

# Programando en ASP

# ¿Qué es ASP?

- **Active Server Pages** (Páginas Activas en el Servidor) es la tecnología para la programación del lado del servidor desarrollada por **Microsoft**
- Los servidores que trabajan con ASP son aquellos basados en la tecnología Windows NT
- Para escribir páginas en ASP utilizamos el lenguaje de script **Visual Basic Script**.

# Visual Basic Script

- El lenguaje Visual Basic Script tiene dos partes, una centrada en el desarrollo del lado del servidor y otra centrada en el desarrollo del lado del cliente.
- Nosotros nos centraremos sólo en la parte del lado del servidor.
- Aunque ambas derivan de Visual Basic por lo que son muy parecidas.

# Pasos previos

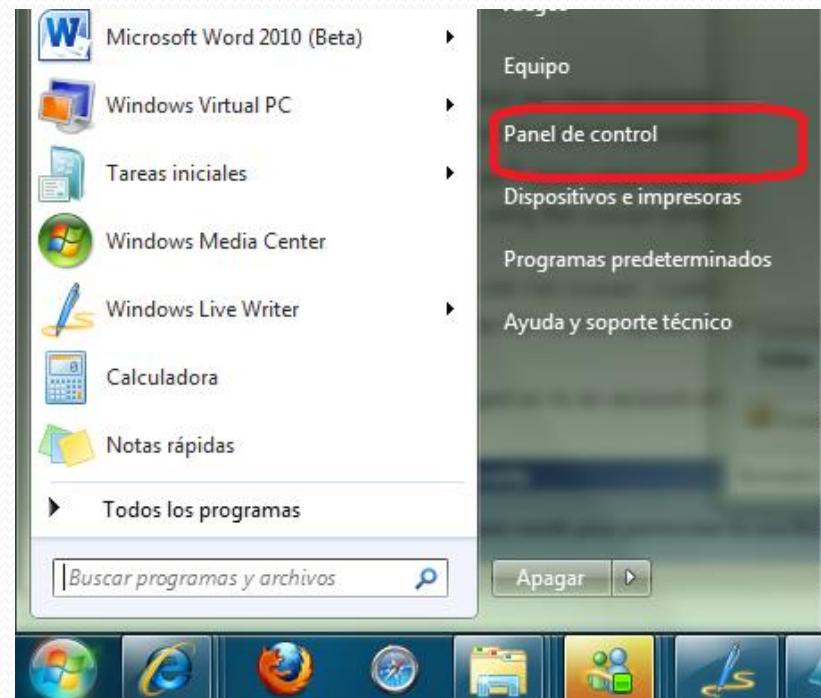
Instalación del Servidor

# ISS

- **Internet Information Server**: es el servidor Web que utilizaremos para trabajar con ASP en modo offline
- Para equipos con la tecnología W95 o W98 la versión que se deben descargar es la versión PWS.
- Para equipos con la tecnología WNT (nuestro caso) el servidor es el IIS.

# Instalación del IIS

- IIS se distribuye de forma gratuita junto con versión de Windows que utilicemos.
- Lo único que debemos hacer es activarla:
- 1. Panel de control



# Instalación del IIS

- 2. Clic en programas

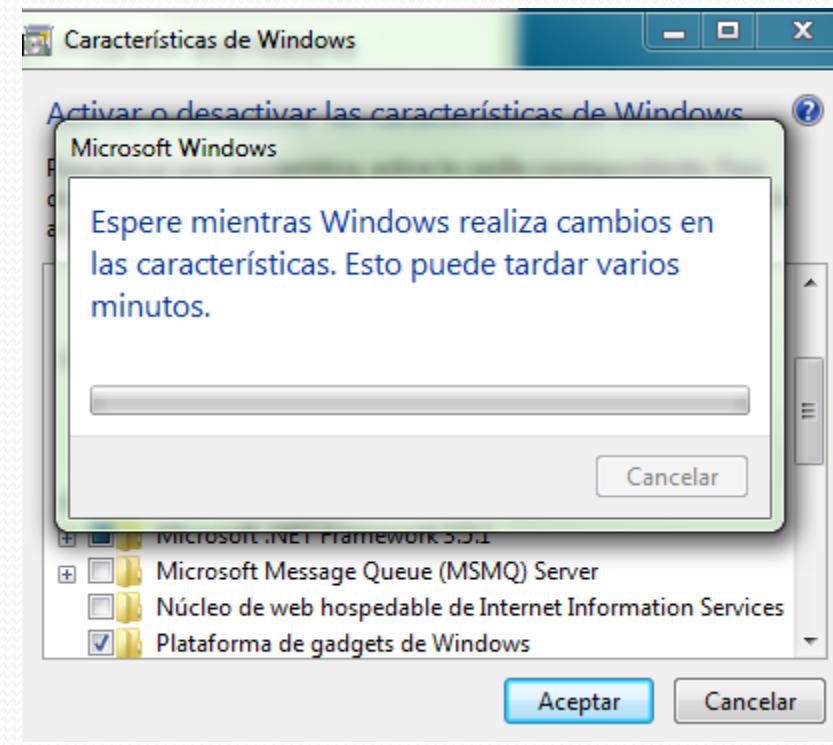
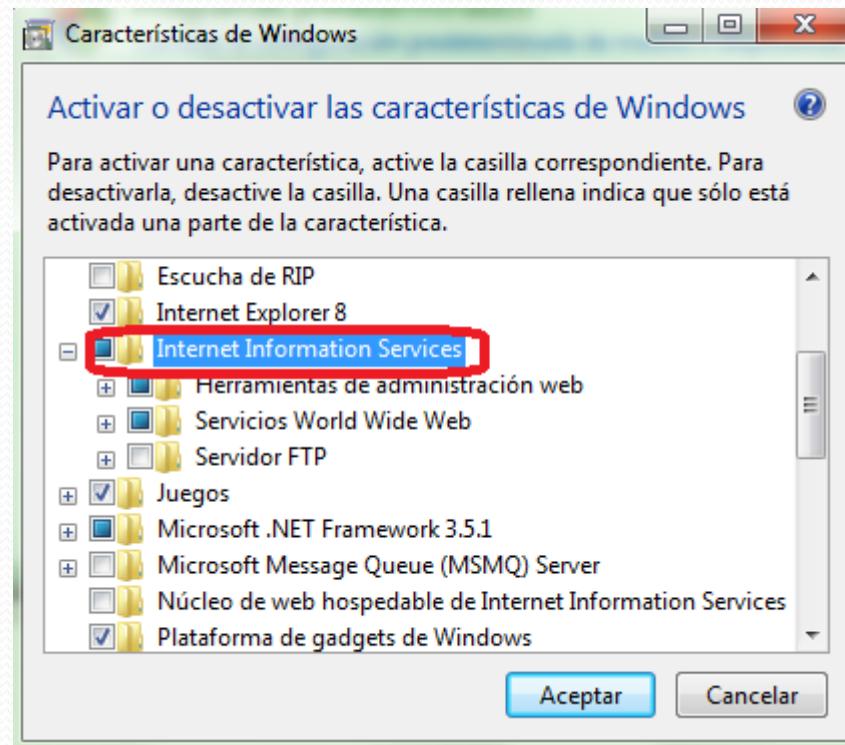


- 3. Activar o desactivar características de Windows



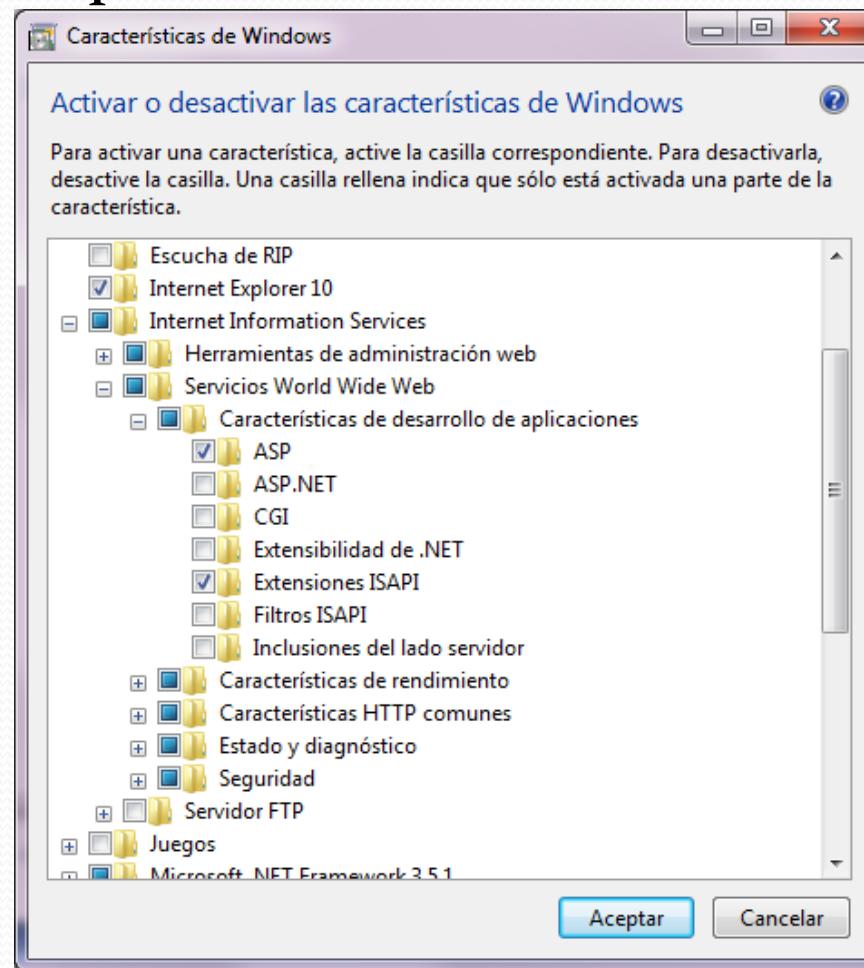
# Instalación del IIS

- 4. Activamos Internet Information Services



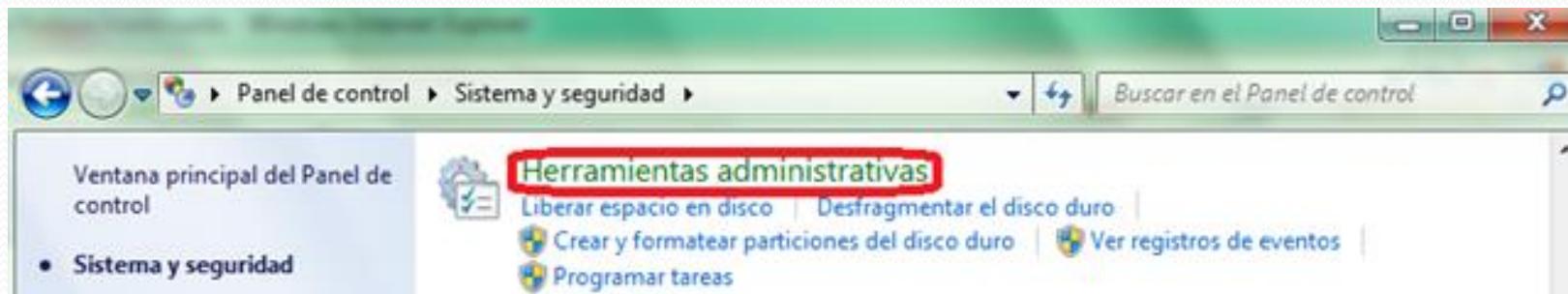
# Instalación del IIS

- Importante: ASP **no viene activo** como lenguaje de programación por lo que deberemos activarlo manualmente:



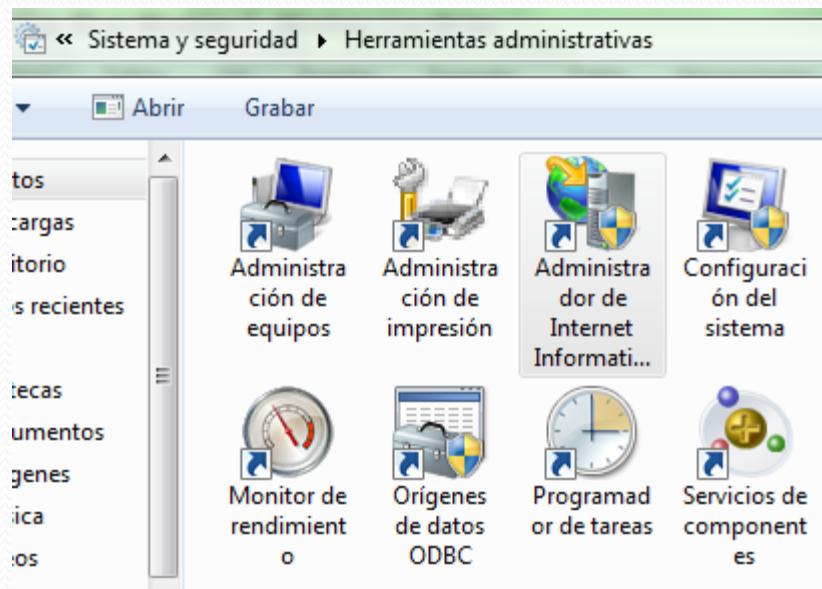
# Instalación del IIS

- En este momento ya está listo.
- Comprobamos: panel de control -> herramientas administrativas



