiten explotar de una forma muy versátil las bases de datos de nuestra aplicación. No entraremos por el momento en consideraciones teóricas al respecto.

Por otra parte, lo scripts ASP deben establecer un dialogo con la base de datos. Este dialogo se lleva a cabo a partir de un idioma universal: el SQL (Structured Query Language) el cual es común a todas las bases de datos. Este lenguaje resulta, como veremos en el manual de SQL [http://www.desarrolloweb.com/manuales/9/], muy potente y fácil de aprender.

En este manual de ASP nos limitaremos a utilizar las instrucciones básicas que serán aprendidas a medida que explicamos las diferentes formas de actuar sobre una base de datos a partir de páginas ASP.

La base de datos que ha sido utilizada en estos ejemplos es MS Access. No es por supuesto la única si bien es la más corriente en pequeños PCs y resulta absolutamente operativa siempre que las tablas no sean astronómicamente grandes. Esperamos poder ofreceros próximamente también un pequeño curso de Access en el que explicar los principios rudimentarios necesarios para poder servirnos de él. No obstante, esta aplicación resulta suficientemente fácil e intuitiva como para poder prescindir de dicho curso por el momento.

Así pues, para sumergirnos en estos capítulos siguientes, hemos de cumplir los siguientes requisitos técnicos:

-Instalar el PWS [http://www.desarrolloweb.com/articulos/245.php?manual=8]
-Enlazar la base de datos con ODBC [http://www.desarrolloweb.com/articulos/246.php?manual=8]
-Instalar el MS Access (viene dentro del pack Office)
-Disponer de un browser y un editor (MS IE y Home Site son nuestras humildes recomendaciones)

La base de datos y los scripts que usamos en los ejemplos pueden ser descargados aquí [http://www.desarrolloweb.com/articulos/reportajes/capitulos/asp/bddpack.zip].

Selecciones en una tabla de base de datos con ASP

Dentro de una base de datos, organizada por tablas, la selección de una tabla entera o de un cierto numero de registros resulta una operación rutinaria.

A partir de esta selección se puede posteriormente efectuar toda una serie de cambios o bien realizar una simple lectura.

El siguiente script nos permite realizar la lectura de la tabla clientes contenida en nuestra base de datos. A primera vista todo nos puede parecer un poco complejo, pero nada más lejos de la realidad.

```
<HTML>
<HEAD>
<TITLE>Lectura de registros de una tabla</TITLE>
</HEAD>
<BODY>
<h1><div align="center">Lectura de la tabla</div></h1>
<br>
<br>
<%
'Antes de nada hay que instanciar el objeto Connection
Set Conn = Server.CreateObject("ADODB.Connection")

'Una vez instanciado Connection lo podemos abrir y le asignamos la base de datos donde vamos a efectuar las operaciones
Conn.Open "Mibase"

'Ahora creamos la sentencia SQL que nos servira para hablar a la BD
sSQL="Select * From Clientes Order By nombre"

'Ejecutamos la orden
set RS = Conn.Execute(sSQL)

'Mostramos los registros%>
<table align="center">
<tr>
<th>Nombre</th>
<th>Teléfono</th>
</tr>
<%
Do While Not RS.Eof
%>
```

```
<tr>
<td><%=RS("nombre")%></td>
<td><%=RS("telefono")%></td>
</tr>
<%
RS.MoveNext
Loop

'Cerramos el sistema de conexion
Conn.Close
%>

</table>

<div align="center">
<a href="insertar.html">Añadir un nuevo registro</a><br>
<a href="actualizar1.asp">Actualizar un registro existente</a><br>
<a href="borrar1.asp">Borrar un registro</a><br>
</div>

</BODY>
</HTML>
```

Antes de nada, si lo que deseamos es interrogar una base de datos, lo primero que hay que hacer es obviamente establecer la conexión con ella. Esto se hace a partir del objeto Connection [http://www.desarrolloweb.com/articulos/2340.php] el cual es "invocado", o más técnicamente dicho instanciado, por medio de la primera instrucción en la cual el objeto toma el nombre arbitrario de la variable Conn.

