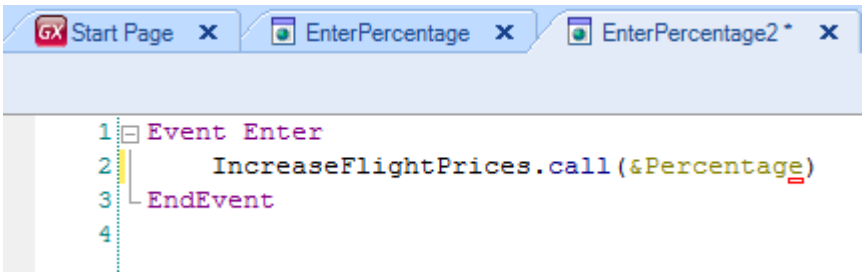
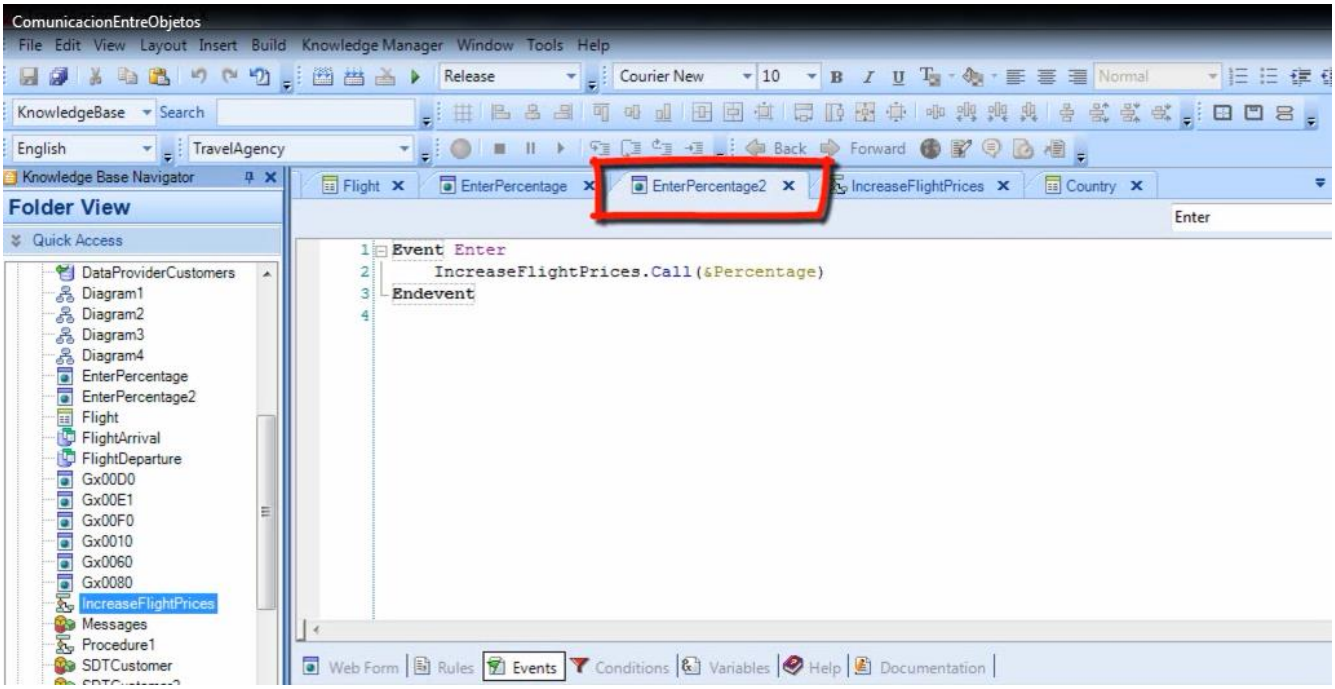
Comunicación entre objetos

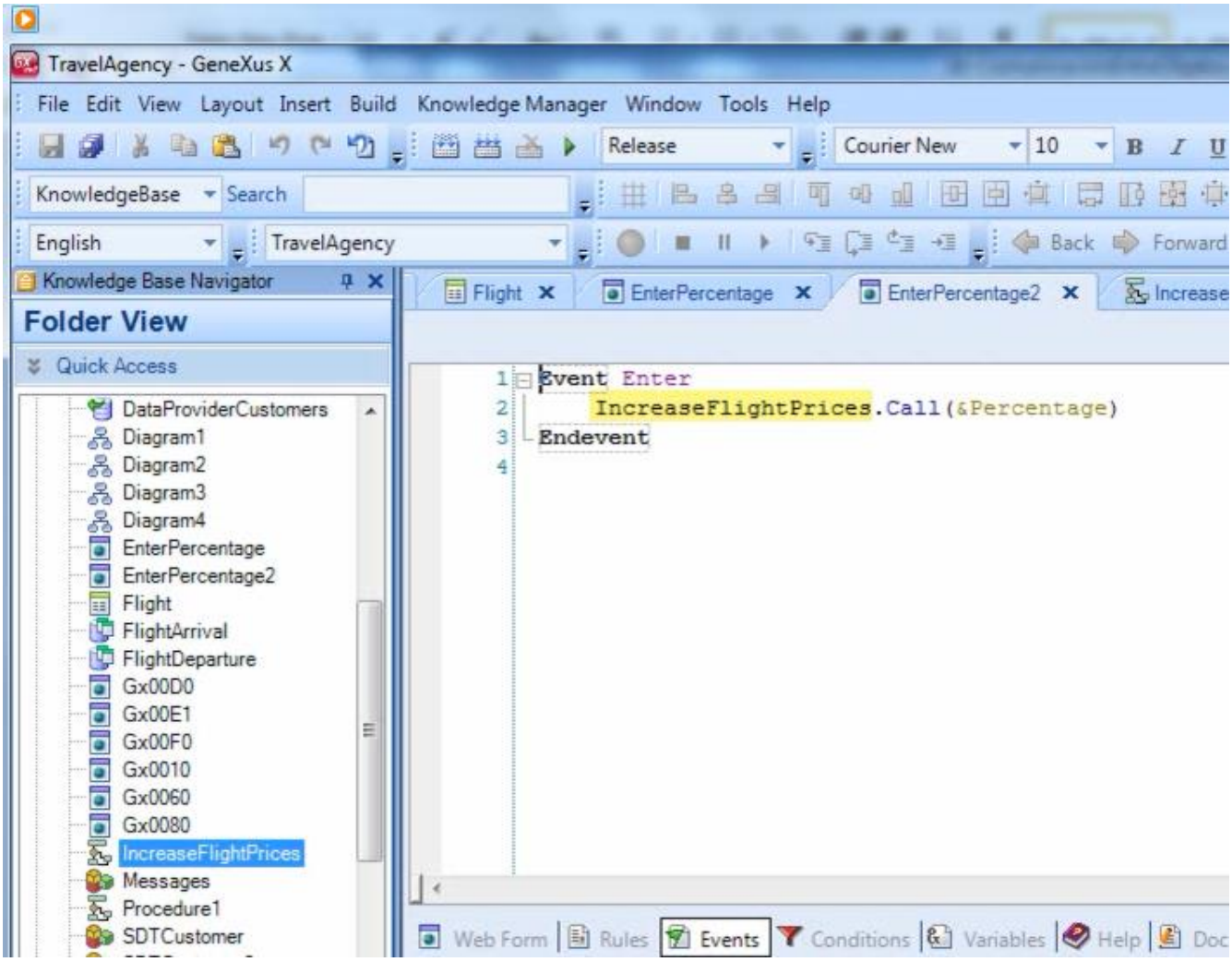


En situaciones anteriores nos hemos encontrado con la necesidad de llamar a un objeto, desde otro.

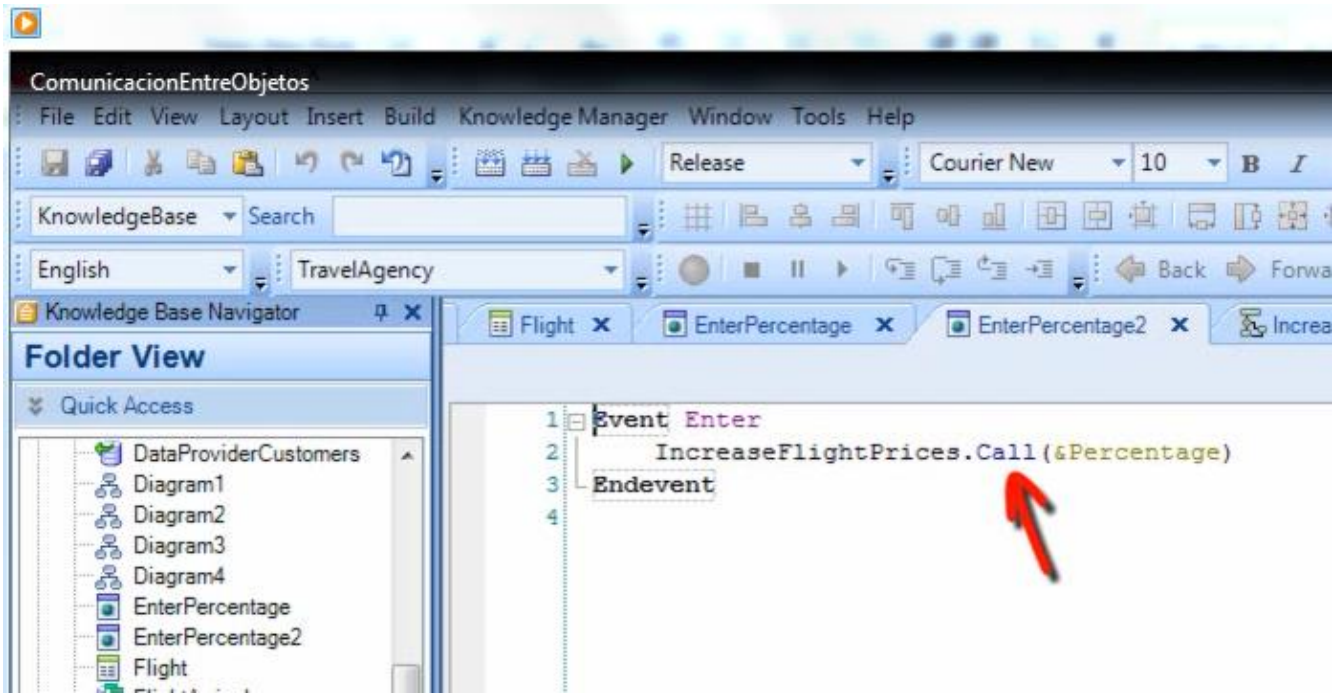
Por ejemplo en el evento Enter del web panel “EnterPercentage2”,



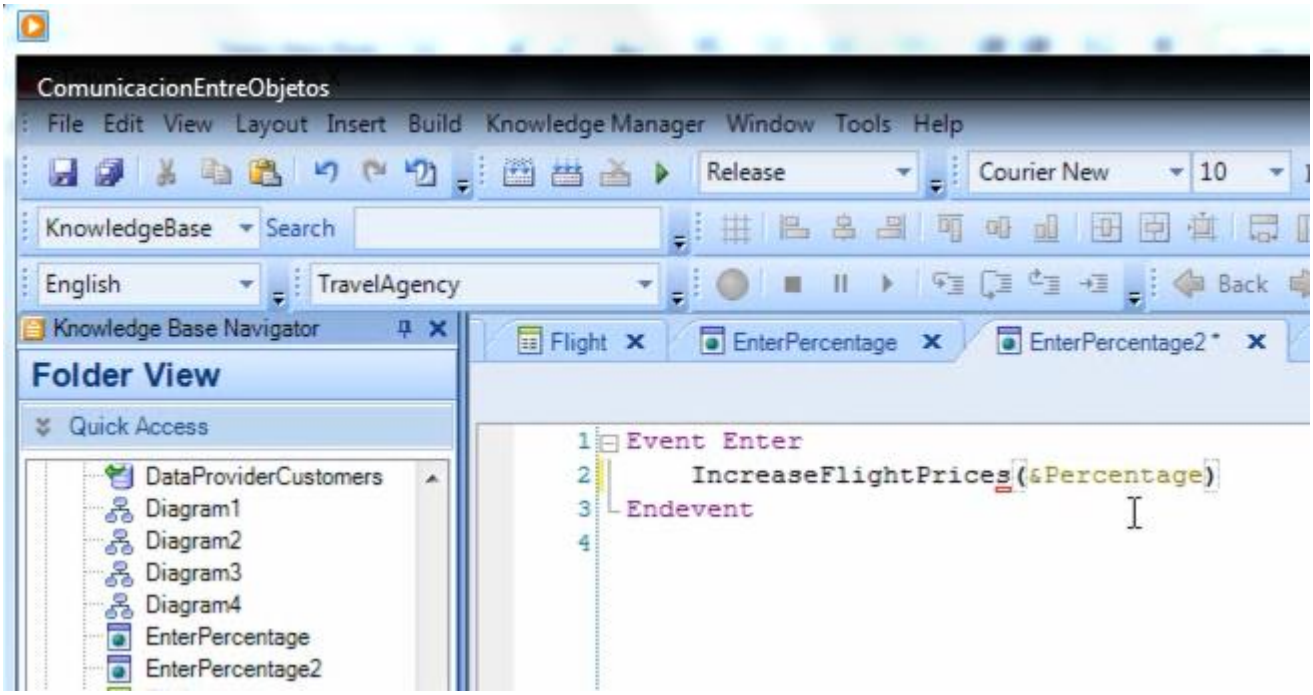
estamos llamando al procedimiento “IncreaseFlightPrices”



con el método call :



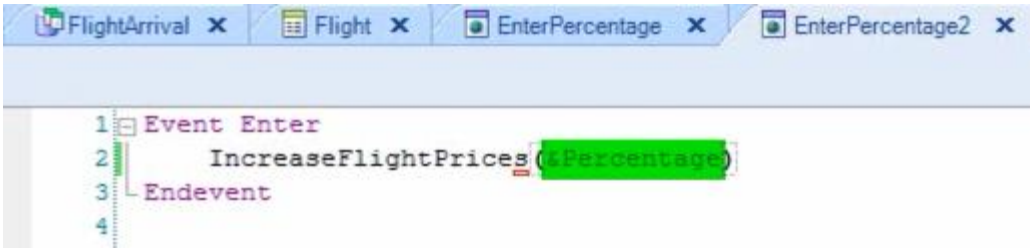
La sintaxis en las llamadas, permite que podamos omitir el call y el funcionamiento será exactamente el mismo, ya que GeneXus tiene la inteligencia para determinar que estamos llamando a un objeto; así que eliminemos el punto y el call.



Recordemos, que cuando llamamos a un objeto podemos pasarle datos que tenemos en el objeto llamador,

para que el objeto llamado los conozca y pueda usarlos.

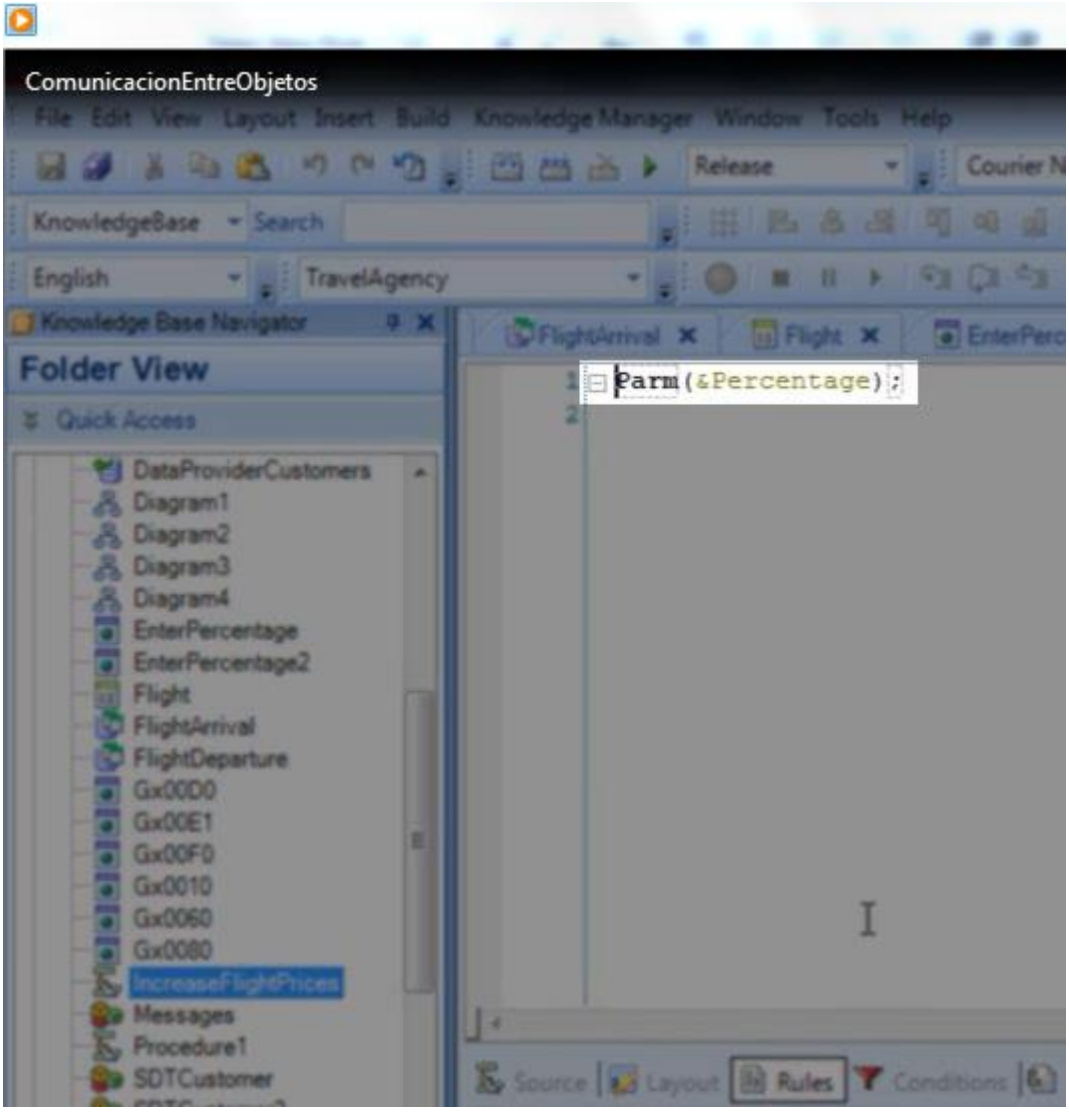
En este ejemplo, estamos pasándole al procedimiento



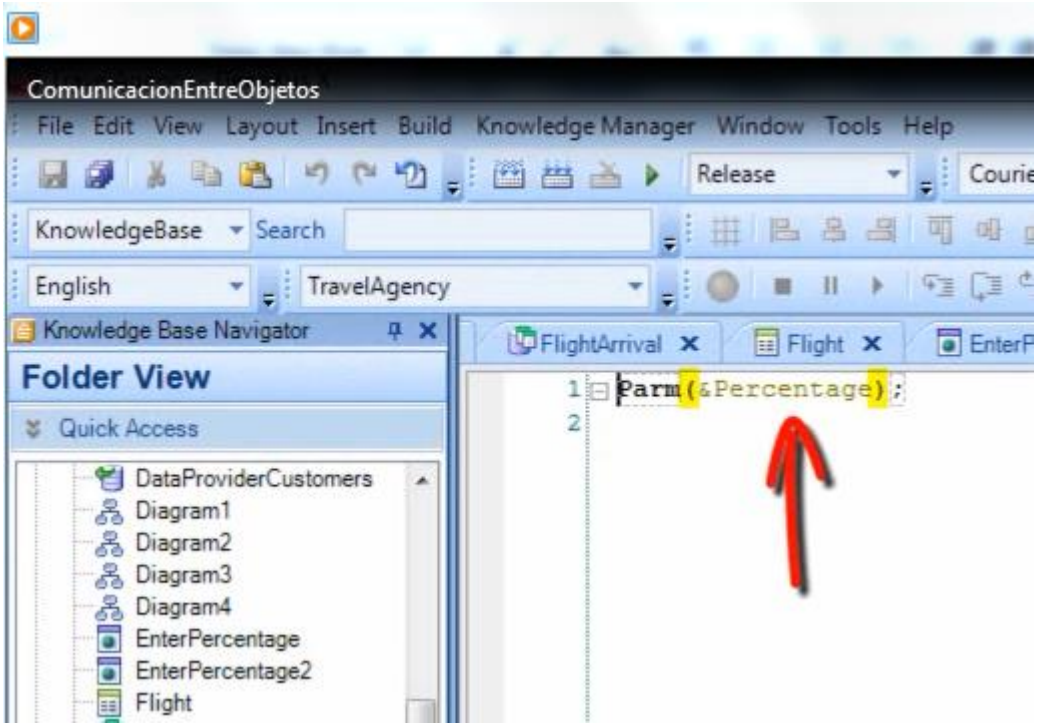
el valor del porcentaje ingresado por el usuario en el web panel, almacenado en la variable &percentage.

En este caso el valor lo tenemos en una variable. Si tuvieramos el dato en un atributo, incluiríamos dentro del paréntesis al atributo que corresponda, o en caso de tener que pasar dos o más valores, enviaríamos varios atributos y/o variables separados por coma.

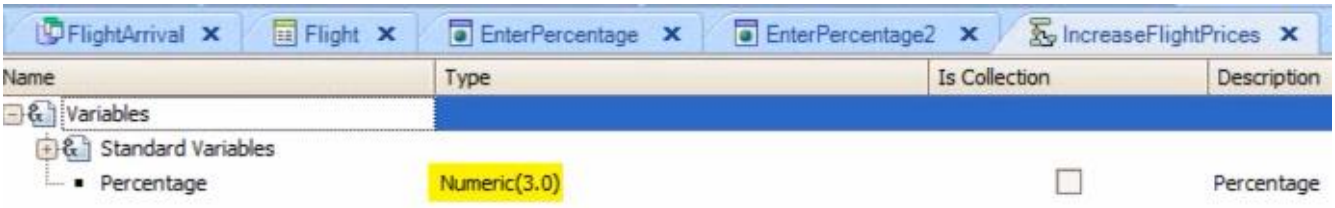
Como ya hemos visto en el objeto llamado, declaramos los datos recibidos en la regla parm.



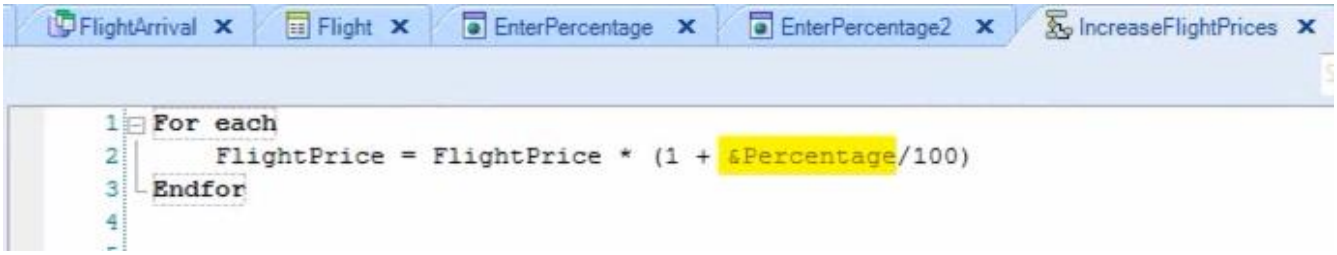
Vemos en esta regla, a la variable &percentage entre paréntesis.



En este objeto hemos definido a la variable con el mismo nombre y tipo de datos con que se conoce en el web panel. El nombre podría haber sido distinto, en cambio el tipo de datos debe coincidir.



Recordemos que la variable recibida con el valor del porcentaje, la utilizamos en el source del procedimiento.



Si al llamar a un objeto, no se le envían datos, en el objeto llamado **no hay que declarar una regla parm**.

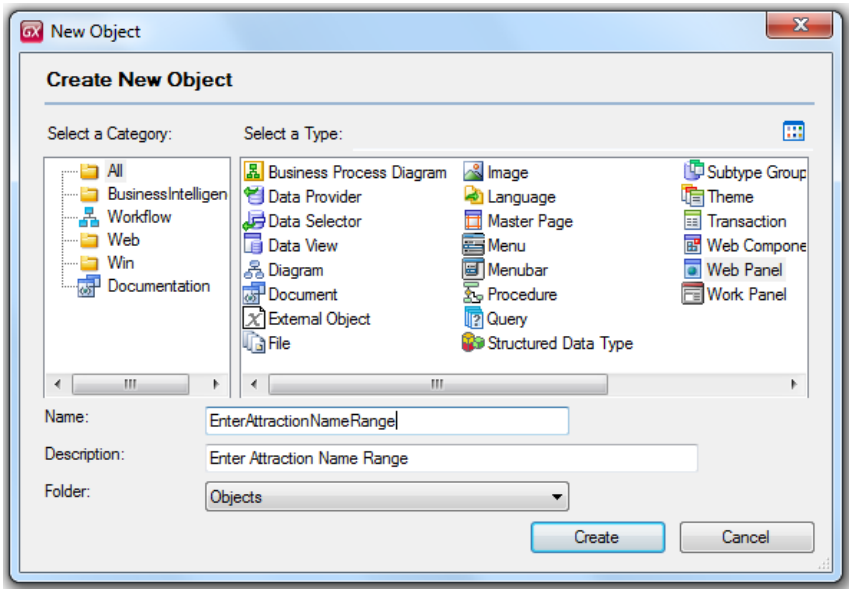
```
1  parm(&Percentage);
2
```

Y si se envía más de un dato, hay que recibirlos dentro de la regla parm en el mismo orden, separados por coma.