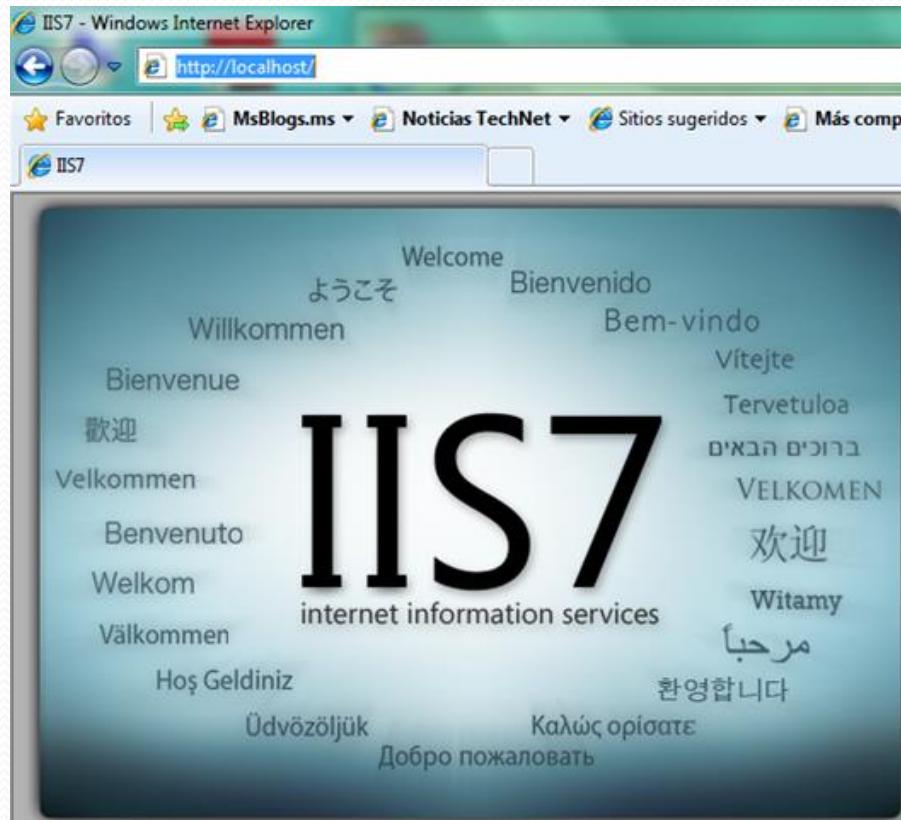
# Instalación del IIS

- Si aparece Administrador de Internet Information Services (IIS) es que ya está disponible.



# Instalación del IIS

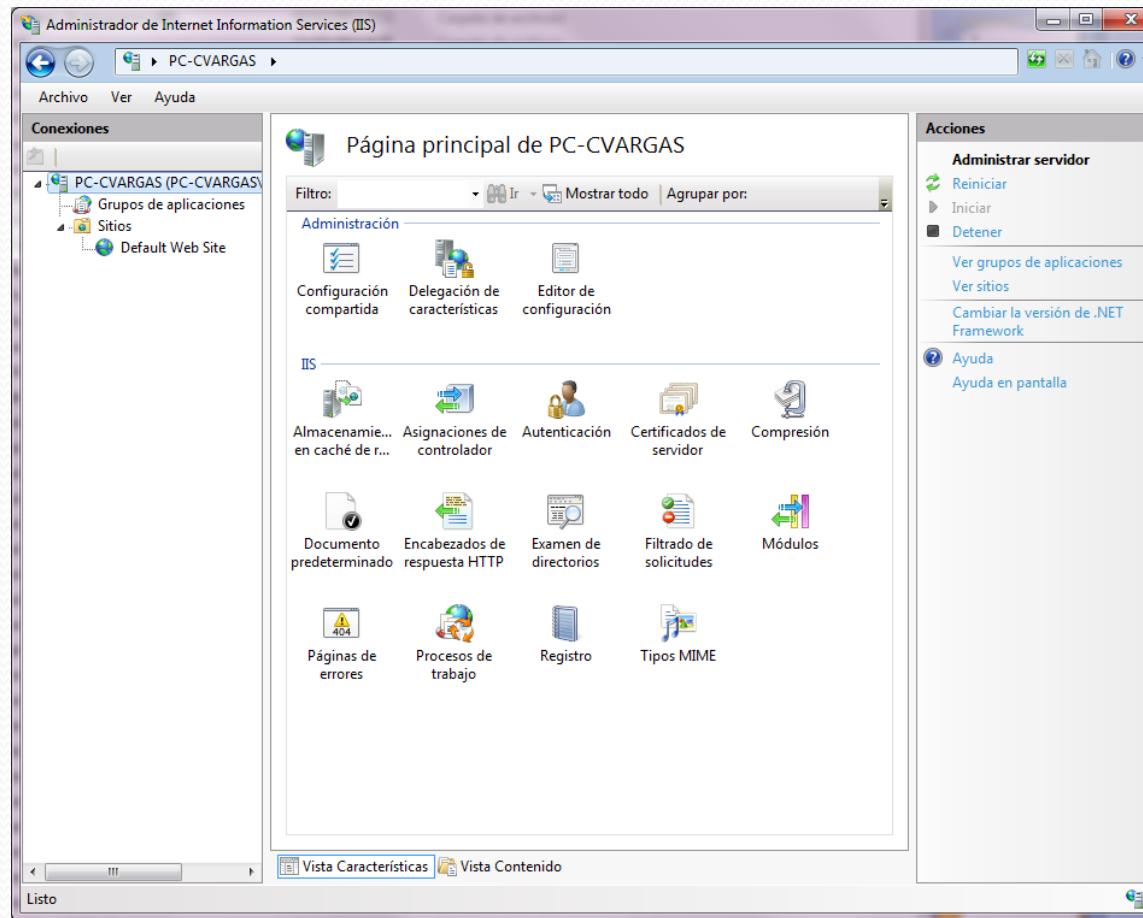
- Por ultimo para estar 100% seguros que funciona, abrimos una ventana de Internet Explorer y en la barra de direcciones escribimos **localhost** .



