Subroutines

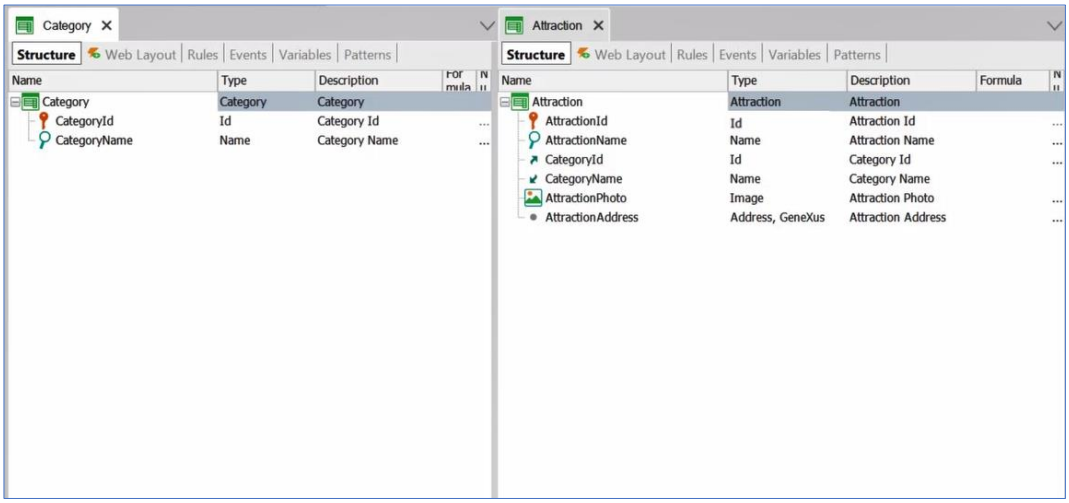
GeneXus

Veremos en este video, que son las subrutinas en GeneXus. Veremos su funcionamiento e implementación y analizaremos para que nos pueden resultar útiles.

Una subrutina es básicamente un bloque de código, el cual podremos invocar cuantas veces queramos, siempre que sea dentro del mismo objeto.
De esta forma, podremos ejecutar ese mismo bloque desde varios lugares del objeto, o desde un mismo lugar, pero en varias ocasiones. La subrutina nos permite escribir ese código una sola vez, asignarle un nombre, y luego simplemente invocarla por el nombre dado.

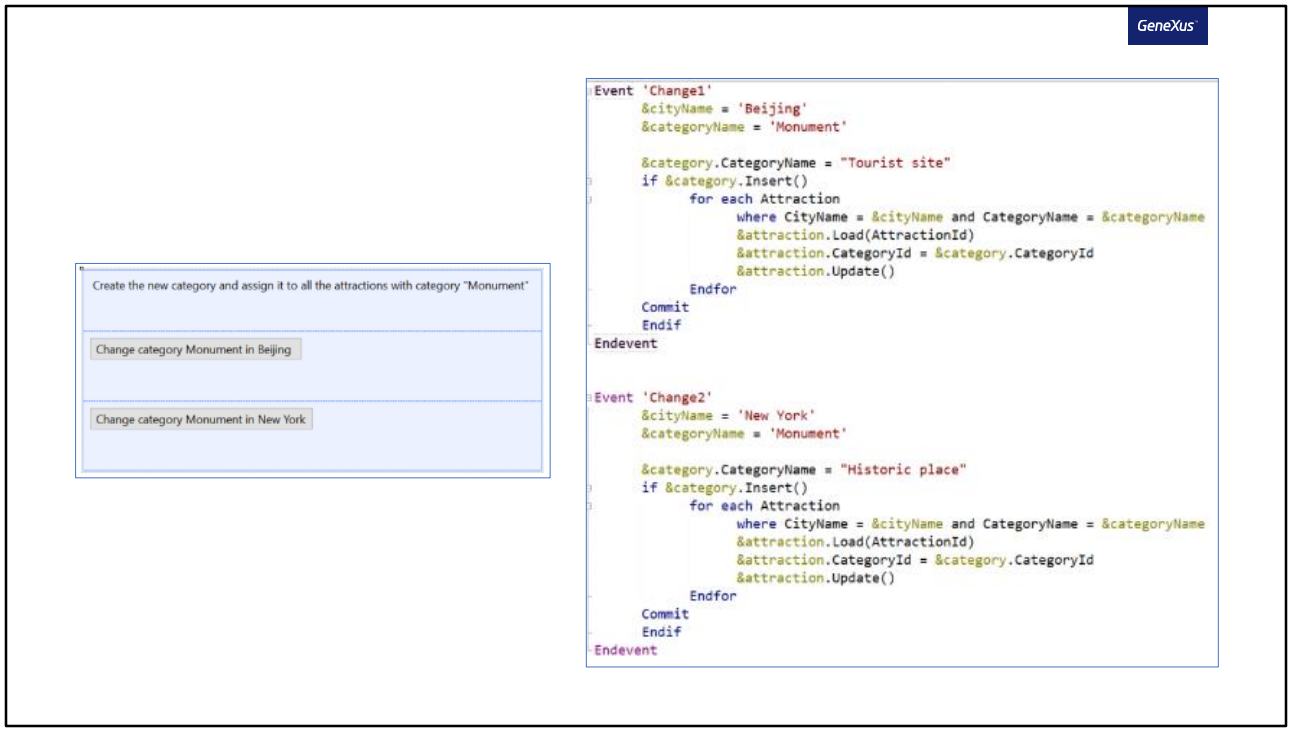
Las subrutinas pueden ser utilizadas en todos los objetos que acepten programación, a excepción de los Data Providers. Estos son: Web Panels, Procedimientos, Panels, Transacciones.