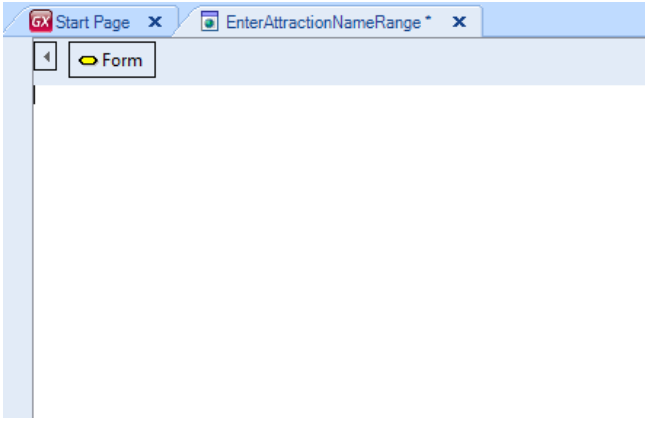
Veamos un ejemplo.

Vamos a definir un web panel para que el usuario ingrese un rango inicial y final de nombres de atracciones que desee listar. Y desde el web panel llamaremos a un procedimiento, para que liste todas las atracciones cuyos nombres estén incluídos en dicho rango.

New/Object , seleccionamos web panel, y le damos el nombre: EnterAttractionNameRange

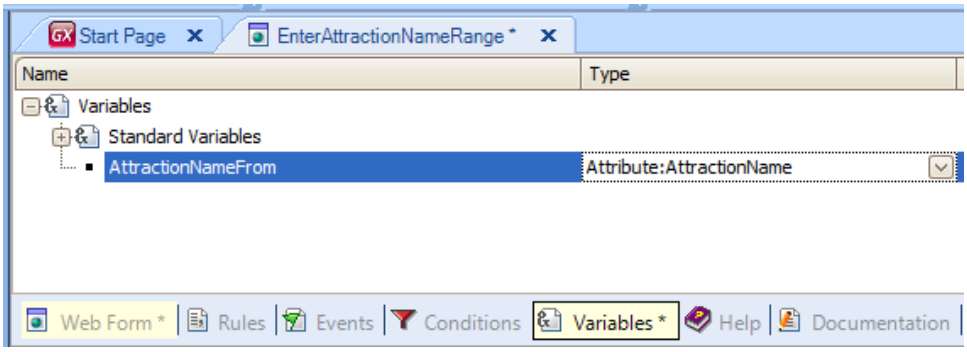


Presionamos Create

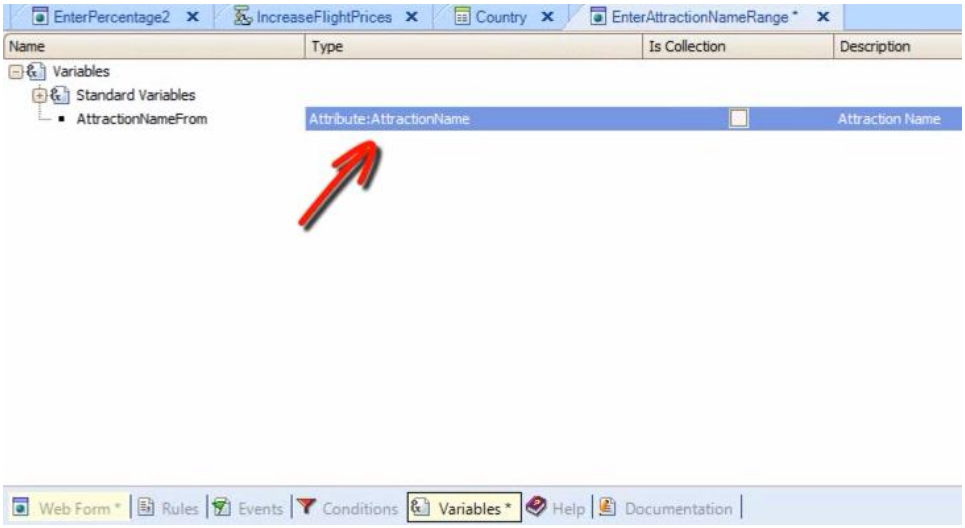


Vamos a la sección de variables para crear las 2 variables que necesitamos.

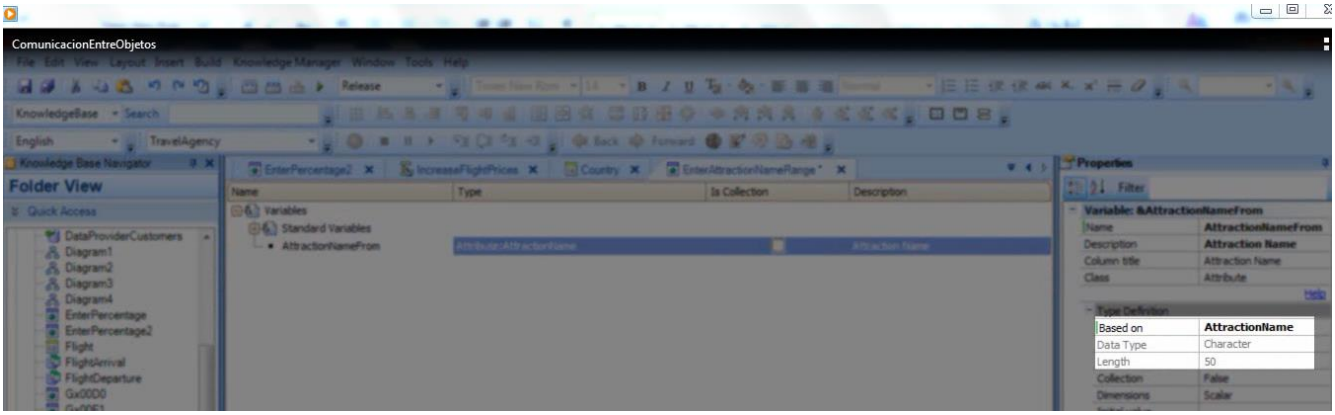
Definimos el nombre de variable: AttractionNameFrom



y se nos sugiere basar la definición de la variable en la definición del atributo AttractionName.

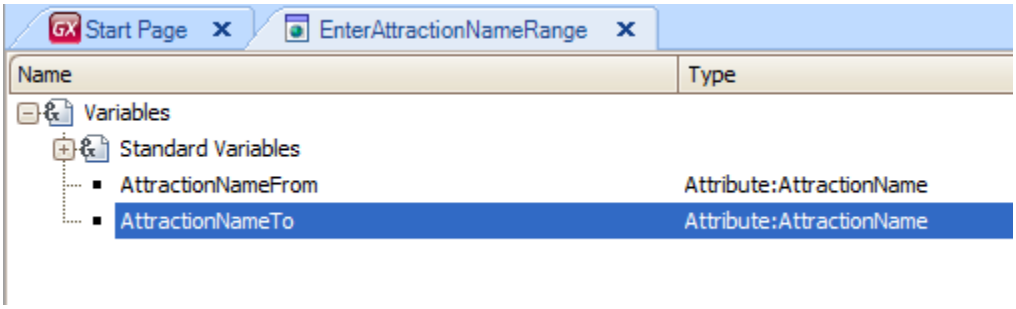


Esto significa que la definición de la variable está enlazada con la definición del atributo



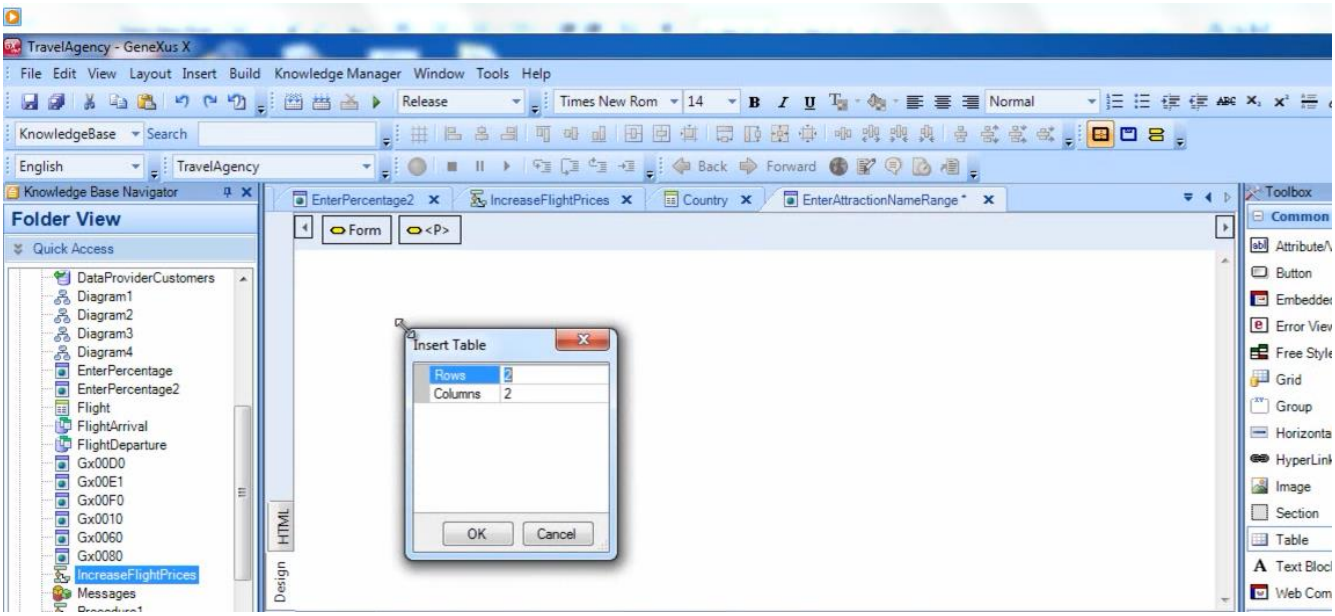
y si en el futuro cambiamos el tipo de datos del atributo, el tipo de datos de la variable cambiará acorde, automáticamente.

Vamos a definir una variable más, de nombre: AttractionNameTo, del mismo tipo:



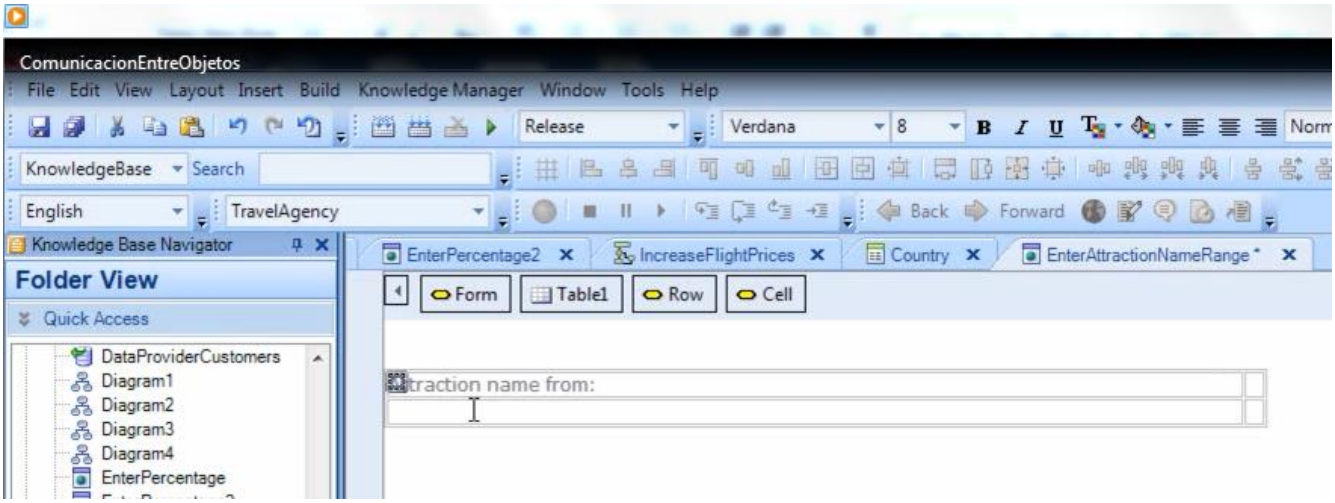
Y ahora vamos al form.

Desde la ToolBox insertamos una tabla y dejamos el tamaño por defecto.

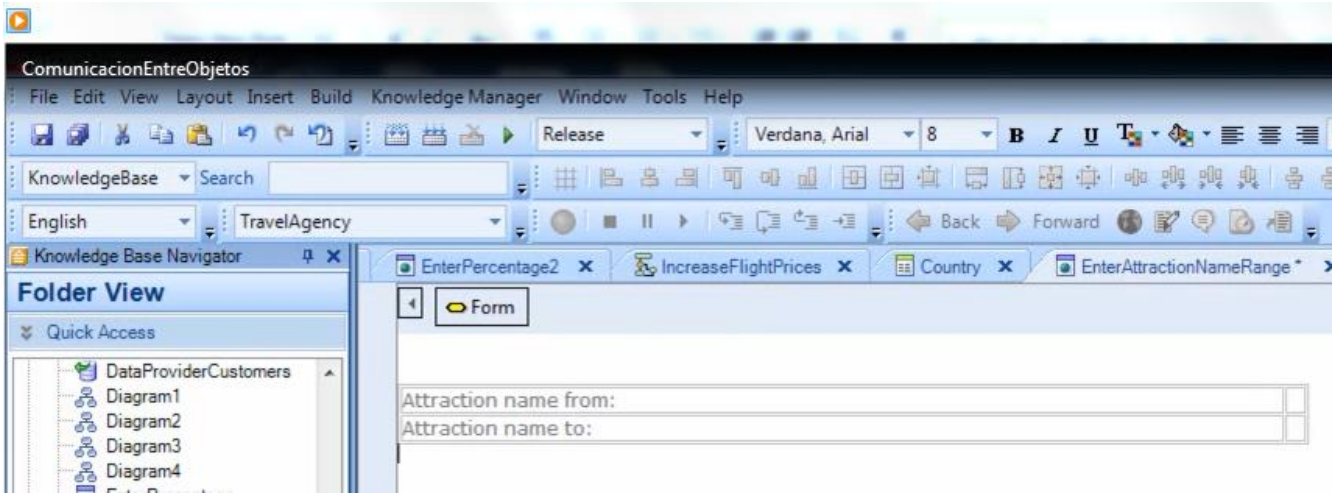


Cuando diseñamos en web, es importante usar tablas para que lo que escribamos o insertemos en la pantalla, quede debidamente alineado.

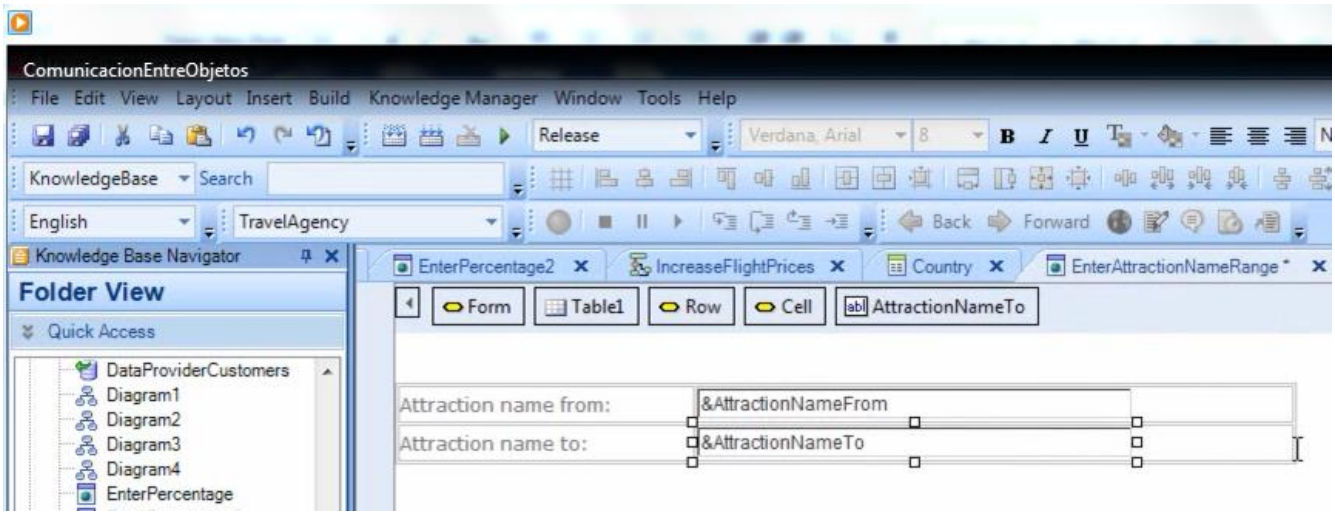
Digitamos “Attraction name from:” en esta celda



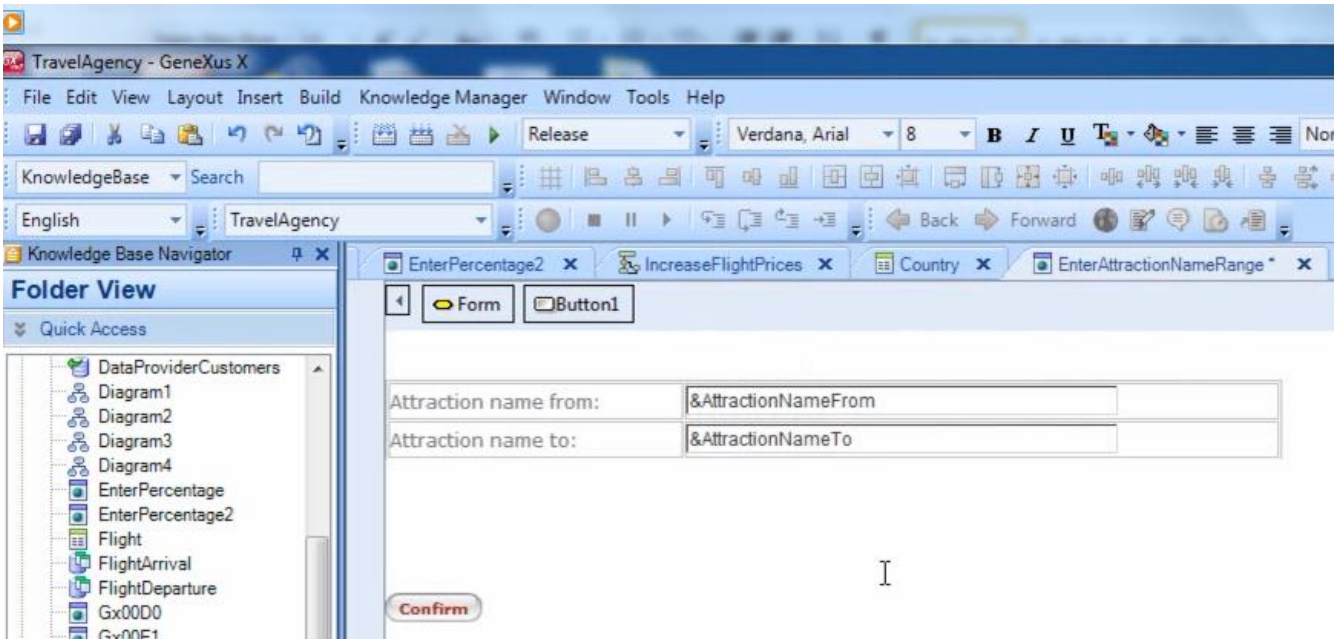
y “Attraction name to:” en esta otra.



e insertamos las variables que recién definimos

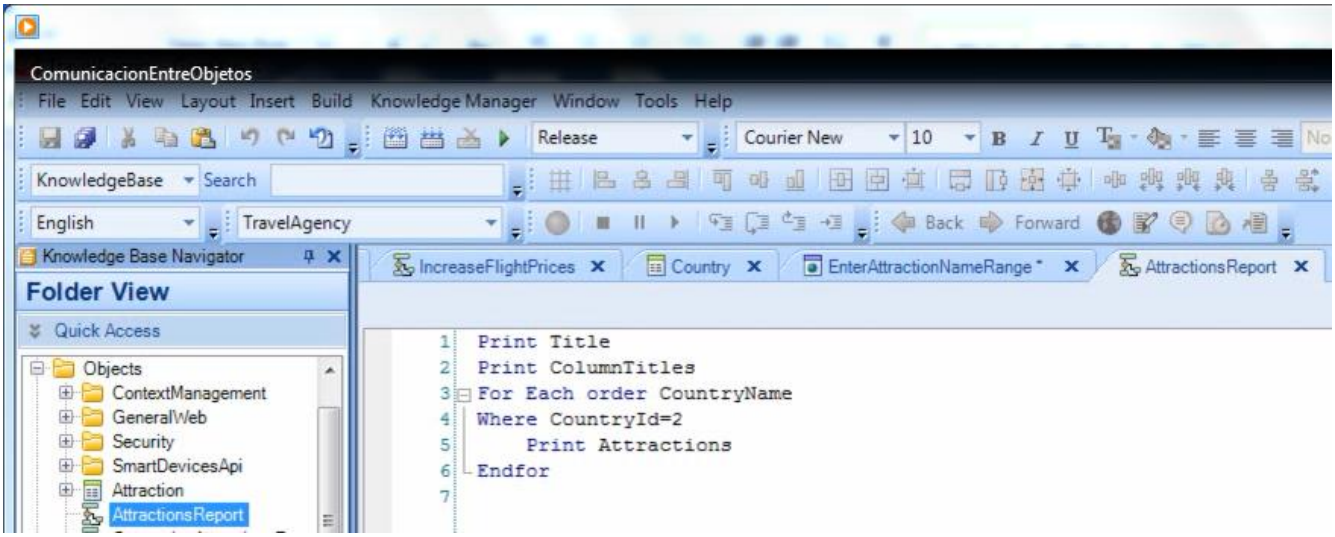


Insertamos también un botón

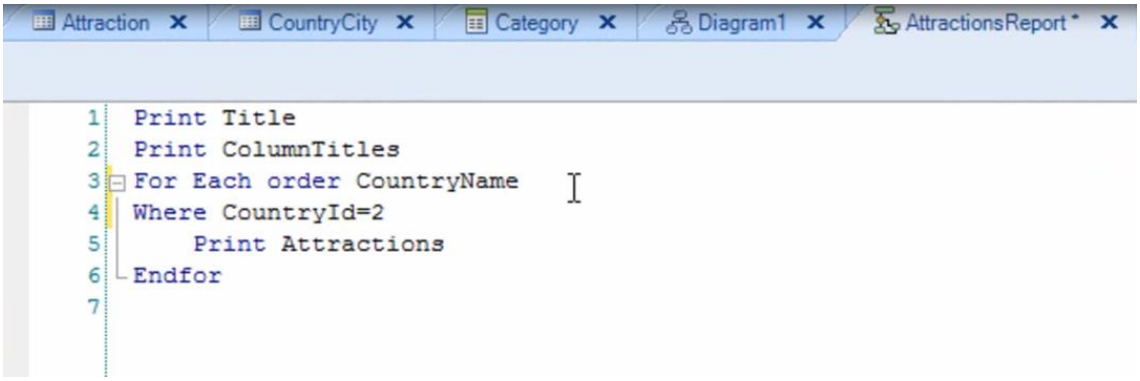


y dejaremos por un momento el web panel así, para pasar a resolver el procedimiento al cual llamaremos en el evento asociado al botón.

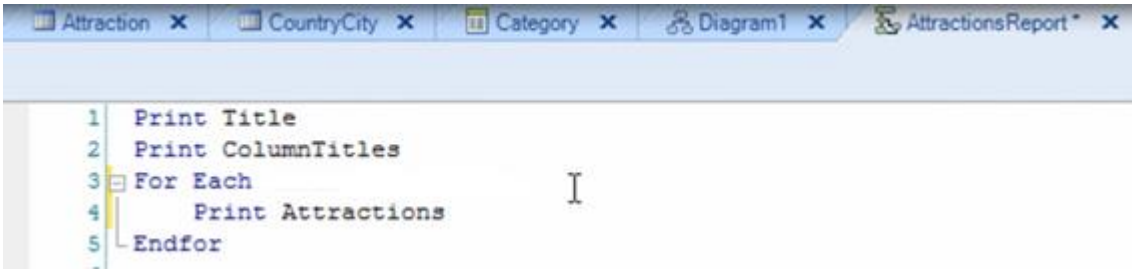
En la ventana Folder View ubicamos al procedimiento AttractionsReport



presionamos botón derecho del mouse, elegimos **Save As …**le damos el nombre "AttractionsReport2" y obtenemos una copia, a la cual le haremos unos pocos cambios.