# Creando un sitio Web

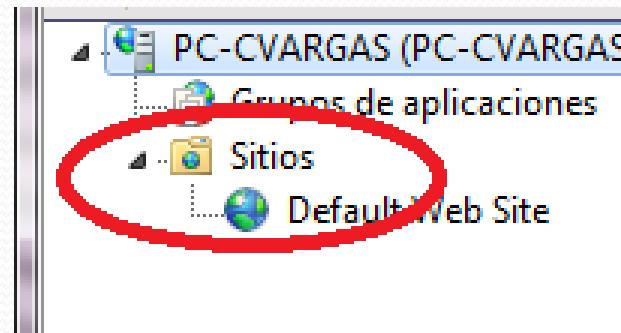
# Creando un Sitio Web

- Para crear un sitio Web ASP, deberemos iniciar el Administrador de IIS



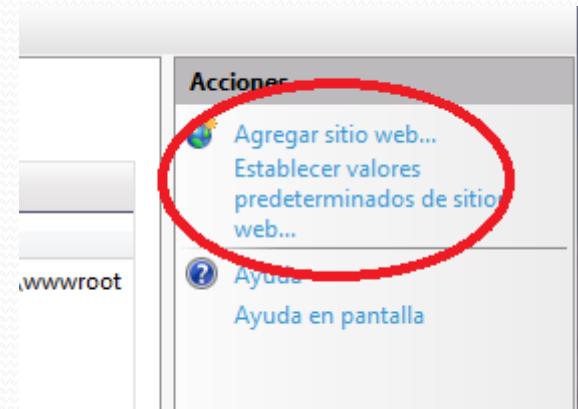
# Creando un Sitio Web

- Como se ve, existe ya un sitio por defecto que es el que hemos visto anteriormente.
- Desde este panel se pueden administrar todos los sitios que haya disponibles así como su configuración.
- Pulsaremos sobre la carpeta **Sitios**.



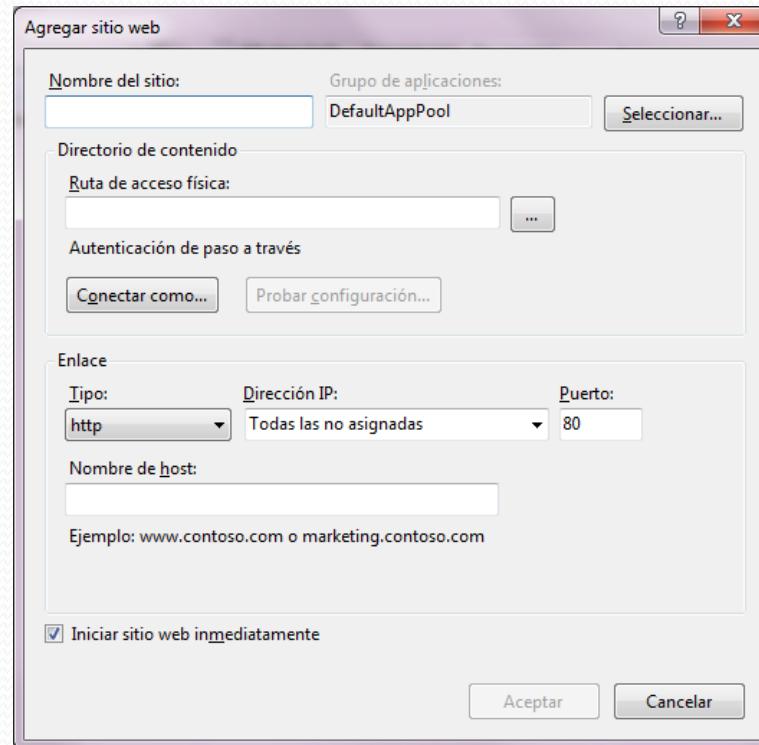
# Creando un Sitio Web

- El servidor IIS **no** exige que los archivos estén en una carpeta en concreto, por lo que podemos crear una carpeta en el lugar que queramos, donde se almacenarán los documentos del nuevo Sitio Web.
- Pulsaremos ahora en Agregar Sitio Web



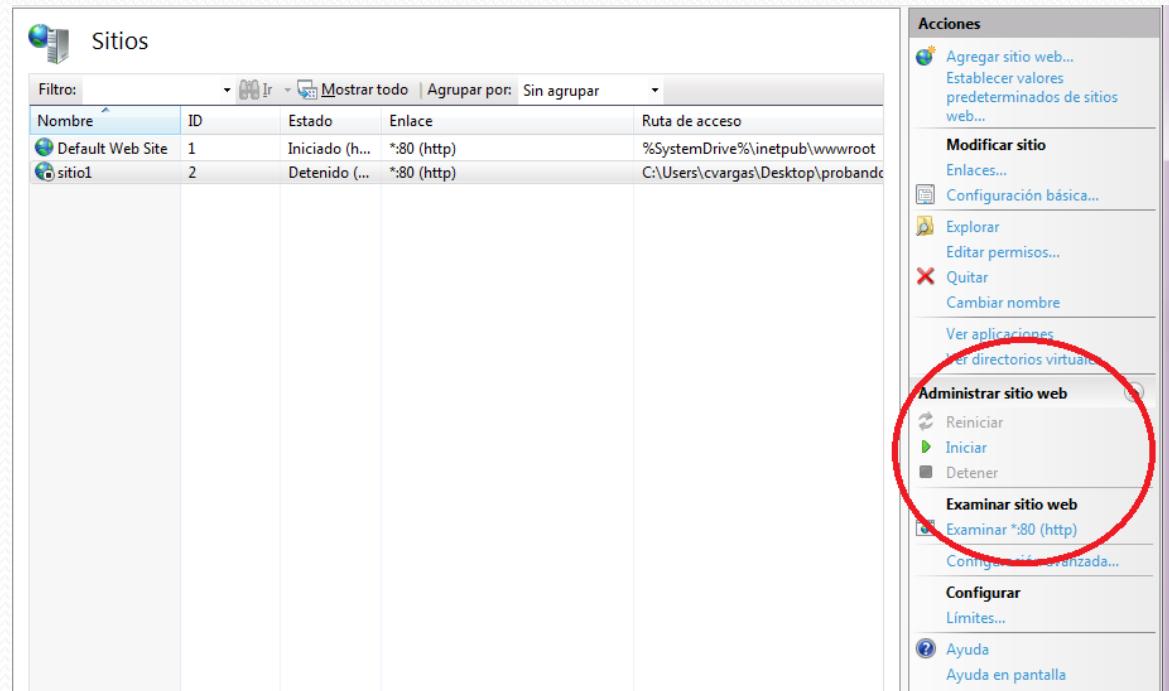
# Creando un Sitio Web

- Elegimos el nombre del Sitio y asignamos la **ruta física** (directorio donde guardaremos los archivos)
- Si tenemos otro servidor web activo, es aquí dónde podemos asignar un puerto diferente para este sitio.



