

El Debugger ABAP

INDICE

Debugger ABAP.....	2
Que es un breakpoint.....	2
Tipos de Breakpoint	2
Ingresando en modo debugging.....	2
Avanzar por el programa ABAP Tipos de breakpoint	4
Bibliografía.....	4

Debugger ABAP.


Es el proceso de identificar y corregir errores de programación. En inglés se conoce como *debugging*, porque se asemeja a la eliminación de *bichos* (*bugs*), manera en que se conoce informalmente a los errores de programación. (definición Wikipedia)
El Debugger es una herramienta imprescindible para los programadores de cualquier lenguaje. Con él es posible probar y depurar aplicaciones que se están desarrollando, visualizar y modificar variables en tiempo de ejecución o comprobar errores que aparecen en aplicaciones que fallan.


Para poder utilizar el debugger ABAP será necesario colocar un Breakpoint en el código que queramos revisar. Un breakpoint sirve para lanzar el debugger ABAP con lo que el programa aparecerá en un nuevo modo, para así poder analizar desde la línea de código donde hemos puesto dicho breakpoint.

Que es un breakpoint / Punto de interrupción.

Como su nombre lo dice es un punto que fijamos en una línea de código en la que queramos detener la ejecución del programa, esto nos permite analizar los datos en ese momento justo antes de ejecutar esa línea de código.
Se pueden colocar tantos como se necesiten, aunque normalmente se colocan en puntos críticos donde consideremos se están produciendo errores en el programa.

Tipos de breakpoint

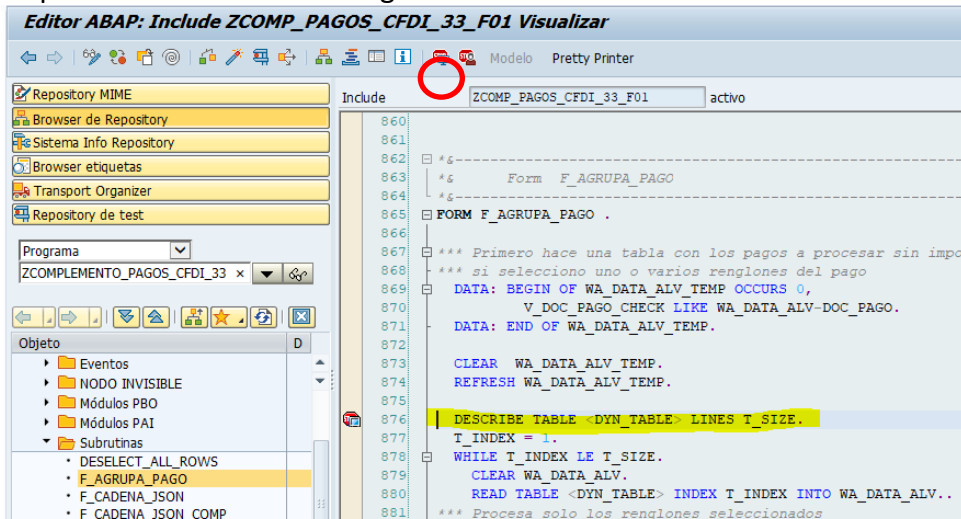
Breakpoint de sesión : Estos puntos de interrupción solo son válidos para la sesión actual, para aplicaciones que se lancen dentro de la misma entrada al sistema. Adicional estos breakpoint son independientes de usuario.

Breakpoint externos: Breakpoint de usuario : Estos puntos de interrupción son válidos para cualquier aplicación que se ejecute en el sistema, independientemente de donde se haya lanzado. Por ejemplo, en aplicaciones Web Dynpro o BSP, que se ejecutan desde un navegador web, solo podrían utilizarse este tipo de breakpoints. Estos suelen tener validez de 2 horas y se asocian a un usuario.

Ingresando en modo debugging

En ABAP hay varias formas de ingresar al modo de debugueo:
Por medio de un Break Point incluido en el código de programa. Como vemos en la imagen siguiente, se está marcado en uno de los cuadros rojos el icono del breakpoint, el cual una vez posicionado en el código, usaremos para indicar que en la línea de código (en este caso la línea 876) queremos realizar un Breakpoint, esto hará que de forma automática aparezca un icono de stop donde seleccionamos.

