Introducción al diseño de Aplicaciones Web con Active Server Pages

Pedro Rufo Martín

INDICE

Páginas Active Server

Páginas Active Server (ASP, Active Server Pages) es un entorno para crear y ejecutar aplicaciones dinámicas e interactivas en la Web.

Se puede combinar páginas HTML, secuencias de comandos y componentes ActiveX para crear paginas y aplicaciones Web interactivas.

El modelo de Páginas ASP

Las páginas ASP comienzan a ejecutarse cuando un usuario solicita un archivo **.asp** al servidor Web a través del explorador. El servidor web llama a ASP, que lee el archivo solicitado, ejecuta las secuencias de comandos que encuentre y envía los resultados al explorador del cliente.

Puesto que las secuencias de comandos se ejecutan en el servidor, y NO en el cliente, es el servidor el que hace todo el trabajo necesario para generar las paginas que se envían al explorador. Las secuencias de comandos **quedan ocultas a los usuarios, estos solo reciben el resultado de la ejecución en formato HTML.**

Desaparece por tanto el problema de si el cliente puede o no ejecutar sentencias de comandos, el servidor Web solo envía el resultado en código HTML standard interpretable por cualquier explorador.

Crear Páginas ASP

Los archivos .asp son archivos de texto normales, no es necesario ningún editor especial para crearlos, puede usarse cualquier editor que genere código ascii.

Un archivo **.asp** puede contener texto, código HTML, código ASP o cualquier combinación de estos. Si no contiene código ASP se comporta como un archivo **.html** normal.

Nota: todos los archivos .asp requieren una parte de proceso por el servidor, por lo cual no es conveniente convertir a .asp los archivos que no contengan código.

Software necesario para la ejecución de Páginas Active Server

Para la implantación de un servidor Web que soporte ASP el software necesario es, si lo que estamos configurando es un servidor de alto rendimiento:

- WINDOWS NT 4.0 , 2000 o XP
- IIS 4.0 o 5.0 (INTERNET INFORMATION SERVER 4.0 - 5.0) Ó IIS3.0 + ASP.EXE

Para desarrollo o sistemas Intranet los requerimientos son más sencillos:

- WINDOWS 95 + PERSONAL WEB SERVER 1.0 + ASP.EXE
- WINDOWS 98 o Millenium + PERSONAL WEB SERVER 4.0

Tanto IIS como Personal Web Server pueden descargarse desde la web de Microsoft.

Nota: Personal Web Server 4.0 esta incluido en algunas de las distribuciones de Windows 98 en el directorio ADD-ONS\PWS

Internet Information Server

1. Introducción
2. Instalación de protocolo TCP/IP
3. Seguridad del sitio Web

Introducción

IIS es el software estándar que soporta comunicaciones Internet en Windows NT.

No es el único, ni proporciona todos los servicios posibles; sin embargo su importancia es enorme al haberse convertido en uno de los más extendidos; haciendo fuerte competencia a los servidores basados en plataformas UNIX.

El auge viene de la mano de la fuerte penetración de Windows NT, complementándose muy adecuadamente con este desde le punto de vista comercial y técnico.

Proporciona unas buenas prestaciones en equipos con muy diferentes prestaciones de hardware.

Es especialmente ventajoso en su utilización en redes Intranet debido a la compatibilidad y posibilidades de uso conjunto con los productos de la familia Microsoft (Word, Access, Odbc, etc.)

Todo indica que el conjunto NT-IIS-Explorer será utilizado de forma creciente para la publicación de datos en Intranet/Internet.

Los servicios básicos que nos proporciona IIS4 son WWW, FTP, Correo y NEWS.

La instalación de IIS es sobre NT Server 4.0, aunque se puede instalar sobre Workstation o Windows 95-98 la versión PWS 4 con la consiguiente perdida de prestaciones.

Workstation puede servir muy adecuadamente como banco de pruebas y aprendizaje.

Una de las principales ventajas de IIS4 es el soporte nativo de páginas ASP (también se soportan en IIS3 mediante la actualización pertinente).

Para publicar en **Intranet** necesitamos:

- Tarjeta adaptadora de red
- Un servidor DNS o WINS si deseamos usar nombres en vez de direcciones IP numéricas.

Para publicar en **Internet** necesitamos:

- Una tarjeta de comunicaciones
- Una conexión a Internet
- Una dirección IP registrada en un DNS

Todo esto nos lo proporciona los Proveedores de Servicios Internet (ISP), junto con la dirección IP de gateway de su servidor, a través del cual se realizaran los encaminamientos de la información.

Instalación del protocolo TCP/IP

Se configura a través de la ventana *Red* en el *Panel de Control* de Windows, en esta ventana configuramos los servicios, protocolos, adaptadores y enlaces.

En la pestaña *Protocolos* seleccionamos TCP/IP, si no aparece, lo añadiremos con el botón *Agregar*.

Una vez escogido pulsamos *Propiedades* para configurarlo.

Pestaña *Dirección IP:*

Configuramos por cada tarjeta:

- Adaptador (Tipo de tarjeta)
- Dirección IP
- Mascara de subred
- Gateway

Pestaña *Dirección DNS:*

- Nombre de Host + Dominio (identificación de la maquina que estamos configurando)
- Orden de búsqueda del servicio Dns
- Orden de búsqueda de sufijo de dominio (opcional)

Todos estos parámetros nos los proporciona nuestro proveedor ISP.

Mecanismos de seguridad

La seguridad de un sitio Web es especialmente importante, debido a la necesidad de garantizar su utilización por usuarios remotos.

IIS 4.o utiliza la seguridad de Windows NT y en algunos casos la amplia.

Se recomienda el uso del sistema de archivos NTFS de NT por su mayor seguridad.
Windows NT basa su seguridad en el sistema de *usuarios y contraseñas*, el uso adecuado de estas contribuye a mantener el equipo seguro.

La mayor parte de las peticiones de paginas Web son realizadas por clientes anónimos, en este caso, el servidor web se encarga de suplantar al usuario real mediante el uso de la cuenta del usuario anónimo.

Mecanismo de seguridad en una petición:

1. Comprobación de la dirección IP del cliente por IIS.
2. Comprobación de usuario y contraseña.
3. Comprobación de los permisos de acceso a archivos establecidos en el sistema NTFS.

Si cualquiera de estas comprobaciones es errónea, la petición no tendrá éxito.

Administración de la cuenta de usuario anónimo.

Cuando se instala IIS se crea automáticamente en NT el usuario anónimo con el nombre IUSR_Nombre del equipo y con la misma contraseña aleatoria que en IIS y el derecho de *Inicio de Sesión en Local.*

Conviene revisar los derechos de los grupos que tienen los grupos *Todos* e *Invitados* a los que pertenece el usuario anónimo.