# Creando un Sitio Web

- En este momento, ya tenemos dos sitios creados, pero sólo uno debe estar activo (si comparten puerto).
- Podemos **activar** o **desactivar** sitios desde la columna de la derecha.
- Ya podemos incluir archivos en el directorio asignado al Sitio Web



# Creando un Sitio Web

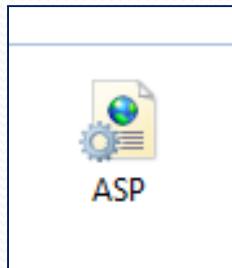
- Por último, debemos indicar al IIS que queremos ver los errores en el navegador.
- Si no hacemos esto, cada vez que ocurra un error, el sistema nos devolverá un mensaje genérico del que no podremos sacar mucha información

An error occurred on the server when processing the URL. Please contact the system administrator.

If you are the system administrator please click [here](#) to find out more about this error.

# Creando un Sitio Web

- Abrimos el IIS
- Seleccionamos el sitio Web donde queremos activar esta opción.
- Hacemos clic con el botón derecho sobre las propiedades de configuración de ASP
- Elegimos Abrir características

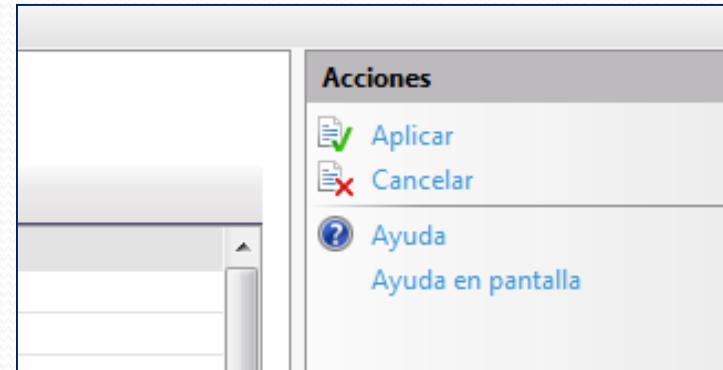


# Creando un Sitio Web

- Buscaremos la opción “Enviar errores al explorador” y la pondremos a TRUE

Compilación	
Lenguaje de script	VBScript
Propiedades de depuración	
Calcular números de línea	True
Capturar excepciones de componentes COM	True
Ejecutar en funciones de finalización de forma anónima	True
Enviar errores al explorador	True
Habilitar depuración de cliente	False
Habilitar depuración de servidor	False

- Ya sólo nos queda aplicar los cambios



# Creando un sitio Web

- A partir de aquí, IIS nos dará información más detallada sobre el error ocurrido

Error de compilación de Microsoft VBScript error '800a0401'

Se esperaba un final de instrucción

/bucles.asp, línea 111

```
response.write ("</table>");
```

-----^

# Aclaración

- Si no queremos crear un Sitio Web nuevo, el sitio que viene creado por defecto está en la ruta:
- **C:/inetpub/wwwroot**
- Introduciendo aquí los documentos (igual que en **htcdoc**) podremos ejecutarlos.



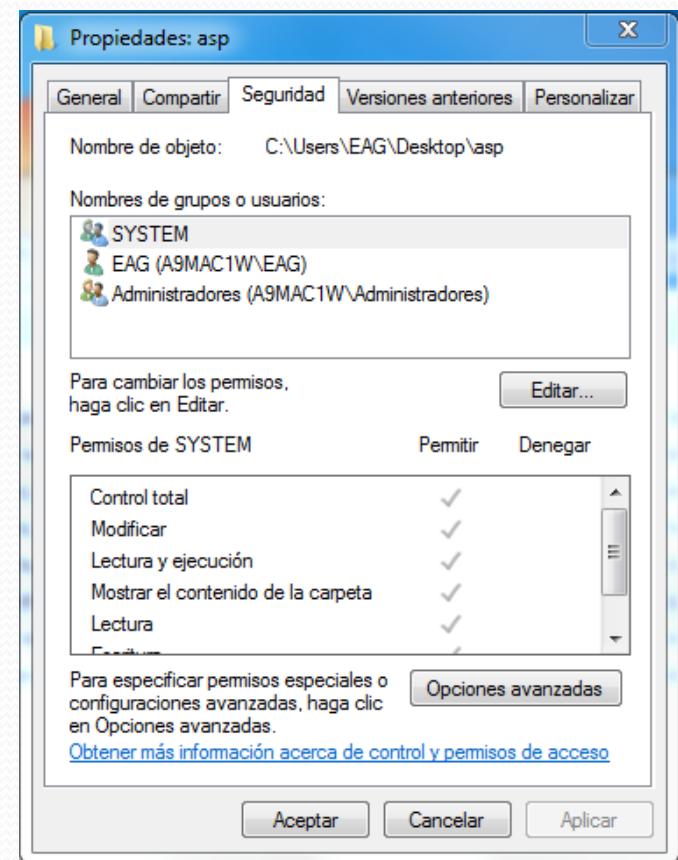
# MUY IMPORTANTE

# Dando permisos

- Es importante que el usuario con el que estemos autentificados en Windows, tenga permisos de **control total, modificar y lectura y ejecución** en la carpeta en la que estén guardados los documentos del Sitio Web.
- Para dárselosharemos lo siguiente:

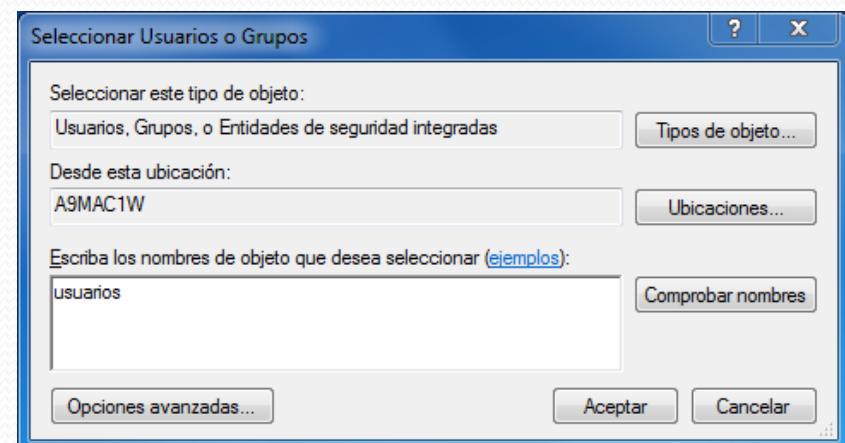
# Dando permisos

- Abrimos las opciones de la carpeta que vayamos a utilizar como directorio raíz.
- Abrimos la pestaña **Seguridad**



