

## Active Server Pages.

Mediante ASP se pueden crear y ejecutar aplicaciones sobre la Web. En esta sección se explican los fundamentos básicos para utilizarlo. Recuerde que para poder probar código ASP se **necesita** tener instalado un **servidor** que lo gestione. Tenga en cuenta que la mayoría de las explicaciones de esta sección son con **Visual Basic Script**, pero que también puede trabajarse sobre otros lenguajes. Hay disponibles **ejemplos en la sección** correspondiente de ejemplos.



- 1.- ¿Que es y que se necesita para utilizarlo?. 28-08-2000
- 2.- Software para interpretar ASP. 28-08-2000
- 3.- Estructura de un documento ASP. 28-08-2000
- 4.- Objetos y componentes de ASP. 28-08-2000



- 1.- Introducir comentarios en su código. 28-08-2000
- 2.- Variables y constantes. 28-08-2000
- 3.- Matrices. 28-08-2000
- 4.- Condicional if...then. 02-10-2000
- 5.- Decisiones múltiples con Select Case. 14-11-2000
- 6.- Ciclo: For...Next. 14-11-2000
- 7.- Ciclo: For...in. 14-11-2000
- 8.- Ciclo: Do...Loop. 26-12-2000
- 9.- Ciclo: While...Wend. 26-12-2000



- 1.- Función para saber de que subtipo es un dato. 28-08-2000
- 2.- Convertir entre subtipos. 28-05-2000

### ¿Que es?.

Las páginas ASP (Active Server Pages) son una tecnología que sirve para crear y ejecutar **aplicaciones** del lado del servidor sobre la Web, combinando **código HTML**, **secuencias de comandos** y **componentes ActiveX**.

### ¿Por qué usar ASP?.

La potencia del lenguaje y los componentes ActiveX nos permite desarrollar aplicaciones solucionando el tradicional problema de controlar la aplicación. Esta en si y las sesiones de usuario que quedan integradas en la propia estructura del lenguaje. Además el cliente sólo necesita **un programa navegador**, ya que es el **servidor quien ejecuta los comandos** y devuelve al cliente la página en HTML.

### ¿Qué se necesita para trabajar en ASP?

- Un *editor de texto* sencillo.
- Un *navegador ó browser*.
- Un *intérprete ASP*.
- Algo de *tiempo* para aprenderlo.

## ¿Como funciona el ASP?

Se ejecuta cuando un usuario solicita un archivo .asp al servidor Web con su navegador. El **servidor web** llama a ASP será el encargado de **interpretar** las secuencias de comandos y enviar los resultados al explorador del cliente en HTML (salvo excepciones especiales).

Es importante aclarar que existen generadores de código ASP, pero se pueden escribir con lo que se quiera, siempre que *se guarden en formato Texto* y se les coloque la *extensión asp*. No se trata de renombrarlos con esta extensión, sino de usar la opción que los guarda en este formato. El NOTEPAD está en casi todos los ordenadores tipo windows y trabaja en este formato, pero pueden usarse otros editores como el Texpad.

### Software necesario

Como se ha indicado el **servidor web** llama a ASP será el encargado de **interpretar** las secuencias de comandos y enviar los resultados al explorador del cliente en HTML. Esto requiere que el servidor web integre esta tecnología.

- Si lo va a ejecutar en el sitio **donde hospeda** sus páginas, contacte con el administrador y asegúrese de que **su servidor web las admite**. Normalmente, los servidores gratuitos no le ofrecen esta posibilidad y debe recurrir a uno de pago.
- Si desea instalarlo en **su equipo**, necesita **instalar un servidor web** con esta tecnología. Le recomendamos que utilice el PWS (Personal Web Server) si tiene Windows 95 o 98. Si tiene NT o 2000 el IIS (Internet Information Server). Existen otras posibilidades si tiene otro sistema operativo o no quiere usar productos Microsoft.

Es necesario que nuestro servidor web tenga el código ASP, en un directorio virtual que tenga **permisos de lectura y ejecución**. Recuerde que, ya no se llamará a los ficheros desde el navegador al estilo *c:\web\asp\algo.asp*. Esta forma de llamarlo no usa el servidor web y no interpreta el código. Ahora, en cambio, **se le piden al servidor** de la forma *http://www.suservidor.com/sitio/asp/algo.asp* (o *http://localhost/asp/algo.asp* si es en su propia máquina), para que las interprete como ASP y nos las devuelva procesadas.

Puede consultar como instalar su servidor web en [la sección de redes](#).