El paso siguiente es abrir el objeto Connection [http://www.desarrolloweb.com/articulos/2340.php] y asignarle la base de datos con la que debe contactar. En este caso, la base la hemos llamado Mibase. Este debe de ser el mismo nombre con el que la hemos bautizado cuando hemos configurado los conectores ODCB, además, este nombre no tiene por qué coincidir necesariamente con el nombre del archivo.

Una vez creada la conexión a nuestra base de datos, el paso siguiente es hacer nuestra petición. Esta petición puede ser formulada primeramente y almacenada en una variable (sSQL) para, a continuación, ser ejecutada por medio de la instrucción siguiente.

La petición que hemos realizado en este caso es la de seleccionar todos los campos que hay en la tabla clientes (* es un comodín) y ordenar los resultados por orden alfabético con respecto al campo nombre. Podemos ver más detalladamente este tipo de instrucciones SQL en nuestro manual de SQL [http://www.desarrolloweb.com/manuales/9/].

El resultado de nuestra selección es almacenado en la variable RS en forma de tabla. Para ver la tabla lo que hay que hacer ahora es "pasearse" por esta tabla "virtual" RS la cual posee una especie de cursor que, a menos que se especifique otra cosa, apunta al primer registro de la selección. El objetivo ahora es hacer desplazarse al cursor a lo largo de la tabla para poder leerla en su totalidad. La forma de hacerlo es a partir de un bucle Do While el cual ya ha sido explicado anteriormente y que lo único que hace es ejecutar las instrucciones comprendidas entre el Do y el Loop siempre que la condición propuesta (Not RS.Eof) sea verdadera. Esto se traduce como "Ejecutar este conjunto de instrucciones mientras que la tabla de resultados (RS) no llegue al final (Eof, End of File).

Las instrucciones incluidas en el bucle son, por un lado, la impresión en el documento de los valores de determinados campos (=RS("nombre del campo")) y por otro, saltar de un registro al otro mediante la instrucción RS.MoveNext.

Todo este conjunto de instrucciones ASP viene en combinación con un código HTML que permite su visualización en forma de tabla. Además, se han incluido unos enlaces que apuntan hacia otra serie de scripts que veremos más adelante y que formaran en conjunto una aplicación.

Es interesante ver el código fuente resultante de este script. En este caso el código que ve el

cliente resulta sensiblemente más sencillo.

Para ver funcionar el script pincha aquí
[http://www.guiartemultimedia.com/desarrolloweb/cursoasp/lectura.asp]

Creación de un nuevo registro

En este caso lo que buscamos es crear, a partir de los datos recibidos de un formulario, un nuevo registro en nuestra tabla clientes. Tendremos pues dos archivos diferentes, uno que podría ser un HTML puro en el que introducimos el formulario a rellenar y que nos envía al segundo, un script muy parecido al previamente visto para realizar una selección. He aquí los dos scripts:

```
<HTML>
<HEAD>
<TITLE>Insertar.html</TITLE>
</HEAD>
<BODY>
<div align="center">
<h1>Insertar un registro</h1>
<br>
<FORM METHOD="POST" ACTION="insertar.asp">
Nombre<br>
<INPUT TYPE="TEXT" NAME="nombre"><br>
Teléfono<br>
<INPUT TYPE="TEXT" NAME="telefono"><br>
<INPUT TYPE="SUBMIT" value="Insertar">
</FORM>
</div>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>Insertar.asp</TITLE>
</HEAD>
<BODY>

<%
'Recogemos los valores del formulario
nombre=Request.Form("nombre")
telefono= Request.Form("telefono")

'Instanciamos y abrimos nuestro objeto conexion
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Mibase"

'Ahora creamos la sentencia SQL
sSQL="Insert Into Clientes (nombre,telefono) values ('" & nombre & "','" & telefono & "')"

'Ejecutamos la orden
set RS = Conn.Execute(sSQL)
%>

<h1><div align="center">Registro Insertado</div></h1>
<div align="center"><a href="lectura.asp">Visualizar el contenido de la base</a></div>

<%
'Cerramos el sistema de conexion
Conn.Close
%>

</BODY>
</HTML>
```

Como puede verse, la forma de operar es idéntica a la vista anteriormente para el display de una tabla. En este caso hemos introducido un enlace a este primer script de lectura para ver

cómo los cambios se han hecho efectivos.