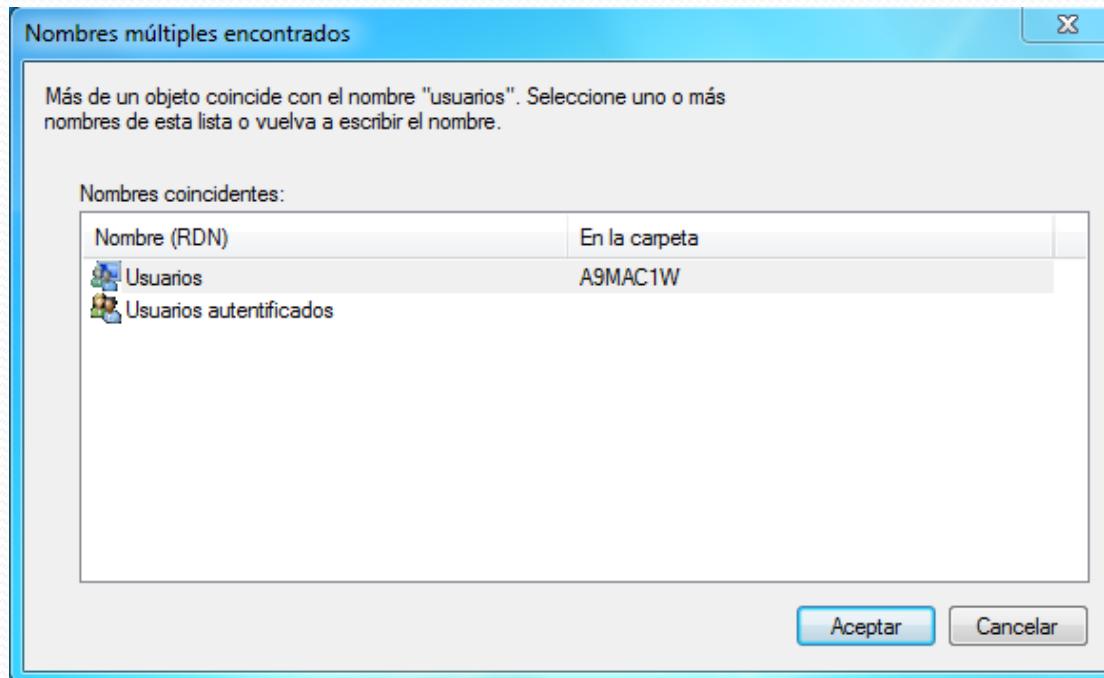
# Dando permisos

- Pulsamos el botón **Editar...** para agregar los nuevos usuarios.
- Agregaremos un nuevo usuario.
- Para asegurarnos que le damos permisos a todos los usuarios que tengan permiso de acceso a Windows debemos elegir el grupo de usuarios de Windows. Para ello, escribimos **usuarios** en el cuadro de texto y le damos a comprobar nombres



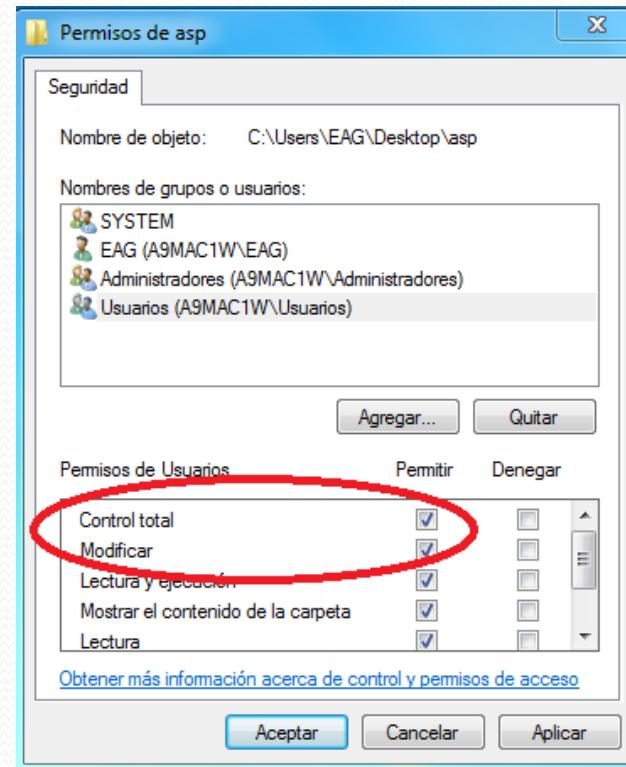
# Dando permisos

- Elegiremos **Usuarios** y aceptamos todo.



# Dando permisos

- Al usuario que hemos creado, le damos permisos de **control total** y **modificación** (el de lectura debe tenerlo por defecto)



# Dando permisos

- En este momento ya podremos visualizar el Sitio Web y todos los documentos que guardemos en este directorio

The background features a dark gray gradient from top to bottom. Overlaid on this are several thin, wavy lines in various colors: yellow, light blue, teal, purple, and pink. These lines curve and overlap each other, creating a sense of depth and motion.

ASP

# ASP

- ASP es principalmente utilizado sirviéndose del lenguaje Visual Basic Script que no es más que una versión light del Visual Basic.
- 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.
- ASP es código embedido, por lo que debe delimitarse como pasaba en PHP.

<%

...

%>

# ASP

- Para indicar si estamos programando en Visual Basic Script o en JavaScript al principio del documento deberemos incluir una de estas líneas:
- `<% @ 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
- Si no se indica nada, por **defecto** se estará usando Visual Basic Script

# ASP

- Se puede incluso programar ASP utilizando los dos lenguajes a la vez, para ello, en vez de utilizar las etiquetas <% y %>, escribiremos:

```
<SCRIPT LANGUAGE = "VBScript">  
    código Visual Basic Script  
</SCRIPT>
```

- O

```
<SCRIPT LANGUAGE = “JavaScript”>  
    código Java Script  
</SCRIPT>
```

# ASP

- Si se utilizan las etiquetas <% y %> el leguaje utilizado será el indicado al principio del documento o si no se indica nada VBScript.