### Estructura de un documento ASP.

Un documento ASP se construye como una página HTML en la que en una parte de ella se inserta el código a interpretar. Esta sección de código queda encerrada entre dos **delimitadores**: `<%` y `%>`.

*Ejemplo: El famoso Hola mundo*

```
<HTML>
<BODY>
  <%= "Hola Mundo"%>
</BODY>
</HTML>
```

### Lenguaje de Programación

ASP viene de forma nativa con dos motores de secuencia de comandos: Microsoft Visual Basic Script (VBScript) y Microsoft Java Script (JScript). No obstante se pueden instalar y utilizar motores de otros lenguajes como REXX y Perl.

Si no se especifica nada en todas las páginas de una aplicación el intérprete supone que esta escrito en el **lenguaje principal de secuencia de comandos** (predeterminado en la ficha Opciones de la Aplicación en el Administrador de Servicios de nuestro servidor web).

Puede **cambiarse** el lenguaje principal de secuencia de comandos en una página, con la directiva `<%@LANGUAGE=Lenguaje%>` al principio del archivo .asp. Nuestro ejemplo podría quedar así:

*Ejemplo: El famoso Hola mundo*

```
<%@LANGUAGE="VBScript"%>
<HTML>
<BODY>
  <%= "Hola Mundo"%>
</BODY>
</HTML>
```

**IMPORTANTE:** De aquí en adelante se los ejemplos y la sintaxis con la que se trabajará será de **Visual Basic Script**, pero recuerde que también puede trabajarse sobre otros lenguajes.

### **Objetos**

Los **objetos ASP** son componentes ActiveX siempre disponibles en el lenguaje (en esto se diferencian básicamente de las DLL de Visual Basic). No hay que crear explícitamente los objetos para emplearlos. De forma natural, ASP maneja objetos como Application, Session, Request, Response y Server.

<b>Objeto</b>	<b>Descripción</b>
Application.	Manipular variables de aplicación, disponibles para todos los usuarios de la aplicación contenida en el mismo directorio virtual.
Session.	Manipular variables de sesiones, disponibles para un sólo usuario.
Request.	Para recoger los datos de un cliente mediante el envío de formularios.
Response.	Gestiona el contenido que se le proporciona al navegador.
Server.	Proporciona una heterogénea gama de funciones. Entre ellas destaca la creación de instancias de objetos ActiveX.

### **Componentes**

En cambio los **componentes ASP** son componentes ActiveX que vienen en DLLs que se crean fuera de ASP y que se pueden generar desde cualquier **lenguaje**. Se han de crear explícitamente. Microsoft Visual InterDev permite manejar objetos como Database Access, File Access, Browser Capabilities, Ad Rotator y Content Linking.

<b>Objeto</b>	<b>Descripción</b>
Database Access. Objeto de datos Active X (ADO).	Permite el acceso a Bases de Datos tipo ODBC.
File Access.	Permite el acceso a ficheros mediante el objeto FileSystem y TextStream.
Browser Capabilities.	Para identificar el navegador y acceder a sus posibilidades.
Ad Rotator. Rotor de anuncios	Gestiona la rotación de espacios publicitarios.
Content Linking. Enlace de contenidos.	Proporciona herramientas para la publicación de contenidos on line.

## Comentarios.

Se hacen poniendo un apóstrofe. El resto de línea no se interpretará como código ASP.

*Ejemplo: El famoso Hola mundo*

```
<HTML>
<BODY>
  <%
  'Saludo al mundo
  Response.write("Hola Mundo")
  %>
</BODY>
</HTML>
```

No pueden ponerse comentarios en expresiones de resultados. El siguiente código daría error.

*Ejemplo: El famoso Hola mundo*

```
<HTML>
<BODY>
  <%= "Hola Mundo" 'Este comentario da error%>
</BODY>
</HTML>
```

## Variables.

Solo hay un tipo de datos: el tipo **Variant**. Es una clase especial de datos que puede contener diferentes tipos de información. Se comporta **según el contexto** como un tipo o como otro.

No necesita la declaración explícita de variables, pero es una buena costumbre. Se utiliza la instrucción DIM. PUBLIC o PRIVATE Puede forzarse la declaración de variables incluyendo la sentencia <% Option Explicit %> al principio de la página.

Puede precisarse el tipo un poco más con los **subtipos** incluidos en el tipo Variant. La función Vartype(Variable) nos permite distinguir entre ellos.