Para que el usuario anónimo funcione correctamente debemos activar *Permitir Anónimos* en las propiedades del servicio Web.

Autenticación

Si se desea, se puede restringir la utilización de los servicios Web de tal forma que únicamente los clientes que proporcionan un nombre de usuario y una contraseña válidos puedan acceder a las paginas solicitadas.

En IIS existen 2 formas de autenticación:

- *Autenticación Básica*: El usuario y la clave se transmiten sin cifrar
- *Autentificación Desafío/Respuesta de Windows NT*: El usuario y la clave se transmiten cifrados; el usuario debe de estar dado de alta en el dominio de la maquina que ejecuta IIS y tener derechos de *Acceso al equipo desde la red*. Es muy adecuado en redes **Intranet**; precisa un cliente Internet Explorer en versión 2 como mínimo.

Generalmente se permiten simultáneamente *Anónimos y mecanismos de autenticación*, en este caso en primer lugar se usa el usuario *Anónimo* y si se produce un error por falta de permisos de acceso a un recurso, el cliente recibe una ventana de dialogo solicitándole las credenciales.

Establecimiento de permisos en los directorios y ficheros de un sitio Web (aspectos Básicos)

De forma genérica un sitio Web reside en

- Un directorio particular
- Los subdirectorios que parten del particular
- Los directorios virtuales

Cada uno de los elementos anteriores, en caso de existir, deberá poseer los suficientes permisos para que el sitio Web funcione correctamente, pero con las restricciones adecuadas para que el equipo este seguro.

Una buena metodología consiste en agrupar los ficheros según su naturaleza y de forma jerárquica; de manera que tengamos separadas distintas aplicaciones en distintos directorios, con sus documentos en subdirectorios.

La asignación general de permisos sigue la siguiente estructura:

- Programas CGI, ISAPI, etc

  Permiso de *Ejecución*
- Páginas ASP

  Permisos de *Lectura* y *Ejecución*
- Documentos estáticos HTML, Imágenes, etc

  Permiso de *Lectura*
- Bases de datos, ficheros auxiliares, etc.

  Permisos de *Lectura* y *Escritura.*

Nota: Se debe tener en cuenta que desde IIS se pueden establecer permisos de Lectura y Ejecución, y desde NT cualquier permiso implementado en NTFS. En caso de discrepancia se toma la opción más restrictiva.

Agregar secuencias de comandos

Como vimos anteriormente, una pagina ASP mezcla en el mismo archivo secuencias de comandos con código HTML standard. Las secuencias de comandos asp se distinguen del resto del texto del archivo mediante **delimitadores** (*un delimitador es un carácter o secuencia de caracteres que marca el principio o final de una unidad*).

En el caso de HTML , dichos delimitadores son los símbolos "<" y ">" que enmarcan las etiquetas Html. ASP utiliza los delimitadores

"**<%**" y "**%>**"

para enmarcar las secuencias de comandos.

Veamos esto con un ejemplo sencillo:

```
<HTML>
<BODY>
Hola, bienvenido a mi página, estamos a : <%=Now()%>
</BODY>
</HTML>
```

La función NOW() de VBScript devuelve la fecha y hora actuales.

Cuando el servidor Web procese la pagina nos devolverá el siguiente resultado al explorador:

Hola, bienvenido a mi página, estamos a : 4/1/2000 14:25:55 PM

Como vemos, el cliente, no recibe el código asp, recibe el resultado de la ejecución de dicho código.

Dentro de los delimitadores ASP se puede incluir cualquier instrucción, expresión, procedimiento u operador válido para el lenguaje de programación que estemos usando. Un ejemplo de esto seria una página que nos mostraría:

Buenos días

si la ejecutamos antes de las 12:00 o:

Buenas Tardes

si la ejecutamos después de esta hora, el código seria el siguiente:

```
<HTML>
<BODY>
<% If Time( ) >= #12:00:00 AM# And Time( ) <#12:00:00 PM# Then%>
Buenos días
<% Else%>
Buenas Tardes
<% End If%>
</BODY>
</HTML>
```

Establecer el lenguaje de la aplicación

ASP viene de forma nativa con dos motores de secuencia de comandos Microsoft Visual Basic Scriptig Edition (VBScript) y Microsoft JScript. Puede instalar y utilizar motores de otros lenguajes como REXX y Perl.

Para establecer el lenguaje principal de secuencia de comandos en todas las páginas de una aplicación, establezca la propiedad Lenguaje ASP predeterminado en la ficha Opciones de la Aplicación en el Administrador de Servicios Internet.

Para establecer el lenguaje principal de secuencia de comandos en una única página, hay que agregar la directiva <%@ LANGUAGE%> al principio del archivo .asp. La sintaxis de esta directiva es la siguiente:

```
<%@ LANGUAGE=Lenguaje_secuencia_comandos %>
```

donde *Lenguaje_secuencia_comandos* es el lenguaje principal de secuencia de comandos que va a establecer en esa página concreta. El valor de la página invalida el valor global de todas las páginas de la aplicación.

Todos los ejemplos de estas páginas están escritos en VBScript

Comentarios en VBScript

VBScript acepta comentarios marcados con apóstrofes. Estos comentarios se eliminan al procesarse la secuencia de comandos y no se envían al explorador.

```
'Esta linea y las siguientes son comentarios.
'La función ImprimirTabla imprime los elementos de una matriz.
Call ImprimirTabla (mimatriz( ))
```

No puede incluir comentarios en expresiones de resultados. Por ejemplo, la primera línea que sigue funciona, pero la segunda no, porque empieza con <%=

```
<% i=i+l 'incrementa i. Esta instrucción funciona.%>
<%= i 'imprime el valor i. Esta instrucción no funciona.%>
```

Distinguir entre mayúsculas y minúsculas

VBScript no distingue entre mayúsculas y minúsculas.

Tipos de datos en VBScript

Vbscript solo tiene un tipo de datos llamado **Variant.**

El tipo Variant es una clase especial de datos que puede contener diferentes tipos de información, se comporta como un número cuando se utiliza en un contexto numérico, y como una cadena de caracteres cuando se usa en un contexto de cadena, no obstante podemos forzar a que los números se comporten como cadenas poniéndolos entre comillas (" ").

Aunque solo existe un tipo de datos, podemos hacer distinciones mas precisas acerca de la naturaleza de la información a través de los ***Subtipos*** incluidos en el tipo Variant, además vbscript pone a nuestra disposición funciones para convertir los datos de un tipo a otro.

