Tipos de datos estructurados

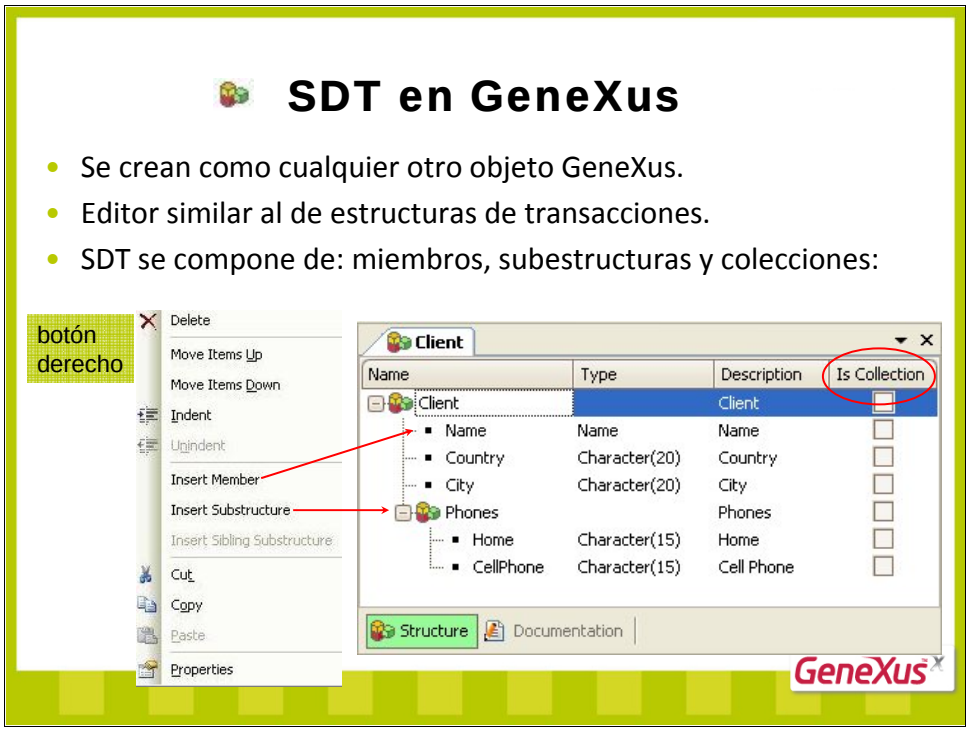
SDT 

SDT: Introducción

- Lenguajes de programación manejan:
  - Tipos de datos **simples** (Numeric, Character, etc.)
  - Tipos de datos **compuestos**.
- Ejemplo de Tipos de datos compuestos (registros o tipos de datos estructurados)

```
Type Client = Record
                Name:     Character(30)
                Country:  Character(20)
                City:     Character(20)
                Phones:   Record
                            Home:      Character(15)
                            CellPhone: Character(15)
                          end;
              end;
```

Luego se definen variables con este tipo de datos y se trabaja con ellas...



El editor de tipos de datos estructurados es similar al editor de transacciones.

Contiene:

Propiedad **Name**, con el nombre que identifica al miembro, subestructura o colección.

Propiedad **Type**, en la cual se debe seleccionar un tipo de dato simple, un dominio, o un tipo de datos estructurado que ya se haya definido en la KB (propiedad Type solo adquiere valor si se está definiendo un miembro y no una subestructura o colección).

Propiedad **Is Collection**, para indicar si el miembro representa una lista (en seguida veremos un ejemplo).

Obsérvese que una subestructura es un miembro compuesto, en lugar de ser uno simple. Es decir, es, en particular, también él, un tipo de datos estructurado.

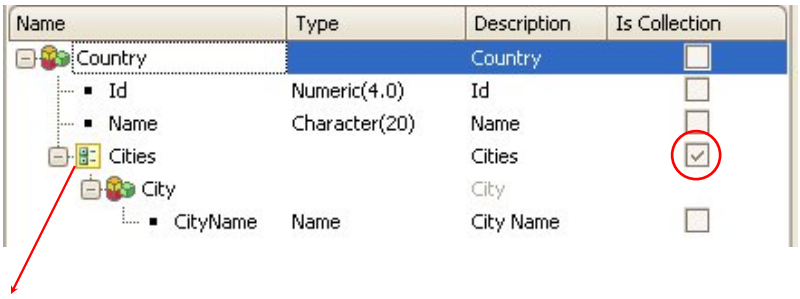
Haciendo botón derecho sobre un miembro de la estructura, se despliega la ventana que se ve a la izquierda, donde se puede insertar otro miembro, o una subestructura.