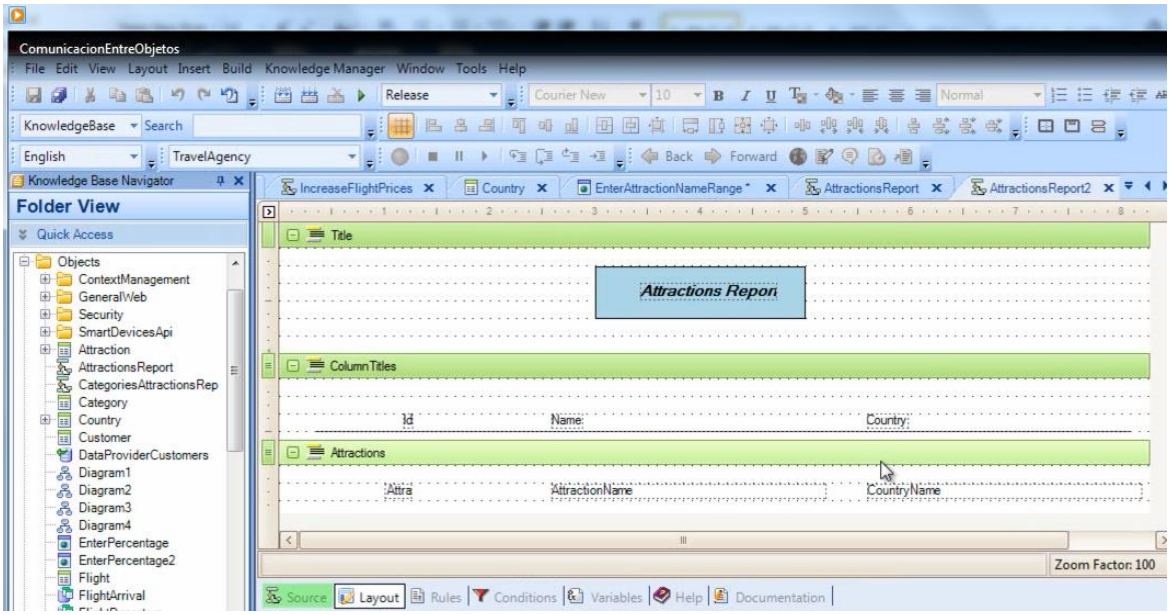


Borramos el order y el where.

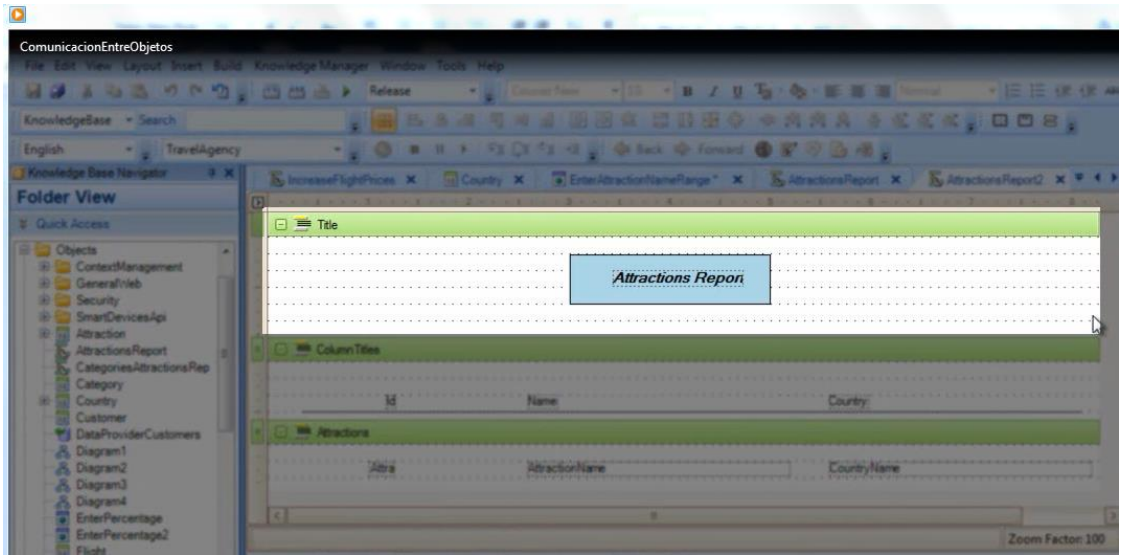


De esta forma, el listado imprime todas las atracciones.

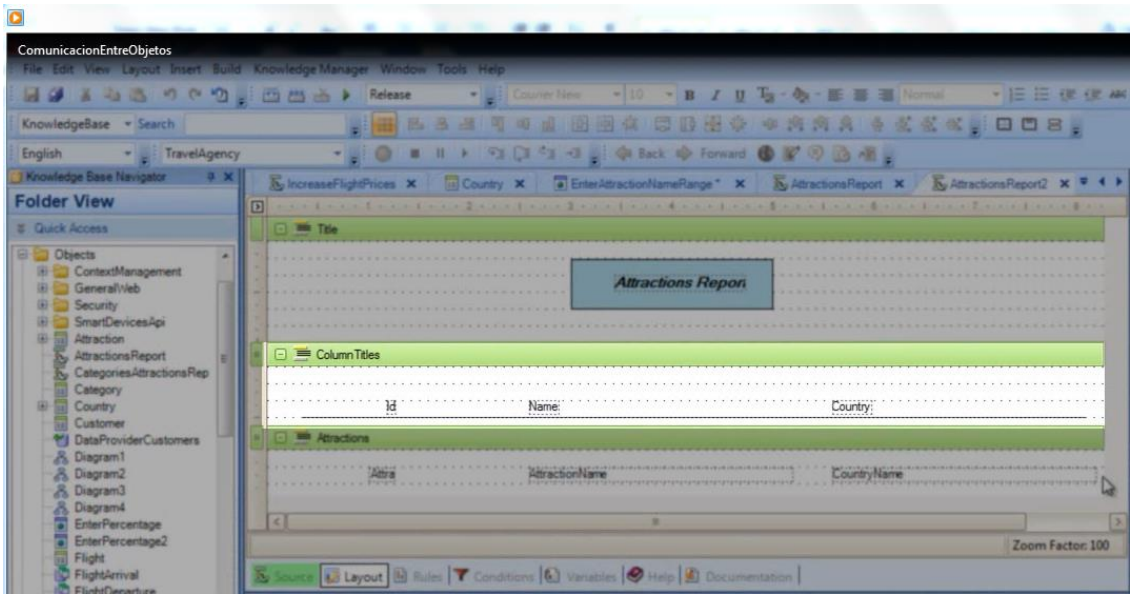
Vamos a ver el layout



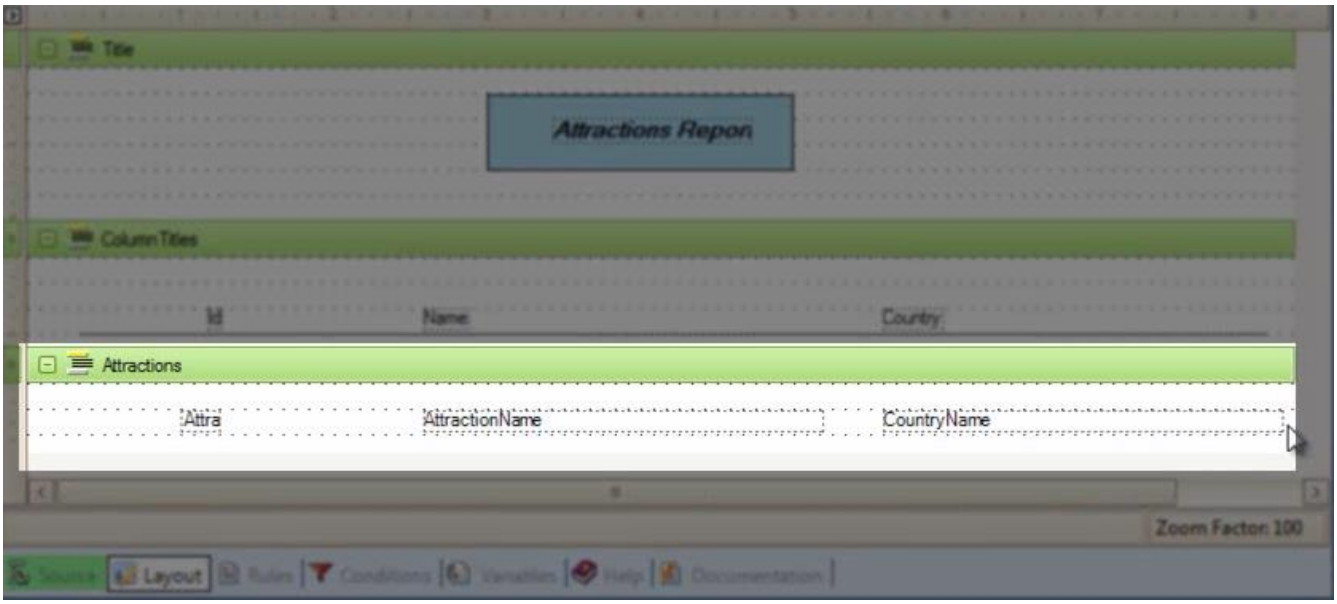
contiene 3 printblocks… éste imprime el título del reporte



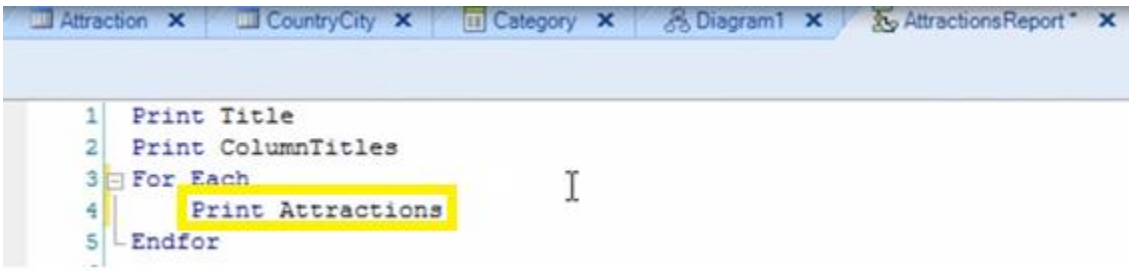
éste imprime los títulos de las columnas



y éste es el único que contiene atributos y es el que se imprime dentro del For each.



Vamos nuevamente al source, y vemos que el For each contiene solamente la invocación al printblock Attractions

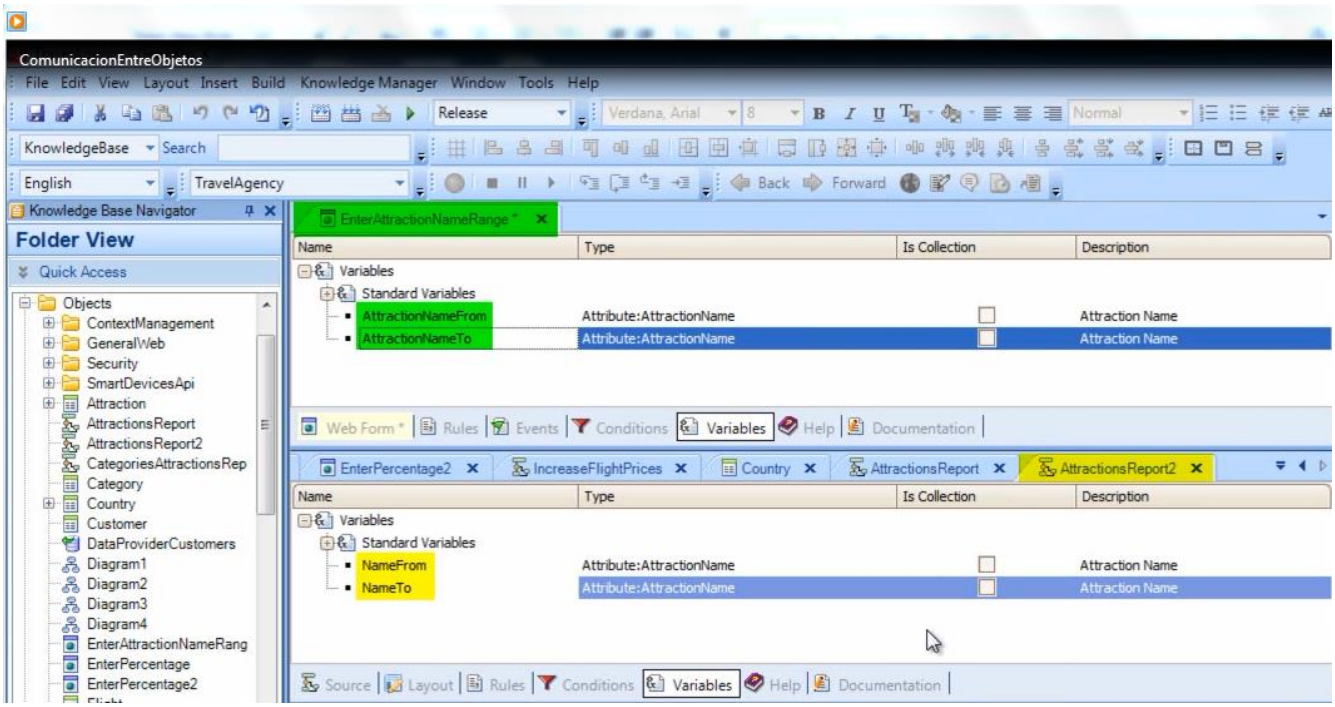


Definiremos ahora, que este procedimiento reciba el rango inicial y final de nombres de atracciones, y usaremos a los datos recibidos para filtrar en el For each.

Vamos a la sección de variables del procedimiento y definimos 2 variables:

- NameFrom, basado en el atributo AttractionName
- Y NameTo, también basado en el atributo AttractionName

Observemos que le hemos dado a las variables, distintos nombres respecto a los nombres de variables que definimos en el web panel.



Lo importante es que **los tipos de datos** enviados y recibidos, coincidan!!

Ahora, vamos a la sección Rules del procedimiento y escribimos: parm(&NameFrom,&NameTo);

AttractionsReport2

```
1 parm(&NameFrom,&NameTo);
2 output_file('AttractionsReport2.pdf','pdf');
3
```

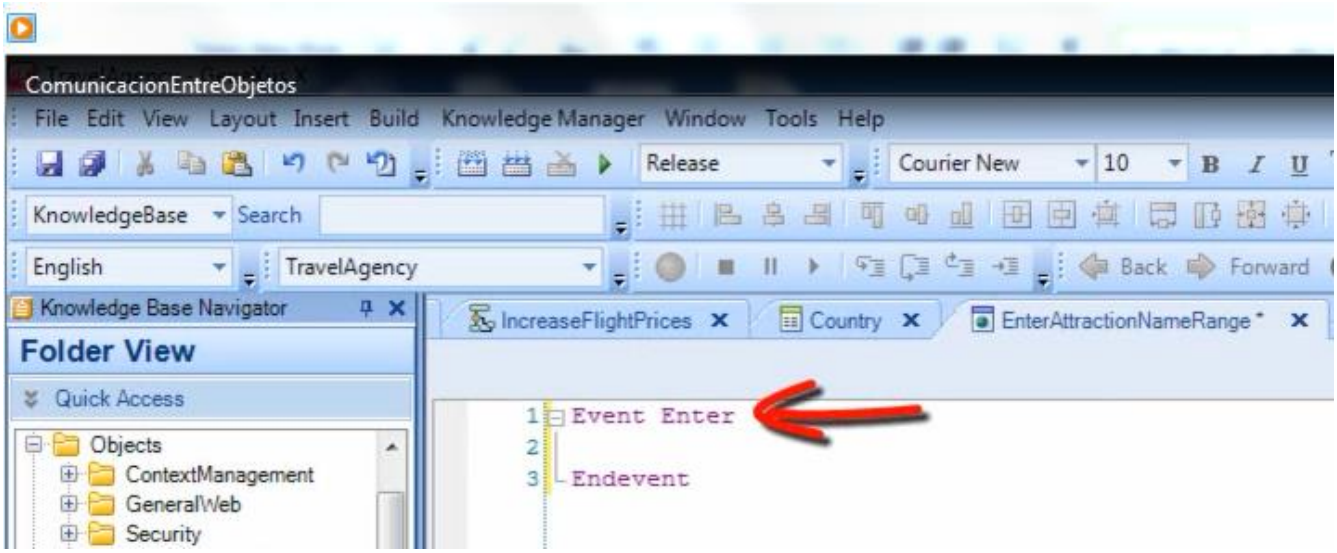
Estas variables que recibimos en el procedimiento, las vamos a usar para filtrar en el For each. Así que volvemos al source y escribimos… where AttractionName>=&NameFrom.. enter.. y where AttractionName>=&NameTo

```
1 Print Title
2 Print ColumnTitle
3 for each
4     where AttractionName >= &NameFrom
5     where AttractionName <= &NameTo
6     print Attractions
7 endfor
```

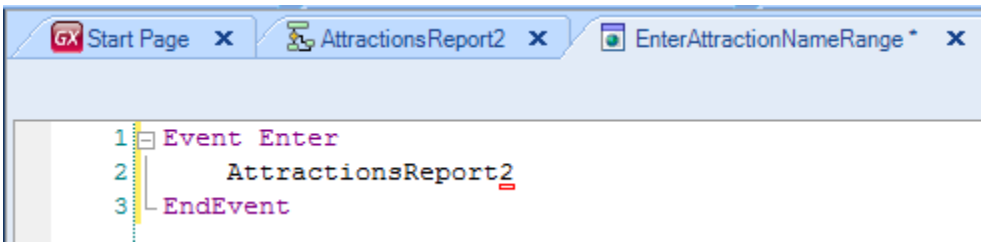
Dado que este procedimiento fue salvado a partir de otro, las propiedades y la regla necesaria para que la impresión salga en formato PDF ya están configuradas.

Con esto el procedimiento está listo y solamente nos resta llamarlo desde el web panel.

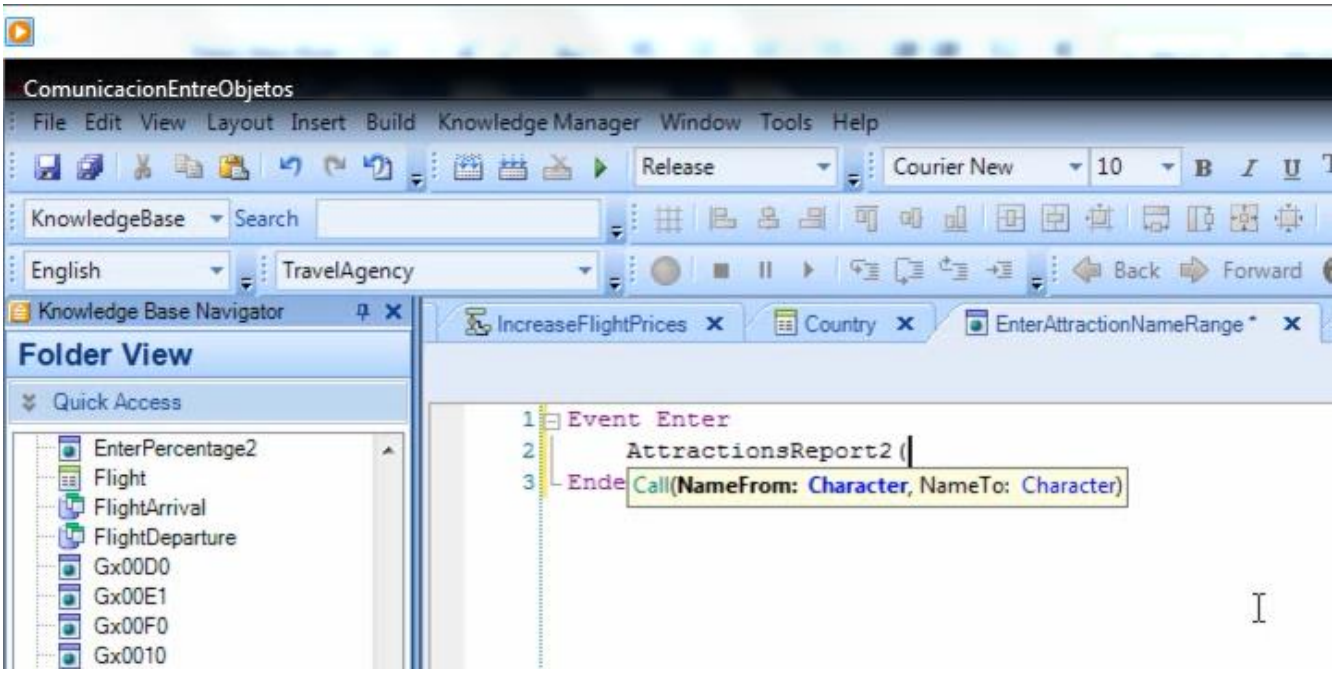
Vamos al web panel y presionamos doble clic sobre el botón para que veamos el evento asociado



Insertamos el nombre del procedimiento AttractionsReport2

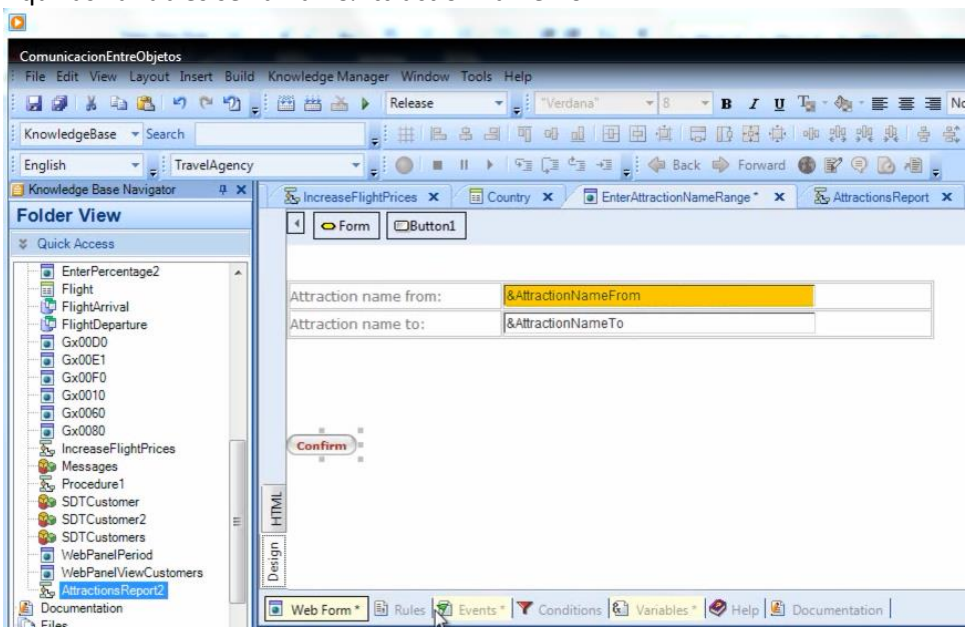


y pegado al nombre del procedimiento, agregamos un par de paréntesis,

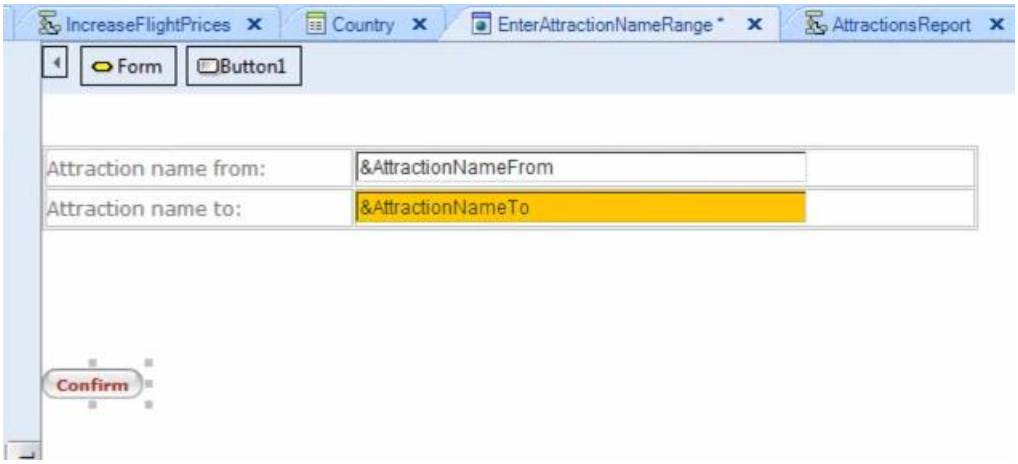


dentro de los cuales vamos a incluir los datos que queremos enviar, separados por coma.