Subtipo	Descripción	Valor de Vartype
Empty	Variable sin inicializar	0
Null	Variable intencionadamente vacia	1
Boolean	Dos valores posibles **True** o **False**	11
Byte	Entero entre 0 y 255	17
Integer	Entero entre -32.768 y 32.768	2
Currency	Numero entre -922.337.203.685.477,5808 y 922.337.203.685.477,5807	6
Long	Numero entre -2.147.483.648 y 2.147.483.647	3
Single	Numero de precisión simple	4
Double	Numero de doble precisión	5
Date	Fecha entre 1-1-100 y 31-12-9999	7
String	Cadena de longitud variable hasta 2.000.000.000 de caracteres.	8
Object	Contiene un Objeto	9
Error	Contiene un numero de error	10

Conversión de Tipos

Todas la funciones de conversión de tipos tienen la misma sintaxis:

Funcion(expresión), siendo *expresión* el dato que se desea convertir.

Cbool	Convierte una expresión a tipo Boolean
Cbyte	Convierte una expresión a tipo Byte
Clnt	Convierte una expresión a tipo Integer
Clng	Convierte una expresión a tipo Long
Csng	Convierte una expresión a tipo Single
Cdbl	Convierte una expresión a tipo Double
Ccur	Convierte una expresión a tipo Curency
Cdate	Convierte una expresión a tipo Date
Cstr	Convierte una expresión a tipo String

- También podemos podemos conocer el subtipo de una variable mediante la función **Vartype(variable)** que nos devuelve el valor referenciado en la tercera columna de la tabla 1

Variables en VBScript

Vbscript no necesita la declaración explicita de variables, pero es conveniente su declaración para evitar errores (se puede forzar la declaración de variables incluyendo la sentencia <% Option Explicit %> al principio de la pagina .asp).

Para declarar una variable se utiliza la instrucción DIM. PUBLIC o PRIVATE. Por ejemplo:

<% Dim Mivariable %>

Restricciones del los nombres de variables:

- Debe comenzar con un carácter alfabético
- No puede contener un punto
- No debe superar los 255 caracteres

Asignación de valores a una variable escalar (variable que contiene un **único** valor):

```
MiVariable = "pepito"
```

Nota: al asignar valores a las variables debemos atenernos a las siguientes normas:

- *Los valores de cadena se asignan entre comillas -> MiVariable = "pepito"*
- *Los valores numericos se asignan sin comillas -> MiVariable = 33*
- *Los valores de fecha se asignan entre almohadillas -> MiVariable = #12-1-1999#*

Declaración de Matrices:

Se declaran del mismo modo que las escalares, con la diferencia de que las matrices utilizan paréntesis ( ) a continuación del nombre de la variable; dentro del paréntesis pondremos el número de elementos de que consta la matriz.

```
Dim MiMatriz(10)
```

NOTA IMPORTANTE: Vbscript numera los elementos a partir del 0, lo que implica que una matriz definida como MiMatriz(5) tendría auténticamente 6 elementos: MiMatriz(0), MiMatriz(1), MiMatriz(2), MiMatriz(3), MiMatriz(4), MiMatriz(5)

Para asignar un valor a una posición cualquiera de la matriz simplemente nos referiremos al indice de la matriz que queremos actualizar:

```
MiMatriz(2) = 122
```

Lo mismo para recuperar un valor almacenado:

```
MiVariable = MiMatriz(2)
```

Las matrices en VBScript pueden tener hasta 60 dimensiones separadas por comas, por ejemplo, la siguiente instrucción define una matriz de 6 filas y 11 columnas:

```
Dim MiMatriz(5,10)
```

También podemos definir matrices que cambien de tamaño durante la ejecución de la secuencia de comandos (matrices dinamicas), para ello las declararemos sin poner el numero de dimensiones *Dim MiMatriz()* y determinaremos las dimensiones con la sentencia **Redim:** *Redim Mimatriz(22).* Si queremos conservar los valores almacenados en la matriz cuando variamos su tamaño debemos añadir la sentencia **Preserve** :

```
Redim Mimatriz(12)
.....
Redim Preserve Mimatriz(20)
```

Constantes en VBScript

Son variables que nunca cambian, se definen con la sentencia **CONST**

Const Miconstante = "texto que nunca cambia"

Objetos integrados de ASP

- **Objeto Application:** el objeto Application se utiliza para compartir información entre todos los usuarios de una aplicación.

- **Objeto Request:** el objeto Request se utiliza para tener acceso a la información que se pasa en las peticiones HTTP. Entre dicha información se incluyen los parámetros que se pasan desde los formularios HTML mediante el método POST o el método GET, cookies y certificados de cliente.

- **Objeto Response:** el objeto Response se utiliza para controlar la información que se envía al usuario. Esto incluye el envío de información directamente al explorador, la redirección del explorador a otra dirección URL o el establecimiento de valores de las *cookies*.

- **Objeto Server:** el objeto Server proporciona acceso a los métodos y las propiedades del servidor. El método utilizado más frecuentemente es el que crea una instancia de un componente ActiveX (Server.CreateObject).

- **Objeto Session:** el objeto Session permite almacenar la información necesaria para una determinada sesión de usuario. Las variables almacenadas en el objeto Session no se descartan cuando el usuario pasa de una página a otra dentro de la aplicación, si no que dichas variables persisten durante todo el tiempo que el usuario tiene acceso a las páginas de la aplicación. También puede utilizar los métodos de Session para terminar explícitamente una sesión y establecer el periodo de tiempo de espera de inactividad de las sesiones.

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") %>
```

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.

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)

Objeto Request

El Objeto Request recupera los valores que el cliente pasa al servidor durante una petición HTTP.
Dependiendo de la forma en que enviemos los datos al servidor tendremos que utilizar una u otra de las diversas colecciones del objeto Request. Las mas típicas son:

- **FORM** recupera datos enviados desde un formulario mediante el método POST.
- **QUERYSTRING** recupera datos enviados como cadena de consulta HTTP.
- **COOKIES** recupera los valores de las Cookies.

Sintaxis General:

Request.coleccion(elemento)

Ejemplos:

FORM

Supongamos que enviamos la información desde el siguiente formulario:

```
<form method="POST" action="recibir.asp">
<p>Nombre: <input type="text" name="Nombre" size="20"></p>
<p>Nacionalidad: <input type="text" name="Nacionalidad" size="20"></p>
<p><input type="submit" value="Enviar" name="Enviar"></p>
</form>
```

En nuestra página "recibir.asp" podriamos usar la siguiente secuencia:

```
Hola Sr/a <%=request.form("nombre")%> <br>
Asi que usted es de nacionalidad <%=request.form("nacionalidad")%>
```

Con lo que el resultado seria:

Hola Sr/a Julian
Asi que usted es de nacionalidad francesa

QUERYSTRING

Supongamos que enviamos la información en forma de cadena de consulta (Notar que una cadena de consulta HTTP esta especificada por las parejas de valores que siguen al signo "?"):

```
<a href="recibir.asp?nombre=Julian&nacionalidad=francesa">
```

En nuestra página "recibir.asp" podriamos usar la siguiente secuencia:

```
Hola Sr/a <%=request.querystring("nombre")%> <br>
Asi que usted es de nacionalidad <%=request.querystring("nacionalidad")%>
```

Con lo que el resultado sería:

Hola Sr/a Julian
Asi que usted es de nacionalidad francesa

Objeto Response

El Objeto response se usa para enviar resultados al navegador cliente o establecer valores de Cookies.

Sintaxis general:

Response.metodo [valor]

Entre los métodos mas interesantes del objeto Response estan los siguientes:

WRITE

El método Write escribe una cadena de resultado en el navegador cliente (Nota: cuando se usa la sintaxis <%=variable%> estamos usando implícitamente el método Response.Write).

Ejemplo:

```
<% response.write "<center>Hola mundo</center>" %>
```

obtenemos

Hola mundo

REDIRECT

El método Redirect hace que el explorador se conecte con una dirección URL diferente.

Nota: debemos usar este metodo antes de enviar cualquier resultado al navegador cliente, en caso contrario produce un error.

Ejemplo:

```
<% response.redirect "www.renfe.es"%>
```

El navegador se dirigirá a la URL especificada

Objeto Server

El objeto Server nos proporciona acceso a métodos y propiedades del servidor.

Propiedades:

ScriptTimeout

Especifica la cantidad maxima de tiempo que puede tardar la ejecución de una secuencia de comandos (Tiempo maximo que puede tardar en ejecutarse una página dada).

Sintaxis

Server.ScriptTimeout= nº de segundos

Ejemplo:

```
<% Server.ScriptTimeout=120 %>
```

La página puede ejecutarse durante 120 segundos antes de que el servidor la termine.

Metodos:

CreateObject

Crea una instancia de un componente ActiveX en el servidor.

Sintaxis

Server.CreateObject (IdProg)

IdProg es el identificativo del tipo de componente que queremos crear, nos viene suministrado por el fabricante del componente.

Ejemplo:

```
<% set Mitabla = CreateObject("ADODB.Recordset") %>
```

Instancia un objeto de tipo recordset y lo asigna a la variable "Mitabla".

Objeto Session

El objeto Session permite almacenar la información necesaria par una sesión de usuario contra nuestra aplicación ASP. Las variables que almacenemos en el objeto Session no se pierden al cambiar de página, si no que se mantienen hasta que el cliente sea eliminado por el servidor.

Las variables de Session de un cliente solo pueden ser accedidas por ese cliente.

El servidor crea automáticamente el objeto Session cuando un usuario que no tenga actualmente una sesión solicita una pagina Web de la aplicación.

Nota: el servidor elimina un cliente bien cuando desde una pagina ASP se invoca el método Abandon o bien cuando este cliente lleva 20 minutos sin actividad en nuestra aplicación.

Creación de una variable en Session

Sintaxis

Sesion("Nomvariable")= valor

Ejemplo:

```
<% Session("Color")="Rojo" %>
```

Para recuperar ese valor:

```
<% ColorFavorito=Session("Color") %>
```

Esto nos almacenaría el valor "rojo" en la variable 'ColorFavorito"

En el objeto Session 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 cestacompra(2)
cestacompra(0) = 1
cestacompra(1) = 8
cestacompra(2) = 22
Session("Cesta") =cestacompra%>
```

con estas instrucciones almacenaríamos TODA la matriz en la variable de sesión "Cesta"

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

```
<% Micesta=Session("Cesta")%>
```

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

Metodos:

Abandon

Destruye todos los objetos y variables almacenados en el objeto Session.

Ejemplo:

```
<% Session.Abandon %>
```

El archivo Global.asa

El archivo Global.asa es un fichero de texto situado en el **directorio raiz** de nuestro servidor Web, es decir, en el directorio de comienzo de nuestras páginas.

Es un archivo de comandos que nos permite la automatización de los cuatro eventos básicos de nuestro servidor.

La estructura es siempre la misma:

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

Sub Application_OnStart
........
End Sub

Sub Application_OnEnd
........
End Sub

Sub Session_OnStart
........
End Sub

Sub Session_OnEnd
........
End Sub

</SCRIPT>
```

Eventos:

Application_OnStart

El evento Application_OnStart se ejecuta antes de que se cree la primera nueva sesión; es decir justo cuando el primer cliente pide una pagina de nuestro servidor.

Application_OnEnd

El evento Application_OnEnd se ejecuta cuando la aplicación termina.

Session_OnStart

El evento Session_OnStart se ejecuta cuando el servidor crea una nueva sesión; esta secuencia de comandos es ejecutada antes de enviar la página solicitada al cliente.

Session_OnEnd

El evento Session_OnEnd se ejecuta cuando se abandona o se supera el tiempo de espera de una sesión.

Ejemplo de Global.asa

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

Sub Application_OnStart
dim mitabla()
redim mitabla(9)
application("tabla")=mitabla
End Sub

Sub Application_OnEnd

End Sub

Sub Session_OnStart
paginaInicio="/appl/index.html"
response.redirect paginaInicio
End Sub

Sub Session_OnEnd

End Sub

</SCRIPT>
```

Cookies

Las cookies son el mecanismo que nos permite guardar información relativa a un usuario a lo largo de sus distintos accesos a nuestras páginas.

Nos permite integrar funcionalidades como:

- Personalización de opciones de cliente
- Personalización en función de las características del cliente
- Cestas de compra
- Etcétera.
-

Las cookies se almacenan en los equipos de los clientes, esto hay que tenerlo en cuenta por las posibles faltas de integridad de datos que pudieran ocurrir.

ASP implementa la posibilidad de usar cookies para crear o destruir información que se almacena en los equipos de los clientes.

Las cookies se transmiten en las cabeceras cuando se realiza la comunicación http y es el navegador el encargado de almacenarlas.

Las cookies se implementan como una colección y se usan mediante los objetos integrados Request y Response.

Tiempo de vida de una cookie

Por defecto una cookie tiene un ámbito de sesión, es decir, tiene de vida el tiempo en que esta activo el navegador.