Valor de Vartype	Subtipo	Descripción	Rango
0	Empty.	Sin inicializar.	-
1	Null.	Asignado valor Nulo.	-
2	Integer.	Entero.	-32.768 y 32.768
3	Long.	Entero largo.	-2.147.483.648 y 2.147.483.647
4	Single.	Número de simple precisión.	-
5	Double.	Número de doble precisión.	-
6	Currency.	Número.	-922.337.203.685.477,5808 y 922.337.203.685.477,5807
7	Date.	Fecha.	1-1-100 y 31-12-9999
8	String.	Cadena de caracteres.	Longitud hasta 2.000.000.000
9	Object.	Objeto.	-
10	Error.	Número de error.	-
11	Boolean.	Boleano.	True o False.

### ***Asignaciones.***

Los valores string se asignan entre comillas, los numericos sin comillas y las fechas entre almohadillas (#).

*Ejemplo: Declarar y asignar variables*

```
<HTML>
<BODY>
<%
'Saludo al mundo
Dim Saludo
Saludo = "Hola Mundo"
Response.write(Saludo)
%>
</BODY>
</HTML>
```

### ***Constantes.***

Las constantes se definen con la sentencia CONST y no cambian de valor.

### ***Matrices.***

Se tratan de forma análoga a las variables sólo que utilizan un paréntesis para indicar el elemento de la serie. Si una variable se declara con un valor n, contendá n+1 porque numera apartir del 0.

*Ejemplo: Matriz para ver los días de la semana.*

```
<HTML>
<BODY>
<%
'Dias de la semana
Dim Semana(6)
Semana(0) = "Lunes "
Semana(1) = "Martes "
Semana(2) = "Miercoles "
Semana(3) = "Jueves "
Semana(4) = "Viernes "
Semana(5) = "Sabado "
Semana(6) = "Domingo "
Response.write(Semana(0))
Response.write(Semana(1))
Response.write(Semana(2))
Response.write(Semana(3))
Response.write(Semana(4))
Response.write(Semana(5))
Response.write(Semana(6))
%>
</BODY>
</HTML>
```

Las matrices en VBScript pueden tener hasta 60 dimensiones separadas por comas. Podemos **cambiar el**

**tamaño** en tiempo de ejecución si se han declarado sin indicar el número de elementos que tiene. Para ello usaremos Redim. Si se quieren mantener los valores al redimensionarla usaremos además Preserve.

*Ejemplo: Matriz para ver los días de la semana.*

```
<HTML>
<BODY>
<%
'Dias de la semana
Dim Semana()
Redim Semana(4)
Semana(0) = "Lunes "
Semana(1) = "Martes "
Semana(2) = "Miercoles "
Semana(3) = "Jueves "
Semana(4) = "Viernes "
'Añado el fin de semana
Redim Preserve Semana(6)
Semana(5) = "Sabado "
Semana(6) = "Domingo "
Response.write(Semana(0))
Response.write(Semana(1))
Response.write(Semana(2))
Response.write(Semana(3))
Response.write(Semana(4))
Response.write(Semana(5))
Response.write(Semana(6))
%>
</BODY>
</HTML>
```

### *if...else*

Permite en función de la evaluación de una condición ejecutar un bloque de sentencias u otro.

```
If Condición Then
    bloque con sentencia/s A
[ Else
    bloque con sentencia/s B ]
End If
```

### *Parámetros*

Condición	Si se cumple, se ejecuta el bloque con sentencia/s A. En caso contrario y utilizando el "else" se ejecutará el bloque con sentencia/s B.
Bloque con sentencia/s	Bloque con una o más sentencias.

### **Código de ejemplo**



*Ejemplo: Código ASP:* Asigna a i un valor y escribe un mensaje diferente en función de que valga 10 o no. Córdelo y péguelo en su página asp.

```
<%  
  Dim i  
  i = 1  
  
  If i=10 Then  
    response.write "i vale 10<BR>"  
  Else  
    response.write "i no vale 10<BR>"  
  End if  
  
  i = 10  
  
  if i=10 Then  
    response.write "Ahora i vale 10<BR>"  
    response.write "Es lo normal<BR>"  
  End if  
>%
```

## ***Case***

En función del valor resultado de la evaluación de una expresión (habitualmente una variable) ejecutará un bloque de sentencias y/u otro.

```
Select Case ( Expresión )  
  [Case etiqueta :  
    [ bloque con sentencia/s ]]  
  [Case etiqueta :  
    [ bloque con sentencia/s ]]  
  ...  
  [Case Else :  
    [ bloque con sentencia/s ]]  
]  
End Select
```

## ***Parámetros***

Expresión	Valor comparado con las etiqueta.
Etiqueta	Si coincide con la expresión, se ejecuta el bloque con sentencia/s. Si no coincide con ninguna etiqueta y se usa el Else, se ejecutará ese bloque de sentencias.
Bloque con sentencia/s	Bloque con una o más sentencias.

