1. Disparar y manejar eventos

En objetos ABAP hay ciertos métodos que se conocen como disparadores (triggers) y otros que se conocen como manejadores (handlers). Los triggers son los métodos que disparan un evento, mientras que los handlers son los métodos que se ejecutan cuando ocurre un evento

Para disparar un evento una clase tiene que:

* Declarar el evento en la parte declarativa.
* Disparar el evento en uno de sus métodos.

Los eventos se declaran en la parte declarativa de una clase o en una interface. Sentencia para declarar eventos dependientes de instancias:

**EVENTS <EVT> EXPORTING… VALUE(<EI>) TYPE TYPE [OPTIONAL]..**

Para declarar eventos estáticos se usa la siguiente sentencia:

**CLASS-EVENTS <EVT>...**

Cuando se declara un evento se puede usar la adición EXPORTING para especificar parámetros que se pasan al manejador del evento. Los parámetros se pasan por valor.

Los eventos dependientes de instancia siempre contienen el parámetro implícito SENDER, el cual tiene el tipo de una referencia al tipo o a la interface en la cual el evento es declarado.

Un evento dependiente de instancia en una clase puede ser disparado por cualquier método en la clase. Para disparar un evento en un método se usa la siguiente sentencia:

**RAISE EVENT <EVT> EXPORTING… <EI> = <F I>...**

Por cada parámetro formal que no esté definido como opcional se tiene que pasar el correspondiente parámetro real en la adición EXPORTING. La referencia a sí mismo ME es pasada automáticamente al parámetro implícito SENDER.

Es importante no confundir este concepto de eventos en la programación orientada a objetos con los eventos del entorno de ejecución ABAP, tales como son el evento, LOAD OF PROGRAM, INITIALIZATION, START OF SELECTION o END OF SELECTION debemos dejar bien en claro que son conceptos totalmente diferentes.

1. Eventos manejadores

Los eventos se usan para ejecutar una serie de métodos que tienen que:

* Estar definidos como eventos manejadores (handler) de ese evento.
* Estar registrados en tiempo de ejecución para el evento.

Para declarar un método manejador de eventos dependiente de instancia se usa la siguiente sentencia:

**METHOD <METH> FOR EVENT <EVT> OF <CIF> IMPORTING.. <EI> ..**

Para métodos estáticos se usa la misma sentencia con CLASS-METHODS en vez de METHODS. La interface de un método manejador de eventos solo puede contener parámetros formales definidos en la declaración del evento. Los atributos de los parámetros también son adoptados por el evento. El método manejador de eventos no tiene por qué usar todos los parámetros pasados en la sentencia RAISE EVENT.

1. Registro de métodos manejadores de eventos

Para permitir a un método manejador de eventos reaccionar a un evento, se tiene que determinar en tiempo de ejecución el disparador al cual va a reaccionar. Esto se hace con la siguiente sentencia:

**SET HANDLER...<HI>...[FOR]...**

Hay 4 tipos de eventos:

* **Eventos dependientes de instancia declarados en una clase**
* **Eventos dependientes de instancia declarados en una interface**
* **Eventos estáticos declarados en una clase**
* **Eventos estáticos declarados en una interface**

La sintaxis y el efecto de la sentencia **SET HANDLER** depende de cual de los cuatro casos que detallamos tenga lugar

Para un evento **dependiente de instancia** se tiene que usar la adición **FOR p**ara especificar la instancia para la cual se quiere registrar el manejador. Se puede especificar una sola instancia como disparador usando una variable referenciada:

**SET HANDLER...<HI>...FOR <REF>.**

También se puede registrar el manejador para todas las instancias que puedan dispara el evento:

**SET HANDLER...<HI>...FOR ALL INSTANCES.**

No se puede usar la adición FOR para **los eventos estáticos**:

**SET HANDLER...<HI>...**

El registro se aplica automáticamente a la clase entera o a todas las clases que implementan la interface que contiene el evento estático. En el caso de las interfaces, el registro también se aplica a las clases que aún no han sido cargadas cuando el manejador se registra.

1. Coordinación en el manejo de eventos

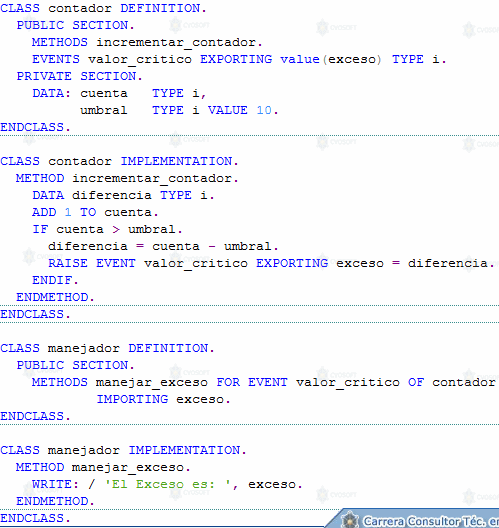
Después de la sentencia RAISE EVENT, todos los métodos manejadores registrados son ejecutados antes de que la siguiente sentencia sea procesada (manejo de eventos sincrónico). Si un método manejador de eventos desencadena eventos, los correspondientes métodos manejadores de eventos son ejecutados antes de que el método manejador original continúe.

Para evitar la posibilidad de un bucle infinito, actualmente los eventos solo se pueden anidar 64 niveles.

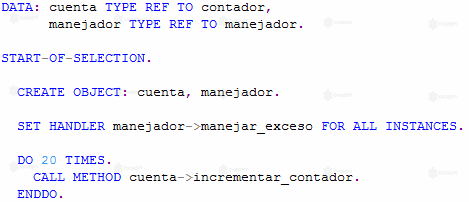
Debido a que los manejadores de eventos son registrados dinámicamente, no se puede saber el orden en el que serán procesados. Por esto se deben programar todos los manejadores de eventos como si fuesen a ejecutar todos simultáneamente.

Los eventos están sujetos a la misma estructura de visibilidad de los demás componentes de una clase. Por lo tanto pueden ser públicos, protegidos o privados. Los métodos manejadores de eventos también tiene atributos de visibilidad, la visibilidad de un evento determina quién está autorizado para controlar o manejar el evento. Los eventos que son públicos pueden ser utilizados por cualquiera, los eventos que son protegidos solo pueden ser utilizados por usuarios dentro de la clase o sus subclases y los eventos que son privados solo pueden ser utilizados dentro de la clase.

La visibilidad de un método manejador de eventos determina quien está autorizado para registrar el método, es decir, en que lugares podremos utilizar las sentencias SET HANDLER. Los métodos manejadores de eventos que son públicos pueden ser utilizados en cualquier lugar. Los métodos manejadores de eventos que son protegidos pueden ser utilizados por el usuario dentro de la clase o sus subclases y los manejadores de eventos que son privados solo pueden ser utilizados por el usuario dentro de la clase. Los métodos manejadores de eventos pueden tener la misma visibilidad o más restringida de aquellos eventos a los que se refieren.

1. Ejemplo del manejo de eventos

La clase CONTADOR implementa un contador. Se desencadena el valor\_critico cuando el valor UMBRAL es excedido y se visualiza la diferencia. La clase MANEJADOR puede manejar las excepciones en la clase CONTADOR. El manejador es registrado en tiempo de ejecución para todas las variables referenciadas que apunten al objeto.

Luego en el evento START-OF-SELECTION se declarará:

Salida por pantalla:

