LOGALI

TEORÍA

Eventos en orientación a objetos

SAP ABAP Programación Orientada a Objetos

Contenido

1. Conceptos

Además de atributos y métodos, las clases y sus instancias pueden contener un tercer tipo de componente: los eventos. Los eventos de instancia pueden ser desencadenados por parte de las instancias de la clase, pero los eventos de instancia estáticos pueden ser desencadenados por la clase en sí.

Los eventos también se pueden definir como componentes de interfaz. Si se dan las circunstancias adecuadas, los métodos de programa de control pueden reaccionar ante el desencadenamiento de este evento. Esto significa que el sistema en tiempo de ejecución puede llamar estos métodos de control después de que el evento fue desencadenado. En otras palabras, el "cliente" generalmente no llama el método de control de manera directa. Esto provoca un concepto de modelación completamente diferente. Mientras está desarrollando la clase que desencadena el evento, no necesita saber nada acerca de la clase que lo está tratando. La clase de desencadenamiento envía un mensaje específico a todas las clases y, si es necesario, a sus instancias. Al momento del desarrollo, no se conocen los tipos de programas de control ni la cantidad de programas de control que podrían usarse.

Debido a la definición de método de control, el rango de resultados posibles puede limitarse. Sin embargo, los resultados que pueden producirse solo podrán determinarse después de que el evento fue desencadenado. Un evento puede tener parámetros de exportación, lo que significa que, al contrario que una llamada de método explícita, el programa de llamada determina el protocolo.

2. Secciones de visibilidad en tratamiento de eventos

Los eventos están sujetos al concepto de visibilidad y pueden ser, por lo tanto, públicos, protegidos o privados. Los métodos de control de eventos también tienen atributos de visibilidad.

Las siguientes secciones de visibilidad determinan dónde se pueden tratar los eventos:

- PUBLIC

Se pueden acceder a ellos desde cualquier lugar.

- PROTECTED
  Solo pueden tratarse dentro de esta clase o de sus subclases.
- PRIVATE
  Sólo pueden tratarse dentro de su clase.

La visibilidad del método de control es la siguiente:

- PUBLIC
  En cualquier lugar del programa.
- PROTECTED
  Puede tratarse dentro de esta clase o de sus subclases.
- PRIVATE
  Sólo puede tratarse dentro de su clase.

3. Desencadenar y tratar eventos

Dependiendo del estado de su aplicación, es posible que no necesite programar todos los pasos cada vez. La separación de causa y efecto en su programación debería reflejarse en la manera en que construye aplicaciones complejas. A menudo, el evento ya está desencadenado y todo lo que debe hacer es crear otro programa de control de eventos.

Dentro de una clase, los eventos de instancia se definen utilizando la sentencia EVENTS, mientras que los eventos estáticos se definen utilizando la sentencia CLASS-EVENTS. Los eventos solo pueden tener parámetros de exportación que deben transmitirse por valor.

Una clase o instancia puede desencadenar un evento en tiempo de ejecución mediante la sentencia RAISE EVENT. Tanto los eventos de instancia como los eventos estáticos pueden desencadenarse en métodos de instancia. Puede desencadenar eventos estáticos en los métodos estáticos. Cuando se desencadena un evento, los métodos e control que se registran para este evento se llaman en secuencia. Estos métodos de control pueden desencadenar más eventos propios.

4. Sintaxis

Los eventos de instancia o los métodos estáticos pueden definirse dentro de una clase como eventos de tratamiento. Para ello, debe especificar el evento con la sentencia FOR EVENT y la clase o interfaz en el cual se definió el evento con la sentencia OF. Si el evento contiene parámetros de exportación y desea poder dirigirse a ellos sintácticamente, debe haber especificado los parámetros de exportación inmediatamente después de IMPORTING en la definición de método. Los parámetros de exportación de la firma del método del programa de control son los que se pueden incluir de manera explícita como parámetros de exportación del evento asociado. Los parámetros se introducen por el método de programa de control durante la definición del evento. El objeto que desencadena el evento determina el protocolo.

Además de los parámetros de exportación definidos de manera explícita, el parámetro de importación predefinido SENDER siempre puede enumerarse. Al utilizar este parámetro, puede situar una referencia al objeto de desencadenamiento de evento en el método de programa de control. Por lo tanto, los métodos de programa de control se llaman normalmente por parte de eventos desencadenados RAISE EVENT. Sin embargo, también pueden llamarse explícitamente (CALL METHOD).

