ebimap
Edicom Business Integrator Mapping Tool

Edicom Business Integrator Mapping tool
Anexo - Sintaxis EdiwinScript

edicom
connecting business

Índice de Contenido

EDIWINSCRIPT

1.1 VARIABLES

Las variables en EDIWINScript se declaran usando la siguiente sintaxis `@variable := contenido`; o mediante la función **`Crear`**`(@variable)` y pueden ser instanciadas mediante la misma sintáxis `@variable`.

```
@x := mundo
@y = "Hola!" & @x;
```

1.2 TIPOS DE DATOS BÁSICOS

En EDIWINScript no existen tipos de datos todas las variables se tratan como cadenas de caracteres y son las propias funciones las que disciernen el tipo de datos analizando el contenido

1.3 OPERADORES

El operador '+' esta sobrecargado y puede ser usado para la concatenación de cadenas y la suma aritmética y convertir cadenas a números.

```
// Concatenar dos cadenas
@a := 'Esto';
@b := ' y eso';
Muestra(@a + @b);  // muestra 'Esto y eso'

// Sumar dos números
@x = 2;
@y = 6;
Muestra(x + y); // muestra 8

// Sumando una cadena y un numero actúa como una concatenación
Muestra( x + '2'); // muestra 22
```

Operadores aritméticos

```
//Operadores binarios
+....Suma
-....Resta
*....Multiplicación
/....División (devuelve un valor en coma flotante)
```

Asignación

```
:=    Asignación
@x := 1;
Muestra( x );
// muestra: 1
```

Operadores de comparación

```
=....Igual
<>   Distinto
>    Mayor que
>=   Mayor o igual que
<    Menor que
<=   Menor o igual que
```

Operadores booleanos

EdiwinScript tiene dos operadores lógicos booleanos: AND/Y (AND lógico), OR/O (OR lógico), además dispone de sus funciones homologas, ademas de la funcion No para simular el operador lógico NOT.

En EdiwinScript a diferencia de JavaScript hay que tener en cuenta que las operaciones lógicas se evalúan totalmente de izquierda a derecha.

```
AND   and
Y     and
OR    or
O     or
```

Operadores con cadenas

```
:=    Asignación
+     Concatenación
```

```
&     Concatenación
//Ejemplos
@str := "ab" + "cd";    // "abcd"
@str := @str & "e";     // "abcde"
```

1.4 ESTRUCTURAS DE CONTROL

INSTRUCCIONES DE CONTROL DE FLUJO *IF/SI*

si <cond> **entonces**

sentencias;

sino

sentencias;

finsi;

if *<cond>* ***then***

sentencias;

else

sentencias;

endif*;*

Ejemplo

```
si @Cond entonces
Muestra('hola1');
Muestra('hola2');
sino
Muestra('adios1');
Muestra('adios2');
finsi;
```

INSTRUCCIÓN DE CONTROL DE FLUJO *SI/IF* (SINTAXIS MAP4)

Si(<Cond>,"Instrucciones SI","Instrucciones SINO"**)**;

Ejemplo:

```
Si(@Cond,"Muestra('hola1');Muestra('hola2')",
"Muestra('adios1');Muestra('adios2')");
```

EL BUCLE *PARA/FOR*

para @Var:=ini **incrementar|decrementar** <limite> **hacer**

sentencias;

finpara

for *@Var:=ini* ***do|downto*** *<limite>* ***do***

sentencias;

endfor

Ejemplo:

```
@res:='';
for @i:=1 to 10 do
      @res:=@res+@i+chr(13);
endfor;
Muestra(@res);
```

INSTRUCCIÓN DE CONTROL DE FLUJO *FOR* (SINTAXIS MAP4)

for(<ini>,<cond>,<inc>,"exp1;...;expl"**)**;

Ejemplo:

```
@res:='';
for(@i:=1, @i <10, @i:=@i+1, "
@res:=@res+@i+chr(13); ")
Muestra(@res);
```

EL BUCLE *MIENTRAS/WHILE*

mientras <cond> **hacer**

sentencias;

finmientras;

while *<cond>* ***do***

sentencias;

endwhile;

Ejemplo:

```
@res:='';
@i:=1;
while @i <= 10 do
@res:=@res+@i+chr(13);
@i:=@i+1;
endwhile;
Muestra(@res);
```

EL BUCLE *REPETIR/REPEAT*

repetir

sentencias;

hasta <cond>;

repeat

sentencias;

***until** <cond>;*

Ejemplo:

```
@res:='';
@i:=1;
repeat
@res:=@res+@i+chr(13);
 @i:=@i+1;
until @i >10;
Muestra(@res);
```

FUNCIONES

EDIWINScript no permite la definición de nuevas variables pero si la llamada a funciones predefinidas con la siguiente sintáxis..

nombre-funcion(arg1, arg2, arg3);

Ejemplo:

```
Muestra(@A);
muestra(@B);
```

1.5 OTRAS CONSIDERACIONES

CASE SENSITIVE

EDIWINScript es sensible a mayúsculas y minúsculas en cuanto al contenido de las variables y nombre de los registros, pero no en cuanto a los identificadores (Nombres de variable, función).

ESPACIOS EN BLANCO Y PUNTO Y COMA

Los espacios, tabuladores y saltos de línea usados fuera de constantes de tipo string son espacios en blanco. A diferencia de C, los espacios en blanco pueden causar efectos laterales en la semántica del script. Debido a una técnica llamada "inserción de punto y coma", cualquier sentencia esta bien formada si cuando aparece un salto de línea esta completa (como si apareciera un punto y coma antes del salto de línea). Se recomienda terminar las sentencias con saltos de línea para evitar la aparición de efectos laterales.

COMENTARIOS

La sintaxis de los comentarios es la misma que en C++ o Java

```
// comentario
{ comentario
   multi-línea}
```

ACCESO A REGISTROS EN EBIMAP

Para acceder al valor de campos o registro de EBIMAP se usa la siguiente notación:

```
@a := [UNH.0062]
// a tiene el valor del campo UNH.0062
```