Aquí las variables se llaman &AttractionNameFrom



y &AttractionNameTo

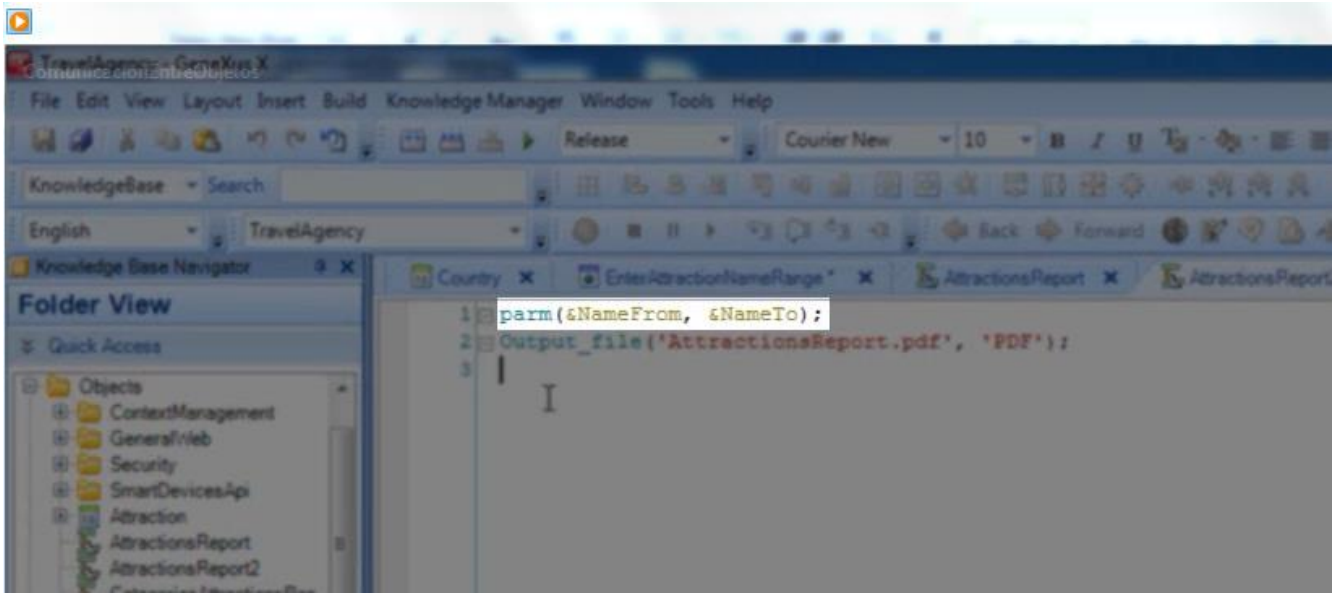


así que las enviaremos con estos nombres.

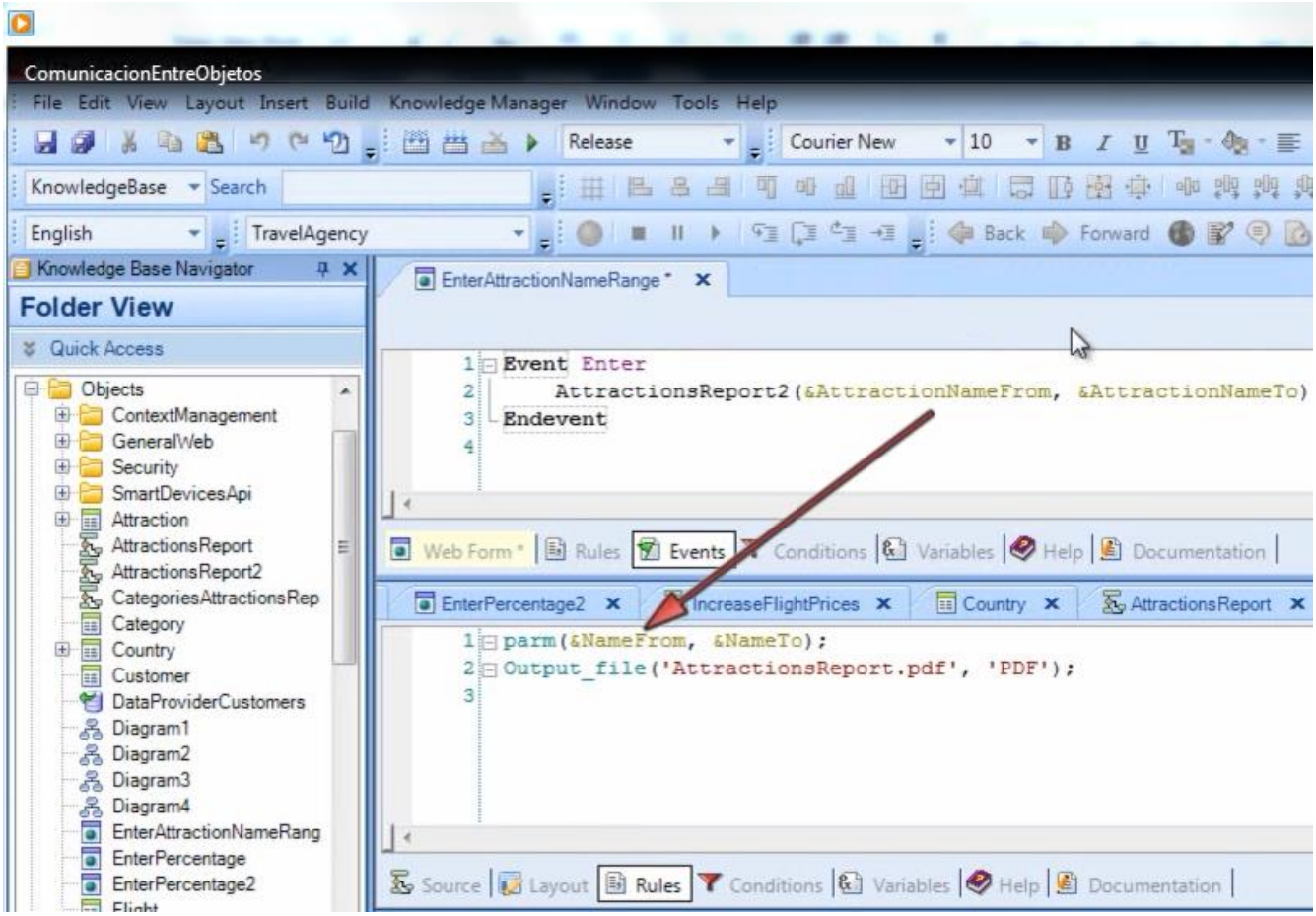
Dentro de los paréntesis digitamos ampersand y elegimos &AttractionNameFrom… digitamos una coma, ampersand nuevamente y elegimos &AttractionNameTo.

Queda completa la llamada al procedimiento, al cual le enviamos el rango de nombres de atracciones que el usuario digitó en el form en estas 2 variables.

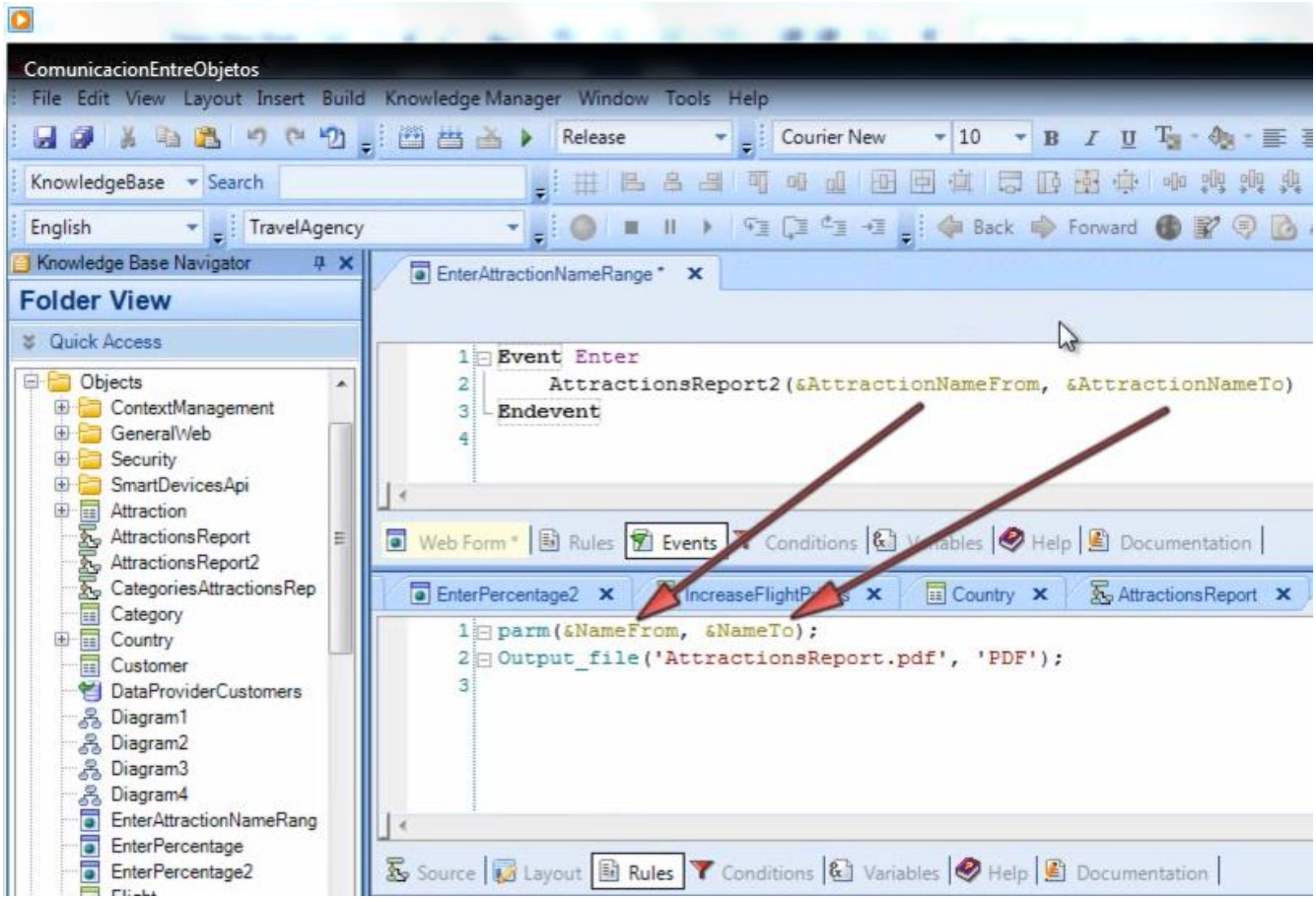
Si vamos al procedimiento, la regla parm, recibe 2 variables:



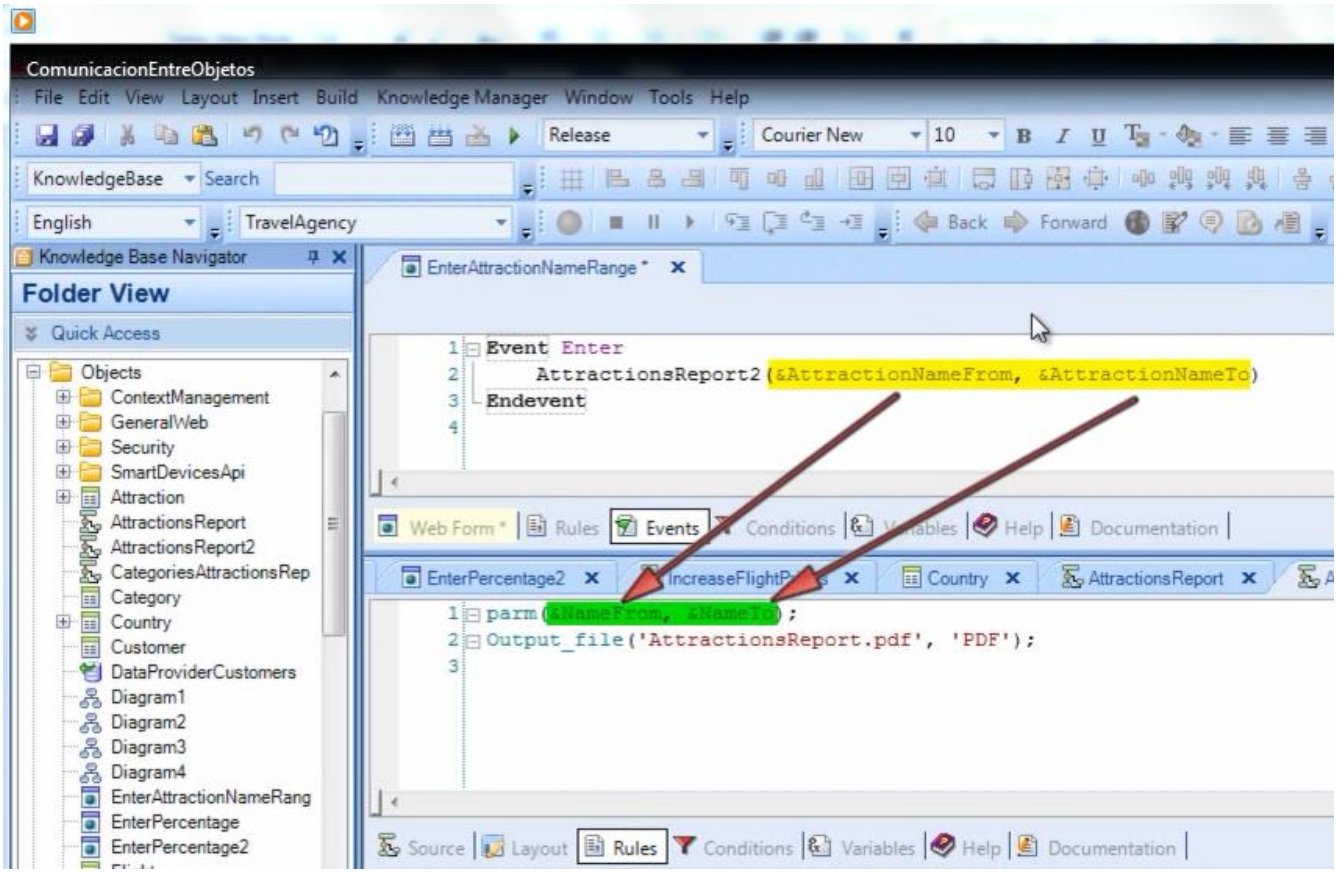
El dato que se envió en primer lugar, viene en primer lugar



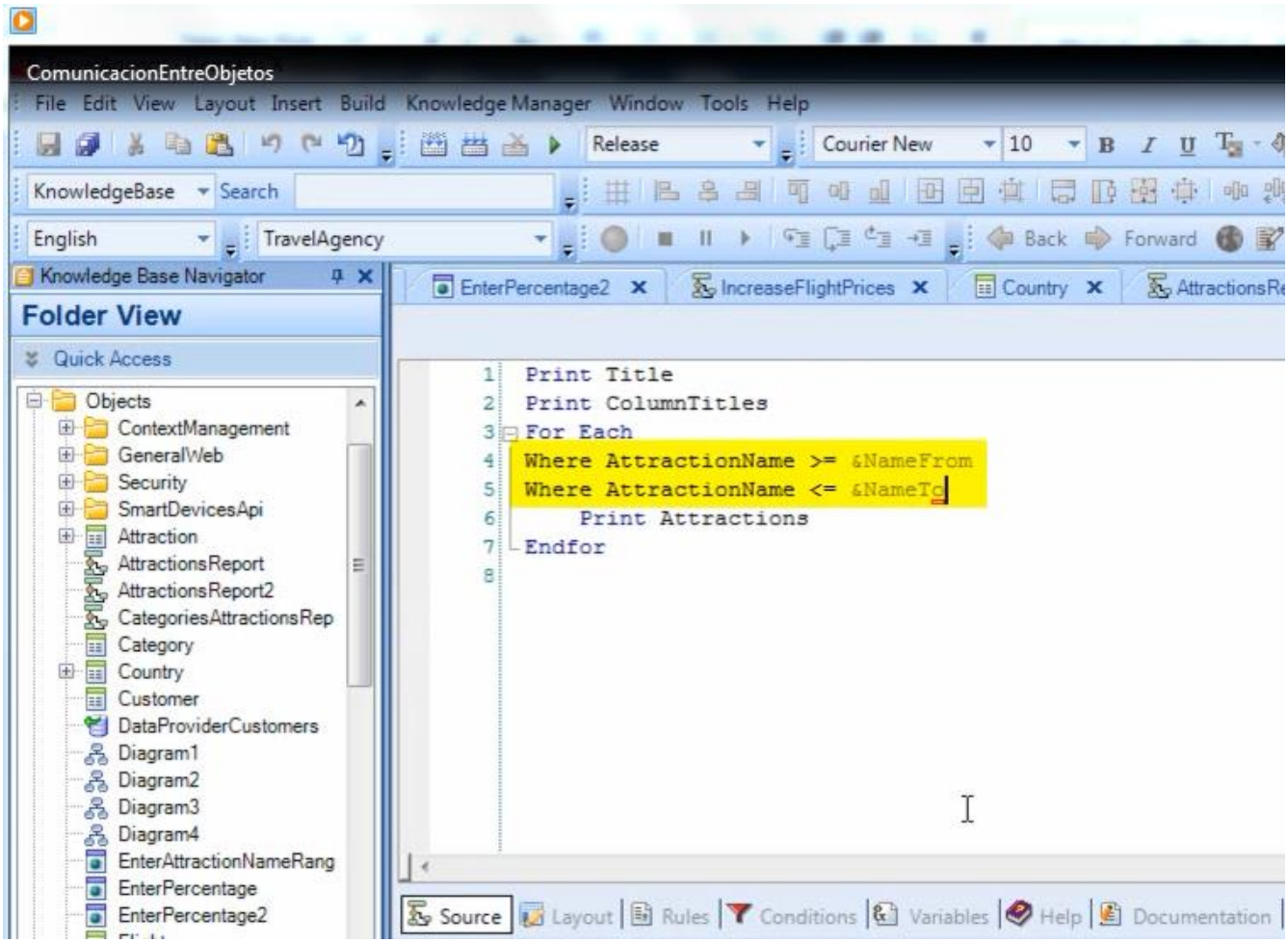
… y así sucesivamente llegan los datos al objeto llamado, en el orden que se enviaron.



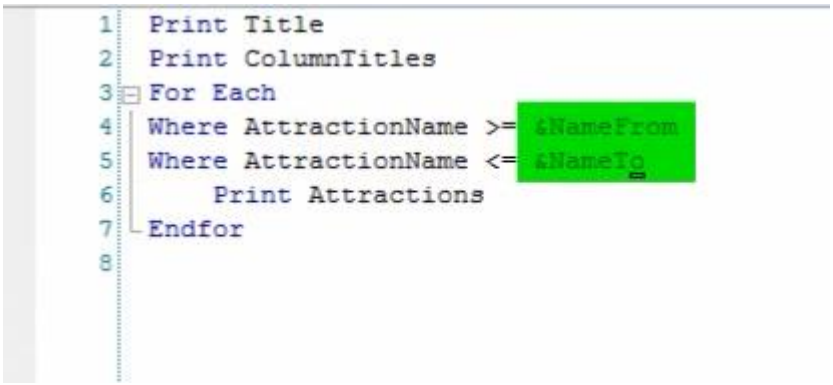
Recalcamos que no importa el nombre de las variables, sino el orden de las mismas. Siempre conviene usar nombres relacionados como hicimos aquí a efectos de entender mejor el código.



Los valores recibidos en las variables los usamos para filtrar en el For Each.



Observemos que usamos los nombres de las variables definidas **en este procedimiento**

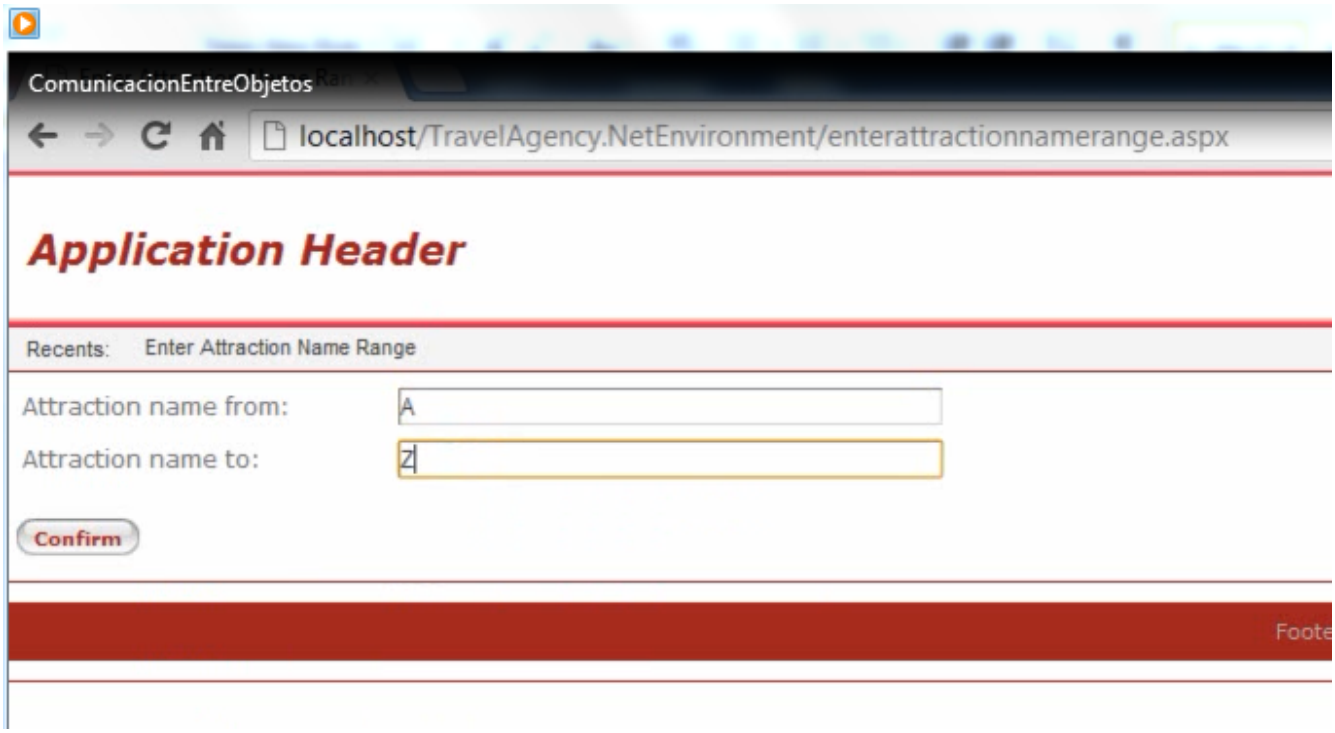


y no los nombres usados en el objeto llamador.

Veamos en funcionamiento todo esto que hemos hecho y explicado. Presionamos F5

Ejecutamos el web panel “EnterAttractionNameRange”.

Queremos ver a las atracciones cuyos nombres comienzan entre A y Z

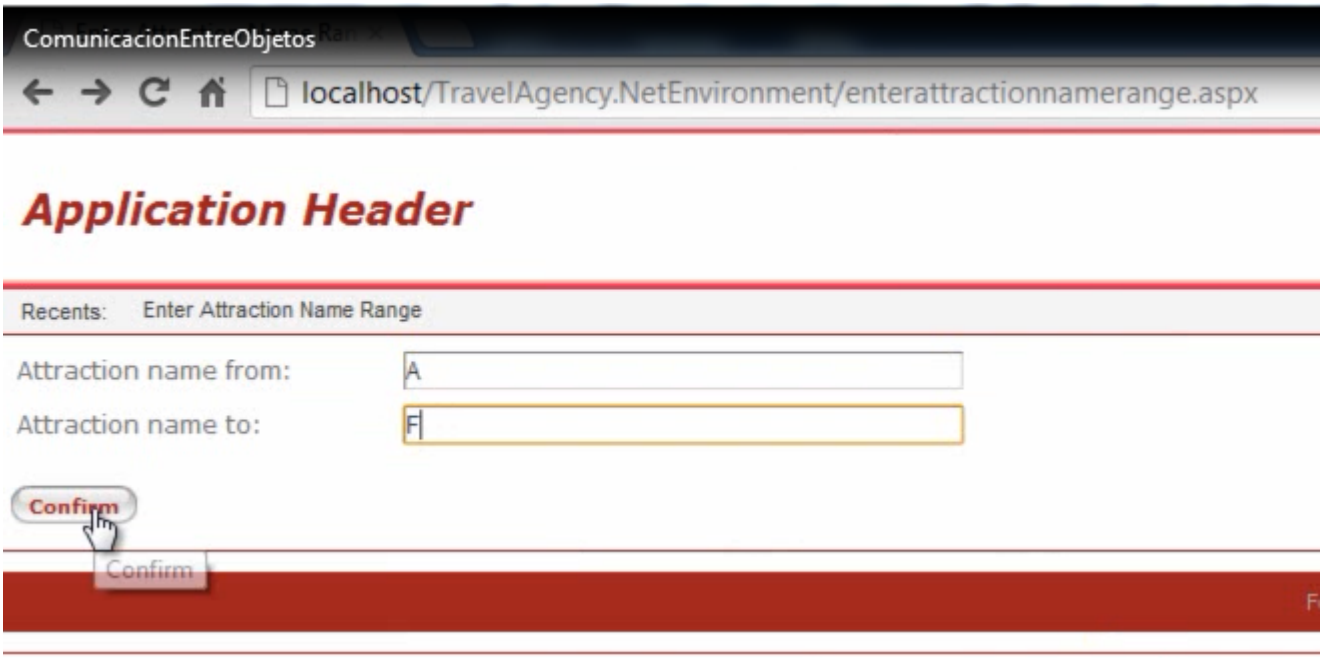


presionamos el botón… y vemos que se listan todas las atracciones.

Attractions Report

Id	Name	Country
1	Louvre Museum	France
2	Great Wall	China
3	Eiffel Tower	France
4	The Christ Redeemer	Brazil
5	The Smithonian Institute	United States
6	Egypt Pyramides	Egypt

Ahora acotamos un poco más el rango. Ponemos entre A y F

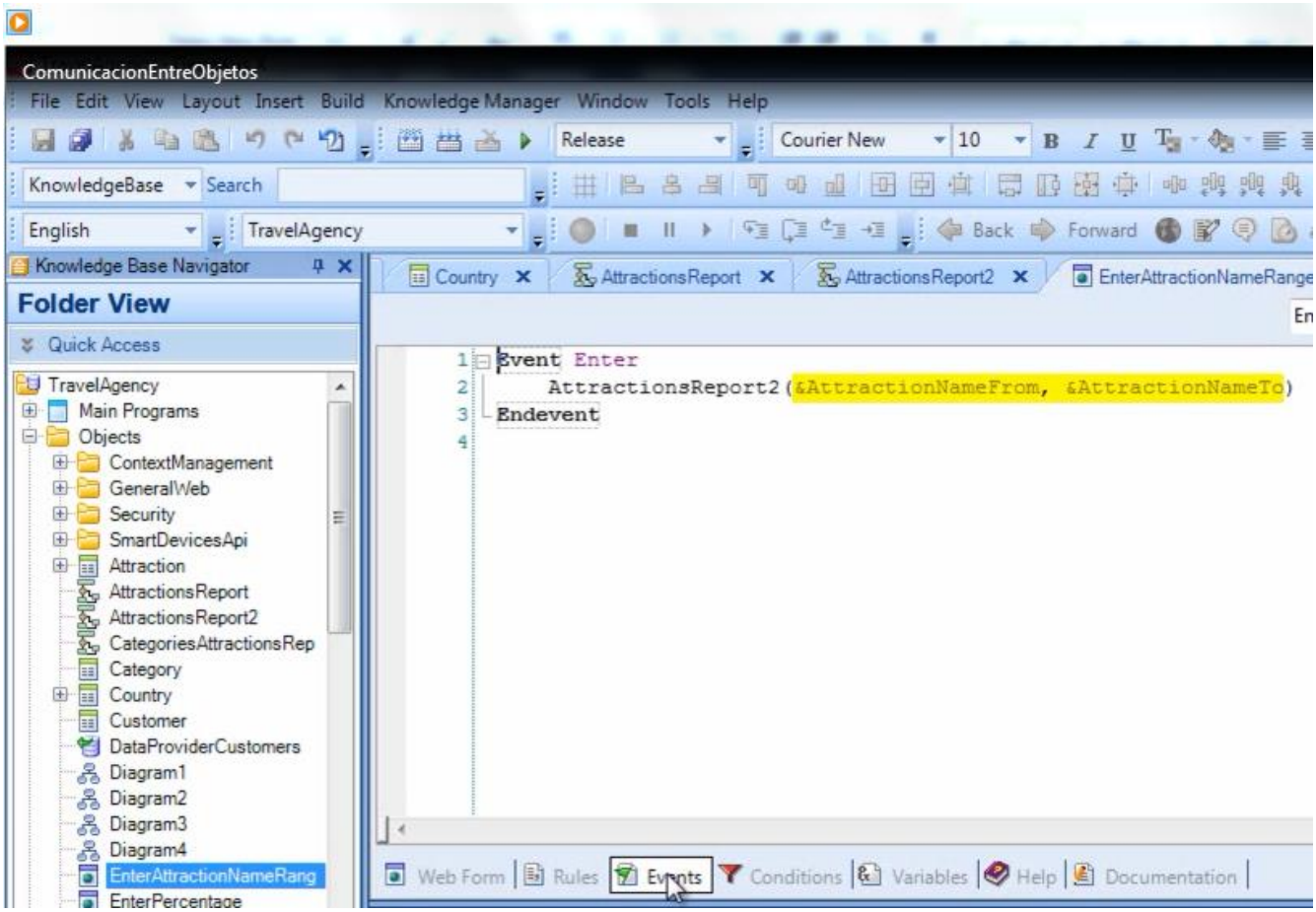


y vemos que salen solamente la Torre Eiffel y las pirámides de Egipto.

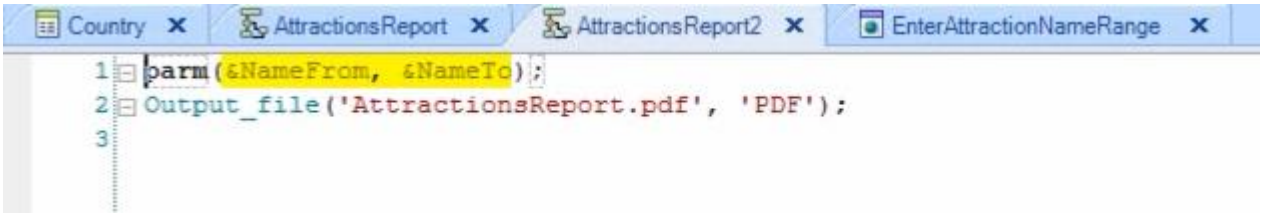
Attractions Report

Id	Name	Country
3	Eiffel Tower	France
6	Egypt Pyramides	Egypt

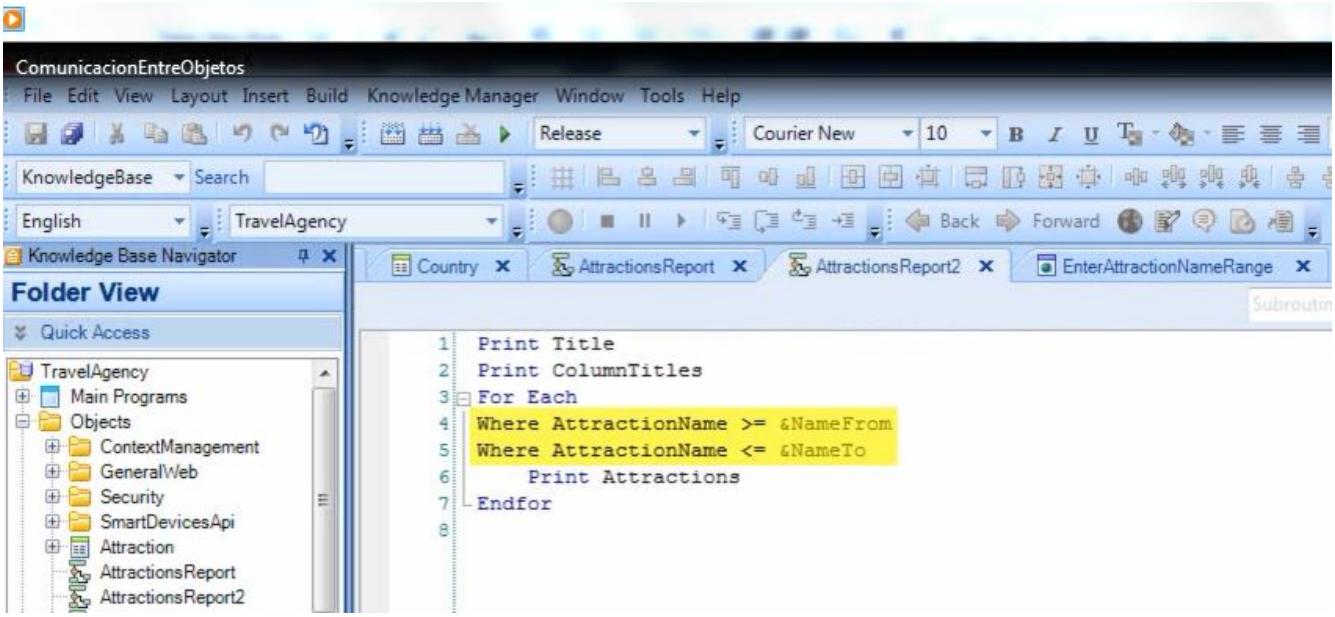
Hemos visto en ejecución, una llamada de un objeto a otro pasándole 2 parámetros al objeto llamado.



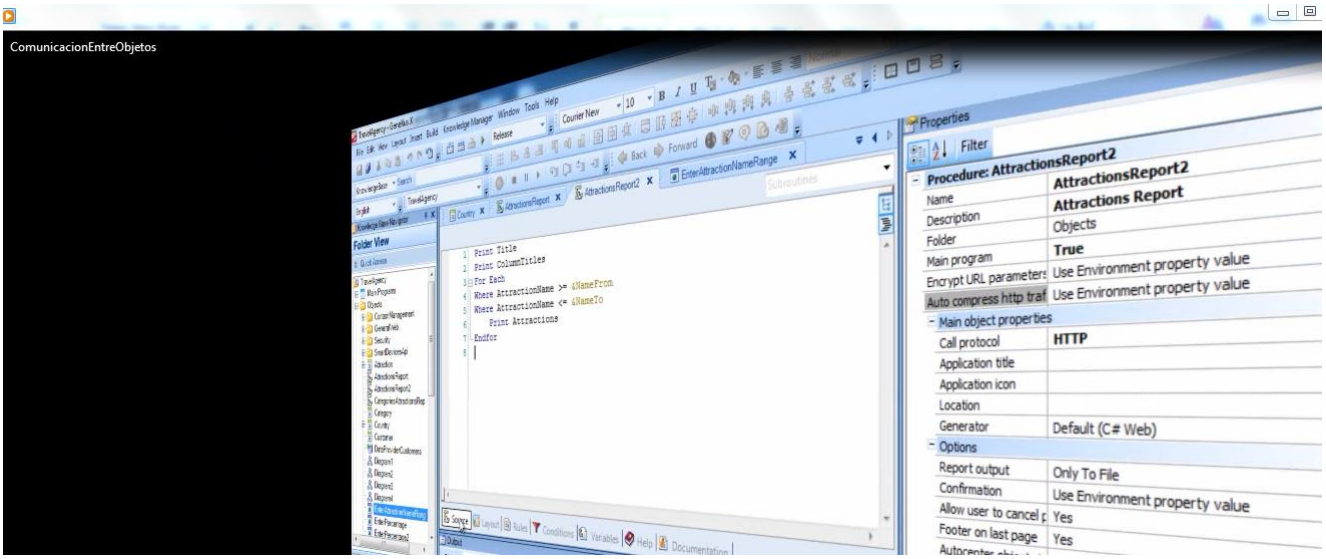
Los datos recibidos

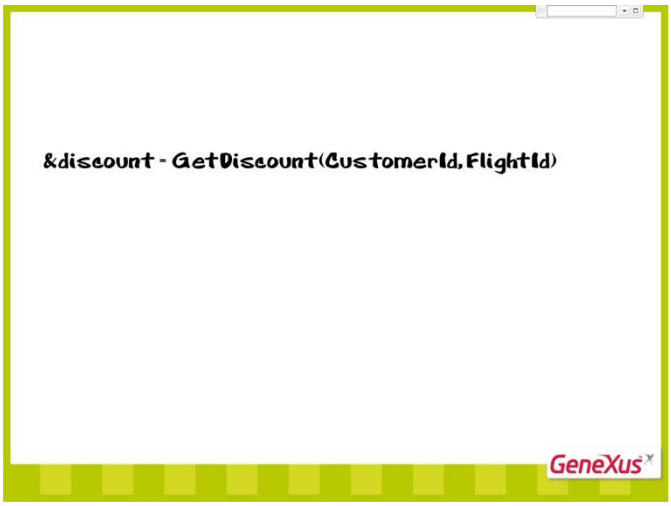


se usaron para filtrar en el comando For Each



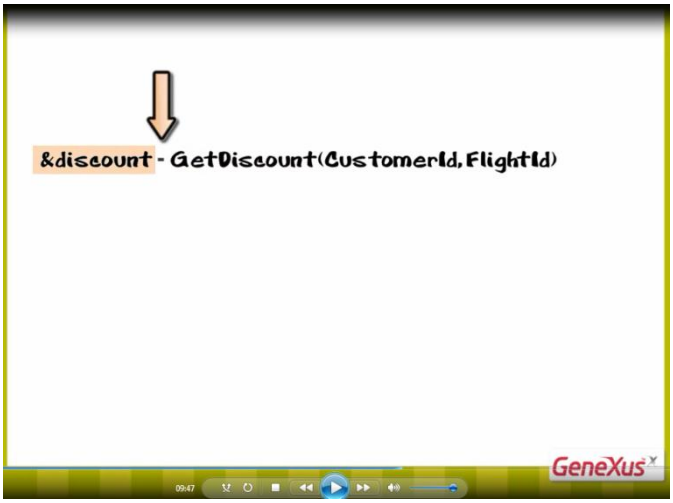
Pasaremos ahora a conocer, otra manera posible de llamar, en particular, a objetos procedimientos o data providers.



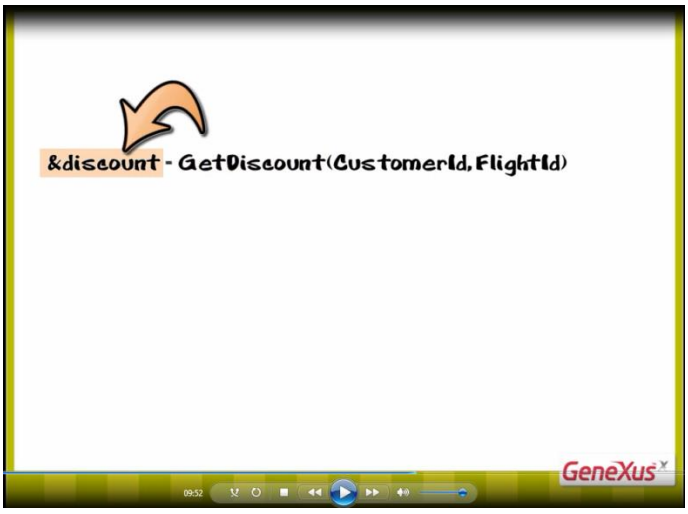


En este ejemplo, estamos llamando a un procedimiento de nombre GetDiscount.

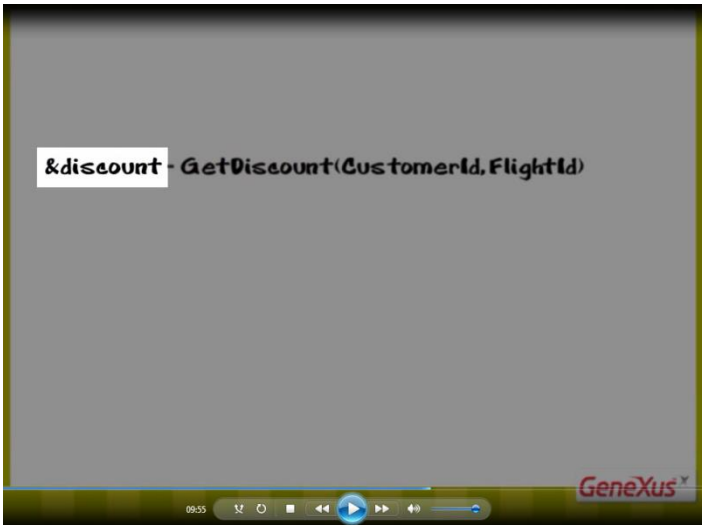
Observemos que delante de la llamada al procedimiento hay una variable y un signo de igual.



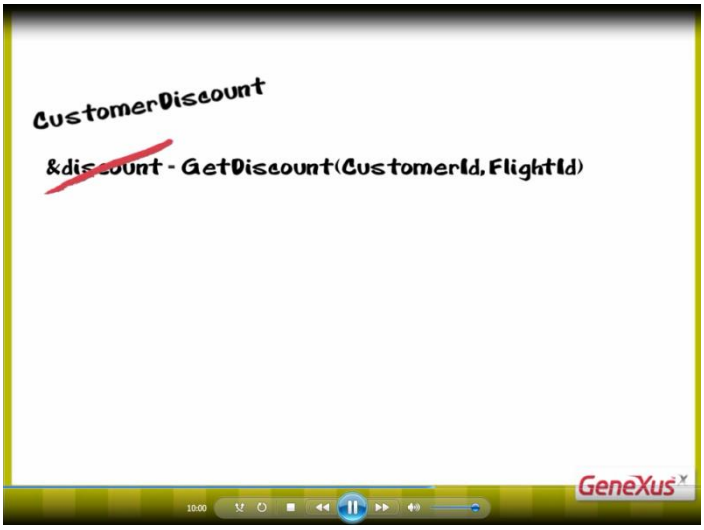
Esto se debe a que el objeto llamado **devuelve un valor**.



A la izquierda del signo de igual

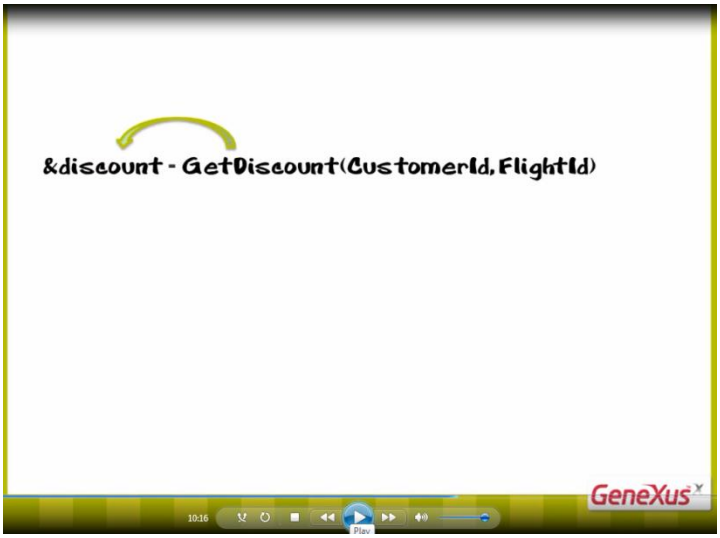


puede haber una variable que reciba dicho valor o dependiendo del objeto y sección del mismo, podría haber un atributo recibiendo el valor.



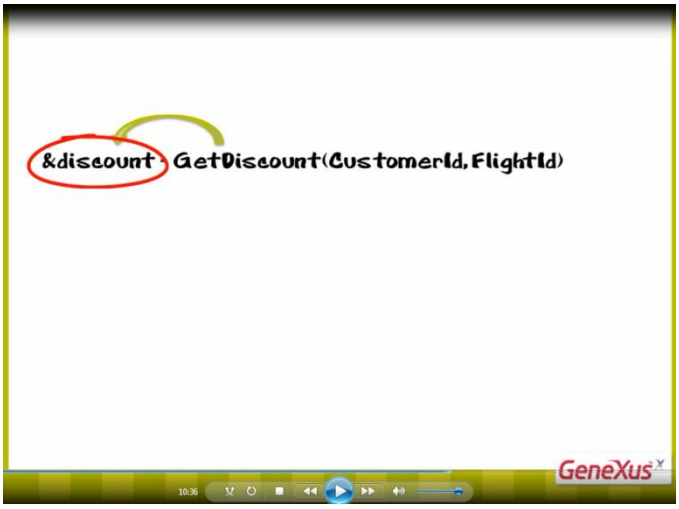
En este caso, no nos interesa entrar en detalles de lo que hace el procedimiento GetDiscount.

Por el nombre del procedimiento, el hecho de que devuelve un valor y el nombre de la variable que recibe el valor que devuelve el procedimiento



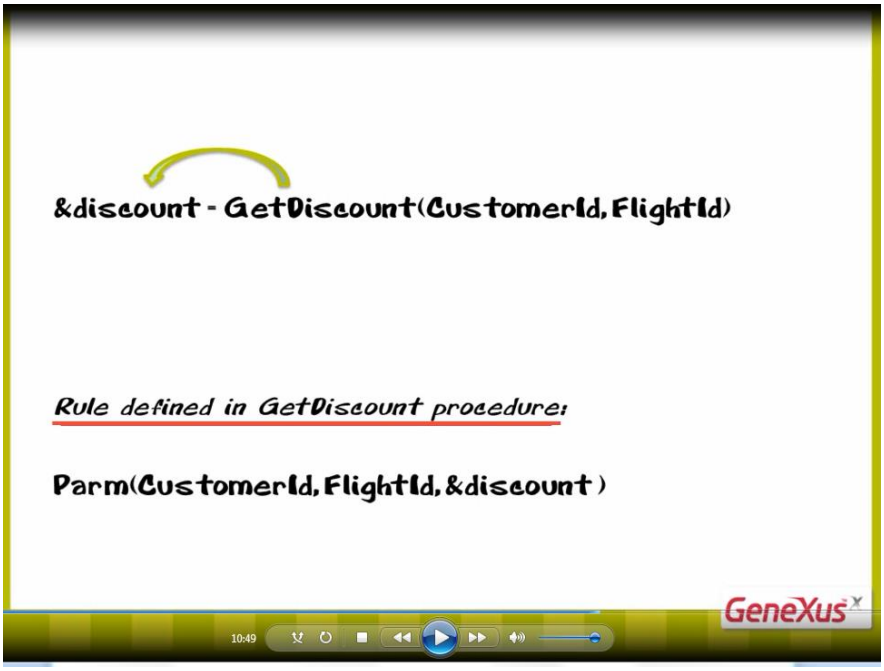
podemos deducir que:

- el procedimiento nos devolverá un descuento…
- que se envían al procedimiento 2 datos: cierto identificador de cliente y cierto identificador de vuelo…
- y que el procedimiento utilizará dichos datos, evaluará y calculará lo que corresponda y retornará un descuento.
-

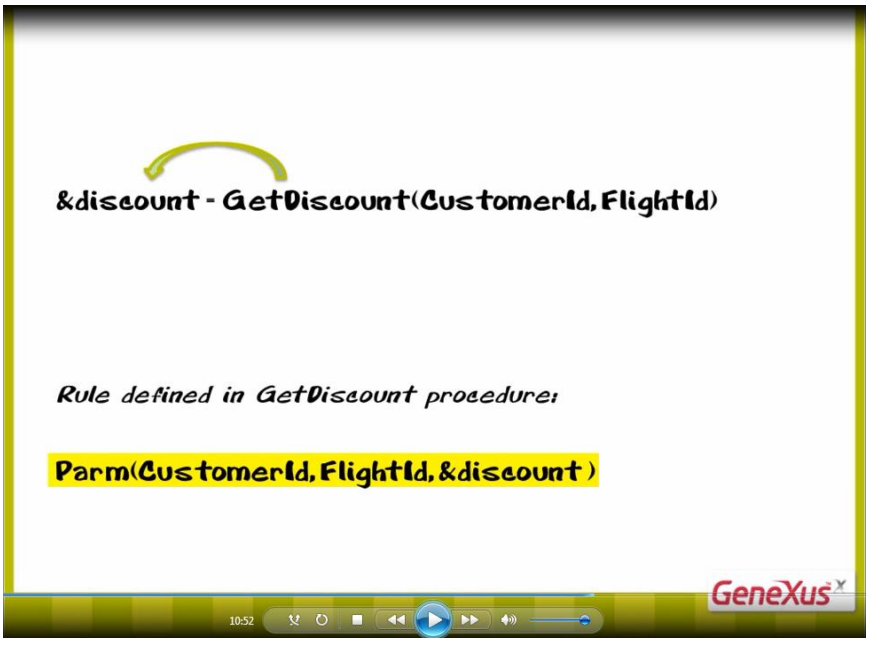


Ahora, es importante que veamos cómo se declara en el objeto llamado, **la regla parm**, cuando en la sintaxis de la llamada, el objeto devuelve un valor.

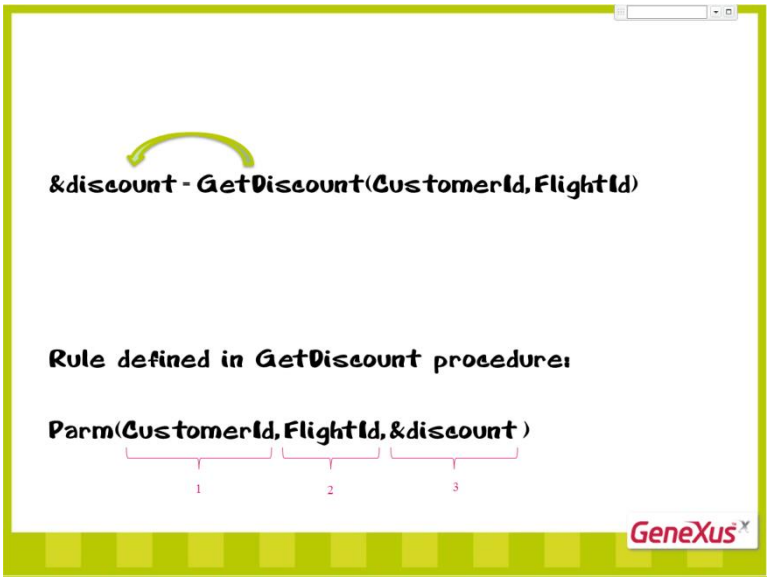
En la sección de reglas del procedimiento GetDiscount



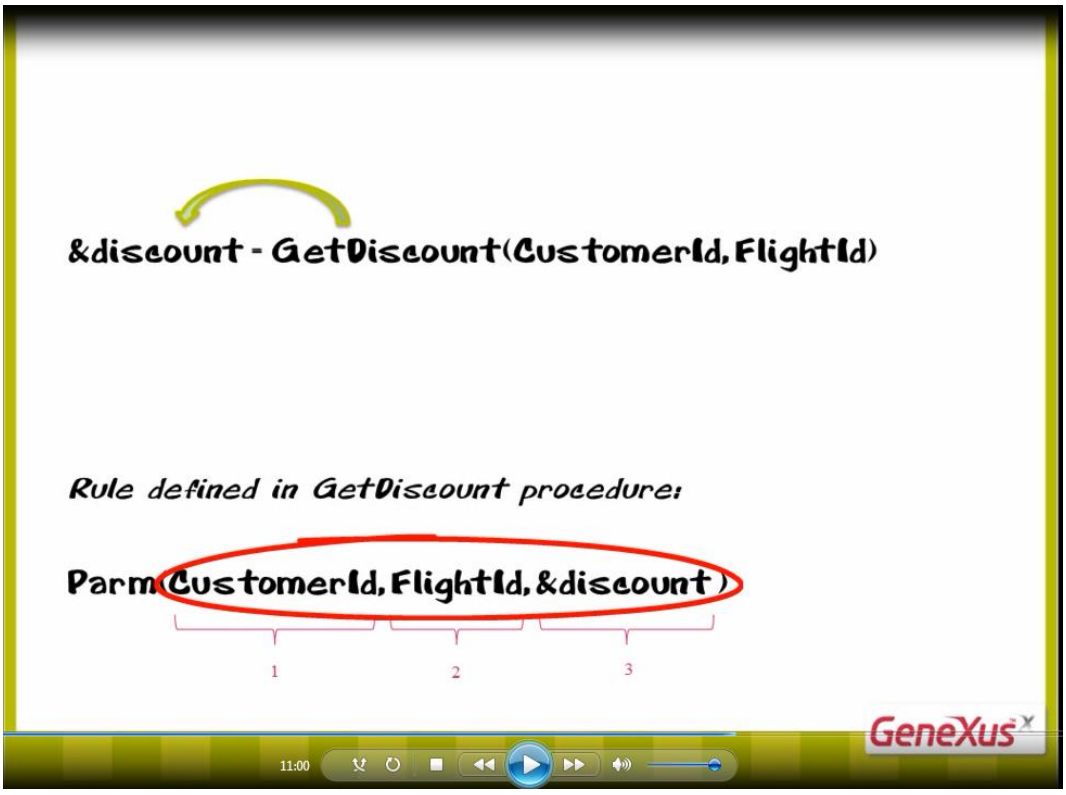
declaramos esta regla parm



con 3 parámetros.

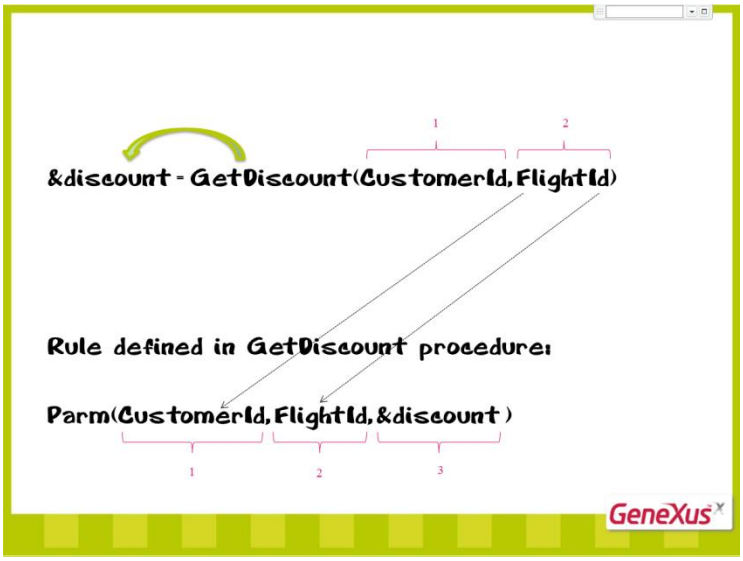


El término “parámetros” lo usamos para referirnos a datos que se envían y reciben entre 2 objetos que uno llama al otro.



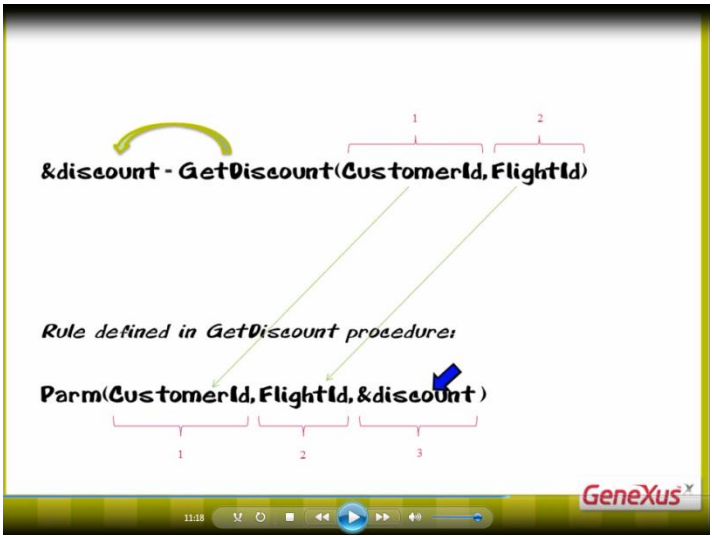
Así podemos hablar en forma genérica independientemente de si se envían o reciben variables, atributos o valores fijos.

Veamos ahora como es el funcionamiento: Los dos parámetros enviados,

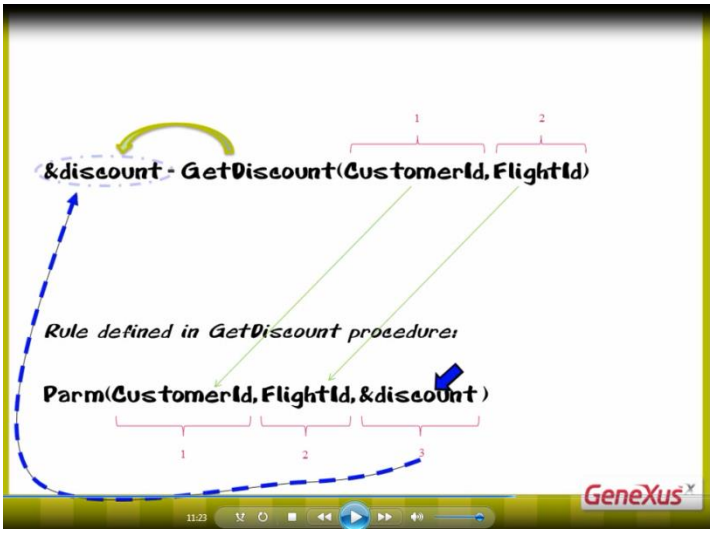


se reciben en orden.

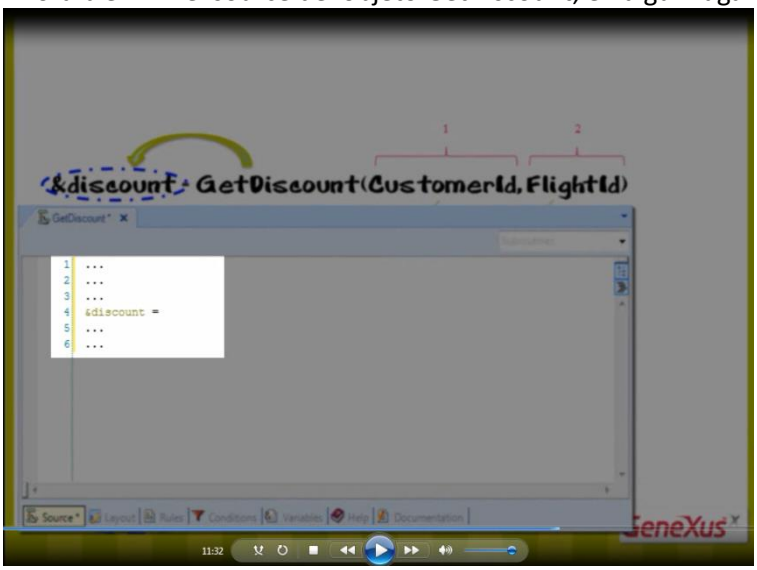
Y el tercer parámetro que definimos en la regla parm



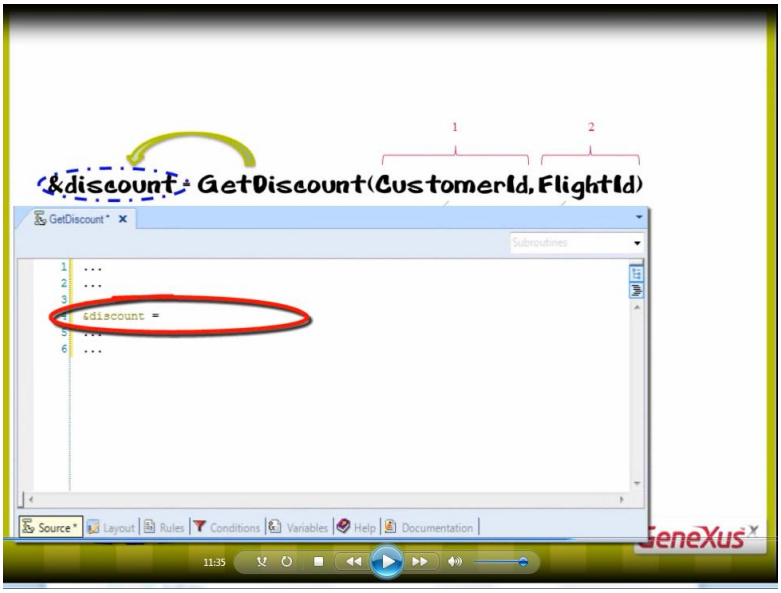
corresponde al que almacena el valor retornado, en la variable &discount de la invocación.



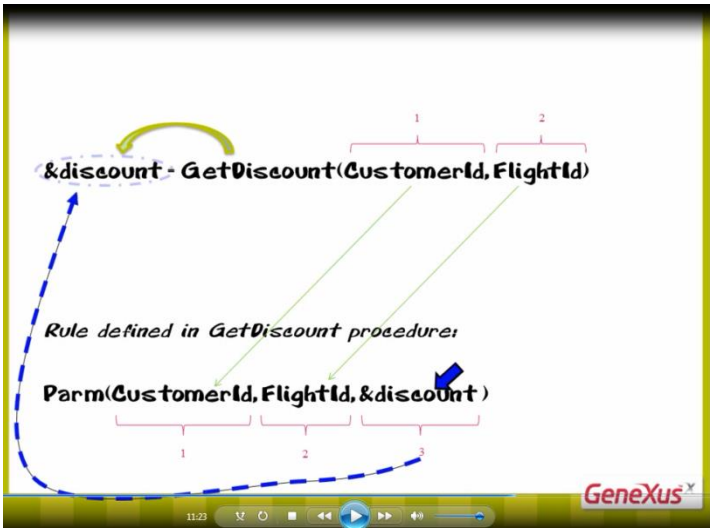
Ahora bien: En el source del objeto GetDiscount, en algún lugar del conjunto de instrucciones



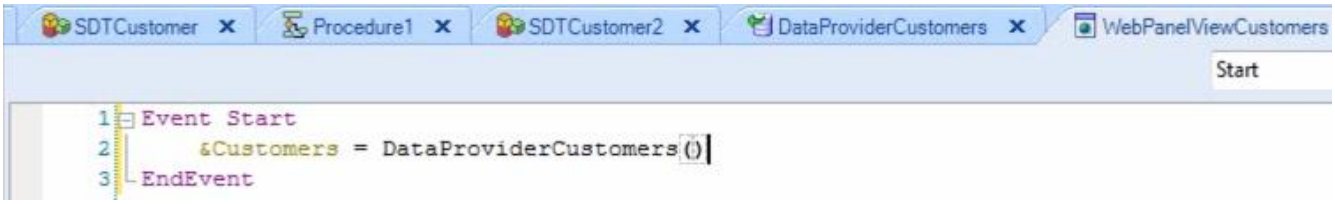
hay que asignarle valor a la variable &discount,



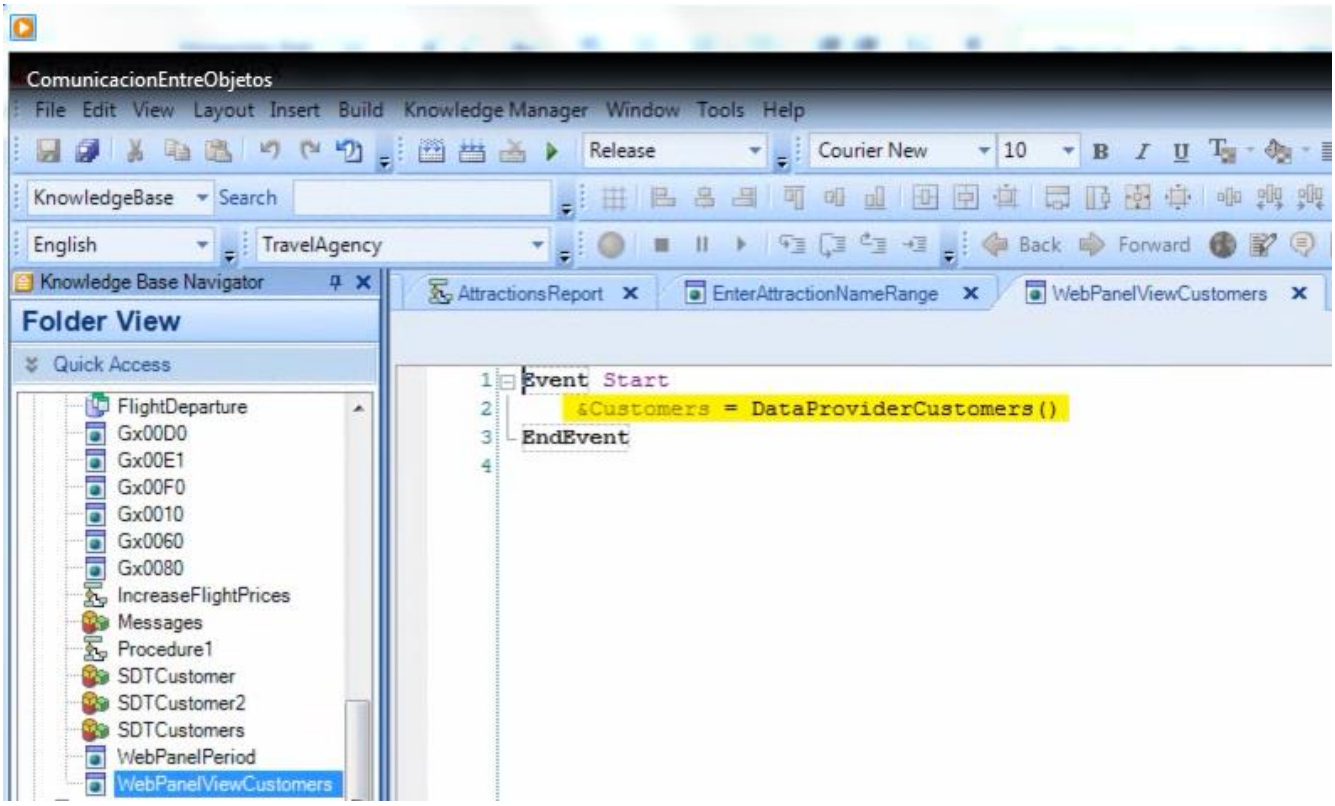
para que dicho valor pueda ser devuelto y asignado a la variable que se encuentra a la izquierda del signo de igual.



Anteriormente ya hemos llamado a un objeto Data Provider de esta manera.

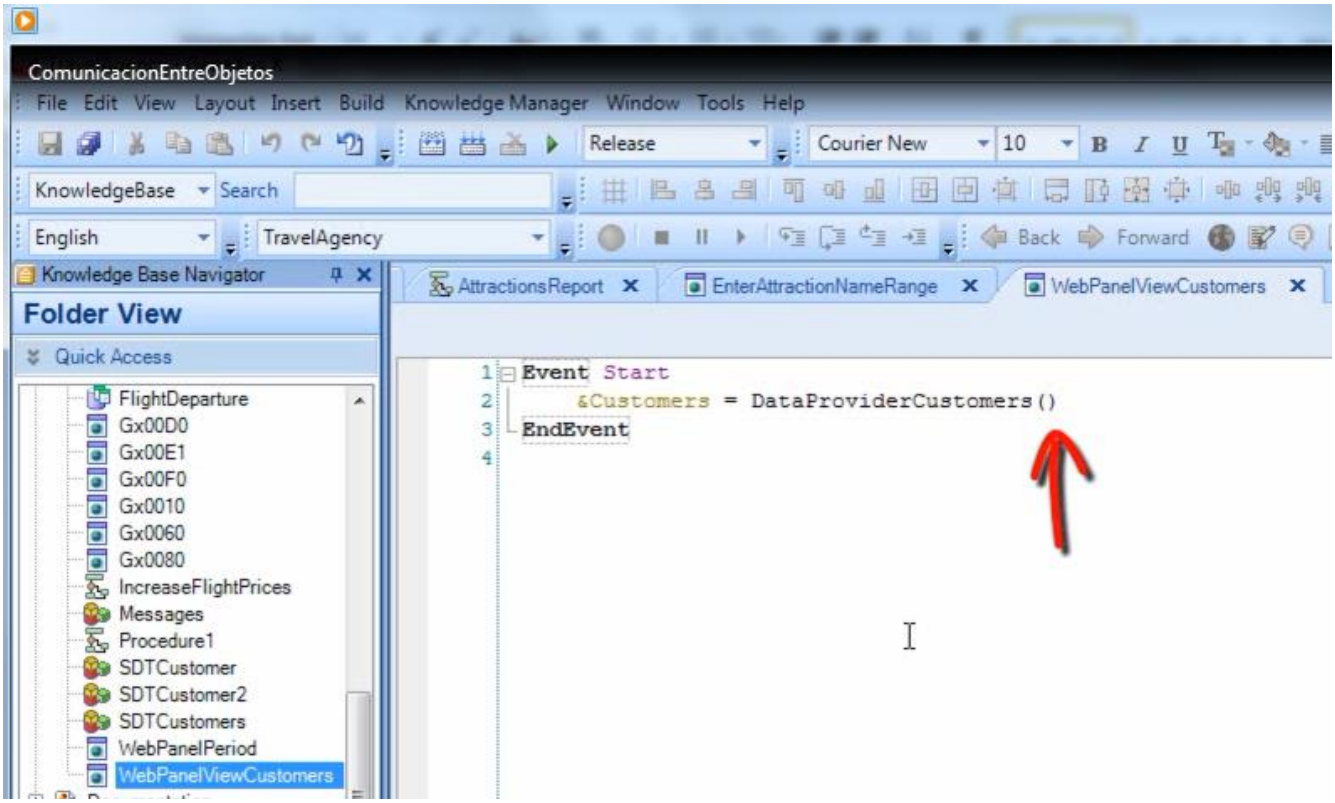


Esta sentencia



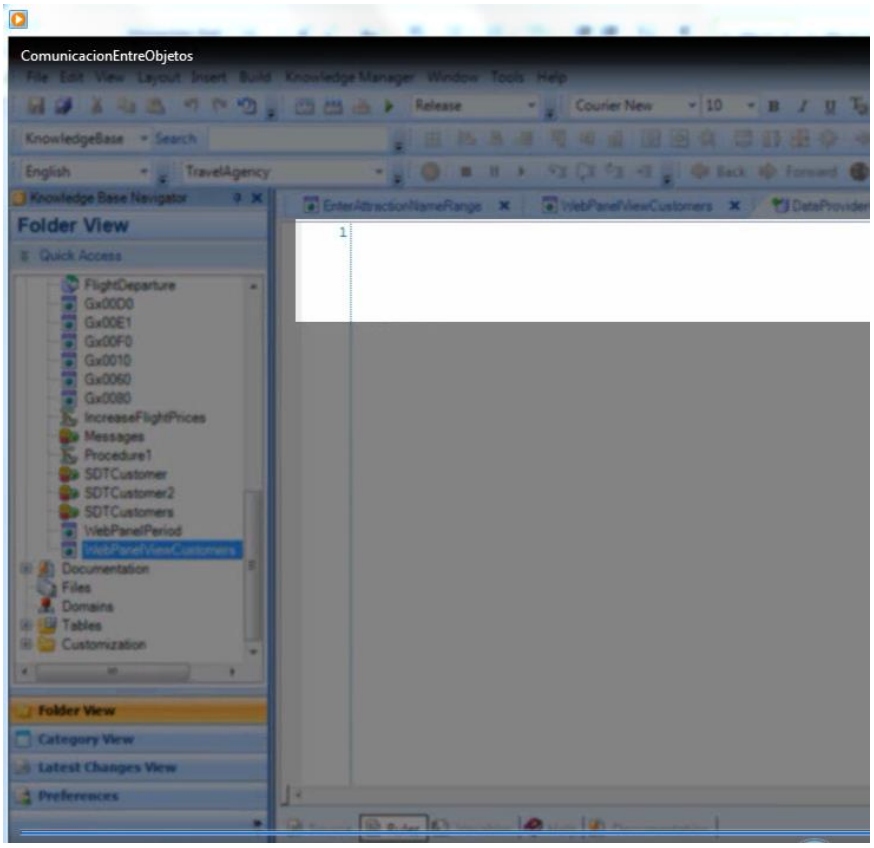
le está asignando a la variable &Customers, definida como una colección de clientes, lo que devuelve el DataProviderCustomers.

En este caso al objeto llamado no se le envían parámetros

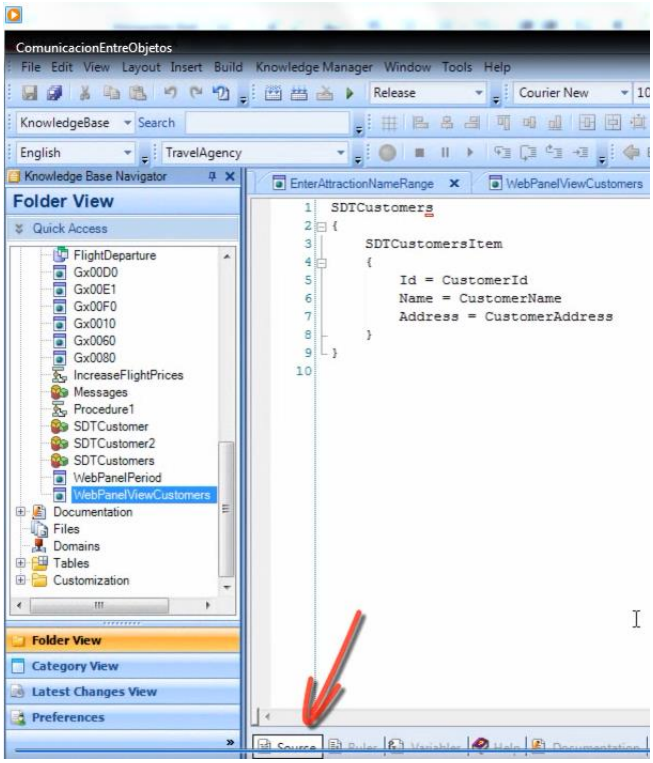


ya que los paréntesis no contienen nada adentro.

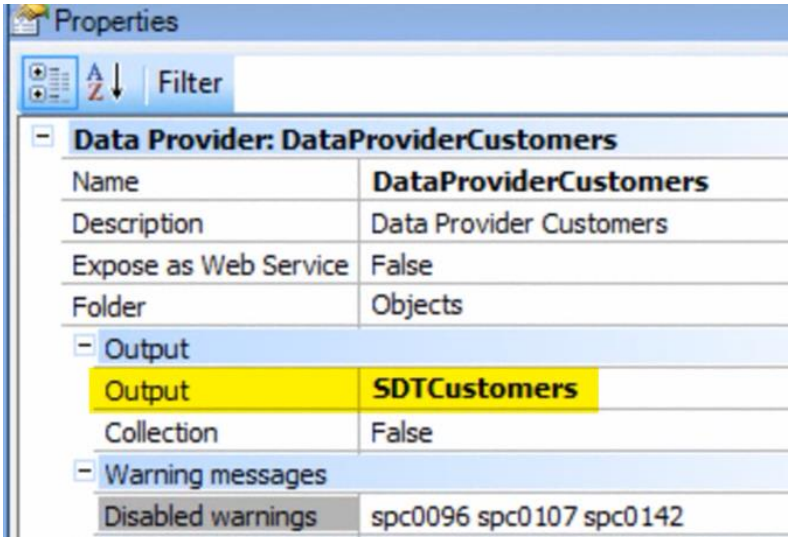
Por lo tanto el Data Provider llamado no tiene declarada una regla parm.