Veamos su funcionamiento en un simple ejemplo.



Lo haremos sobre una aplicación que estamos realizando para una agencia de viajes, entre sus transacciones tenemos una de nombre Category, para registrar las diferentes categorías, y otra de nombre Attraction. En esta última, además de sus atributos propios, tiene los atributos CategoryId y CategoryName de la transacción Category.

Veamos uno de los Web Panels de la aplicación, y analicemos su implementación.

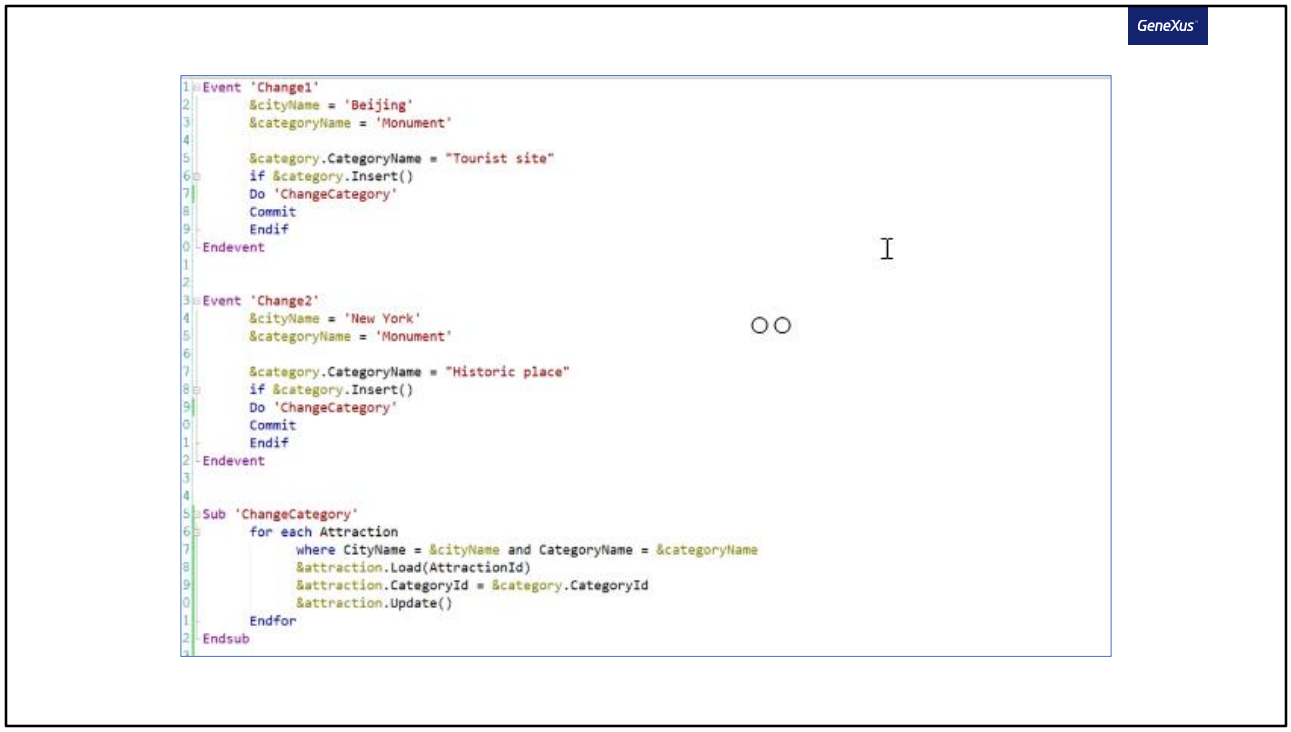


En el Layout vemos dos botones, uno de nombre "Change category Monument in Beijing" y el otro "Change category Monument in New York". El primero tendrá como nombre de evento Change1 y el segundo Change2.

