Artículo
Nancy Martínez · 22 jul, 2020 Lectura de 8 min

El arte de mapear Globals para Clases (5 de 3)

Ejemplos de Mapeos

Obviamente, si tienes un cuatro artículo en la trilogía, debes apostar por las ganancias económicas y escribir el quinto, ¡así que aquí está!

Nota: Hace muchos años, Dan Shusman me dijo que el mapeo de globals es una forma de arte. No existe una manera correcta o incorrecta de hacerlo. El modo en que interpretes los datos te llevará al tipo de mapeo que realizas. Como siempre, existe más de una forma de llegar a la respuesta final. Según vayas revisando mis ejemplos, verás que hay algunos en los que se mapean el mismo tipo de datos, pero de distintas maneras.

Al final de este artículo hay un archivo zip con la recopilación de ejemplos de mapeo que he escrito para los clientes a lo largo de los años. A continuación, destacaré un par de ellos, que resaltan algunas de las cosas a las que he hecho referencia en mis 4 artículos anteriores sobre el mapeo. En este artículo no hay la misma cantidad de detalles que los otros 4. Por eso, si algo no está claro, coméntamelo y lo explicaré con más detalle.

Especificación de Row ID:

Clase de ejemplo: Mapping.RowIdSpec.xml

Lo prometí varias veces en los artículos anteriores. Necesitas definir esto solamente cuando las expresiones del subíndice no sean campos simples. En mi ejemplo, estoy tomando el valor que está almacenado en el subíndice y lo multiplicándolo por 100, por lo tanto, cuando busque en el global necesitaré tener 1.01 pero quiero que el valor lógico sea 101. Entonces, el Subíndice 2 tiene una expresión de {ClassNumber}/100 y el RowIdSpec es {L2}*100. Siempre me equivoco cuando lo hago por primera vez. La expresión del subíndice toma el valor lógico y lo utiliza en la variable $ORDER() del global, así que se debe dividir entre 100. La especificación de RowId toma el valor del global y construye el valor lógico para multiplicarlo por 100.

También puedes hacer esto escribiendo Next Code e Invalid Condition, que realizarán las multiplicaciones y divisiones.

Expresión del subíndice Tipo "Other" / Next Code / Invalid Condition / Acceso a datos:

Clase de ejemplo: Mapping.TwoNamespacesOneGlobal.xml

¡Esta Clase es una mina de oro! Utiliza la mitad de las cosas sobre las que quiero hablar. Esta clase recorre el mismo global en 2 namespaces diferentes. No podemos utilizar el mapeo al nivel del subíndice porque ambos namespaces utilizan los mismos valores del subíndice. En vez de eso, utilizaremos una sintaxis extendida del Global Reference, ^¦namespace¦GlobalName, primero haciendo un bucle sobre el global en el namespace USER y después haciendo un bucle sobre el mismo global en el namespace SAMPLES. La IDKey para esta tabla contendrá 2 partes: Namespace y Sub1.

Expresión del Subínidice "Other": En el subíndice del nivel 1 no vamos a hacer un $ORDER() o

$PIECE(), sino que vamos a configurar {L1} para 1 de 2 valores codificados: USER o SAMPLES. Aquí no se usa ningún global, por el que el Tipo es ‘ Other ’ .

Next Code: Si estás usando una expresión del subíndice tipo ‘ Other ’ deberás proporcionar Next Code (muy bien, podrías escribir una expresión compleja para hacerlo, pero de cualquier manera debes proporcionar el código). La primera vez que Next Code llama a {L1} se establecerá para ‘ Start Value ’ , el valor predeterminado es la cadena vacía. Tu próximo código debe establecer que {L1} equivale a la cadena vacía cuando haya terminado de hacer un bucle. En este ejemplo {L1} se establecerá en “ USER ” , “ SAMPLES ” y “ ” las tres veces que se llama. El Next Code en el subíndice del nivel 2 se establecerá a “ ” para los 2 valores correctos devueltos por el subíndice del nivel 1. En este caso, el Next Code simplemente es un $$ORDER() sobre el global con la Extended Reference.