Recordemos que cuando arrastramos al source de este Data Provider



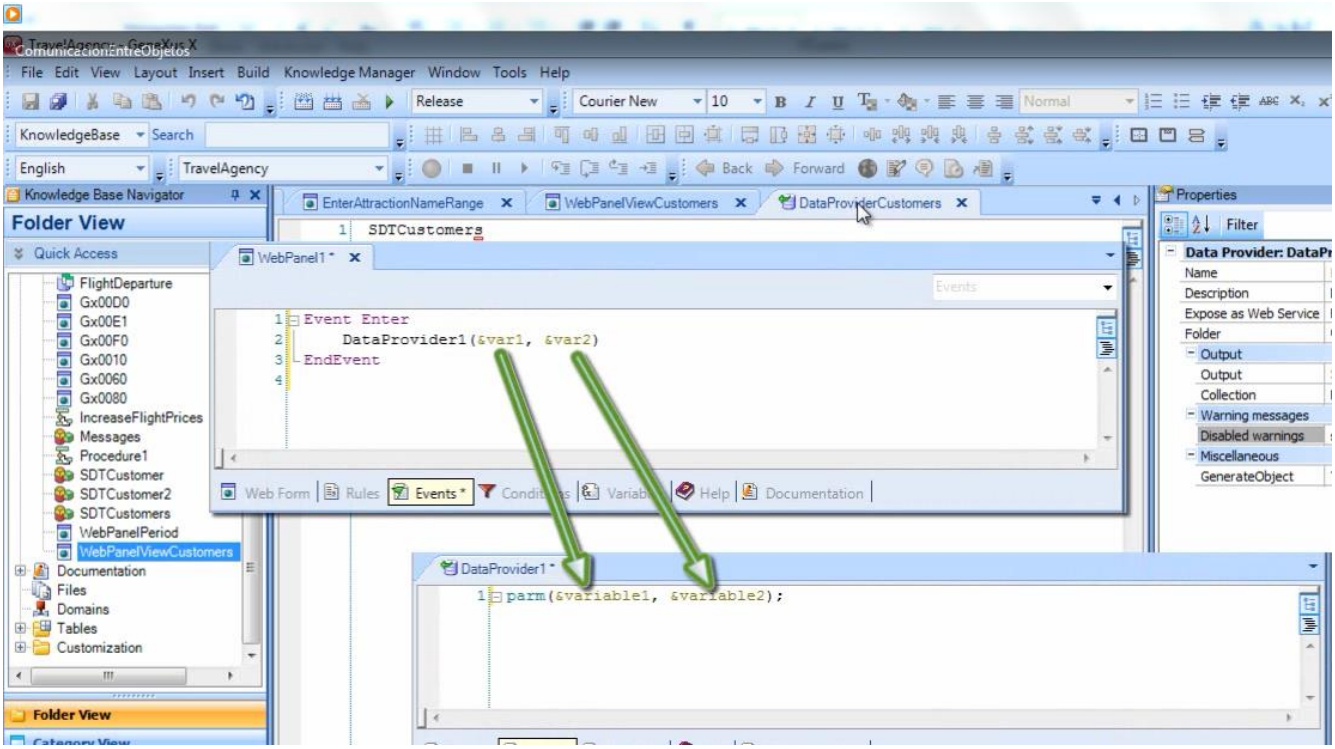
el SDT que queríamos cargar, automáticamente se completó la propiedad Output del Data Provider



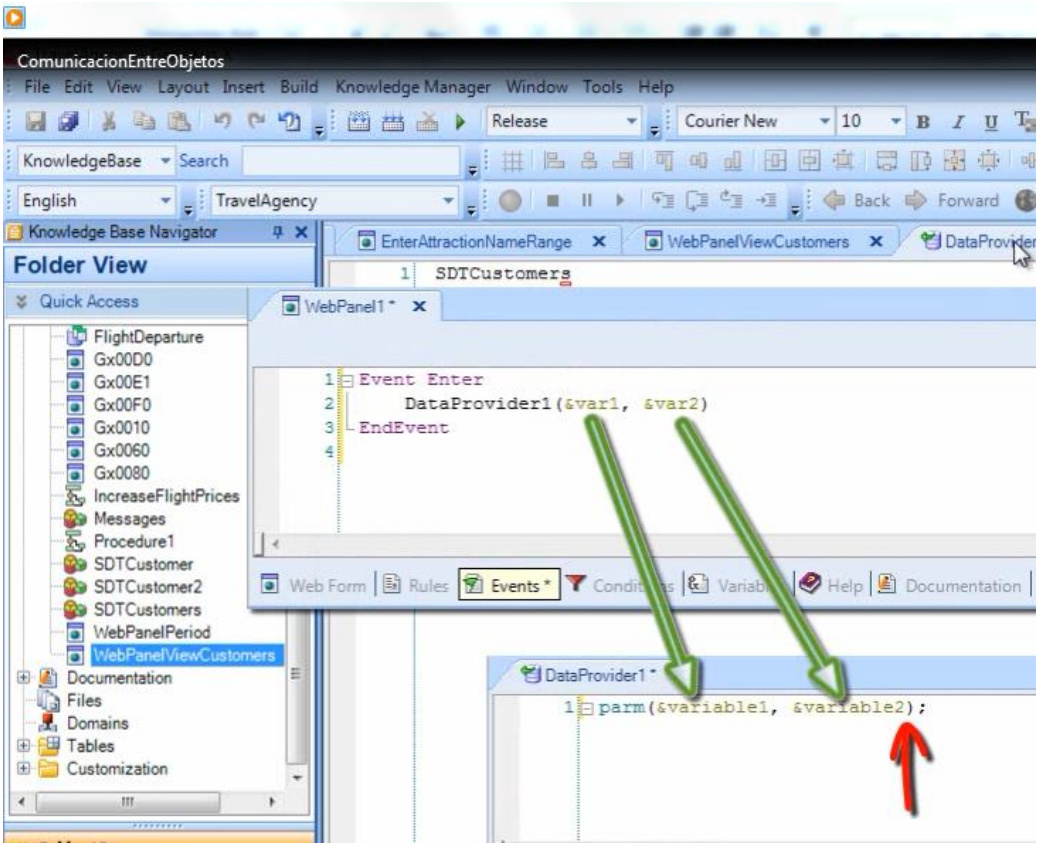
con el nombre del SDT que arrastramos.

Tal como el nombre de la propiedad Output lo describe, quedó configurado así en el objeto, lo que el mismo retorna, o en otras palabras: la salida del Data Provider.

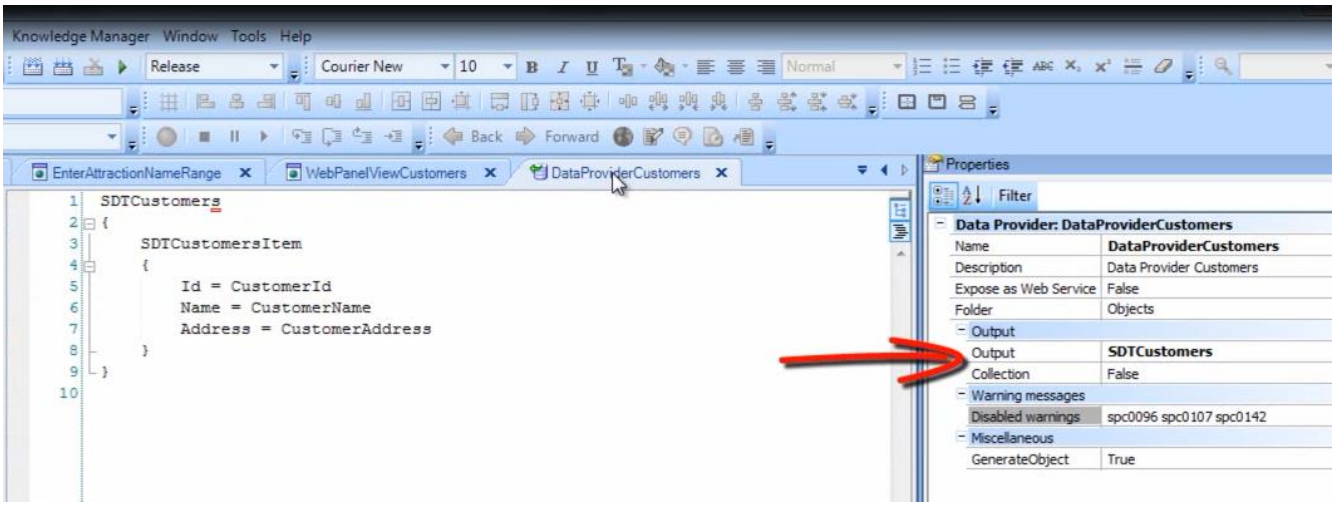
Por este motivo, en objetos Data Provider hay que recibir en la regla parm, la misma cantidad de parámetros que se envían en la llamada

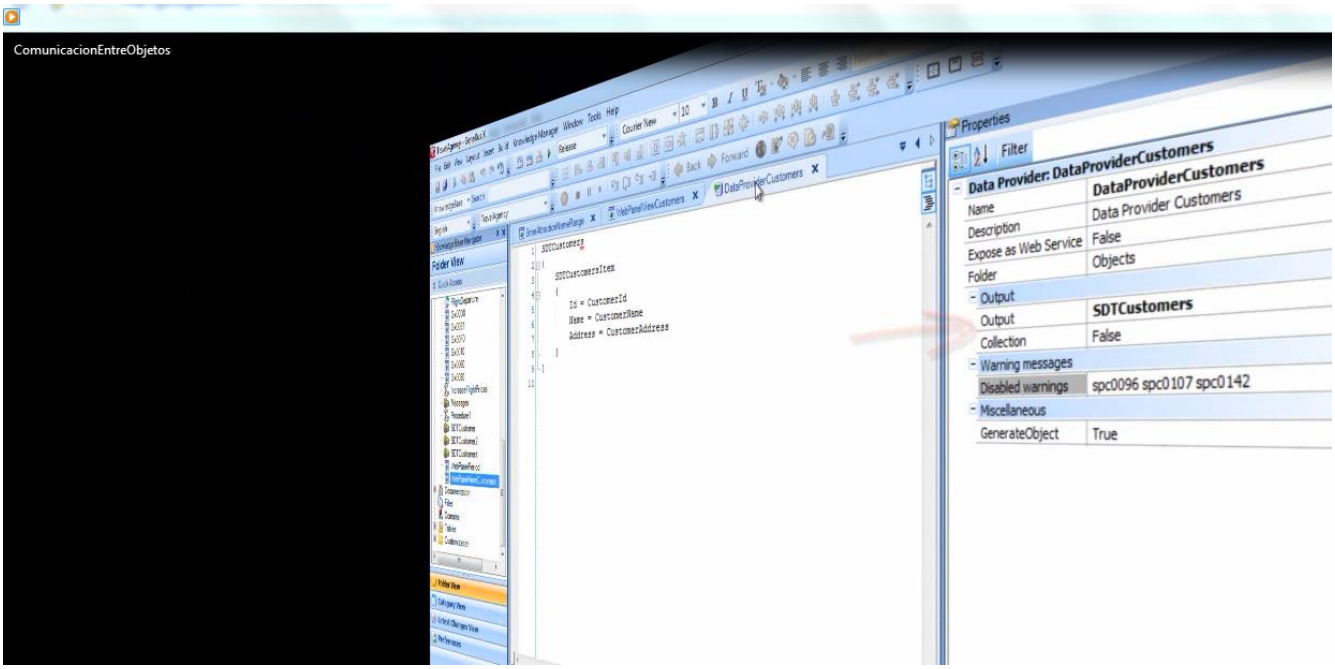


sin agregar un parámetro extra de retorno al final

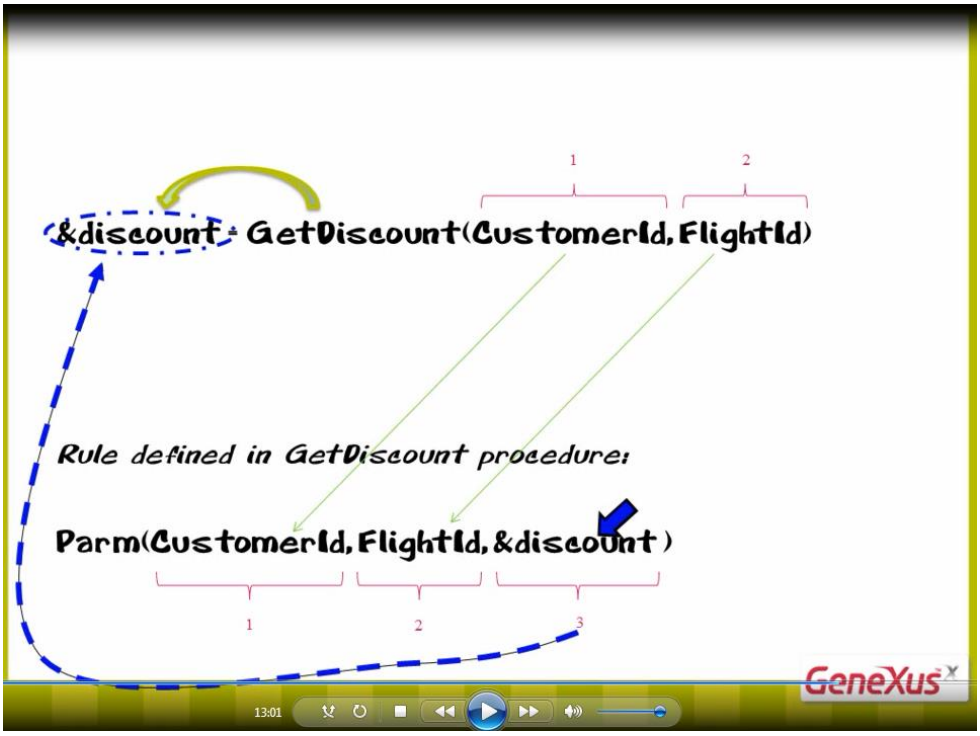


ya que la salida queda definida por la propiedad Output del data provider



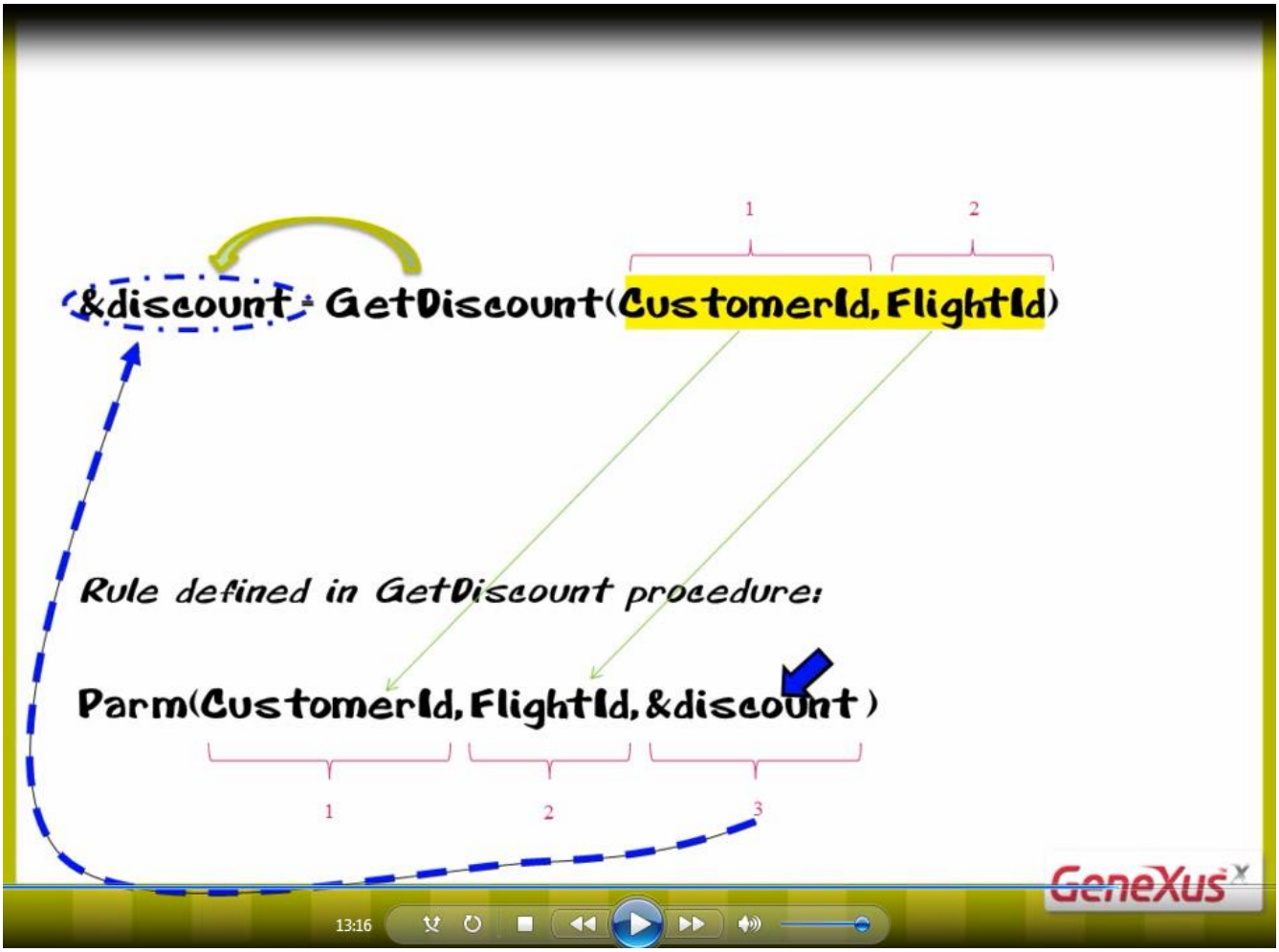


Para finalizar, vamos a observar una última cosa.



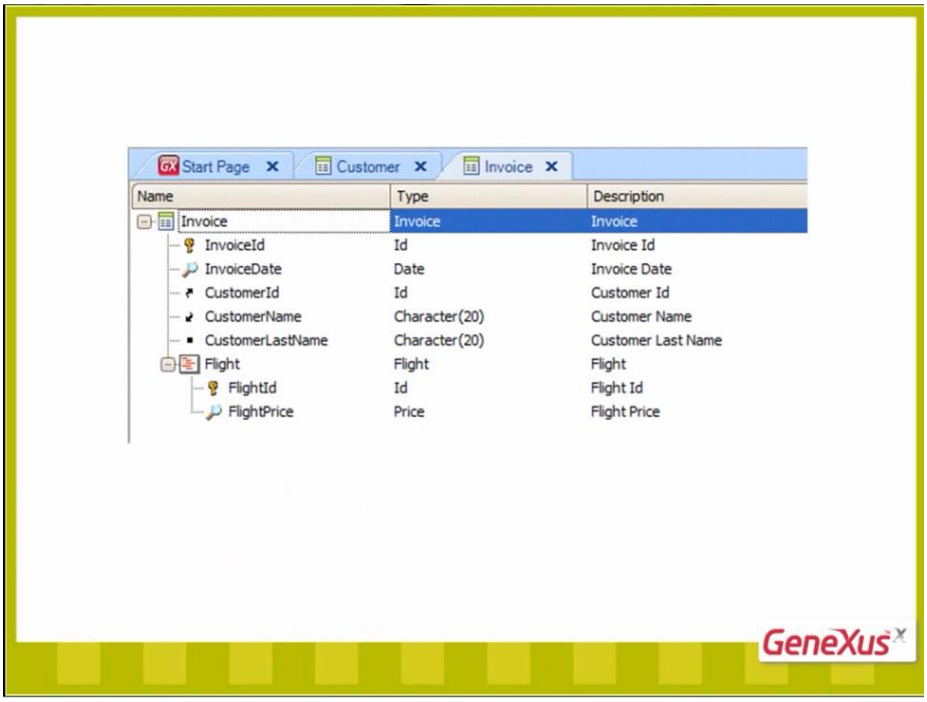
En este ejemplo, algunos parámetros son variables y otros, atributos. ¿Cómo elegimos qué mandar y qué recibir?

A la hora de enviar datos a un objeto que se llama, no se generan dudas: Si el dato se tiene en un atributo

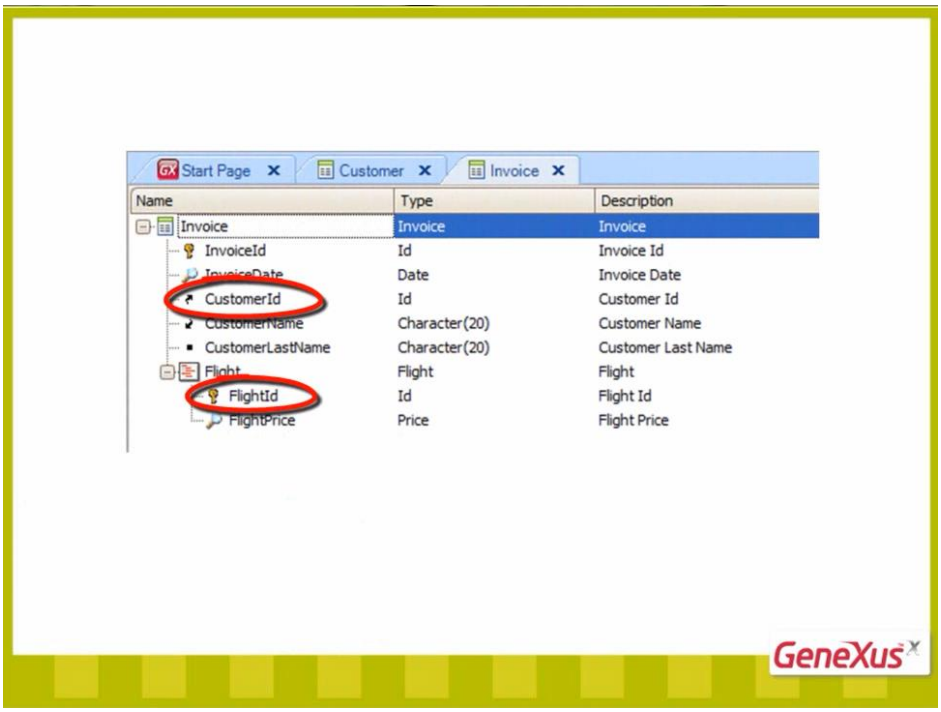


se incluye como parámetro al atributo; y si se encuentra en una variable, se utiliza la variable.