Tip: Si se desea crear un SDT con exactamente la misma estructura que la de una transacción, entonces en lugar de definir uno a uno todos los miembros, subestructuras y colecciones, alcanza con arrastrar (hacer Drag & Drop) el nombre de la transacción desde el Folder View hacia la estructura en edición del SDT.

De la misma forma, si se desea que un miembro de la estructura corresponda a un atributo, puede seleccionarse y arrastrarse el atributo desde el Work With Attributes (ventana editable desde opción View del menú de GeneXus) o insertarse con el diálogo Insert/Attribute del menú de GeneXus.

SDT en GeneXus

- Ejemplo conteniendo colección:

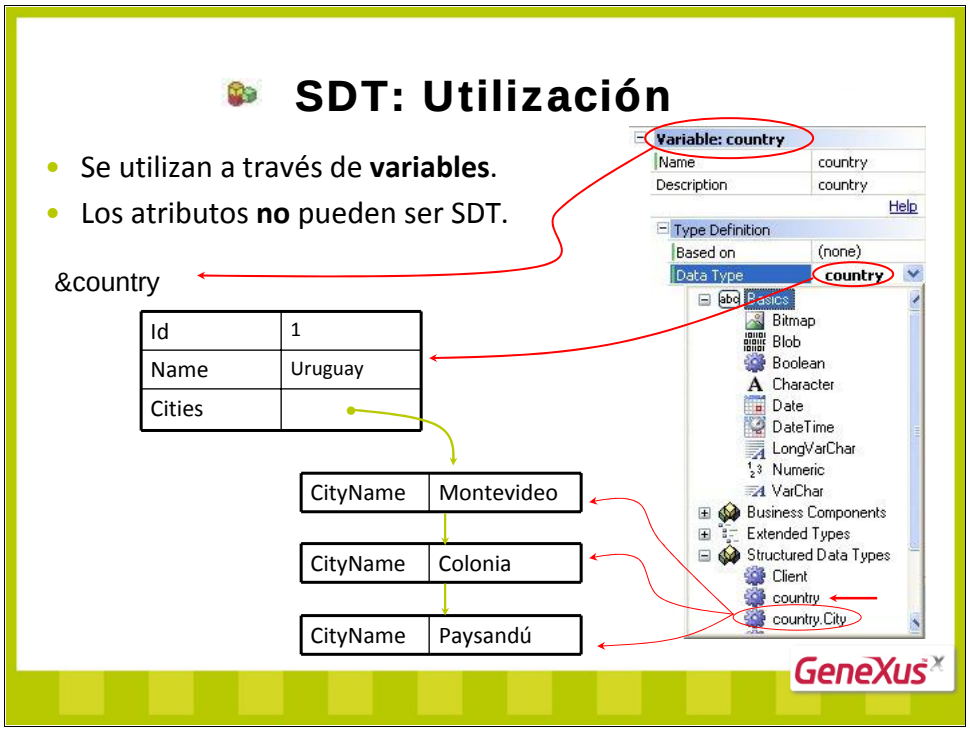


Un country tiene muchas ciudades...

Marcando el check box Is Collection se abrirá una rama de la estructura como puede verse, donde se le pedirán dos nombres: el de la colección en sí, y el de cada ítem de la misma.

Como se verá a continuación, cuando se define una colección, junto con el SDT se estará creando implícitamente otro, que corresponderá a los items de la colección.

Esto se debe a que de esta forma se podrá luego definir una variable del tipo de datos del ítem, para luego agregarla a la colección.



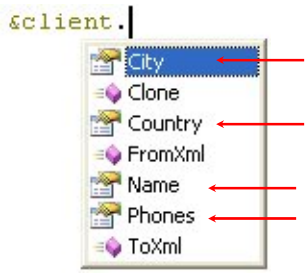
A la derecha puede verse el diálogo de propiedades de una variable &country que se está definiendo dentro de algún objeto.

El SDT Country definido en la KB tal como se aprecia en la página anterior, da lugar a la creación de dos tipos de datos estructurados: uno correspondiente al propio tipo de datos “country” y otro correspondiente a los ítems de la colección “country.City”. El por qué de este último caso se debe a que uno podría querer definir una variable solo de ese tipo de datos, para luego agregarla con el método Add que ya mencionaremos, a la colección.

Obsérvese que la variable &country se ha definido del tipo de datos “country” que aparece en la lista obtenida al hacer clic en el combo box de la propiedad “Data Type” del diálogo de definición de propiedades de la variable.

SDT: Utilización

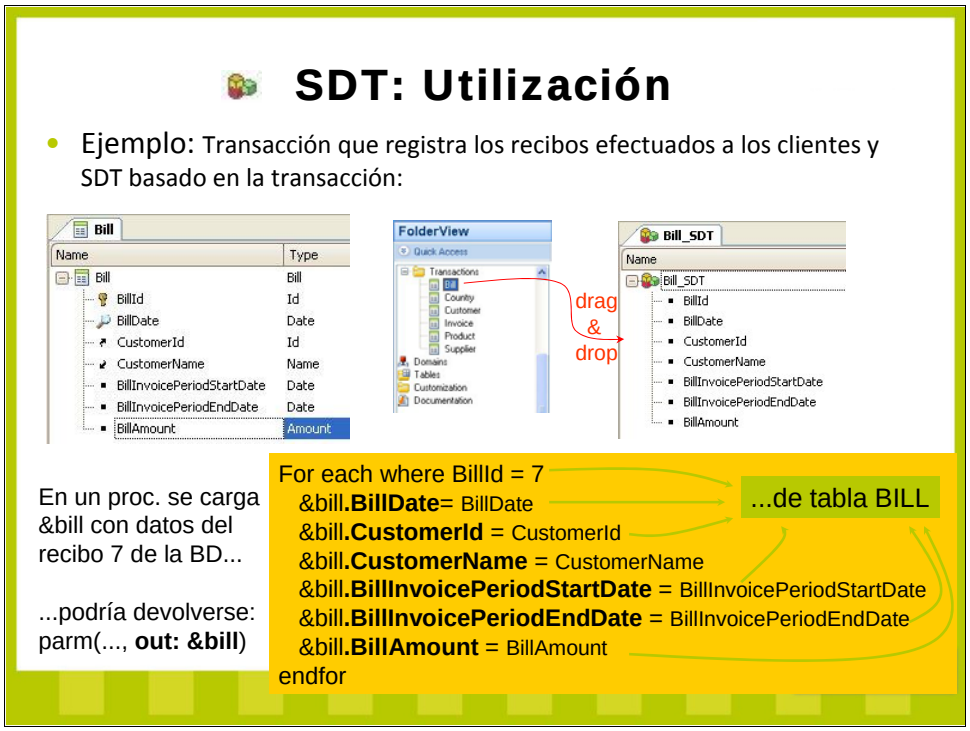
- Para datos no repetitivos, se accede a cada miembro mediante el nombre, como propiedad de la variable.



Name	Julia James	
Country	Uruguay	
City	Montevideo	
Phones	Home	555-155.55.44
	CellPhone	092-155.12.33

&client.**Name** = 'Julia James'
&client.**Country** = 'Uruguay'
&client.**City** = 'Montevideo'
&client.**Phones.Home** = ...
&client.**Phones.CellPhone** = ...

GeneXus



Aquí se presenta un ejemplo con el que continuaremos trabajando en lo que sigue. Agregamos a nuestra realidad una transacción de recibos. Supongamos que una vez al mes, se lanza un proceso de generación de recibos, en el que, tras elegir un período de facturación (usualmente todo el mes anterior) para cada cliente se sumarizan todos los montos de las facturas que se le efectuaron en dicho período, y se le genera un recibo (autonumber). La generación del recibo será automática (la realizará nuestro sistema); ese es un tema que veremos en breve.

Por ahora, supongamos que necesitamos un procedimiento que devuelva los datos de un recibo determinado de los generados automáticamente como explicamos recientemente (por ejemplo, el de id. 7). Una opción es acceder mediante un for each a la tabla BILL creada a partir de la transacción de igual nombre, y junto con la regla parm:

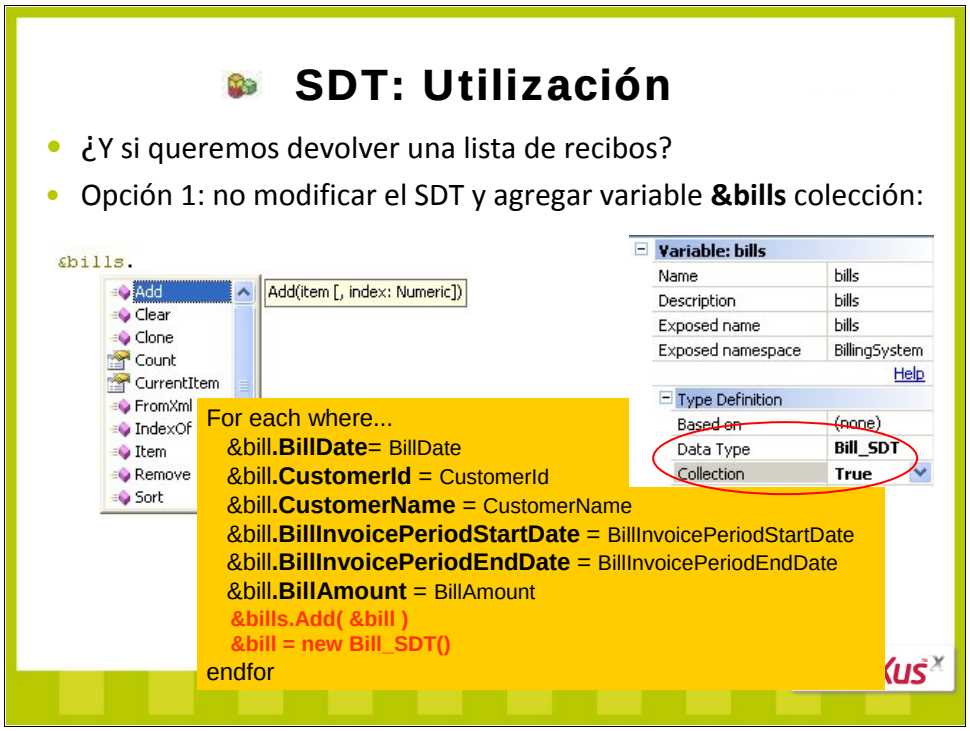
parm(out: BillDate, out: CustomerId, out: CustomerName, out: BillInvoicePeriodStartDate, out: BillInvoicePeriodEndDate, out: BillAmount);

implementar lo pedido.

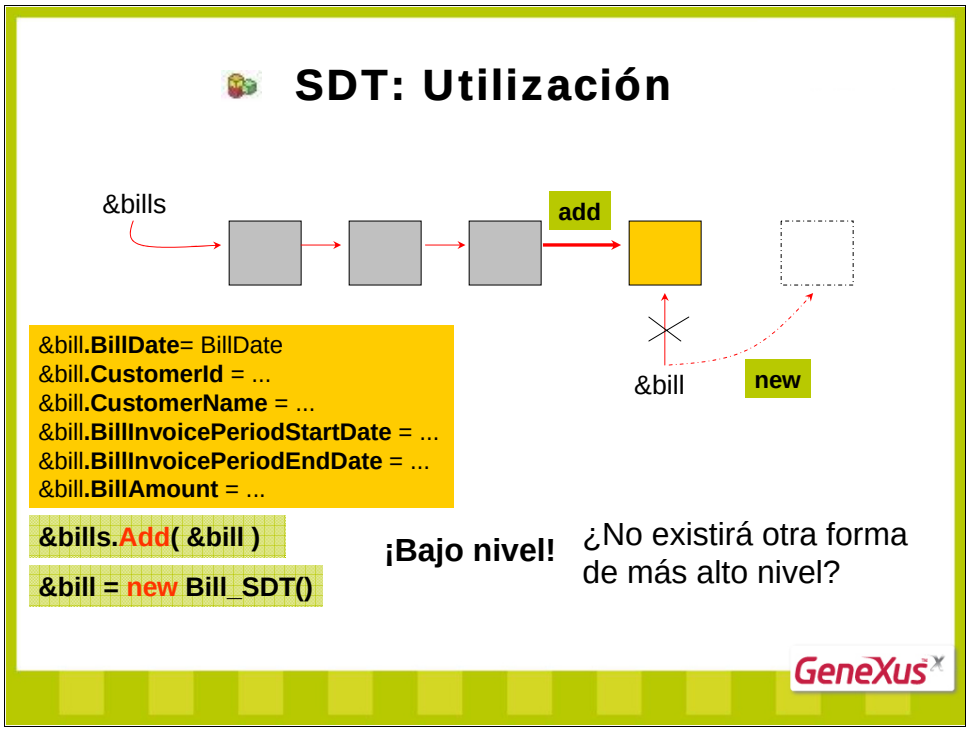
La otra opción, es devolver toda esa información en una sola variable ¡estructurada!

parm(out: &bill);

cargada como se muestra arriba. Para ello se define un SDT basado en la transacción Bill (los SDTs no pueden tener el mismo nombre que una transacción, razón por la cuál le llamamos BILL_SDT). Para no tener que ingresar uno a uno los miembros del SDT de igual nombre que los atributos de la transacción, alcanza con arrastrar la transacción Bill desde el Folder View, dentro de la estructura del SDT y automáticamente se inicializará como se muestra arriba.

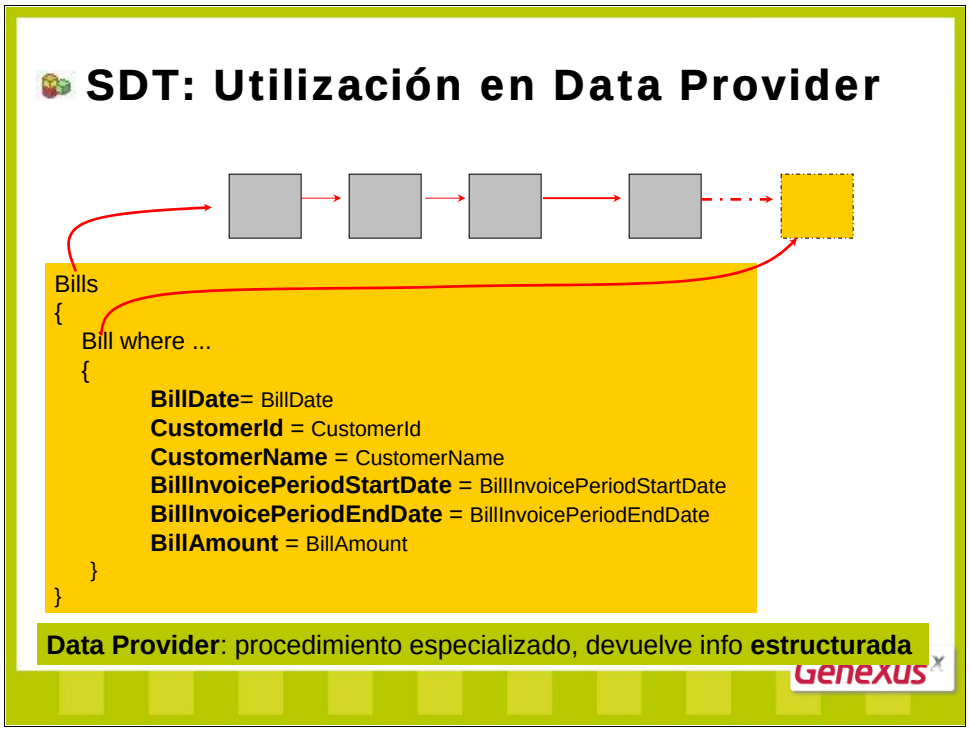


Supongamos que queremos devolver una lista de recibos (por ejemplo, los que se hayan efectuado en un rango de fechas dado).



Hay que pedir nuevo espacio de memoria para la variable &bill, para la siguiente iteración.

Como veremos en breve, existe un modo mucho más sencillo, de más alto nivel, DECLARATIVO, de obtener la colección de SDTs cargada, sin tener que preocuparnos de realizar operaciones de bajo nivel, como agregar una variable a la colección y pedir memoria...

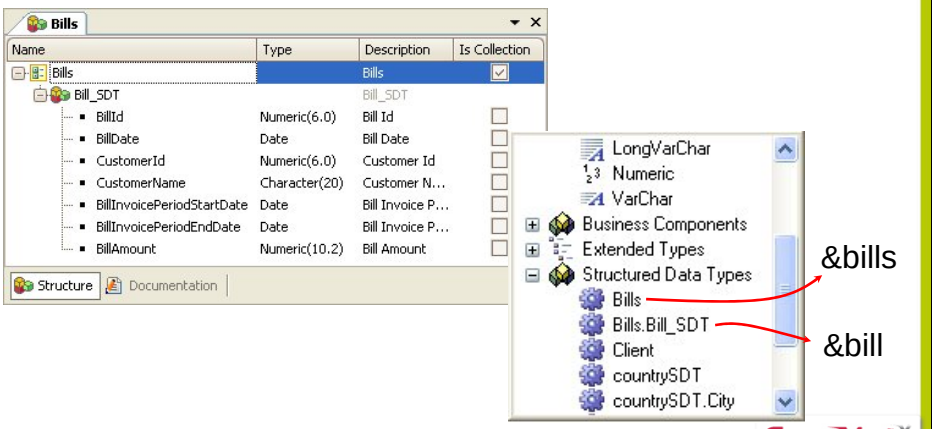


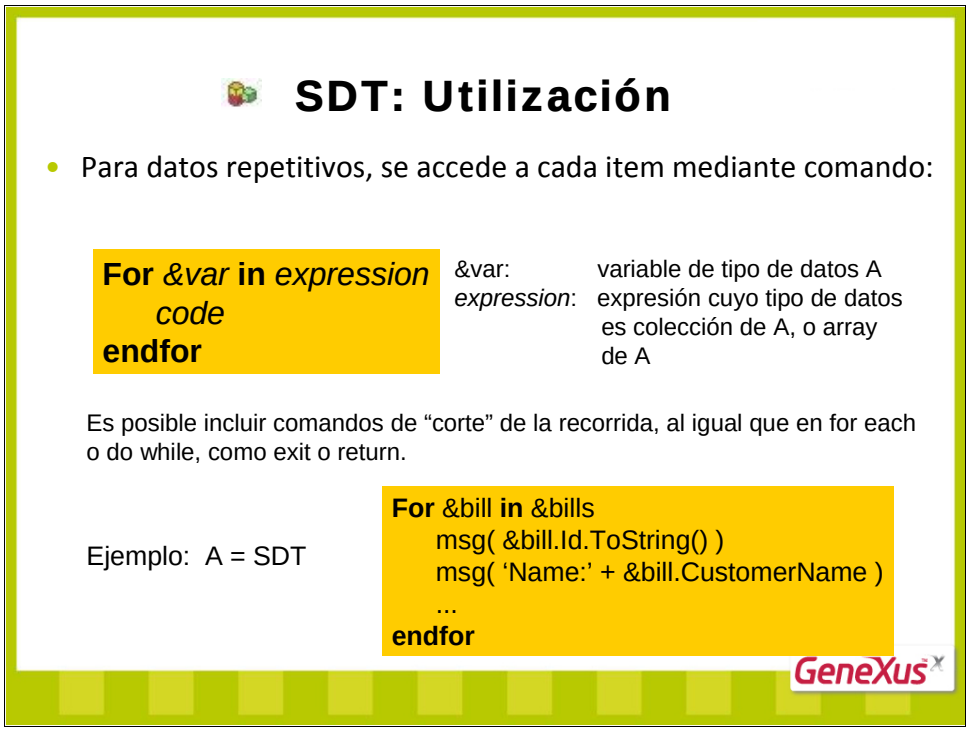
...este modo declarativo, por tanto de alto nivel, de cargar una colección de SDTs se conoce con el nombre de Data Provider. Podemos pensarlo como un procedimiento especializado, que devolverá siempre información estructurada (ya sea simple como colección).

Aquí presentamos el ejemplo, que luego ampliaremos cuando entremos de lleno en este tema.

SDT: Utilización

- Opción 2: modificar el SDT para que sea colección y luego trabajar de la misma forma, definiendo las variables:





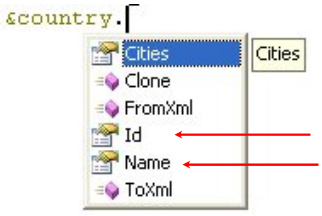
La variable *&var* va tomando los valores de cada posición de la lista.

No es posible obtener la posición del ítem durante la recorrida, para esto es necesario definir un variable que actúe como contador.