Podemos variar el tiempo de vida de una cookie mediante el atributo expires**.**

Sintaxis:

Response.Cookies(cookie)[(clave)|.atributo] = valor
Request.Cookies(cookie)(clave)

Ejemplos:

Enviar una cookie simple

```
<% Response.cookies("color")="morado" %>
```

Recuperar el valor de esa cookie

```
<% ColorFavorito=Request.cookies("color")%>
```

Enviar una cookie con claves

```
<% Response.cookies("color")("fondo")="morado"%>
<% Response.cookies("color")("texto")="blanco"%>
```

Recuperar una cookie con claves

```
<% Request.cookies("color")("fondo")%>
```

Nos recuperaría el valor ***morado***

```
<% Request.cookies("color")("texto") %>
```

Nos recuperaría el valor ***blanco***

Nota: Cuando usamos Response *para escribir una cookie, si esta ya existía se sobrescribe.*

Atributos:

Expires

Establece el día de caducidad de una cookie

Ejemplos:

Hacer que una cookie caduque inmediatamente

```
<% Response.cookies(cookie).expires="1/1/1990"%>
```

Hacer que una cookie caduque cierto día

```
<% Response.cookies(cookie).expires="12/12/2000"%>
```

Componentes ActiveX

Los componentes ActiveX se han diseñado para que se ejecuten en el servidor Web como parte de las aplicaciones Web, proporcionan funcionalidad a las aplicaciones, como el acceso a ficheros, Bases de datos, etcétera.

Existen componentes ActiveX para tareas muy diversas, en esta páginas mostraremos como operar con algunos de los que se incluyen por defecto en la instalación de ASP.

Adrotator	Inserción de publicidad rotatoria
FileSystemObject	Acceso a ficheros en el servidor
TextStream	Acceso a ficheros en el servidor
ActiveX Data Object	Acceso a bases de datos

Componente ADRotator

El componente ADRotator automatiza la rotación de imágenes de anuncio en una página Web. Cada vez que un cliente abre o recarga la página este componente presenta una nueva imagen según las definiciones especificadas en un archivo.

Archivos necesarios:

- **Archivo Rotator Schedule:** es un archivo de texto que contiene la agenda de presentación de los anuncios.
- **Archivo de redirección:** es un archivo .asp que implementa la redirección a la URL anunciante.

Creación del objeto AdRotator

```
<% Set Rotacion=Server.CreateObject("MSWC.AdRotator") %>
```

Propiedades:

Border

Permite especificar si los anuncios se presentan enmarcados.

```
<% objeto.border=tamaño %>
```

Clickable

Permite especificar si los anuncios se presentan como hipervinculos.

```
<% objeto.clickable= True o False %>
```

TargetFrame

Permite especificar el marco de destino del hipervinculo.

```
<% objeto.TargetFrame= nombre del marco destino %>
```

Metodos:

GetAdvertisement

Recupera el siguiente anuncio del fichero Schedule.

Sintaxis

```
Objeto.GetAdvertisement (url del fichero Shedule)
```

Estructura del fichero ScheduleEl fichero Schedule esta dividido en 2 secciones **separadas por un asterisco "*"**. La primera sección es la especifica los parámetros comunes para todas las imágenes que se muestren; la segunda los ficheros , localizaciones y parametros propios de cada una de las imagenes.

Sintaxis de la primera sección:

REDIRECT	Especifica la url que se encargara de hacer la redirección, generalmente una página .asp.
WIDTH	Ancho en píxel del anuncio
HEIGHT	Alto en píxel del anuncio
BORDER	Ancho en píxel del borde del anuncio

Sintaxis de la segunda sección:

Url de la imagen a mostrar
Url de la pagina a redireccionar
Texto alternativo de la imagen
Ponderación de apariciones del anuncio con respecto del total

Ejemplo:

```
REDIRECT        /util/redirect.asp
WIDTH           300
HEIGHT          50
BORDER          2
*
/imagenes/logo1.gif
http://www.transcontinental.com
El viaje de tus sueños
20
/imagenes/logo5.jpg
http://www.pastelito.fr
Dulces de calidad
30
```

Ejemplo de fichero de redirección

```
<% response.redirect (request.querystring("url")) %>
```

Ejemplo de una página completa

```
<html>
<head><title>Uso de AdRotator</title></head>
<body><h2>Uso de AdRotator</h2>
<% Set Rotacion=Server.CreateObject("MSWC.AdRotator") %>
<=% Rotacion.GetAdvertisement("adrot.txt") %>
</body>
</html>
```

Componente FileSystemObject

El componente FSO nos permite abrir y crear ficheros de texto en el servidor.

Este componente consta de 22 métodos, de los cuales podemos seleccionar 2 que son los que nos van a permitir leer o escribir en archivos de texto existentes en el servidor o crear dichos archivos.

Para crear un objeto FSO la sintaxis es la misma que para cualquier otro componente ActiveX

```
<% Set MiFSO=Server.CreateObject("Scripting.FileSystemObject") %>
```

Cuando abrimos o creamos un fichero de texto mediante FSO este nos devuelve una instancia del objeto TextStream que es la que representa el archivo físico y con la cual trabajaremos.

Metodos:

CreateTextFile

Crea un archivo físico y devuelve la instancia de TextStream con la cual trabajaremos.

Sintaxis

```
<% Set MiFichero=MiFSO.CreateTextFile("Nombre Fichero",Sobreescribir") %>
```

Nombre Fichero	Nombre del fichero a crear.
Sobreescribir	Admite los valores TRUE o FALSE, si el fichero ya existe y el valor dado es TRUE se crea de nuevo, si no , devuelve un error.

OpenTextFile

Abre un archivo físico y devuelve la instancia de TextStream con la cual trabajaremos.

Sintaxis

```
<% Set MiFichero=MiFSO.OpenTextFile("Nombre Fichero",modo,crear") %>
```

Nombre Fichero	Nombre del fichero a abrir.
Modo	Indica si queremos abrir el fichero para lectura (**1**) , para escritura (**2**) o para escribir nuevos registros al final del fichero(**8**)
Crear	Admite los valores TRUE o FALSE, si el fichero no existe y el valor dado es TRUE se crea.

Ejemplo:

Apertura de fichero para lectura:

```
<% Set MiFichero=MiFSO.OpenTextFile("c:\Fichero_nuevo.txt",1,true") %>
```

Objeto TextStream

El objeto TextStream nos sirve para manejar ficheros de texto en el servidor. La creación de este objeto se realiza a partir de un objeto FileSystemObject y gracias a alguno de sus métodos.

Una vez creado, disponemos de un objeto TextStream que representa un archivo físico abierto, ya sea para lectura o escritura.

Este objeto dispone de 9 métodos:

Metodos:

Close

Cierra el archivo.