La construcción de la sentencia SQL se hace por fusión de los distintos elementos constitutivos. La forma de fusionarlos mediante el símbolo **&**. Todo lo que sea texto tiene que ir entre comillas. Sería interesante introducir una línea suplementaria en vuestro código para imprimir la sSQL formada. La línea sería del siguiente tipo:

Response.Write sSQL

Esta línea iría situada evidentemente después de haber construido la sentencia.

Para ver funcionar el script pincha aquí
[http://www.guiartemultimedia.com/desarrolloweb/cursoasp/insertar.html]

Actualización de un registro existente

Para mostrar cómo se actualiza un registro presente en nuestra base de datos, vamos a hacerlo a partir de un caso un poco más complejo para que empecemos a familiarizarnos con estas operaciones. Realizaremos un par de scripts que permitan cambiar el numero de teléfono de las distintas personas presentes en nuestra base. El nombre de estas personas, así como el nuevo numero de teléfono, serán recogidos por medio de un formulario.

El archivo del formulario va a ser esta vez un script ASP en el que efectuaremos una llamada a nuestra base de datos para construir un menú desplegable donde aparezcan todos los nombres. La cosa quedaría así:

```
<HTML>
<HEAD>
<TITLE>Actualizar1.asp</TITLE>
</HEAD>
<BODY>
<div align="center">
<h1>Actualizar un registro</h1>
<br>

<%
'Instanciamos y abrimos nuestro objeto conexion
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Mibase"
%>

<FORM METHOD="POST" ACTION="actualizar2.asp">
Nombre<br>
<%
'Creamos la sentencia SQL y la ejecutamos
sSQL="Select nombre From clientes Order By nombre"
set RS = Conn.Execute(sSQL)
%>
<select name="nombre">
<%
'Generamos el menu desplegable
Do While not RS.eof%>
   <option><%=RS("nombre")%>
   <%RS.movenext
Loop
%>
</select>
<br>
Teléfono<br>
<INPUT TYPE="TEXT" NAME="telefono"><br>
<INPUT TYPE="SUBMIT" value="Actualizar">
</FORM>
</div>

</BODY>
</HTML>
```

La manera de operar para construir el menú desplegable es la misma que para visualizar la tabla. De nuevo empleamos un bucle Do While que nos permite mostrar cada una de las opciones.

El script de actualización será muy parecido al de inserción:

```
<TITLE>Actualizar2.asp</TITLE>
</HEAD>
<BODY>

<%
'Recogemos los valores del formulario
nombre=Request.Form("nombre")
telefono= Request.Form("telefono")

'Instanciamos y abrimos nuestro objeto conexion
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Mibase"

'Ahora creamos la sentencia SQL
sSQL="Update Clientes Set telefono='" & telefono & "' Where nombre='" & nombre & "'"

'Ejecutamos la orden
set RS = Conn.Execute(sSQL)
%>

<h1><div align="center">Registro Actualizado</div></h1>
<div align="center"><a href="lectura.asp">Visualizar el contenido de la base</a></div>

<%
'Cerramos el sistema de conexion
Conn.Close
%>

</BODY>
</HTML>
```

Nada que comentar al respecto salvo la estructura de la sentencia SQL que en este caso realiza un Update en lugar de un Insert. Aconsejamos, como para el caso precedente imprimir el valor de sSQL de manera a ver cómo queda la sentencia una vez construida.