En la sección eventos, vemos el código asignado justamente a estos dos eventos. Veamos el primero.
Se declaran dos variables, cityname y categoryname, y se le asigna el texto Beijing y monument respectivamente.
Luego tenemos una variable category, que es del tipo Business Component Category, a la que se le dice que el nombre de la categoría será Tourist site. Y posteriormente intentará insertar esta nueva categoría en la tabla Category.
Si se insertó el registro correctamente, entonces se recorrerá la tabla attraction, y filtrará solo los registros en los que CityName tenga igual valor a la variable CityName y que CategoryName tenga igual valor a la variable categoryName.
Para esos registros se actualizará la categoría de la atracción, cambiando por la que acabamos de ingresar, en este caso Tourist Site.

El segundo botón, hará lo mismo que el primero, pero con diferentes datos. El nombre de la categoría a insertar será Historic Place, y se filtraran las atracciones de Nueva York que tengan como categoría Monument.

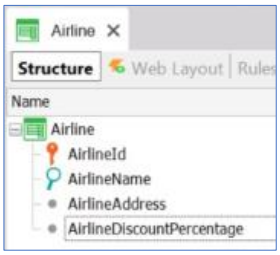
Si observamos, el primer y el segundo evento, tienen un bloque de código exactamente igual.
En este caso, podríamos declarar ese código en un único lugar, y simplemente llamarlo desde el evento que queramos. Vamos a hacerlo.



El comando Sub nos permitirá definir una subrutina, y a continuación debemos asignarle un nombre. Con endsub marcamos la finalización de la misma.
Dentro debemos ingresar el código que invocaremos luego. Lo copiamos, y lo pegamos aquí.
Eliminamos ese bloque en los eventos, y debemos en su lugar llamar a la subrutina. Esto lo hacemos mediante el comando Do, seguido del nombre.
Con esta implementación, el funcionamiento será exactamente el mismo al de antes de declarar la subrutina.

De esta manera, logramos modularizar nuestro código, quedando más claro y facilitando la lectura del mismo. Otra ventaja es que, si debemos cambiar algo de este bloque, lo hacemos una única vez y se aplica en todos los lugares donde es llamado. Y de esta forma, tenemos la posibilidad de reutilizar este código mediante la subrutina las veces que queramos, siempre que sea dentro de este mismo objeto, en este caso el Web Panel InsertCategoriesAndAttractions.

Veamos ahora este otro ejemplo, para comprender un poco más su funcionamiento.

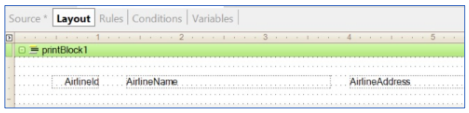



```
&Id = 1

For each Airline
      Where AirlineId = &Id
      AirlineAddress = '77 West Wacker Drive, Chicago'
      &NextId = AirlineId + 1
      do 'ChangeName'
Endfor

sub 'ChangeName'
      for each Airline
            where AirlineId = &NextId
            AirlineName = 'American Airlines, Inc'
      endfor
endsub
```

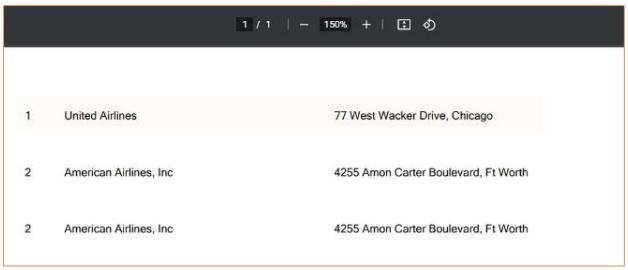
En nuestra aplicación, tenemos la transacción Airline, donde registramos las diferentes aerolíneas, con los siguientes atributos.