Invalid Condition: Tanto los subíndices de nivel 1 como los del nivel 2 tienen a Next Code, lo que significa que ambos necesitan de una Invalid Condition. Dado un valor para este nivel de subíndice, la condición debería devolver 1 si es un valor incorrecto. Para {L1} si el valor no es “ USER ” o “ SAMPLES ” devolverá 1. Por ejemplo

SELECT * FROM Mapping.RowIdSpec WHERE NS = “ %SYS ”

no devolverá ninguna fila, debido al Invalid Condition para {L1}

Acceso a datos: Supongamos que tienes un global con una IdKey que se basa en 2 propiedades en un global como ^glo({sub1},{Sub2}). Si proporcionas un valor para {Sub1} antes de iniciar el bucle en {Sub2} haremos una verificación para ver si hay datos en el nivel anterior mediante un $DATA(^glo({Sub1}). En este ejemplo no tenemos un global en el subíndice del nivel 1, por lo que necesitaremos que nos digan lo que debemos probar. Esta es la expresión Acceso a Datos: ^¦{L1}¦Facility. En el siguiente ejemplo también se necesita una expresión de Acceso a Datos y podría ser más fácil de comprender. Si este ejemplo te resulta complejo, mira el siguiente.

Acceso a Datos / Referencia Full Row:

Clase de Ejemplo: Mapping.TwoNamespacesOneGlobal2.xml

En esta clase se mapean los mismos datos que en la anterior. La diferencia está en el subíndice de nivel 1. En lugar de codificar los valores para el namespace, esta clase tiene un segundo global que contiene los namespace en los que queremos recorrer: ^NS("Claims",{NS}). Esto no solamente simplifica el mapeo, sino que lo hace más flexible, ya que puedes añadir un nuevo namespace simplemente configurando un mapeo global en vez de modificar el mapeo de las clases.

Acceso a Datos: En el mapeo, el ‘ Global ’ se define como ^NS, ya que este es el global que estamos recorriendo en los subíndices del nivel 1 y 2. Para el subíndice del nivel 3 queremos cambiar a ^¦{NS}¦Facility({Sub1}). Antes que podamos usar un Global diferente en Next Code necesitamos definirlo en el ‘ Acceso a Datos ’ . {L1} solamente era una restricción en ^NS global y no se utiliza en ^Facility global, así que simplemente lo adaptaremos. {L2} ahora se utiliza como la referencia para el namespace en la sintaxis del Extended Global: ^¦{L2}¦Facility. En esta clase, no se define el ‘ Next Code ’ (por lo que tampoco era necesario hacerlo en el ejemplo anterior). El compilador de la clase toma el ‘ Acceso a Datos ’ y lo utiliza para generar el $ORDER() que se necesita en este nivel

Referencia Full Row: Es similar a ‘ Invalid Condition ” , y se utiliza para hacer referencia a la fila cuando se utilizan todas las partes de la IdKey: ^¦{L2}¦Facility({L3}). Creo que podríamos continuar sin definir la ‘ Referencia Full Row ’ para esta clase, pero no está de más. Si tienes un ‘ Next Code ’ que modifique el valor lógico de un subíndice antes de recorrer el global, tendrás que definir la ‘ Referencia Full Row ’ . Por ejemplo, como mencioné anteriormente, la clase RowIdSpec podría haberse hecho usando el ‘ Next Code ’ . Si has seguido esa ruta, la ‘ Referencia Full Row ’ habría sido ^Mapping({L1},{L2}/1.00)

Acceso al subíndice Type Global:

Clase de ejemplo: Mapping.TwoGlobals.xml

En esta clase se muestran datos de dos globals diferentes: ^Member y ^Provider. En el último ejemplo comenzamos en un global y posteriormente cambiamos a un global diferente para obtener una fila. En este caso tenemos dos globals, donde ambos contienen filas. Recorreremos todas las filas del primer global y luego recorreremos las filas en el otro global**.**

Expresión del subíndice Type Global: Cuando definiste el subíndice para el primer nivel como ‘ Global ’ , el mapa debe tener la Propiedad Global establecida en"*". Apuesto a que podríamos utilizar este estilo de mapeo para crear Mapping.TwoNamespacesOneGlobal. ¡Cuando realices estos mapeos, recuerda que eres un artista!