5. Registrar eventos

Los eventos se registran mediante la sentencia SET HANDLER. La inscripción sólo está activa durante el tiempo de ejecución del programa. Con los eventos de instancia, FOR es seguido por la referencia al objeto que desencadena el evento.

El suplemento ACTIVATION 'X' es opcional durante el registro. Para deshacer el registro, utilice ACTIVATION ' '. Es posible registrar varios métodos con una sentencia SET HANDLER:

```
SET HANDLER: ref_handler_1->on_eventname_1 ...
             ref_handler_n->on_eventname_n FOR ...
```

Atención: Si se han registrado varios métodos en un evento, la secuencia en la que los métodos de programa de control se llaman no está definida, es

decir, no existe ninguna secuencia garantizada en la que se llamen los métodos de programa de control.

Con el suplemento ALL INSTANCES, un control de eventos puede registrarse para todas las instancias de la clase que define al evento de instancia. Éste es el único modo de registrar a objetos que aún no han sido creados.

6. Ejemplo práctico para definir y lanzar un evento

6.1. Ejemplo de la declaración de un evento dentro de una clase

```
CLASS cl_main DEFINITION.
  PUBLIC SECTION.
    EVENTS: limite_alcanzado.
ENDCLASS.
```

El levantamiento de un evento se hace a través de la palabra clave RAISE EVENT.

```
RAISE EVENT limite_alcanzado.
```

6.2. Manejo de los Eventos

Primero de todo lo que se necesita es crear un objeto receptor. Así, cuando se declara el método en la clase del receptor, se define que método es el método de control del evento, y esto se consigue con la palabra clave FOR EVENT.

```
CLASS cl_manejador_evento DEFINITION.
 PUBLIC SECTION.
   METHODS: on_limite_alcanzado
    FOR EVENT limite_alcanzado OF lcl_main.
ENDCLASS.
```

Lo que hemos definido es un método ON_LIMITE_ALCANZADO que se va encargar de tratar la acción necesaria en cuando salta el evento LIMITE_ALCANZADO de la clase CL_MAIN. Es decir es el método que se va ejecutar en cuando la clase lanza el evento con RAISE EVENT.

6.3. Configuración del manejador

La parte importante que siempre tenemos que tener en cuenta es que la el evento se va ejecutar si algún receptor quiere controlar el evento y por esto tiene que informar al sistema sobre su intención. En ABAP, puede hacerlo a través de SET HANDLER.

```
CREATE OBJECT gr_manejador_evento.
  SET HANDLER gr_manejador_evento->on_limite_alcanzado FOR gr_main.
```

Con este último paso hemos avisado al sistema que el objeto receptor CL_MANEJADOR_EVENTO va controlar el evento LIMITE_ALCANZADO de la clase CL_MAIN, que es el objeto que levanta el evento.

6.4. Ejemplo de la muestra

Veamos el siguiente ejemplo para entender cómo se desencadenan los eventos.

```
CLASS lcl_main DEFINITION.
  PUBLIC SECTION.
    DATA: lv_contador TYPE i.
    METHODS: procesar IMPORTING iv_num TYPE i.
    EVENTS: limite_alcanzado.
ENDCLASS.


CLASS lcl_main IMPLEMENTATION.
  METHOD procesar.
    lv_contador = iv_num.
    IF iv_num GE 2.
      RAISE EVENT limite_alcanzado.
    ENDIF.
  ENDMETHOD.

ENDCLASS.


CLASS lcl_manejador_evento DEFINITION.
  PUBLIC SECTION.
    METHODS: on_limite_alcanzado
      FOR EVENT limite_alcanzado OF lcl_main.
ENDCLASS.


CLASS lcl_manejador_evento IMPLEMENTATION.
```

```
  METHOD on_limite_alcanzado.
    WRITE: 'Evento Procesado'.
  ENDMETHOD.
ENDCLASS.
```

Código de ejecución:

```
  DATA: lo_main          TYPE REF TO lcl_main.
  DATA: lo_event_handler TYPE REF TO lcl_manejador_evento.


  lo_main = NEW ().
  lo_event_handler = NEW ().
  SET HANDLER lo_event_handler->on_limite_alcanzado FOR lo_main.

  lo_main->procesar( 5 ).
```