## **Código de ejemplo**



*Ejemplo: Código ASP:* Reacciona ante diferentes valores de i mostrando diversos mensajes. Córtelo y péguelo en su página ASP.

```
<%  
  
  ' Declaración de variables  
  Dim fruta  
  
  fruta = "manzana"  
  
  Select Case ( fruta )  
    Case "manzana" :  
      response.write "Me gusta la "  
      response.write "manzana"  
    Case "pera" :  
      response.write "No me gusta la "  
      response.write "pera"  
    Case Else :  
      response.write "&iexcl;&iexcl;Vaya!!, a mi me gusta la "  
      response.write "manzana"  
  End Select  
  response.write "<BR>"  
  
  fruta = "melocotón"  
  
  Select Case ( fruta )  
    Case "manzana" :  
      response.write "Me gusta la "  
      response.write "manzana"  
    Case "pera" :  
      response.write "No me gusta la "  
      response.write "pera"  
    Case Else :  
      response.write "&iexcl;&iexcl;Vaya!!, a mi me gusta la "  
      response.write "manzana"  
  End Select  
  response.write "<BR>"  
  
%>
```

## Funcionamiento del código de ejemplo

Me gusta la manzana  
¡¡Vaya!!, a mi me gusta la manzana

## *Ciclo For*

Crea un ciclo con un bloque de sentencias que se repiten un número fijo de veces basándose en un valor de una variable índice.

```
For Expresión inicialización To Valor final [Step Expresión incremental]  
  [ bloque con sentencia/s ]  
[Exit For]  
  [ bloque con sentencia/s ]  
Next
```

## Parámetros

Expresión inicialización	Asignación de variable índice. Habitualmente es un contador.
Valor final	Último valor del contador, con el que se repetirá el ciclo.
Expresión incremental	Valor incremental utilizado para actualizar el contador. Si es positivo o 0, se ejecuta el bucle mientras que el contador <=Valor final, si es negativo, se ejecuta el bucle mientras que el contador >=Valor final.
Bloque con sentencia/s	Bloque con una o más sentencias.

## Código de ejemplo

*Ejemplo: Código ASP:* Cuenta hasta 10 de dos en dos. Córtelo y péguelo en su página ASP.

```
<%  
For i = 0 To 10 Step 2  
  response.write i & " borreguito&lt;BR&gt;"  
  if (i=10) then  
    response.write "Ya tengo todos los borreguitos&lt;BR&gt;"  
  end if  
Next  
%>
```

## Funcionamiento del código de ejemplo

0 borreguito  
2 borreguito  
4 borreguito  
6 borreguito  
8 borreguito  
10 borreguito  
Ya tengo todos los borreguitos

## Ciclo For...in

Iteración sobre todos los contenidos de una colección o un array. Se ejecutará el bloque de sentencias para cada contenido.

```
For Each Elemento In Grupo  
  [ bloque con sentencia/s ]  
[Exit For]  
  [ bloque con sentencia/s ]  
Next [elemento]
```

## Parámetros

Elemento	Variable utilizada para repetir los elementos de la colección o matriz. Para colecciones, elemento sólo puede ser una variable Variant, una variable genérica Object o cualquier variable de objeto de automatización específica. Para matrices, elemento sólo puede ser una variable Variant.
Grupo	Nombre de una colección de objetos o matrices.
Bloque con sentencia/s	Bloque con una o más sentencias.

## Código de ejemplo

*Ejemplo: Código JavaScript:* Devuelve el valor de las propiedades de "personas"

```

<%
  Dim Personas
  Set Personas = CreateObject("Scripting.Dictionary")
  Personas.Add "0", "Pepe "
  Personas.Add "1", "Luis"
  Personas.Add "2", "Francisco"

  For Each I in Personas
    Response.write Personas.Item(I)
    Response.write "<BR>"
  Next
%>

```

## Funcionamiento del código de ejemplo

Pepe  
Luis  
Francisco

## *Do...Loop*

Es un ciclo que se repetirá mientras se cumpla una condición o bien hasta que una condición se cumpla.

```

Do [{While | Until} Condición]
  { bloque con sentencia/s }
[Exit Do]
  { bloque con sentencia/s }
Loop