Para ver funcionar el script pincha aquí
[http://www.guiartemultimedia.com/desarrolloweb/cursoasp/actualizar1.asp]

Borrado de un registro

Otra de las operaciones elementales que se pueden realizar sobre una base de datos es el borrar un registro. Para hacerlo, SQL nos propone sentencias del tipo Delete. Veámoslo con un ejemplo aplicado a nuestra agenda. Primero, crearemos un menú desplegable dinámico como para el caso de las actualizaciones:

```
<HTML>
<HEAD>
<TITLE>Borrar1.asp</TITLE>
</HEAD>
<BODY>
<div align="center">
<h1>Borrar un registro</h1>
<br>
<%
'Instanciamos y abrimos nuestro objeto conexion
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Mibase"
%>

<FORM METHOD="POST" ACTION="borrar2.asp">
```

```
Nombre<br>
<%
'Creamos la sentencia SQL y la ejecutamos
sSQL="Select nombre From clientes Order By nombre"
set RS = conn.execute(sSQL)
%>
<select name="nombre">
<%
'Generamos el menu desplegable
Do While not RS.eof%>
   <option><%=RS("nombre")%>
   <%RS.movenext
Loop
%>
</select>
<br>
<INPUT TYPE="SUBMIT" value="Borrar">
</FORM>
</div>

</BODY>
</HTML>
```

El siguiente paso es hacer efectiva la operación a partir de la ejecución de la sentencia SQL que construimos a partir de los datos del formulario:

```
<HTML>
<HEAD>
<TITLE>Borrar2.asp</TITLE>
</HEAD>
<BODY>
<%
'Recogemos los valores del formulario
nombre=Request.Form("nombre")

'Instanciamos y abrimos nuestro objeto conexion
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Mibase"

'Ahora creamos la sentencia SQL
sSQL="Delete From Clientes Where nombre='" & nombre & "'"

'Ejecutamos la orden
set RS = Conn.Execute(sSQL)
%>

<h1><div align="center">Registro Borrado</div></h1>
<div align="center"><a href="lectura.asp">Visualizar el contenido de la base</a></div>

<%
'Cerramos el sistema de conexion
Conn.Close
%>

</BODY>
</HTML>
```

Para ver funcionar el script pincha aquí
[http://www.guiartemultimedia.com/desarrolloweb/cursoasp/borrar1.asp]

ActiveX Data Object

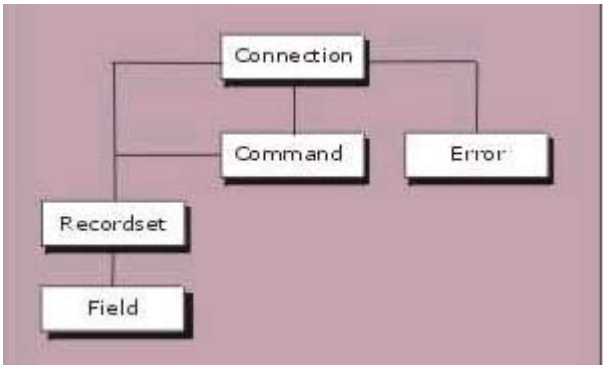
Una de las características mas interesantes de ASP es su facilidad para el manejo de bases de Datos que residen en el servidor. Esto lo conseguimos mediante el uso de ADO (ActiveX Data Object) de una forma fácil, rápida y con un mínimo consumo de recursos del sistema.

ADO usa ODBC para el acceso a bases de datos. lo que nos independiza de la tecnología de las mismas; esto implica que podemos cambiar la tecnología de la base de datos y si mantenemos

la misma estructura de datos, nuestras aplicaciones desarrolladas con ADO pueden seguir funcionando sin cambiar ni una sola línea de código.

Para desarrollo podemos crear nuestras fuentes de datos en Microsoft Access, pero en entornos de producción con gran afluencia de clientes deberemos de usar gestores de bases de datos mas potentes, como Oracle, Microsoft Sql Server, etc.

ADO esta formado por varios objetos organizados de forma jerárquica (cada uno de ellos con sus métodos y propiedades específicos) de los cuales vamos a estudiar los que considero mas interesantes.



Objetos

Connection

Nos proporciona una conexión a una base de datos ODBC desde una página ASP. Esta conexión nos permitirá efectuar las operaciones que deseemos sobre la base de datos.

Es el objeto primario de ADO, ninguno de los otros objetos puede existir si este no es declarado de forma explicita o implícita (en algunos de los ejemplos veremos que no existe una declaración del objeto Connection, pero debemos de tener en cuenta que siempre existe, si es necesario ADO lo declarará por si mismo).