En este objeto procedimiento, tenemos el siguiente código en el source.
A través de este for each, se recorrerá la tabla Airline, filtrando por la aerolínea con id 1, ya que es el valor que le asignamos a esta variable.
A ese registro, le queremos actualizar su dirección, mediante el atributo AirlineAddress.
Luego tenemos una variable de nombre nextId, a la cual, le asignaremos el valor de AirlineId más uno, en este caso será dos. Y posteriormente llamamos a la subrutina ChangeName.
Esta subrutina realiza un for each también sobre la table Airline, filtrando por el registro que tenga a AirlineId igual a dos.
Si se encuentra ese registro, se actualizará su nombre.

Ahora, cuando se termina de ejecutar la subrutina, y se vuelve al for each principal, ¿en qué registro estaremos parados? Si acá tuviéramos un atributo de la transacción Airline, por ejemplo este (AirlineDiscountPercentage), y le asignamos un valor, ¿a qué registro aplicará? ¿Al registro con id uno, que es donde estábamos parados antes de llamar a la subrutina? ¿O al registro con id dos?




```
&Id = 1

For each Airline
      Where AirlineId = &Id
      AirlineAddress = '77 West Wacker Drive, Chicago'
      &NextId = AirlineId + 1
      Print PrintBlock1
      do 'ChangeName'
      Print PrintBlock1
      AirlineDiscountPercentage = 25
Endfor

sub 'ChangeName'
      for each Airline
            where AirlineId = &NextId
            AirlineName = 'American Airlines, Inc'
            Print PrintBlock1
      endfor
endsub
```



Veamos esto declarando tres atributos en el layout, para mostrar el Id, el nombre y la dirección de la aerolínea.

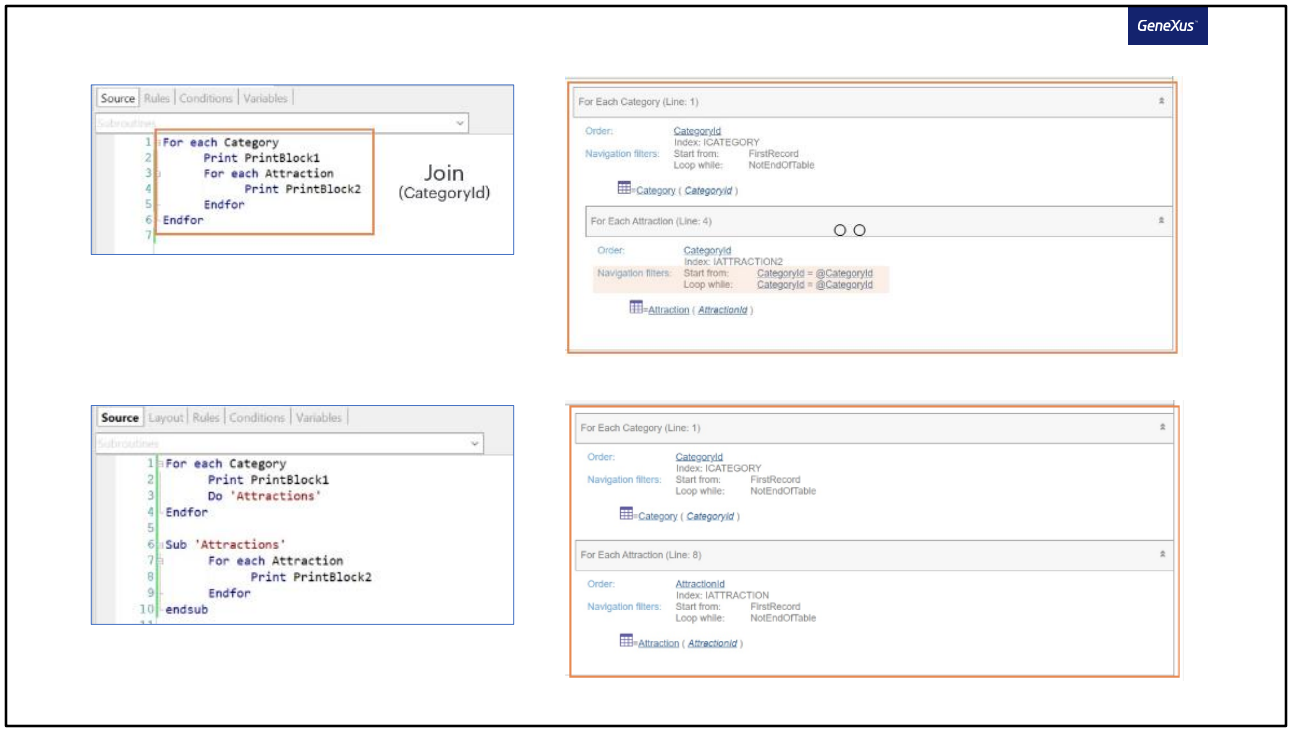
Y en el source, agregamos tres Print Printblock, para ver en que aerolínea estamos posicionados en cada momento. Ponemos uno antes de llamar a la subrutina, otro durante ejecución de la subrutina, y otro inmediatamente después que salimos de ella.