Hola mundo

# Hola mundo

- Para empezar debemos indicar que Visual Basic Script
  - No diferencia entre mayúsculas y minúsculas.
  - No tiene ';' al final de cada línea, por lo que sólo puede haber una instrucción por línea
  - La función que muestra por pantalla datos es Response.write(cadena)
- El script que muestra el mensaje 'Hola Mundo' sería:

```
1  <HTML>
2  <HEAD>
3  <TITLE>Cuadro de mensaje</TITLE>
4  </HEAD>
5  <BODY>
6  <% response.write("Hola mundo")
7  %>
8  </BODY>
9  </HTML>
```

# Comentarios

- Los comentarios en VBScript se indican iniciando la línea con una ‘:

```
<%  
' esto es un comentario de una linea  
    response.write("Hola mundo")  
%>
```

- No existen comentarios de varias líneas, se deberá colocar la ‘ cada vez que se necesite.

# Operadores de concatenación

- Como cualquier otro lenguaje VBScript permite concatenar cadenas literales con el contenido de las variables.
- Si queremos concatenar un literal con una variable cuyo contenido es una cadena utilizaremos +
- Si queremos concatenar un literal con una variable cuyo contenido es numérico utilizaremos &

# Operadores de concatenación

<8

```
variableNumerica = 1
variableCadena = "hola"
response.write("Concatenando numeros "&variableNumerica)
response.write("Concatenando cadenas "+variableCadena)
```

8>

# Tipos de Datos

<b>Booleano</b>	Es un tipo de datos que contiene un si o un no. se corresponden: TRUE equivale a (-1) FALSE equivale a (0)
<b>Byte</b>	Numérico, entero sin signo hasta 65.000
<b>Currency</b>	Tipo de moneda, se utiliza para manipular de manera exacta valores monetarios, y en general cualquier cálculo que requiera una precisión de hasta 15 dígitos decimales
<b>Fecha</b>	Es un tipo de 64 bits de tamaño que almacena fechas. Se utiliza el formato americano: mes, día, hora.
<b>Double</b>	Coma flotante con doble precisión (64 bits)
<b>Entero</b>	Número entero, con signo. Desde -32.768 hasta 32.767
<b>Entero largo</b>	Este tipo es un valor entero con signo de doble precisión. Como los nuevos ordenadores trabajan con palabras de 32 bits, y no menos, se recomienda usar este tipo antes de el tipo entero normal.
<b>Objeto</b>	El subtipo de objeto es una referencia de puntero de 32 bits a una instancia de de objeto de automatización OLE. Los controles Active-X y java. Utilizan esta sintaxis: <code>Set miobjeto = new oleObjeto</code>
<b>Single</b>	Coma flotante de precisión simple
<b>Cadena</b>	Conjunto continuo de valores de caracteres, de longitud variable.

# Datos y variables

- Como pasaba con PHP, VBScript es un lenguaje débilmente tipado.
- Una variable puede ser de un tipo en un momento y de otro tipo en otro momento.
- En cada caso será del tipo del contenido que tenga.

# Datos y variables

- Como en todos los lenguajes de programación los nombres de variables:
  - Deben empezar por letra.
  - No pueden contener signos de puntuación excepto ‘\_’

# Datos y variables

- En general, el uso de variables en VBScript es el mismo que en cualquier otro lenguaje de programación:

]<>

```
a = 2  
b = 3  
c = a + b  
response.write(c)
```

]>>

# Datos y variables

- En VBS no es necesario declarar las variables, quedan declaradas en el momento de usarlas.
- Sin embargo, VBS permite la opción de forzar la declaración de variables. Así evitamos el problema de usar variables de forma incontrolada.

# Datos y variables

## OPTION EXPLICIT

- Agregando esta línea al principio de un documento, todas las variables deberán ser declaradas antes.
- La declaración de variables se hace anteponiendo al nombre de la variable la palabra **DIM**

```
<%
```

```
OPTION EXPLICIT
```

```
dim i
```

```
dim saludo
```

```
i=0
```

```
response.write("empezando el programa "&i)
```

```
saludo="Hola Mundo"
```

```
response.write("<br>utilizando una variable declarada antes "+saludo)
```

```
%>
```

# Operadores Aritméticos

- Son aquellos que soportan operaciones sencillas

+	Suma
-	Resta
*	Multiplicacion
/	División en coma flotante. Es la división normal. Devuelve un numero real si es el resultado
\	División de enteros Devuelve un numero entero, resultado de la división.
^	Potencia
Mod	Resto de la división

# Operadores de comparación

- Los que ya conocemos

=	Igual
<>	Distinto
>	Mayor
<	Menor
>=	Mayor o igual
<=	Menor o igual

# Operadores lógicos y de cadenas

- De nuevo, los que conocemos

AND	Y lógico
OR	O lógico
Xor	Xor
Not	NO lógico

- Un operador extra es el de concatenación de cadenas.

|<%

```
cadena1 = "primera cadena "
cadena2 = "segunda cadena"

concatenadas = cadena1 & cadena2

response.write(concatenadas)
```

%>

# Ejercicios

# Ejercicios

- Codificar una página en ASP que cree 3 variables de distinto tipo, les asigne valor y las muestre por pantalla.
- Codificar una página en ASP que cree tres variables y muestre la siguiente tabla

Valor 1	
Valor 2	
Valor 3	
$1 + 2$	
$2 * 3$	
$1 / 3$	
$1 + 2 + 3$	
$(2 + 3) / 1$	

# Ejercicios

- Crear una página en ASP que contenga 3 cadenas de caracteres y las muestre en una tabla ordenadas en orden alfabético.

# Arrays

# Arrays

- En VBS es necesario declarar los arrays antes de utilizarlos.
- Al declararlos hay que indicar su tamaño
- Se indica cuál será la última posición del array (empezando por 0)

```
<%  
dim edades (1)  
edades (0) = 4  
edades (1) = 25  
  
response.write(edades (0))  
response.write(edades (1))  
%>
```

# Arrays dinámicos

- Si declaramos un array como **DIM**, éste no podrá redimensionarse durante la ejecución del script.
- Si queremos poder redimensionar un array después de creado, lo declararemos como **REDIM**
- De esta forma, en el momento que lo necesitemos, podemos cambiar el tamaño del array.

```
redim edades(2)
edades(0) = 4
edades(1) = 25
edades(2) = 47
...
...
redim edades(4)
edades(3)=11
edades(4)=15
```

# Arrays dinámicos

- El funcionamiento interno de **REDIM** es el siguiente:
  - Eliminar el array antiguo
  - Crear el nuevo array con el mismo nombre
- Esto implica que se pierde el contenido del primer array creado.
- Para evitar esto, añadiremos la palabra **PRESERVE** a la hora de redimensionar