Sintaxis

```
<% MiFichero.close%>
```

Read

Lee y devuelve un numero especifico de caracteres.

Sintaxis

```
<% MiFichero.read(numero de caracteres) %>
```

ReadAll

Lee y devuelve un archivo completo.

Sintaxis

```
<% MiFichero.ReadAll %>
```

ReadLine

Lee y devuelve una línea completa de un archivo de texto.

Sintaxis

```
<% MiFichero.ReadLine%>
```

Skip

Salta un numero determinado de caracteres al leer un archivo.

Sintaxis

```
<% MiFichero.Skip(numero de caracteres) %>
```

SkipLine

Salta una línea al leer un archivo.

Sintaxis

```
<% MiFichero.SkipLine %>
```

Write

Escribe una cadena de caracteres en un archivo.

Sintaxis

```
<% MiFichero.Write("texto_entre_comillas") %>
```

WriteLine

Escribe una cadena de caracteres en un archivo añadiendo al final un carácter de fin de linea.

Sintaxis

```
<% MiFichero.WriteLine("texto_entre_comillas") %>
```

WriteBlankLines

Escribe un numero especifico de caracteres de nueva línea.

Sintaxis

```
<% MiFichero.WriteBlankLines(numero_de_lineas) %>
```

Ejemplo de escritura en un archivo

```
<HTML>
<HEAD><TITLE>Ejemplo de FSO y TextStream</TITLE></HEAD>
<BODY>
<%
Set Mfso=Server.CreateObject("Scripting.FileSystemObject")
Set MArchivo=Mfso.OpenTextFile("c:\fecha.txt",2,true)
MArchivo.writeline "Hola Mundo, hoy es:"
MArchivo.write date()
MArchivo.close
%>
Creado archivo en C:\fecha.txt con la fecha de hoy
</BODY>
</HTML>
```

Fuentes de datos ODBC

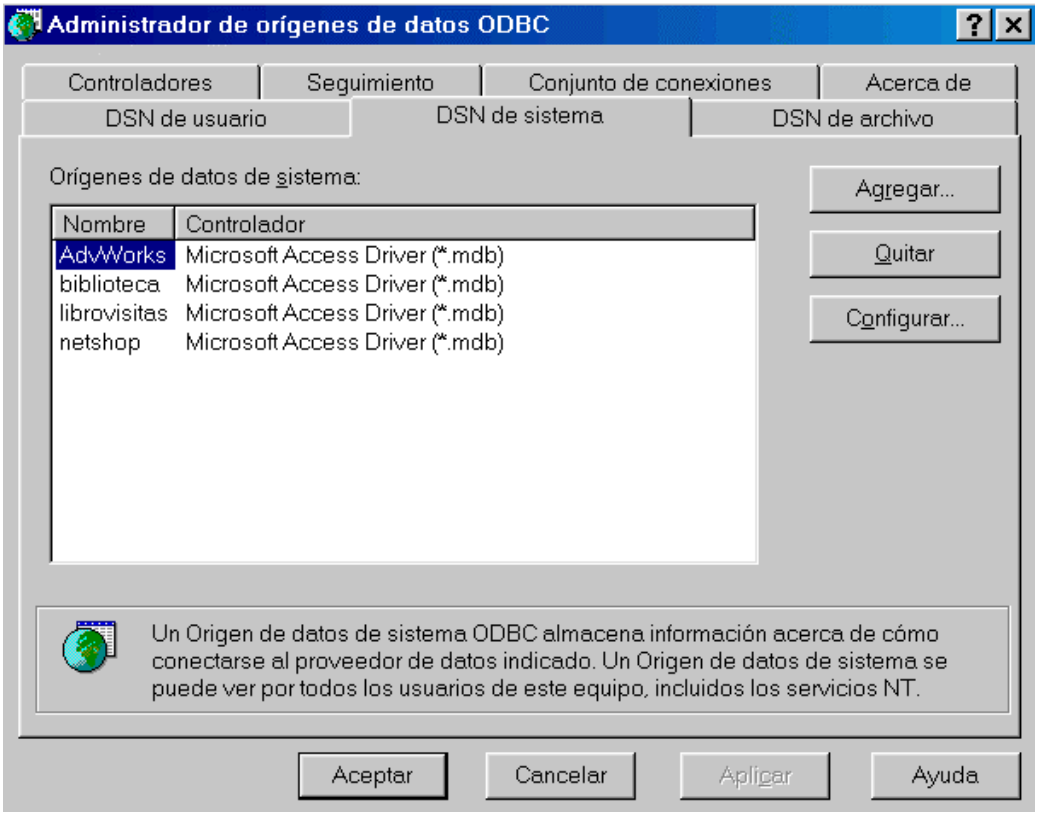
La administración de orígenes de datos ODBC (Open Database Connectivity) es una utilidad general de Windows NT.

Permite que las aplicaciones accedan a los datos a través usando SQL como lenguaje estándar.
Se administran a través de la ventana ODBC del *Panel de Control*.

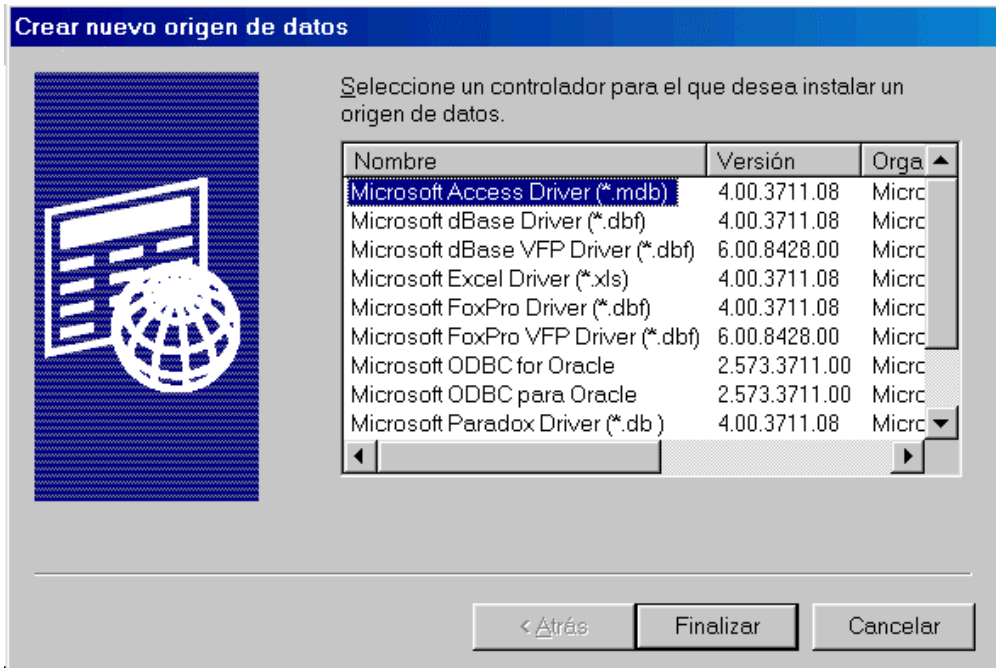
Se pueden configurar tres diferente fuentes de datos ODBC, la forma más interesante es la de DSN del Sistema, que presenta la ventaja de poder ser accedida por cualquier usuario, siendo el tipo usado en aplicaciones ASP.

Configuración de una fuente de datos En este ejemplo declaramos una base de datos Access, el procedimiento es prácticamente igual para cualquier otra tecnología de bases de datos.

Para declarar una base de datos ODBC haremos doble clik en el icono **Fuentes de Datos ODBC** que encontraremos en el panel de control de Windows (Inicio->Configuración->Panel de Control), una vez abierto el programa nos situaremos en la pestaña **DSN de Sistema**

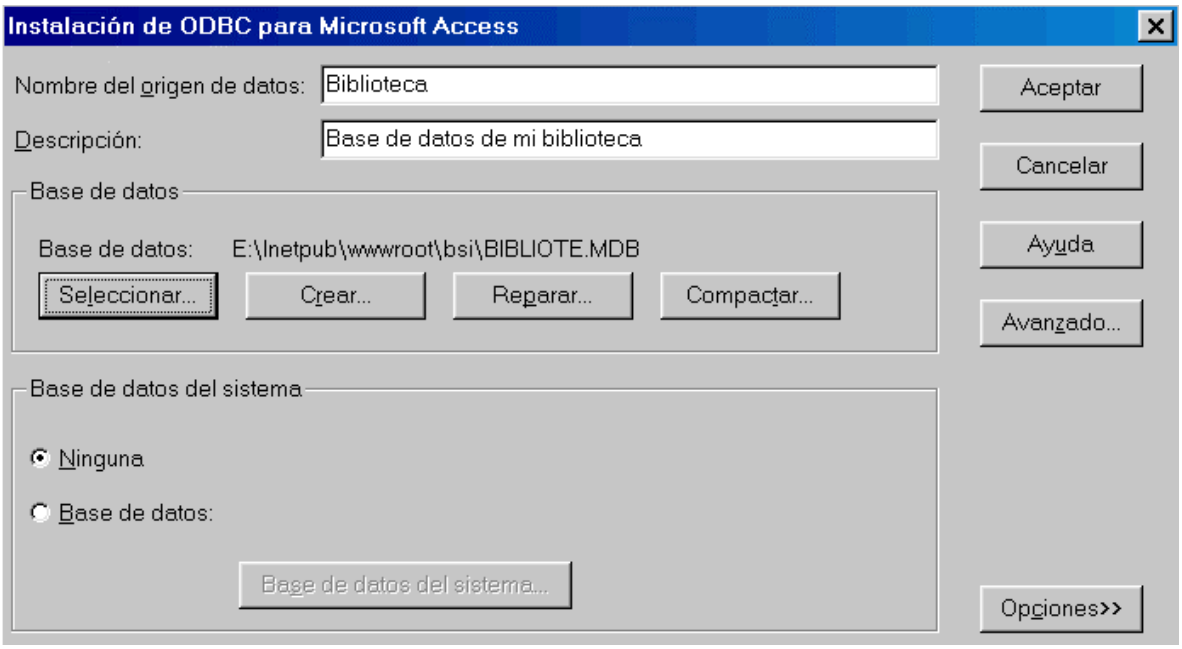


Pulsaremos el botón AGREGAR para añadir una nueva fuente de datos, lo que nos llevara a la pantalla de controladores.



En esta ventana elegiremos el driver adecuado para la tecnología de la base de datos que queremos tratar desde nuestras páginas ASP y pulsaremos finalizar. La siguiente pantalla que se nos muestra nos pedirá la ubicación física de nuestra base de datos y el nombre ODBC con el que queremos acceder a ella.

Nombre de origen de datos: Es el nombre simbólico de nuestra base de datos, será el que utilicemos desde ASP para referirnos a ella.


Instalación de ODBC para Microsoft Access
Nombre del origen de datos:
Biblioteca
Descripción:
Base de datos de mi biblioteca
Aceptar
Cancelar
Ayuda
Avanzado...
Base de datos
Base de datos:
E:\Inetpub\wwwroot\bsi\BIBLIOTE.MDB
Seleccionar...
Crear...
Reparar...
Compactar...
Base de datos del sistema
Ninguna
Base de datos:
Base de datos del sistema...
Opciones>>

Descripción: Es una descripción de la base de datos

Base de datos: Es el camino físico de la base de datos, para establecerlo pulsaremos el botón SELECCIONAR y mediante un explorador elegiremos nuestra fuente de datos.

Una vez hecho esto pulsaremos el botón ACEPTAR y ya tendremos nuestra base de datos disponible para su uso desde nuestras aplicaciones Web.

NOTA: Si nuestro sistema operativo es NT y declaramos una base de datos ACCESS debemos de tener en cuenta que ACCESS crea en el mismo directorio de la base de datos un fichero con extensión ***.ldb*** *cuando cualquier usuario interactúa con la base de datos. Ello nos obliga a dar derechos suficientes al usuario de IIS (IUSR_Nombre del equipo) para manipular el directorio de la base de datos.*

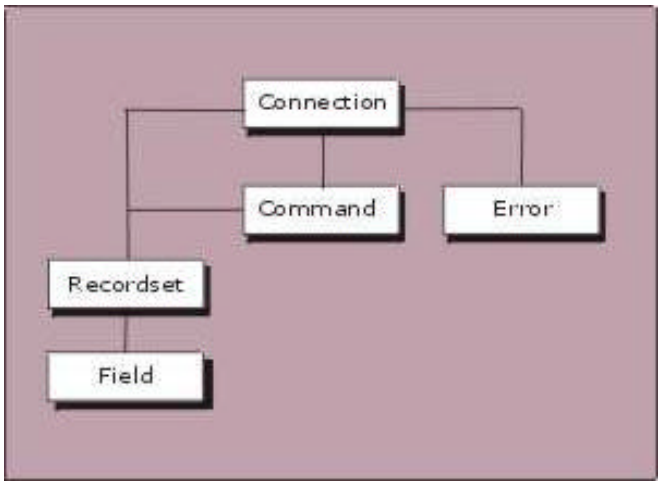
ActiveX Data Object

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, etcétera.