Ejecutemos para probar.
Vemos que en el primer print, antes de llamar a la subrutina, estamos posicionados en la aerolínea con Id 1, ya con la dirección actualizada.
El segundo print que se ejecuta, corresponde al que está dentro de la subrutina. Vemos que se imprime en pantalla el registro con el Id dos, y ya con el nuevo nombre actualizado.
El tercer print, se ejecuta una vez salimos de la subrutina. Y vemos que, en ese momento, seguimos posicionados en el registro con Id dos, que es donde estábamos dentro de la subrutina. Y no en el registro con Id uno, que es donde estábamos posicionados antes de llamar a la subrutina.
Por lo cual, si en este momento actualizamos el atributo AirlineDiscountPercentage con un valor, se hará sobre el registro con Id dos.
Este funcionamiento se da debido a que los atributos declarados, serán globales del objeto. Por tanto, si un atributo toma valor en una determinada sección de un objeto, y posteriormente se llama a una subrutina que también asigna un valor al mismo atributo, al regresar de la subrutina invocada y consultar el valor del atributo tendrá el valor asignado en la subrutina. Esto lo acabamos de ver con el atributo AirlineId.

Las subrutinas no admiten el paso de parámetros, por lo tanto, para intercambiar datos utilizamos variables, las cuales son globales para los objetos.

Si no quisiéramos este comportamiento que acabamos de ver, en lugar de utilizar una subrutina, podríamos llamar a un procedimiento, por ejemplo. Pasándole en ese caso sí por parámetro, la variable por la que queramos luego utilizar para filtrar.

Veamos ahora un tercer ejemplo.



En este caso tenemos el siguiente procedimiento, el cual realiza un for each que navega la tabla Category, y un for each anidado que navegará la tabla Attraction. Como sabemos, en este caso, GeneXus realizará un Join por CategoryId. Ya que toda atracción tendrá una categoría asignada. Y CategoryId es el atributo común que permitirá unir ambas tablas. Veamos esto en el listado de navegación.

Volviendo al source del procedimiento vemos que implementamos dos printblock, uno imprimirá el nombre de la categoría, y el otro el nombre de la atracción.

Si ejecutamos, vemos que efectivamente se imprime el nombre de la categoría, y dentro las atracciones que tengan esa categoría asociada.
Ahora, en caso que no quisiéramos este comportamiento, o sea que no se realice el Join, sino que se imprima la categoría y luego se impriman todas las atracciones, independientemente de a que categoría pertenezcan. ¿Cómo podríamos implementarlo?
Una opción es poner este código dentro de una subrutina.
Veamos ahora el listado de navegación.
Vemos que efectivamente se recorre toda la tabla Category, y luego es recorrida toda la tabla Attraction, del primer al último registro, sin aplicar ningún tipo de filtro.

De esta forma, GeneXus ya no realiza la inferencia automática, y no hace el filtro por CategoryId. Serán dos navegaciones independientes.

Hasta aquí vimos en diferentes ejemplos el uso y funcionamiento de las subrutinas.

GeneXus

Subroutine

- Code Block
- Can be used in:
  - Web Panels
  - Procedures
  - Panels
  - Transactions
- They are defined by the 'SUB' command and invoked with 'DO'
- Attributes are global to the object
- The for each command are not nested.

Hagamos un pequeño resumen.
Las subrutinas:

- Son bloques de código, que nos permiten modularizar el mismo. Pudiendo invocarlas tantas veces queramos dentro del mismo objeto.
- Se pueden utilizar en Web Panels, Procedimientos, Panels, Transaciones
- Son definidas mediante el comando SUB y luego invocadas mediante el comando DO.
- No admiten el paso de parámetros, para intercambiar datos se utilizan variables.
- Si un atributo tiene un valor, y al llamar a la subrutina este cambia. Al regresar de la subrutina y consultar el valor, tendrá el que fue asignado en la misma, ya que los atributos son globales al objeto.
- Si se llama desde dentro de un for each y la subrutina tiene también un comando for each, los mismos no se anidarán, o sea, no se harán inferencias ni filtros.

Para más información sobre este tema, los invitamos a visitar nuestro Wiki.


GeneXus™