La conexión terminará cuando nosotros la cerremos explícitamente con el método close o bien cuando termine la ejecución de la página ASP.

Error

Es una colección en la que se almacenaran los posibles errores del objeto Connection.

Command

Representa un comando SQL que se ejecuta contra la base de datos declarada en el objeto Connection.

Si el resultado de ese comando es un conjunto de datos, estos se almacenaran en un objeto de tipo Recordset.

Recordset

Representa una tabla o el resultado de una consulta ejecutada contra la base de datos. Va a ser nuestro interface natural contra la base de datos.

Como en todo modelo relacional, los datos se nos presentaran en filas y columnas.

Field

El objeto Field representa la información relativa a un campo de un Recordset.

Contiene la colección Fields que representa todos los campos de la tabla, cada miembro de esa colección es un objeto de tipo Field.

Objeto Connection (propiedades y métodos)

Hemos comentado que el objeto Connection [http://www.desarrolloweb.com/articulos/2313.php] nos proporciona una conexión a una base de datos desde una página ASP; ahora vamos a ver como se usa , así como sus propiedades y métodos.

Para establecer la conexión lo primero que hacemos es crear el Objeto Connetion por medio de la propiedad CreateObject de objeto Server:

```
<% Set conexion=Server.CreateObject("ADODB.Connection")%>
```

Una vez establecida la instancia del objeto pasamos a configurarlo mediante sus distintas propiedades y métodos.

Propiedades:

ConnectionString

Especifica la referencia a la base de datos con la cual queremos conectar, conteniendo en una cadena de texto la información necesaria para efectuar esa conexión mediante parejas de valores separadas por ";".

Los valores que podemos asignar son:

Data Source:	DSN=Nombre ODBC de la Base de Datos
Usuario:	UID=Nombre de Usuario
Password:	PWD=Password del usuario para la base de datos

Ejemplo:

```
<% conexion.ConnectionString="DSN=MIOdbc;UID=pepe;PWD=1234" %>
```

Mode

Especifica los permisos de la conexión.

Algunos de los valores mas habituales que podemos asignar son:

1 Establece permiso solo de Lectura
2 Establece permiso solo de Escritura
3 Establece permiso de Lectura/Escritura

Ejemplo:

```
<% conexion.Mode=3 %>
```

Métodos:

BeginTrans

Abre una transacción; todas las operaciones que realicemos a partir de ese momento no serán

efectivas hasta que no cerremos la transacción.

Ejemplo:

```
<% conexion.BeginTrans %>
```

Close

Cierra el objeto

Ejemplo:

```
<% conexion.close %>
```

CommitTrans

Cierra una transacción haciendo efectivos los cambios efectuados dentro de ella.;

Ejemplo:

```
<% conexion.CommitTrans %>
```

Execute

Ejecuta una sentencia SQL contra la base de datos.

Ejemplo:

```
<% Set resultado=conexion.execute (Select * from amigos) %>
```

Open

Abre la conexión con los parámetros especificados en las propiedades.

Ejemplo:

```
<% conexion.open %>
```

RollBackTrans

Deshace todos los cambios efectuados en la base de datos desde el inicio de la transacción.

Ejemplo:

```
<% conexion.RollBackTrans %>
```

Objeto Command de ASP (Propiedades y métodos)

Propiedades:

- **ActiveConnection**

  Especifica el objeto connection al que se refiere este objeto Command.

  Sintaxis

  objcommand.activeconnection=Nombre de la conexión

  Ejemplo:

```
<%Set conexion=Server.CreateObject("ADODB.Connection")
conexion.ConnectionString="DSN=MIOdbc;User=pepe;Password=1234"
conexion.open
objcommand.activeconnection=Conexion %>
```

- **CommandText**

  Es una cadena de texto con el comando a ejecutar.

