LOGALI

TEORÍA

Excepciones

SAP ABAP Programación Orientada a Objetos

Contenido

1. Excepciones basadas en clases

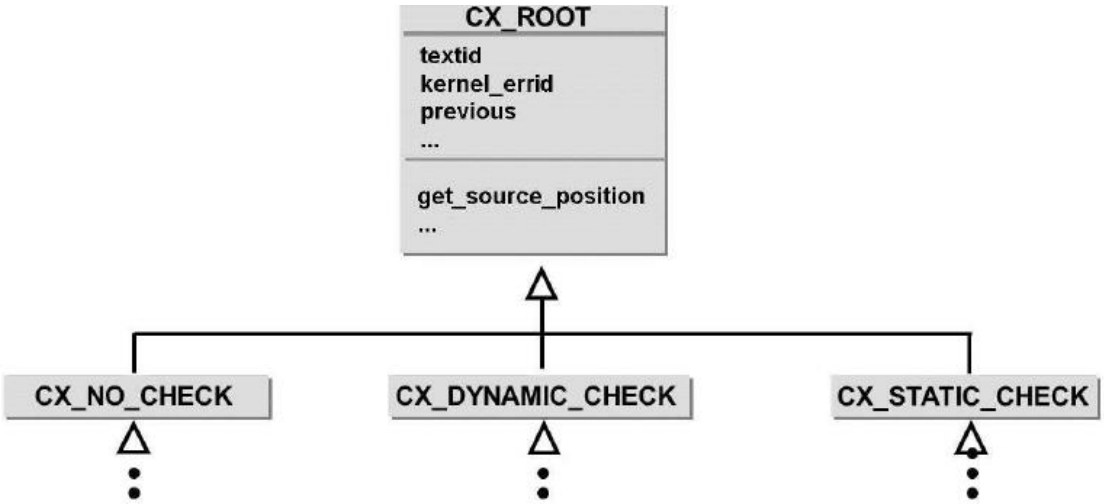
Una excepción es una situación que surge cuando un programa se ejecuta y durante la cual no es posible continuar el flujo normal del programa. SAP Web AS 6.10 presentó un nuevo concepto de excepción de objetos ABAP de manera paralela al concepto existente basado en sy-subrc. Las excepciones y el tratamiento de excepciones se basan ahora en clases. En el nuevo concepto, una excepción se representa con un objeto de excepción. Un objeto de excepción es una instancia de una clase de excepción. Los valores de atributo del objeto de excepción contienen información sobre la situación de error correspondiente. La emisión de una excepción basada en clases normalmente significa instanciar una clase de excepción y fijar los atributos. La gestión de una excepción basada en clases implica la evaluación del objeto de excepción y sus valores de atributo.

Las excepciones basadas en clases se emiten por parte de la sentencia RAISE EXCEPTION o por parte del entorno de tiempo de ejecución. Las excepciones basadas en clases se interceptan y tratan con la estructura TRY...CATCH...ENDTRY.

2. Jerarquía de las excepciones

Puede definir sus propias clases de excepción, pero el sistema ya incluye un rango de clases de excepción predefinidas, particularmente para las excepciones en el entorno del tiempo de ejecución. Generalmente, las clases de excepción se crean globalmente en el generador de clases, pero también se pueden definir clases de excepción local dentro de un programa o clase global.

Todas las clases de excepción se derivan de la clase de excepción CX_ROOT, como se puede ver en la siguiente imagen. Por lo tanto, puede acceder genéricamente a cualquier objeto de excepción mediante una variable de referencia REF TO CX_ROOT.



Sin embargo, una nueva clase de excepción no está permitida para heredarla directamente de CX_ROOT. Debe derivar cualquier clase de excepción nueva de manera directa o indirecta desde una de las subclases CX_ROOT - CX_NO_CHECK, CX_DYNAMIC_CHECK o CX_STATIC_CHECK. A través de esto, todas las clases de excepción se subdividen en tres grupos.

Según el grupo al que pertenece una excepción determinada, esta se tratará de manera diferente por la verificación de sintaxis y el entorno de tiempo de ejecución. El grupo predeterminado es CX_STATIC_CHECK, que garantiza una verificación de sintaxis y estabilidad de programa máximos. Los otros grupos solo deben utilizarse en casos especiales.