En {L1} el tipo de acceso es ‘ Global ’ y el ‘ Next Code ’ devolverá “ Member ” , “ Provider ” y “ ” . Al definir el tipo de acceso como ‘ Global ’ , el compilador sabe que debe utilizar {L1} como el nombre del global, por lo que {L2} y {L3} son simples subíndices. {L4} también es un subíndice pero tiene un código adicional para lidiar con el hecho de que los dos globals almacenan las propiedades en ubicaciones diferentes.

Esta clase muestra una cosa más interesante en la sección correspondiente a los datos del mapeo. Si solamente estuviéramos mapeando el global ^Member, únicamente habría definido los subíndices de tres niveles y la sección de los datos habría tenido valores de 4, 5, 6, 7, 8 y 9 en el nodo. Para tratar con códigos postales que se encuentren en el nodo 9 o en el 16, añadimos el nodo base a la IdKey 4 o 10, y utilizamos +6 en el nodo para obtener un desplazamiento al código postal.

Acceso a la Variables:

Clase de ejemplo: Mapping.SpecialAccessVariable.xml

Acceso a Variables: se pueden definir en los subíndices de cualquier nivel. Puede haber más de 1 por cada nivel. En este ejemplo definimos uno que se llama {3D1}. El 3 significa que se definió en el subíndice del nivel 3, y el 1 significa que es la primera variable que se definió en este nivel. El compilador generará un nombre único para la variable y también se encargará de definirlo y eliminarlo. La variable se define una vez que hayas ejecutado el 'Next Code'. En este ejemplo quiero usar la variable en el 'Next Code' del nivel 4, del modo que la definí en el nivel 3. En este ejemplo utilizo el {3D1} para "recordar" en que punto del bucle nos encontramos, de modo que esto nos permita modificar el valor para una fecha no válida por "*" pero aún podamos regresar al mismo lugar en el bucle.

Mapa de Bits:

Clase de ejemplo: Mapping.BitMapExample.xml

Aunque los índices para mapas de bits son relativamente nuevos, esto no significa que no querrás añadirlos a tu aplicación que esta utilizando el almacenamiento Cache SQL. Puedes añadir un índice para mapas de bits a cualquier clase que tenga una simple IdKey %Integer positivo.

Tú defines el 'Type' como "bitmap" y no incluyes la IdKey como un subíndice. Lo único que debes recordar cuando definas un mapa de bits es que también necesitas una extensión para el mapa de bits. De nuevo, esto no requiere nada especial, la extensión es un índice de valores IdKey, por lo que no se necesitan subíndices y el 'Type" es "bitmapextent".

La clase tiene métodos que puedes llamar, para preservar los índices del mapa de bits según modificas los datos

mediante las Sets y Kills. Si puedes utilizar SQL u Objetos para realizar cambios, los índices se mantendrán automáticamente.

¡Así que algunos de estos ejemplos son un poco complejos! Echa un vistazo a las Clases y mira si tienen sentido para ti. Si no es así, házmelo saber: Brendan@interSystems.com , siempre estaré encantado de ayudar a un artista en apuros. 

Aquí están los ejemplos (archivo ZIP).

Estos son los anteriores artículos sobre el mapeo:

El Arte de mapear Globals para Clases (1 de 3)

El Arte de mapear Globals para Clases (2 de 3)

El Arte de mapear Globals para Clases (3 de 3)

El Arte de mapear Globals para Clases (4 de 3)

#Mapeo #Modelo de datos de objetos #SQL #Caché