Se pone en una línea del código.



Editor ABAP: Include ZCOMP_PAGOS_CFDI_33_F01 Visualizar

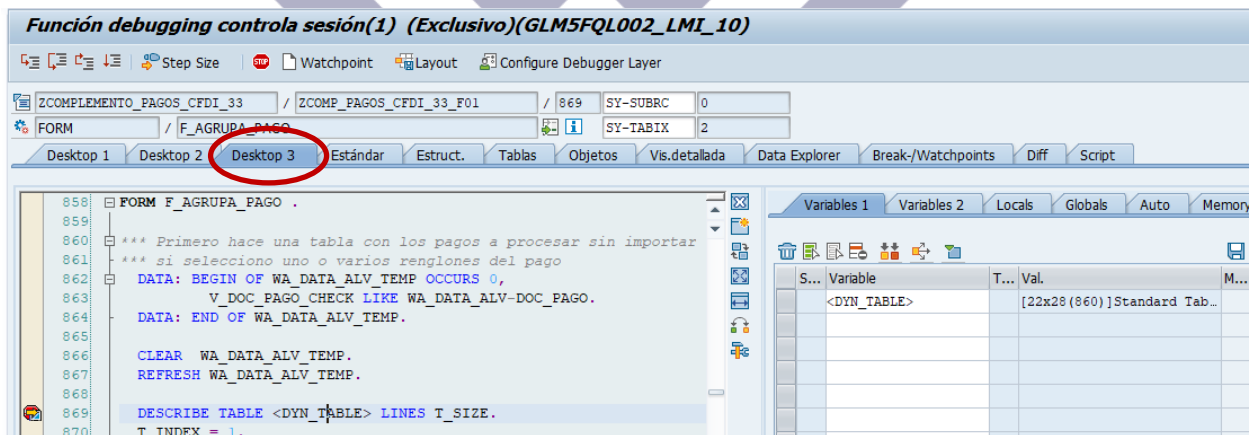
Repository MIME Include ZCOMP_PAGOS_CFDI_33_F01 activo

```

860
861
862 *$-----
863 *$ Form F_AGRUPA_PAGO
864 *$-----
865 FORM F_AGRUPA_PAGO .
866
867 *** Primero hace una tabla con los pagos a procesar sin impo
868 *** si selecciono uno o varios renglones del pago
869 DATA: BEGIN OF WA_DATA_ALV_TEMP OCCURS 0,
870         V_DOC_PAGO_CHECK LIKE WA_DATA_ALV-DOC_PAGO.
871 DATA: END OF WA_DATA_ALV_TEMP.
872
873 CLEAR WA_DATA_ALV_TEMP.
874 REFRESH WA_DATA_ALV_TEMP.
875
876 DESCRIBE TABLE <DYN_TABLE> LINES T_SIZE.
877 T_INDEX = 1.
878 WHILE T_INDEX LE T_SIZE.
879     CLEAR WA_DATA_ALV.
880     READ TABLE <DYN_TABLE> INDEX T_INDEX INTO WA_DATA_ALV..
881 *** Procesa solo los renglones seleccionados
  
```

Y al ejecutar la transacción se detendrá en la línea en la que fijamos el break.

Si nos vamos a la pestaña de “Desktop 3”, a la izquierda nos aparecerá el código de la aplicación en ejecución y a la derecha una tabla con las variables del programa. En las dos primeras pestañas podremos poner las variables que queramos: en “Locals” nos aparecerán todas las variables locales del programa y en “Globals” las globales.