El método GET_SOURCE_POSITION devuelve el nombre del programa principal o el programa include y también el número de línea del código fuente en el que se ha producido la excepción. El método GET_TEXT devuelve un texto de excepción en forma de un string. Este método no se define directamente en CX-ROOT pero sí en la interfaz IF_MESSAGE, que se implementa con CX-ROOT.

3. La estructura de control TRY-ENDTRY

Puede tratar una excepción si la sentencia que la emitió se encuentra en una estructura de control TRY-ENDTRY. Trata la excepción mediante la sentencia CATCH en la estructura TRY-ENDTRY.

El bloque TRY contiene las sentencias para las que es necesario tratar las excepciones. Un bloque CATCH contiene el programa de control de excepciones que se ejecuta si una excepción especificada se ha producido en el bloque TRY asociado.

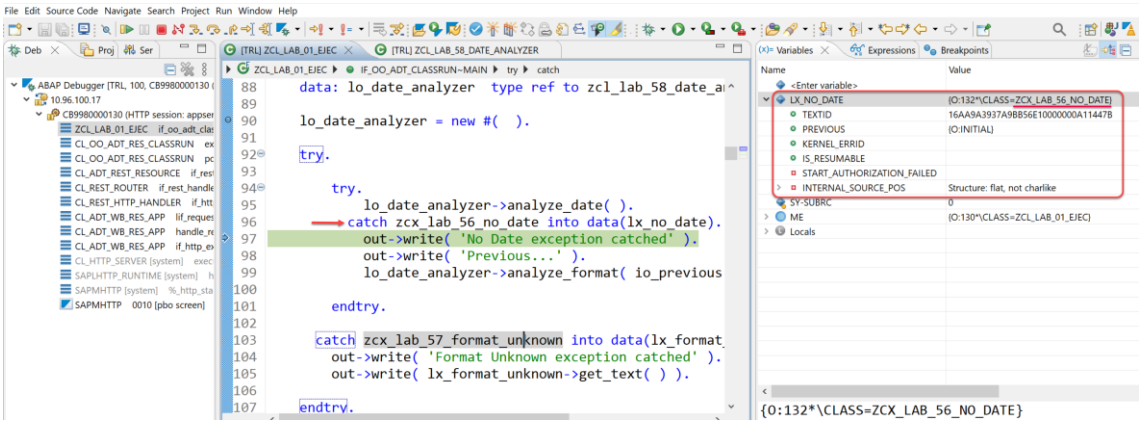
Como todas las estructuras de control en objetos ABAP, puede anidar las estructuras TRY-ENDTRY hasta cualquier profundidad. Particularmente, los bloques TRY, CATCH y CLEANUP pueden contener estructuras TRY-ENDTRY completas.

Puede especificar cualquier número de clases de excepción para la sentencia CATCH. De esta manera, se define un programa de control de excepciones para todas estas clases de excepción y sus subclases.

Si se produce una excepción, el sistema busca la sentencia CATCH coincidente en la estructura TRY-ENDTRY que rodea inmediatamente a la sentencia. Busca a través de los bloques CATCH para la clase de excepción relevante o las clases superiores de la jerarquía de herencia. Si encuentra alguna de las clases de excepción relevantes, el programa navega directamente hacia el programa de control. Si el sistema no encuentra una sentencia CATCH que coincida, busca gradualmente hacia el exterior en las estructuras TRY-ENDTY circundantes. Si no puede encontrar un programa de control dentro del mismo procedimiento, el sistema intenta propagar la excepción al programa de llamada. Este proceso se discutirá de manera más detallada posteriormente.

Si una estructura TRY-ENDTRY contiene un bloque CLEANUP, este bloque se ejecuta cuando se retira la estructura TRY-ENDTRY, porque el sistema no puede encontrar un controlador dentro de la estructura TRY-ENDTRY, pero sí en una estructura TRY-ENDTRY circundante o en un programa de llamadas.

4. Análisis de las excepciones basadas en clases en el Debugger



Si se emite una excepción, el sistema muestra el nombre de la clase de excepción en el campo *Excepción emitida* en modo debugging.

Si un bloque CATCH atrapa la excepción, se visualiza un mensaje de éxito. El puntero de la sentencia actual se mueve entonces a este bloque CATCH.

Si se produce una excepción, aparecerán dos pulsadores. Utilice los pulsadores para analizar el objeto de excepción y navegar al punto en el código fuente donde se produjo la excepción.