  Sintaxis

  objcommand.Commandtext=comando sql a ejecutar

  Ejemplo:

```
<%Set conexion=Server.CreateObject("ADODB.Connection")
conexion.ConnectionString="DSN=MIOdbc;User=pepe;Password=1234"
conexion.open

Set objcommand=Server.CreateObject("ADODB.Command")
objcommand.activeconnection=Conexion

objcommand.commandtext="delete * from socios where pago=0"

%>
```

Metodos:

- **Execute**

  Ejecuta el comando almacenado en la propiedad CommandText..

  Sintaxis

  objcommand.execute

  Ejemplo:

```
<%Set conexion=Server.CreateObject("ADODB.Connection")
conexion.ConnectionString="DSN=MIOdbc;User=pepe;Password=1234"
conexion.open

Set objcommand=Server.CreateObject("ADODB.Command")
objcommand.activeconnection=Conexion objcommand.commandtext="delete * from socios where pago=0"
objcommand.execute%>
```

  Otro Ejemplo almacenando en un recordset:

```
<%Set conexion=Server.CreateObject("ADODB.Connection")
conexion.ConnectionString="DSN=MIOdbc;User=pepe;Password=1234"
conexion.open

Set objcommand=Server.CreateObject("ADODB.Command")
objcommand.activeconnection=Conexion objcommand.commandtext="select * from socios where codigo > 100"


Set resultado=Server.CreateObject("ADODB.recordset")
set resultado=objcommand.execute() %>
```

Objeto Application

El objeto Application se utiliza para compartir información entre todos los usuarios de una aplicación (entendemos por una aplicación ASP todos los archivos .asp de un directorio virtual y sus subdirectorios. Como varios usuarios pueden compartir un objeto Application, existen los métodos Lock y Unlock para asegurar la integridad del mismo (varios usuarios no puedan modificar una misma propiedad al mismo tiempo).

Lock

El método Lock asegura que sólo un cliente puede modificar o tener acceso a las variables de Application al mismo tiempo.

Sintaxis:
ApplicationLock

Unlock

El método Unlock desbloquea el objeto Application para que pueda ser modificado por otro cliente después de haberse bloqueado mediante el método Lock. Si no se llama a este método de forma explícita, el servidor Web desbloquea el objeto Application cuando el archivo .asp termina o transcurre su tiempo de espera.

Sintaxis:
Application.Unlock

Ejemplo:

```
<% Application.Lock
Application("visitas") = Application("visitas")+1
Application.Unlock %>
Eres el visitante numero <%= Application("visitas") %>
```

Puedes ver un ejemplo en acción [http://asptutor.com/asp/ejemplos/application.asp]

En el ejemplo anterior el método Lock impide que más de un cliente tenga acceso a la variable Visitas al mismo tiempo. Si la aplicación no se hubiera bloqueado, dos clientes podrían intentar incrementar simultáneamente el valor de la variable Visitas. El método Unlock libera el objeto bloqueado de forma que el próximo cliente puede incrementar la variable.

Nota Importante:

En el objeto Application pueden almacenarse matrices, pero estas son almacenadas como un objeto, es decir, no podemos almacenar o recuperar un solo elemento de la matriz, si no que cargaremos o recuperaremos la variable con la matriz completa

Ejemplo:

```
<%Dim parametros(2)
parametros(0) = "verde"
parametros(1) = 640
parametros(2) = 480
Application.Lock
Application("Param") =parametros%>
Application.UnLock
```

con estas instrucciones almacenaríamos TODA la matriz en la variable de aplicación "Param"

Para recuperar los valores de la matriz primero recuperamos esta en una variable normal

<%Apliparam=Application("Param")%>

Ahora podremos operar con los valores de la tabla en las variables Apliparam(0), Apliparam(1) y Apliparam(2)

Autores del manual:

Hay que agradecer a diversas personas la dedicación prestada para la creación de este manual. Sus nombres junto con el número de artículos redactados por cada uno son los siguientes:

- **Rubén Alvarez**
  (18 capítulos)
- **Miguel Gonzalez**
  (1 capítulo)
- **Pedro Rufo Martín**
  Webmaster de www.asptutor.com
  http://www.asptutor.com/
  (4 capítulos)

Volver [http://www.desarrolloweb.com/manuales/8]