Función debugging controla sesión(1) (Exclusivo)(GLM5FQL002_LMI_10)

ZCOMPLEMENTO_PAGOS_CFDI_33 / ZCOMP_PAGOS_CFDI_33_F01 / 869 SY-SUBRC 0

FORM / F_AGRUPA_PAGO / SY-TABIX 2

Desktop 1 Desktop 2 Desktop 3 Estándar Estruct. Tablas Objetos Vis.detallada Data Explorer Break-/Watchpoints Diff Script

```

858 FORM F_AGRUPA_PAGO .
859
860 *** Primero hace una tabla con los pagos a procesar sin importar
861 *** si selecciono uno o varios renglones del pago
862 DATA: BEGIN OF WA_DATA_ALV_TEMP OCCURS 0,
863         V_DOC_PAGO_CHECK LIKE WA_DATA_ALV-DOC_PAGO.
864 DATA: END OF WA_DATA_ALV_TEMP.
865
866 CLEAR WA_DATA_ALV_TEMP.
867 REFRESH WA_DATA_ALV_TEMP.
868
869 DESCRIBE TABLE <DYN_TABLE> LINES T_SIZE.
870 T_INDEX = 1.
  
```

S...	Variable	T...	Val.	M...
	<DYN_TABLE>		[22x28 (860)]Standard Tab...	

Para ver el contenido de las tablas o variables, damos doble clic sobre el objeto en este caso es la tabla <DYN_TABLE>, del lado derecho se muestra la tabla y en la columna Val, el contenido de esta, damos doble clic sobre <DYN_TABLE> y mostrará el contenido.

Función debuging controla sesión(1) (Exclusivo)(GLM5FQL002_LMI_10)

Step Size Watchpoint Layout Configure Debugger Layer

ZCOMPLEMENTO_PAGOS_CFDI_33 / ZCOMP_PAGOS_CFDI_33_F01 / 869 SY-SUBRC 0

FORM / F_AGRUPA_PAGO SY-TABIX 2

Desktop 1 Desktop 2 Desktop 3 Estándar Estruct. Tablas Objetos Vis.detallada Data Explorer Break-/Watchpoints Diff Script

Table Contents

Table <DYN_TABLE>

Attributes Standard [22x28(860)]


Insert Column Columns ...


Row	LLAVE [C(10)]	CHECK [C(1)]	ANULADA [C(30)]	BUKRS [C(4)]	KUNNR [C(10)]	NAMEI [C(30)]	DOC_PAGO [C(1... XML [C(30)]	PDF [C(30)]
1			⊗	1002	0000011741	ABDDAN RECICLA...	1400000060	
2			⊗	1002	0000011741	ABDDAN RECICLA...	1400000061	
3			⊗	1002	0000011409	ALTA FABRICACI...	1400000064	
4			⊙	1002	0000011412	CENTRALUM SA D...	1400000065	
5			⊗	1002	0000011412	CENTRALUM SA D...	1400000066	
6			⊙	1002	0000011412	CENTRALUM SA D...	1400000072	
7			⊙	1002	0000011794	PERFILES Y HER...	1400000078	
8			⊗	1002	0000011405	INDUSTRIAL DE ...	1400000083	
9			⊗	1002	0000011459	VIDRIOS EL CAS...	1400000085	
10			⊗	1002	0000030066	ALLIED TUBE &	1400000101	


Para regresarnos a la ventana anterior clic en la pestaña Desktop 3, y podemos continuar con el debug.


Avanzar por el programa ABAP.

Por último, para poder avanzar por el programa, se pueden utilizar las siguientes opciones:

Paso a paso (F5):  paso a paso Se introducirá en la línea de código que estemos, es decir, si es una subrutina o una función navegará hasta ella para poder depurarla. Si es una sentencia simple (IF, MOVE, CLEAR...) pasará a la siguiente línea de código.

Ejecutar (F6):  ejecutar Avanzará a la siguiente línea de código, independiente de que sea una subrutina, función o sentencia simple, ejecutando la subrutina o función en cuestión.

Retornar (F7):  Retornar Si se está dentro de una subrutina o función, saldrá de la misma ejecutándola y parando justo en la línea siguiente a su llamada.

Continuar (F8):  Continuar: El programa se ejecutará hasta encontrar otro breakpoint. Si no existen más, el programa se ejecutará hasta terminar o pasar el control al usuario.

BIBLIOGRAFÍA.

Wikipedia® es una marca registrada de la [Fundación Wikimedia, Inc.](https://www.wikimedia.org/), una organización sin ánimo de lucro.