5. Emisión de excepciones basadas en clases

La emisión de excepciones basadas en clases se realiza mediante la sentencia RAISE EXCEPTION. Existen dos variantes de esta sentencia.

Variantes de la sentencia RAISE EXCEPTION...

RAISE EXCEPTION TYPE <exception_class> [EXPORTING ...].

Esta sentencia crea un nuevo objeto de excepción que es una instancia de la clase <exception_class>.

RAISE EXCEPTION <object_ref>.

Esta sentencia utiliza un objeto de excepción existente al cual señala <object_ref>. El objeto de excepción se creó directamente utilizando una sentencia NEW o se interceptó en una sentencia anterior CATCH... INTO ...statement.

6. Propagación de excepciones

Para propagar una excepción desde un procedimiento, se utiliza generalmente el suplemento RAISING cuando define la interfaz de procedimiento. En métodos de clases locales y subrutinas, especifique directamente el suplemento RAISING al definir el procedimiento. Por ejemplo:

METHODS meth_name ... RAISING cx_... cx_... o

FORM subr_name ... RAISING cx_... cx_....

7. Implementación de excepciones reanudables

Utilice la sentencia RESUME para reanudar un programa inmediatamente después de la sentencia que emitió la excepción en el código fuente.

Debe satisfacer los siguientes requisitos previos para usar la sentencia RESUME:

- La excepción debe interceptarse con la sentencia CATCH en el suplemento BEFORE UNWIND. Esto garantiza que el contexto de la excepción se mantendrá activo para un posible RESUME. Si el bloque CATCH salió sin sentencia RESUME, el sistema borra el contexto de la excepción después de salir del bloque CATCH.

- La excepción deberá emitirse con la variante RAISE RESUMABLE ... de la sentencia RAISE EXCEPTION. Esto prepara el bloque de tratamiento de emisiones para RESUME.

- Si la excepción se propaga, debe marcarla como reanudable en todos los niveles de jerarquía al utilizar el suplemento RAISING

RESUMABLE(...) con el nombre de la clase de excepción entre paréntesis. Esto prepara todos los métodos que propagan la excepción para un posible RESUME.

Consejo:

El programa de control de una excepción determinada verifica, en tiempo de ejecución, si dicha excepción se emitió o propagó como reanudable o no. Todos los objetos de excepción brindan el atributo de instancia pública IS_RESUMABLE, que se fija en 'X' o ' ' por el framework, de acuerdo con el modo en que se emitió y propagó la excepción. Si reanuda una excepción no reanudable, producirá un error en tiempo de ejecución (clase de excepción CX_SY_ILLEGAL_HANDLER).

8. Implementación de la nueva emisión de las excepciones

Puede emitir las excepciones basadas en clase con una de las siguientes variantes de la sentencia RAISE EXCEPTION:

- RAISE EXCEPTION TYPE <exception_class> [EXPORTING ...]

Esta sentencia crea un nuevo objeto de excepción que es una instancia de la clase <exception_class>. De manera opcional, los valores pueden proporcionarse para el constructor utilizando el suplemento EXPORTING.

- RAISE EXCEPTION <object_ref>

Esta sentencia utiliza un objeto de excepción existente, concretamente, cualquier objeto de excepción al cual señala <object_ref>. Este objeto de excepción se creó directamente con una sentencia CREATE OBJECT o, más comúnmente, se interceptó en una sentencia anterior CATCH... Sentencia INTO ... y pasa a la llamada de manera explícita.

Después de interceptar una excepción, el programa puede emitir una segunda excepción, etc. Un objeto de excepción no es válido cuando el programa sale del bloque CATCH. Por lo tanto, el programa de control en el nivel superior solo tiene acceso al último objeto de excepción. Sin embargo, puede concatenar objetos de excepción, es decir, puede dejar que un

objeto de excepción apunte a uno anterior que apunta a su anterior, y así sucesivamente. De este modo, el programa de control de una excepción puede seguir la cadena y evaluar cada objeto de excepción en la secuencia.

Todas las clases de excepción brindan el atributo de instancia pública PREVIOUS. Indica el atributo con REF TO CX_ROOT para que apunte a objetos de excepción arbitrarios. Los constructores de todas las clases de excepción tienen un parámetro de importación, PREVIOUS, del mismo tipo que puede usarse para enlazar un objeto de excepción existente a uno nuevo.