```

También se admite esta sintaxis. En este caso se repetirá al menos una vez.

```

Do
  { bloque con sentencia/s }
[Exit Do]
  { bloque con sentencia/s }

```

Loop [{While | Until} Condición]

## ***Parámetros***

Condición	Se evalúa y en el caso de <b>While</b> mientras sea verdadera se ejecutará el bloque de sentencia/s. Con <b>Until</b> se ejecutará el bloque de sentencia/s si no se cumple. Si condición es Null, condición se considera falsa.
Bloque con sentencia/s	Bloque con una o más sentencias.
Exit Do	Se usa para salir de la estructura de control. Puede usar tantos como se necesiten.

## **Código de ejemplo**

*Ejemplo: Código ASP:* Cuenta hasta 10. Córtele y péguelo en su página ASP.

```
Dim i
i = 0

Do
  response.write i & " borreguito<BR>"
  if (i=10) then
    response.write "Ya tengo todos los borreguitos<BR>"

  end if
  i=i+1
Loop while ( i <= 10)
```

## **Funcionamiento del código de ejemplo**

0 borreguito  
1 borreguito  
2 borreguito  
3 borreguito  
4 borreguito  
5 borreguito  
6 borreguito  
7 borreguito  
8 borreguito  
9 borreguito  
10 borreguito  
Ya tengo todos los borreguitos

## ***While...Wend***

Es un ciclo que se repetirá mientras se cumpla una condición.

```
While Condición
    { bloque con sentencia/s }
Wend
```

## ***Parámetros***

Condición	Se evalúa y mientras sea verdadera se ejecutará el bloque de sentencia/s. Si condición es Null, condición se considera falsa.
Bloque con sentencia/s	Bloque con una o más sentencias.

## **Código de ejemplo**

*Ejemplo: Código ASP:* Cuenta hasta 10. Córtele y péguelo en su página ASP.

```
Dim i
i = 0

While ( i <= 10)
    response.write i& " borreguito<BR>"
    if (i=10) then
        response.write "Ya tengo todos los borreguitos<BR>"
    end if
    i=i+1
Wend
```

## **Funcionamiento del código de ejemplo**

0 borreguito  
1 borreguito  
2 borreguito  
3 borreguito  
4 borreguito  
5 borreguito  
6 borreguito  
7 borreguito  
8 borreguito  
9 borreguito  
10 borreguito  
Ya tengo todos los borreguitos

## ***Funciones: Vartype.***

El tipo **Variant** se comporta **según el contexto** como un tipo o como otro. La función Vartype(Variable) nos permite distinguir entre los diferentes subtipos.

Valor de Vartype	Subtipo	Descripción	Rango
------------------	---------	-------------	-------

0	Empty.	Sin inicializar.	-
1	Null.	Asignado valor Nulo.	-
2	Integer.	Entero.	-32.768 y 32.768
3	Long.	Entero largo.	-2.147.483.648 y 2.147.483.647
4	Single.	Número de simple precisión.	-
5	Double.	Número de doble precisión.	-
6	Currency.	Número.	-922.337.203.685.477,5808 y 922.337.203.685.477,5807
7	Date.	Fecha.	1-1-100 y 31-12-9999
8	String.	Cadena de caracteres.	Longitud hasta 2.000.000.000
9	Object.	Objeto.	-
10	Error.	Número de error.	-
11	Boolean.	Boleano.	True o False.
17	Byte.	Entero.	0 y 255.

### ***Funciones: Conversiones de tipos.***

Podemos convertir de un subtipo a otro con una familia de expresiones a las que se le pasa una expresión, que es el dato o variable que se desea convertir.

<b>Función</b>	<b>Descripción</b>
Cbool	Convierte a Boolean.
Cbyte	Convierte a Byte.
Cint	Convierte a Integer.
Clng	Convierte a Long.
Csng	Convierte a Single.
Cdbl	Convierte a Double.
Ccur	Convierte a Currency.
Cdate	Convierte a Date.
Cstr	Convierte a String.

**Ejemplo:** *Convertir a entero una cadena de caracteres*

```
<HTML>
<BODY>
  <%
    'Convertir a entero una cadena de caracteres
    Dim NumeroS, Numerol
    NumeroS = "12500"
    Response.write("Ahora tengo un String " & NumeroS & "<BR>")
    Numerol = cint(NumeroS)
    Response.write("Ahora tengo un Entero " & Numerol & "<BR>")
  %>
</BODY>
</HTML>
```