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

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 metodos)

Hemos comentado que el objeto Connection 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 conexión=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:	User=Nombre de Usuario
Password:	Password=Password del usuario para la base de datos

Ejemplo:

```
<% conexion.ConnectionString="DSN=MIOdbc;User=pepe;Password=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 %>
```

Metodos:

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.CommitBeginTrans %>
```

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

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

Ejemplo:

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

Objeto Error (propiedades y métodos)

El objeto Error contiene la colección **Errors**, que es la encargada de almacenar los errores que se pudieran producir durante la ejecución de operaciones contra Bases de Datos.

Propiedades:

Description

Descripción del error.

Number

El numero de error.

SQLState

Código de error SQL.

Metodos:

Clear

Elimina los datos del objeto Error.

Ejemplo:

Examinando los posibles datos de la colección Errors

```
.......
.......
Miconexion.open
If Miconexion.Errors.Count > 0 then
For each error in Miconexion.errors then
Response.write Error.Number & " = "& Error.Description
next
End if
```

*Nota: **Count** es una propiedad de la colección Errors.*

Objeto RecordSet

El objeto Recordset es el interface entre los datos obtenidos de nuestras consultas sobre las tablas y nuestras páginas asp. Representa una tabla organizada en filas (registros) y columnas (campos).

La propiedades y métodos de Recordset son muchos, en este capitulo vamos a ver las mas interesantes, para hacerlo un poco mas sencillo de entender vamos a verlos agrupados por la funcionalidad que nos ofrecen.

Definición del tipo de Cursor

Entendemos como cursor el puntero que nos permite desplazarnos por los registros del recordset. Dependiendo del tipo elegido determinaremos los desplazamientos y cambios realizables en los datos.

El tipo de cursor lo definiremos mediante la propiedad CursorType, los posibles valores son:

Denominación	valor	Características
adOpenForwardOnly	0	Es el cursor por defecto, solo nos permite recorrer la tabla de forma secuencial (no se puede volver hacia atrás) y no permite modificaciones en los registros. Por contra es el de menos consumo. No vemos los cambios realizados en la tabla por otro recordset.
adOpenKeyset	1	Nos permite movernos en los dos sentidos, si permite modificaciones en los registros. Vemos los cambios realizados en la tabla por otro recordset a excepción de las nuevas altas.
adOpenDynamic	2	Nos permite movernos en los dos sentidos, si permite modificaciones en los registros. Vemos Todos los cambios realizados en la tabla por otro recordset.
adOpenStatic	3	Nos permite movernos en los dos sentidos, no permite modificaciones en los registros. No vemos los cambios realizados en la tabla por otro recordset.

Definición del tipo de Cerrojo

Entendemos como cerrojo el tipo de bloqueo que efectuaremos en la base de datos cuando modifiquemos un recordset, a fin de evitar que dos o mas usuarios accedan a modificar un mismo registro a la vez.

El tipo de cerrojo lo definiremos mediante la propiedad LockType, los posibles valores son:

Denominación	valor	Características
adLockReadOnly	1	Es el defecto; no permite al usuario modificar los datos de la tabla.
dLockPessimistic	2	Cuando se abra la tabla nadie mas podrá hacerlo, este modo nos asegura la plena integridad de los datos.
adLockOptimistic	3	Cierra la tabla a los demás usuarios cuando se invoque al método Update del objeto recordset; de este modo la Base de datos quedará bloqueada menos tiempo que con el método anterior.

Ejemplo de definición de un recordset para actualizar datos:

```
Const adOpenForwardOnly = 0
Const adOpenKeyset = 1
Const adOpenDynamic = 2
Const adOpenStatic = 3
Const adLockReadOnly = 1
Const adLockPessimistic = 2
Const adLockOptimistic = 3%>

set rs=createobject("ADODB.Recordset")
rs.CursorType = adOpenKeyset
rs.LockType = adLockOptimistic
```

Moviéndose por los datos del RecordSet

Métodos usados:

Método	Características
Move *Num_registros*	Mueve el cursor *Num_registros* hacia abajo si es positivo y hacia arriba si es negativo
MoveFirst	Mueve el cursor al primer registro del Recordset
MoveLast	Mueve el cursor al ultimo registro del Recordset
MoveNext	Mueve el cursor un registro hacia adelante
MovePrevious	Mueve el cursor un registro hacia atrás

Propiedades usadas:

Propiedades	Caracteristicas
PageSize	Establece el numero de registros por página del recordset
	rs.Pagesize=10
AbsolutePage	Mueve el cursor al primer registro de dicha página (es necesario definir anteriormente el pageSize)
	rs.AbsolutePage=2
PageCount	Contiene el numero de páginas del recordset, tomando como base PageSize
	xx=rs.PageCount
Absoluteposition	Mueve el cursor al num_registro especificado
	rs.Absoluteposition=17
RecordCount	Contiene el numero de registros del recordset; Nota: No funciona con el cursor **adOpenForwardOnly**
	xx=rs.recordcount
BOF	Toma el valor True cuando estamos en el primer registro del recordset
EOF	Toma el valor True cuando estamos en el ultimo registro del recordset

Modificando los datos

Métodos usados:

Metodo	Caracteristicas
AddNew	Abre un nuevo registro en el recordset para introducir datos
	rs.Addnew rs("codigo")=1234 rs("titulo")="Todo sobre ASP" rs.Update
Delete	Elimina el registro actual
Update	Actualiza un registro del recordset tras haberlo modificado
	rs("titulo")="Como hacerse rico en 10 minutos" rs.Update

Abriendo y cerrando el recordset

Métodos usados:

Metodo	Caracteristicas
Open *Sql*, *conexion*	Abre el recordset y almacena en el el resultado de *sql* contra la *conexion* set rs=createobject("ADODB.Recordset") rs.CursorType = 1 rs.LockType = 3 Sqltxt="SELECT * FROM libros" rs.open Sqltxt, "DSN=Biblioteca"
Close	Cierra el recordset

Ejemplo de listado de un Recordset:

Listado.asp

```
<%SQLtxt = "SELECT Producto, Cantidad, Precio FROM almacen
set rs = CreateObject("ADODB.Recordset")
rs.Open SQLtxt, "DSN=Mibase"%>
<table>
<%
Do While NOT rs.EOF%>

<tr>
<td><% = rs("Producto")%></td>
<td><% = rs("Cantidad")%></td>
<td align="right"><% = FormatCurrency(rs("Precio"))%></td>
</tr>

<% rs.MoveNext
Loop
rs.Close
</table>
%>
```