<%

```
redim edades(2)
edades(0) = 4
edades(1) = 25
edades(2) = 47
do while (i<2)
    response.write(edades(i)&"<br>")
    i=i+1
loop
redim preserve edades(3)
edades(3) = 11
i=0
do while (i<=3)
    response.write(edades(i)&"<br>")
    i=i+1
loop
```

%>

# Matrices

- Igual que se ha hecho con los arrays, las matrices hay que declararlas antes de usarlas.
- El tamaño se indica entre paréntesis, separados por coma (filas, columnas)

<%

```
dim matriz(1,2)
matriz(0,0) = 4
matriz(0,1) = 25
matriz(0,2) = 45
matriz(1,0) = "hola"
matriz(1,1) = "Cadena"
matriz(1,2) = 88
```

```
response.write(matriz(0,0)&"<br>")
response.write(matriz(0,1)&"<br>")
response.write(matriz(0,2)&"<br>")
response.write(matriz(1,0)&"<br>")
response.write(matriz(1,1)&"<br>")
response.write(matriz(1,2))
```

%>

# Función uBound

- La función **uBound()** permite conocer la posición más alta de un array
- La función **uBound** puede recibir dos parámetros:
  - Nombre del array
  - Número de dimensión de la que se quiere conocer el tamaño

```
|<%  
redim arr(4,2,8,21)  
  
response.write(uBound(arr))  
'mostrará un 4  
  
response.write(uBound(arr,2))  
'mostrará un 2  
  
response.write(uBound(arr,3))  
'mostrará un 8  
  
response.write(uBound(arr,4))  
'mostrará un 21  
%>
```

# Estructuras de control

# Estructuras de control

- VBS soporta la estructuras de control que ya conocemos
- Condicionales
  - If
  - Case
- Repetitivas
  - For
  - While --- wend
  - Do --- loop
- Veamos...

# Condicionales

# Estructuras condicionales (IF)

- La sintaxis de esta estructura es:

```
IF (expresion) then  
    Sentencias
```

....

```
ELSEIF (expresion2) then  
    Sentencias
```

....

```
ELSEIF (expresion3) then  
    Sentencias
```

....

```
END IF
```

# Estructuras condicionales(CASE)

- La sintaxis es la siguiente:

```
SELECT CASE (variable)
```

```
CASE (valor1):
```

```
                          (acción para caso valor1)
```

```
CASE (valor3):
```

```
                          (acción para caso valor2)
```

```
CASE (valor3):
```

```
                          (acción para caso valor3)
```

```
CASE ELSE:
```

```
                          (acción si no se cumple ningún anterior)
```

```
END SELECT
```

Nota: en cada case se podrían poner varios valores separados por comas

```
<%  
dim dia  
dia = 7  
SELECT CASE dia  
CASE 1:  
    response.write("El dia es LUNES")  
CASE 2:  
    response.write("El dia es MARTES")  
CASE 3:  
    response.write("El dia es MIERCOLES")  
CASE 4:  
    response.write("El dia es JUEVES")  
CASE 5:  
    response.write("El dia es VIERNES")  
CASE 6:  
    response.write("El dia es SABADO")  
CASE 7:  
    response.write("El dia es DOMINGO")  
CASE ELSE:  
    response.write("Tiene que ser un dia de la semana en número, del 1 al 7")  
END SELECT  
>%>
```

<%

dia=4

```
SELECT CASE (dia)
    CASE 1,2,3,4,5:
        response.write("Es día de semana")
    CASE 6,7:
        response.write("Es fin de semana")
    CASE ELSE:
        response.write("error en el día de la semana")
END SELECT
```

%>

# Buckles

# Bucles (FOR – NEXT)

- En este caso, la sintaxis cambia un poco con respecto al resto de lenguajes de programación

**FOR (inicio) TO (fin) STEP (paso)  
sentencias**

· · · · ·

**NEXT**

- Si no indicamos STEP, irá de uno en uno
- STEP podría ser un número negativo, si queremos ir hacia atrás

&lt;%

```
response.write("hacia adelante <br>")  
FOR i=1 TO 10  
| response.write(i & "<br>")  
next  
response.write("fin<br>")  
'-----  
response.write("de dos en dos<br>")  
for i=1 TO 10 STEP 2  
| response.write(i & "<br>")  
next  
response.write("fin<br>")  
'-----  
response.write("Hacia atras<br>")  
for i=10 TO 1 STEP -1  
| response.write(i & "<br>")  
next  
response.write("fin<br>")
```

%&gt;

# Bucles (WHILE – WEND)

- En VBS existe el bucle while, en este caso tiene la siguiente estructura:

```
WHILE(condicion)
    sentencias
    .....
WEND
```

```
<8
dim edades(2)
edades(0) = 4
edades(1) = 25
while (i<2)
    response.write(edades(i)&"<br>")
    i=i+1
wend
8>
```

# Bucles (DO – LOOP)

- El bucle DO-LOOP es un bucle muy versátil en VBS.
- Permite generar gran variedad de bucles distintos.
  - Que comprueben la condición antes de ejecutarse
  - Que hagan la primera ejecución y después se comprueben
  - Que se ejecuten mientras se cumpla la condición
  - Que se ejecuten hasta que se cumpla la condición
- Veamos su sintaxis

# Bucles (DO – LOOP)

- El bucle DO – LOOP permite:
  - Combinarse tanto con WHILE como con UNTIL
  - Indicar la condición en la parte del DO o del LOOP
- Tiene, por tanto, la siguiente sintaxis:

**DO [WHILE|UNTIL][(condicion)]  
sentencias**

.....

**LOOP [WHILE|UNTIL][(condicion)]**

# Bucles (DO – LOOP)

- Tendremos entonces 4 posibilidades:

```
<%
dim edades(2)
edades(0) = 4
edades(1) = 25
do while (i<2)
    response.write(edades(i)&"<br>")
    i=i+1
loop
%>
```

```
<%
dim edades(2)
edades(0) = 4
edades(1) = 25
do until (i<2)
    response.write(edades(i)&"<br>")
    i=i+1
loop
%>
```

```
<%
dim edades(2)
edades(0) = 4
edades(1) = 25
do
    response.write(edades(i)&"<br>")
    i=i+1
loop while (i<2)
%>
```

```
<%
dim edades(2)
edades(0) = 4
edades(1) = 25
do
    response.write(edades(i)&"<br>")
    i=i+1
loop until (i<2)
%>
```

# Manual de referencia

# Manual de consulta

- Para cualquier duda sobre ASP el mejor sitio de consulta es el sitio de **w3schools**

<http://www.w3schools.com/vbscript/>

# Programando en ASP