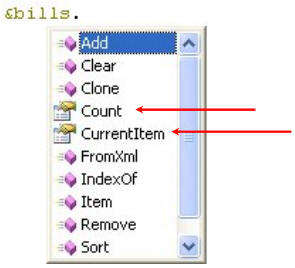
Como puede fácilmente inferirse, este comando es válido para colecciones de cualquier tipo de datos, no solo SDTs.

SDT: propiedades

Los nombres de los miembros de una variable SDT se muestran como propiedades:



Además, para variables collection:



&cVar.**Count**	Retorna la cantidad de items de la colección.
&cVar.**CurrentItem**	Cuando una variable SDT collection se despliega en un form (como grid) permite seleccionar el ítem que se ha seleccionado con el mouse como línea del grid.

GeneXus

Las propiedades Count y CurrentItem solo están disponibles para variables SDT Collection.

SDT: operadores y métodos

&var = **new** *SDT*()	Retorna una nueva referencia o puntero al SDT especificado

&cVar.**Add(** *&item* [**,** *Position*]**)**	Agrega ítem a colección en la posición relativa especificada. Si se omite se agrega al final. Position comienza en 1.
&cVar.**Clear()**	Elimina todos los ítems de la colección
&var.**Clone()**	Crea una nueva área de memoria y copia los datos de la variable en esta: *&var1 = &var2.Clone()*
&var.**FromXML(** *&xml* **)**	Carga variable SDT a partir de string conteniendo una estructura xml.
&cVar.**Item(** *Position* **)**	Retorna referencia al elemento con posición relativa *Position* en la collection.
&cVar.**Remove(** *Position* **)**	Elimina ítem que se encuentre en la posición especificada y corre un lugar todas las posiciones.
&cVar.**Sort(** *memberName* **)**	Ordena los elementos de la colección de acuerdo a miembro.
&var.**ToXml()**	Retorna un string con el formato XML de los datos de la variable SDT (que puede ser collection): *ml = &var.ToXml()*

GeneXus

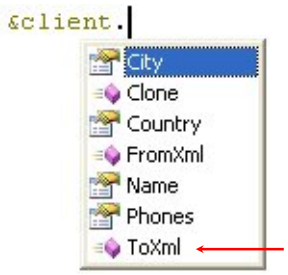
Aquí se presentan la mayoría de los métodos con los que cuentan los tipos de datos estructurados.

Algunos aplican a variables SDT no colección, se representan con *&var*, otros a colecciones, se representan con *&cVar*.

Para la lista completa, así como ejemplos, acceder al wiki o al help de la versión.

SDT: Ejemplo

- Método ToXml:



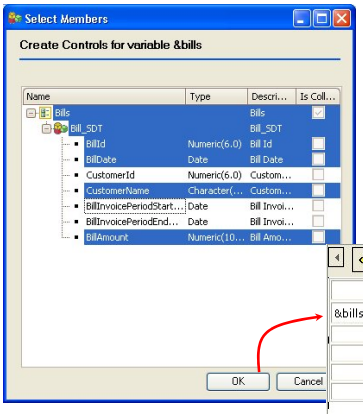
&xml = &client.ToXml()

Name	Julia James	
Country	Uruguay	
City	Montevideo	
Phones	Home	555-155.55.44
	CellPhone	092-155.12.33

```
<Client>
   <Name> Julia James </Name>
   <Country> Uruguay </Country>
   <City> Montevideo </City>
   <Phones>
            <Home> 555-155.55.44 </Home>
            <CellPhone> 092-155.12.33 </Home>
   </Phones>
</Client>
```

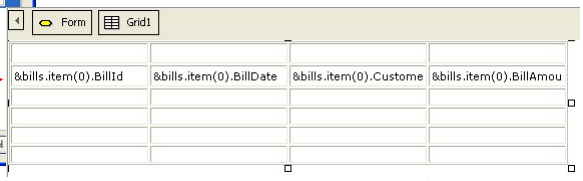
SDT collection en FORM

- Si se inserta en un form una variable SDT collection:



aparece ventana para seleccionar los miembros a insertar

que se presentarán como grid que será cargado automáticamente :



Puede seleccionarse del SDT los miembros que quieren cargarse como columnas del grid. Obsérvese que en nuestro caso hemos omitido los miembros CustomerId, BillInvoicePeriodStartDate y BillInvoicePeriodEndDate.