Un ejemplo de cuando los datos los tenemos en atributos, sería el caso de que la llamada al procedimiento GetDiscount estuviera definida en una regla de una transacción

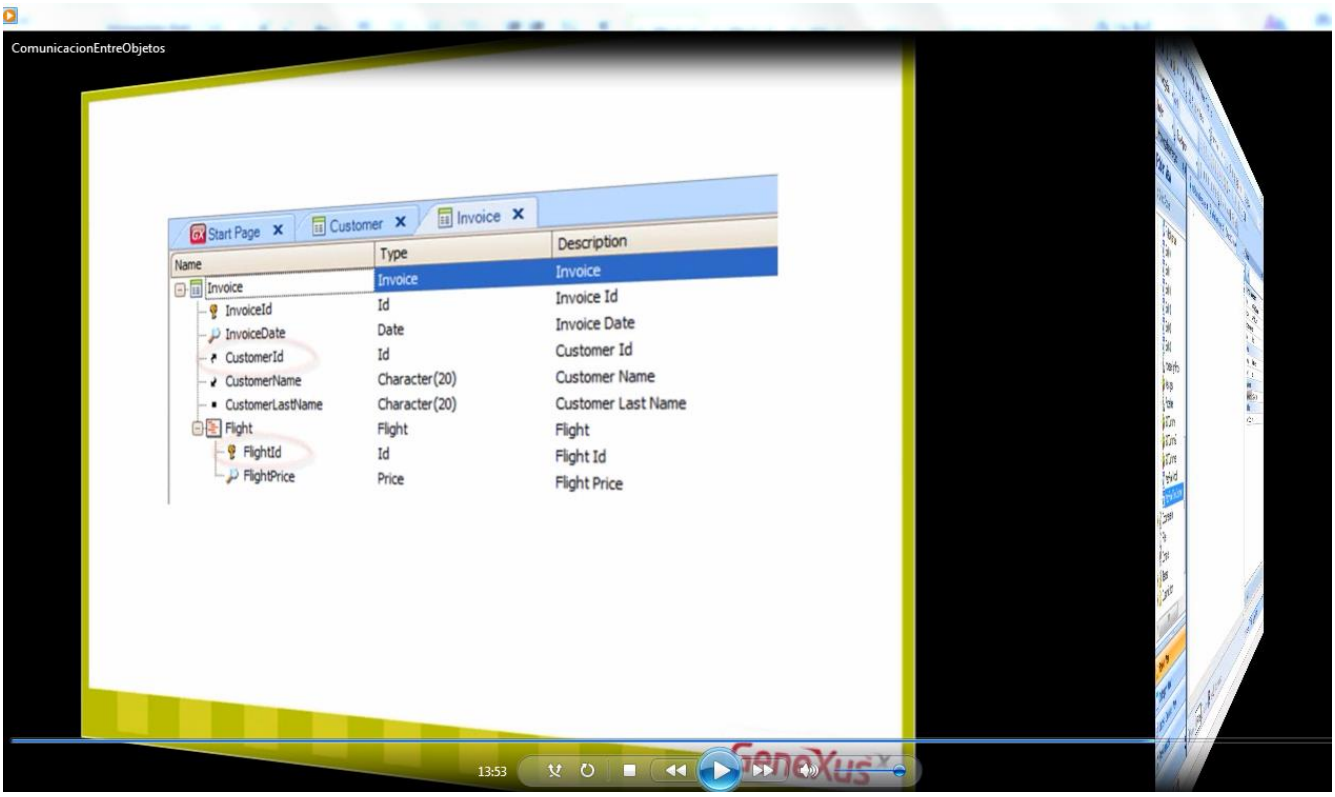


en la cual el usuario ha ingresado un identificador de cliente y un identificador de vuelo

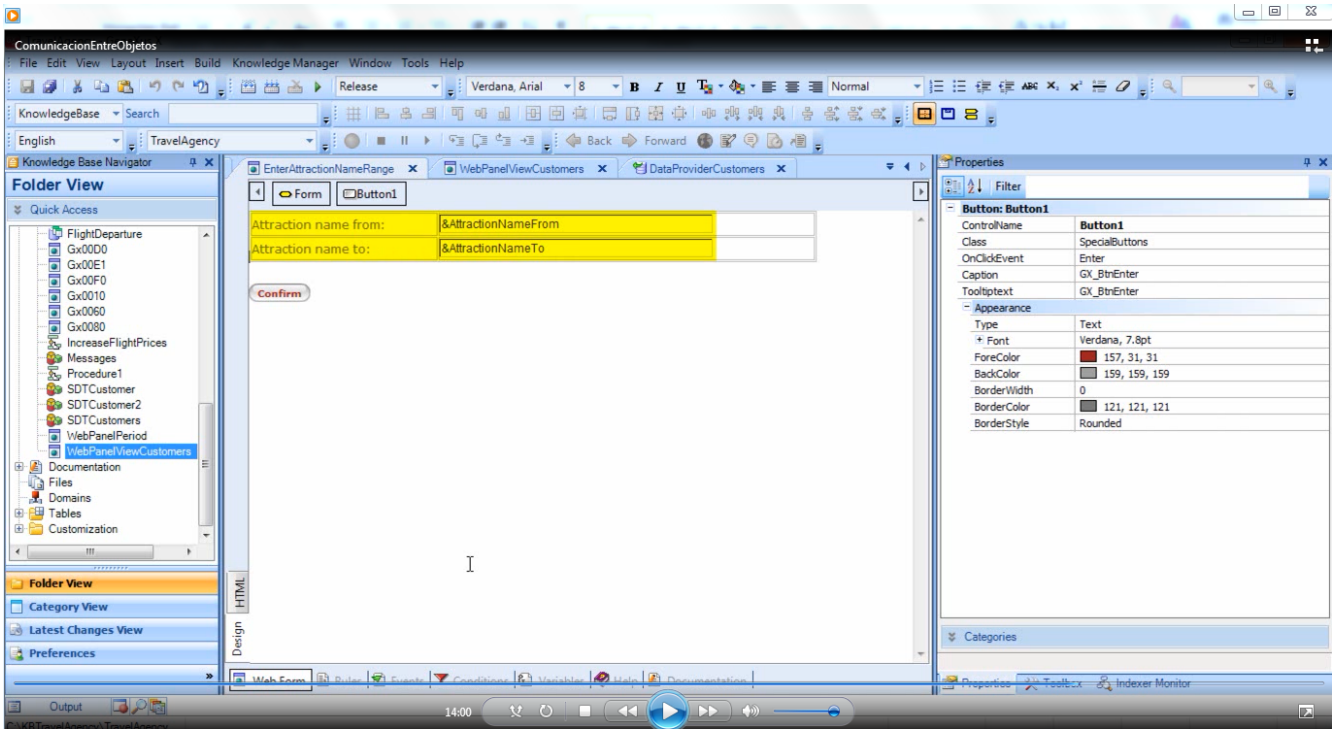


y se tienen los valores a ser enviados al procedimiento, en atributos.

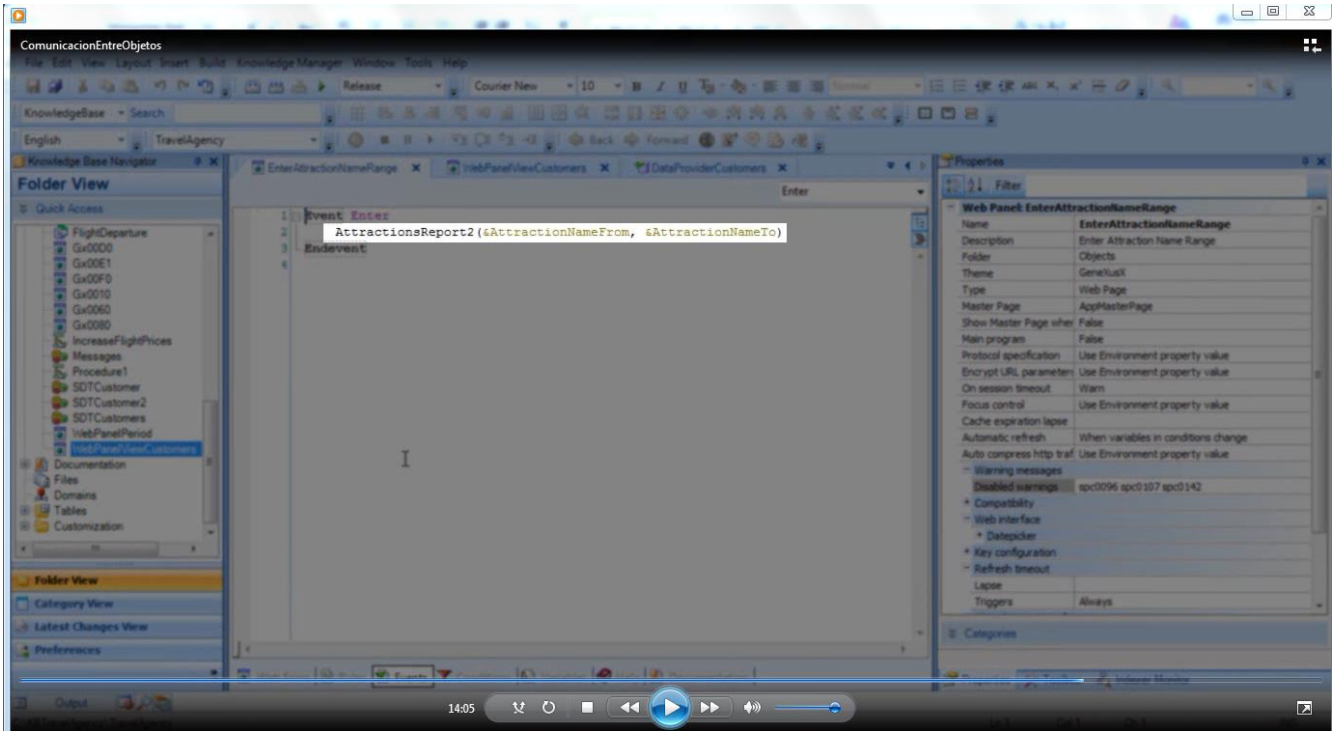
Otro ejemplo de uso de atributos, podría ser si la misma llamada estuviera definida dentro de un For each, el cual navega cierta tabla base, tiene alcance a la tabla extendida de la misma y dispone de esos atributos.



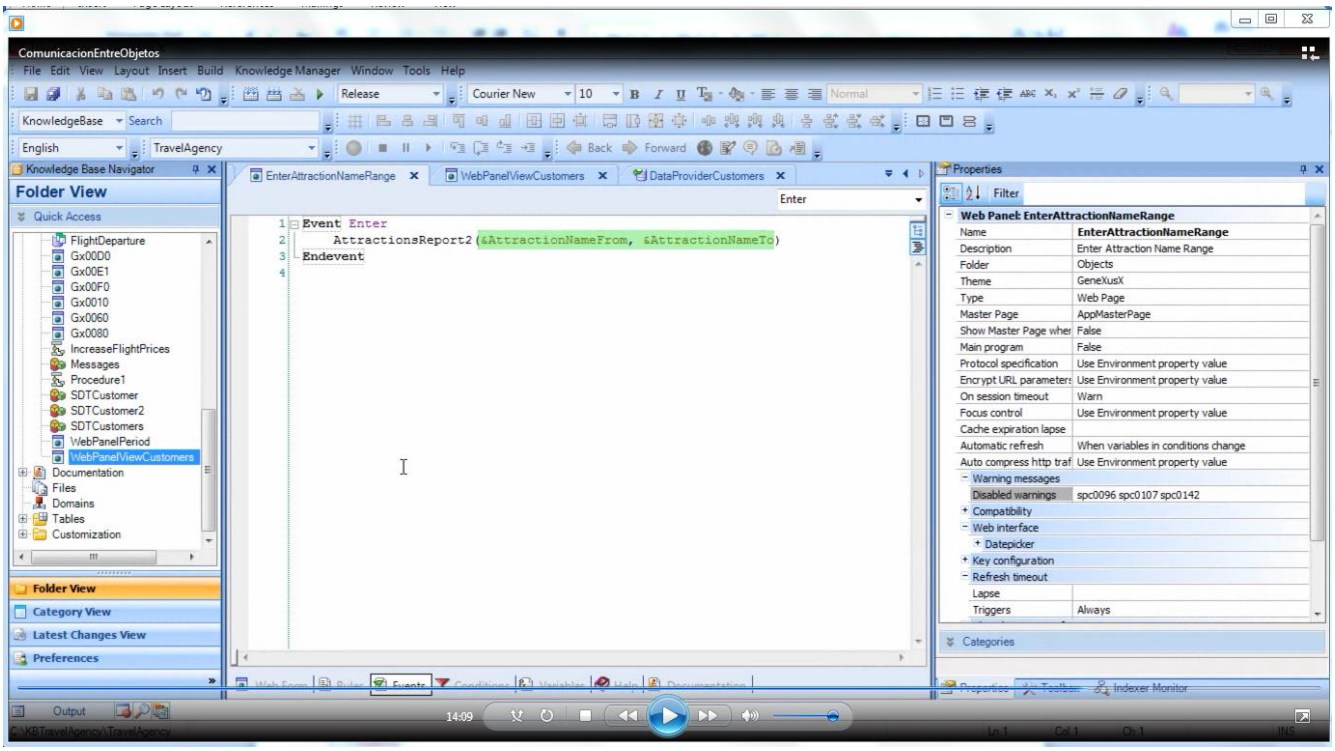
Un ejemplo de cuando los datos los tuvimos en variables, fue en el web panel EnterAttractionNameRange



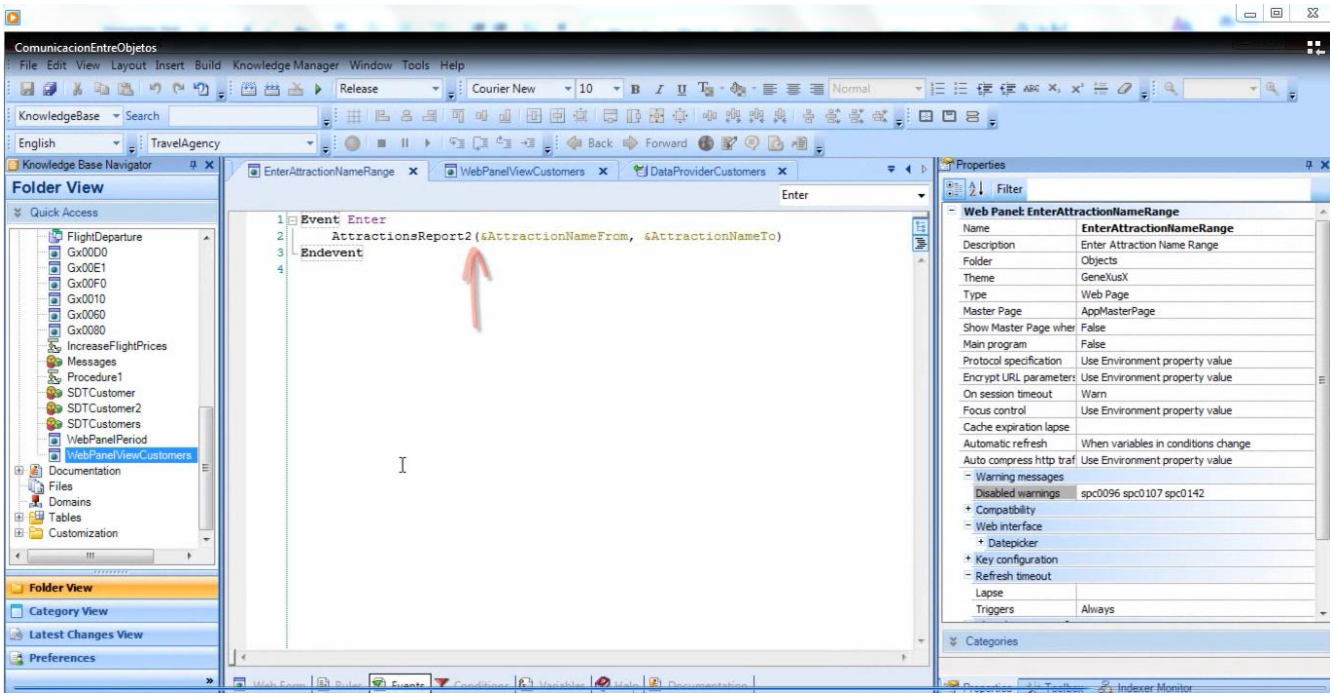
donde ingresábamos un rango de atracciones y al confirmar

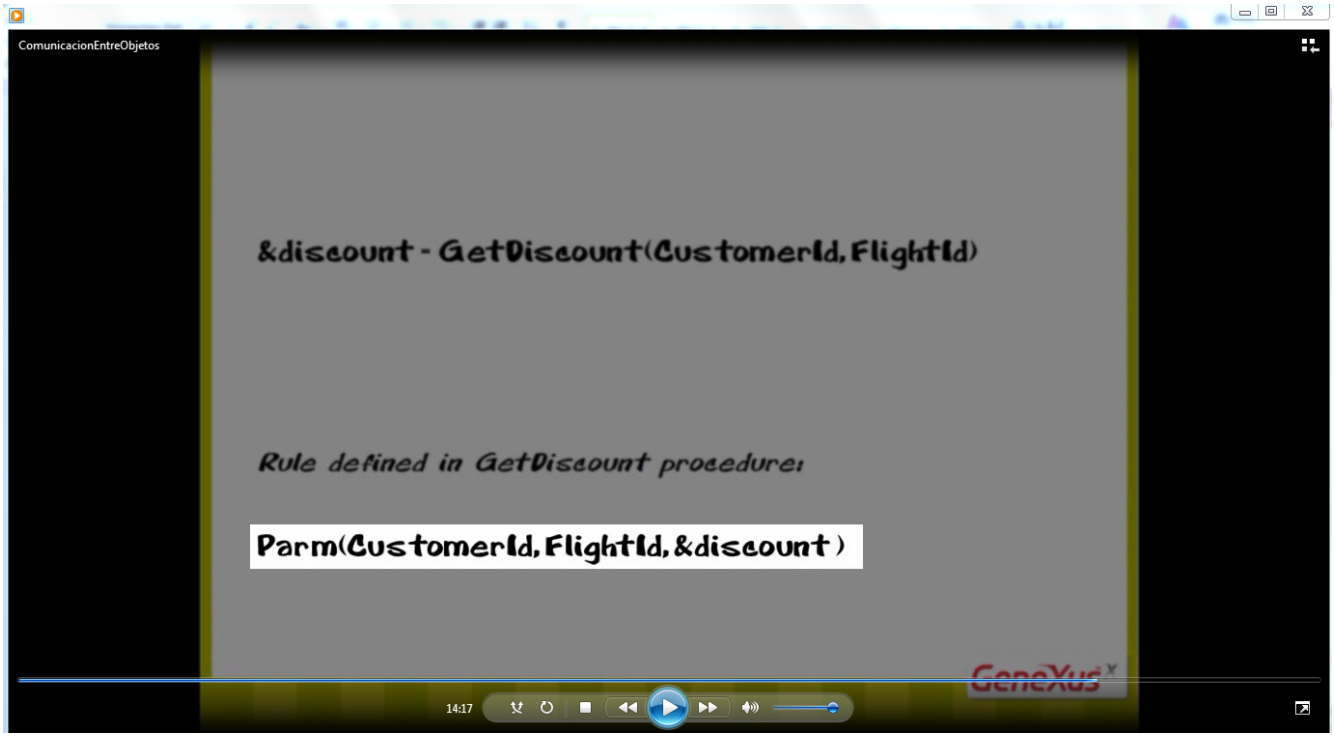


invocábamos a un procedimiento pasándole dicho rango que lo disponíamos en 2 variables



Recordemos que aquí podemos omitir el call.



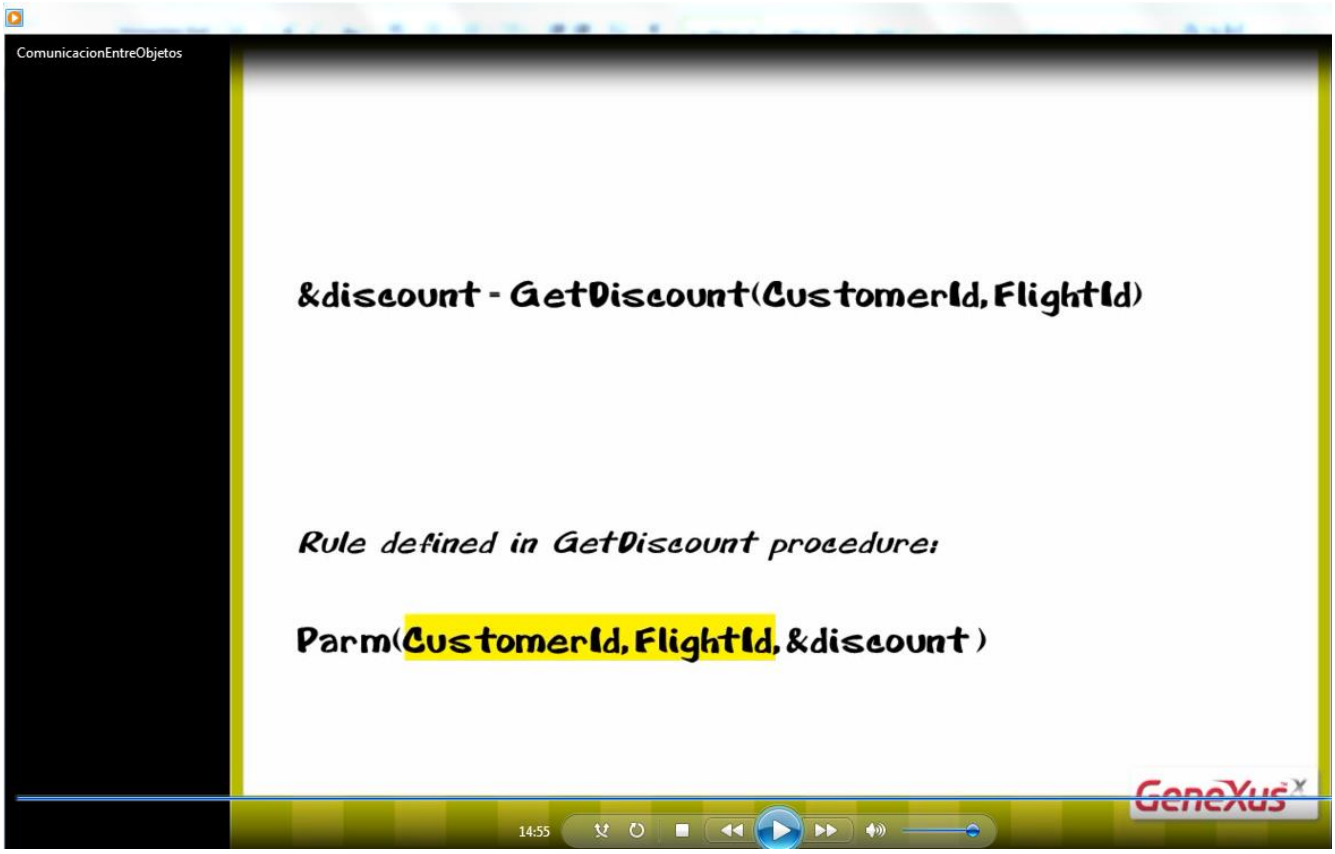


En lo que respecta a la declaración de la regla parm en el objeto invocado, podemos decidir **para cada parámetro recibido**, si declarar un atributo o una variable, independientemente de cómo haya sido enviado.

¿Cuál es la diferencia entre usar una variable o un atributo en la regla **parm** del objeto invocado?

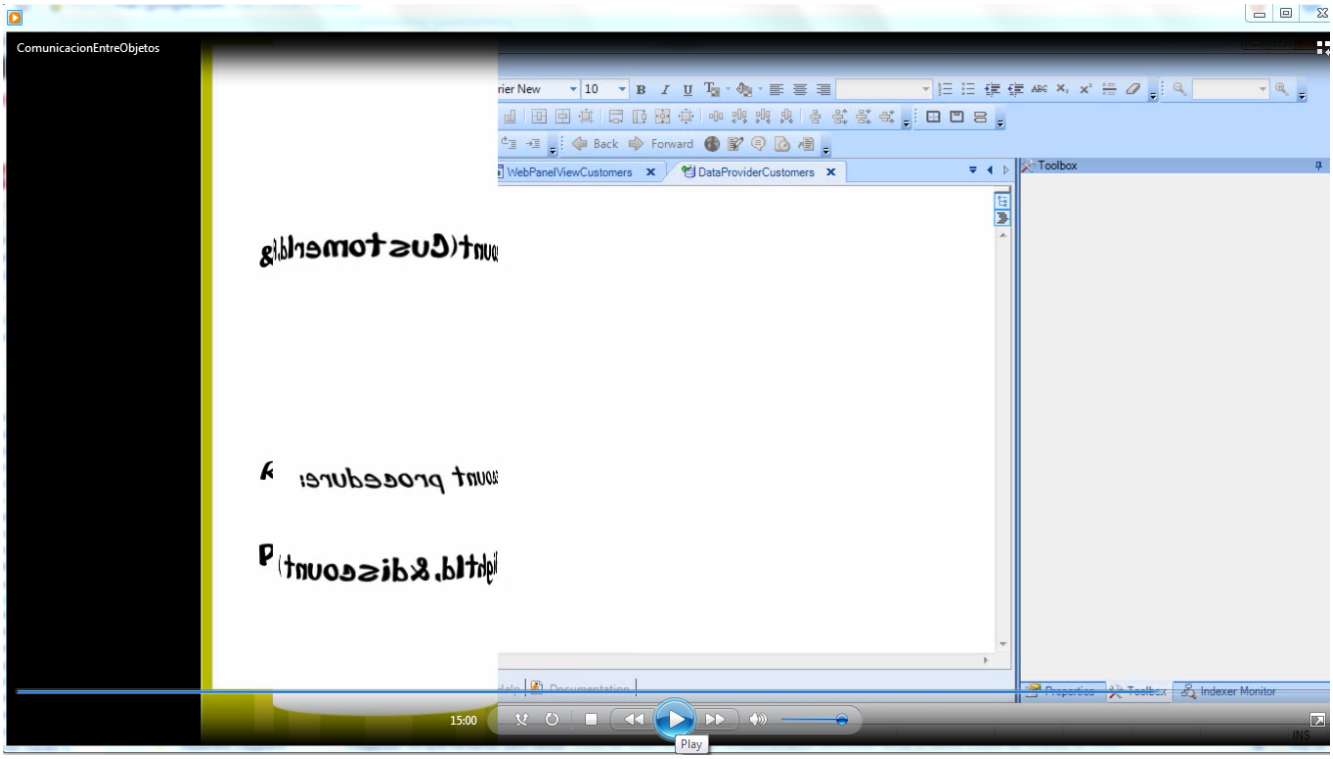
Si se recibe el valor en una variable, la misma se podrá utilizar libremente en la programación: se la podrá utilizar como condición de filtro por igualdad, por mayor, mayor o igual, menor, menor o igual… se la podrá utilizar para alguna operación aritmética, o lo que se necesite realizar con la misma.

Si en cambio se recibe el valor en un atributo,

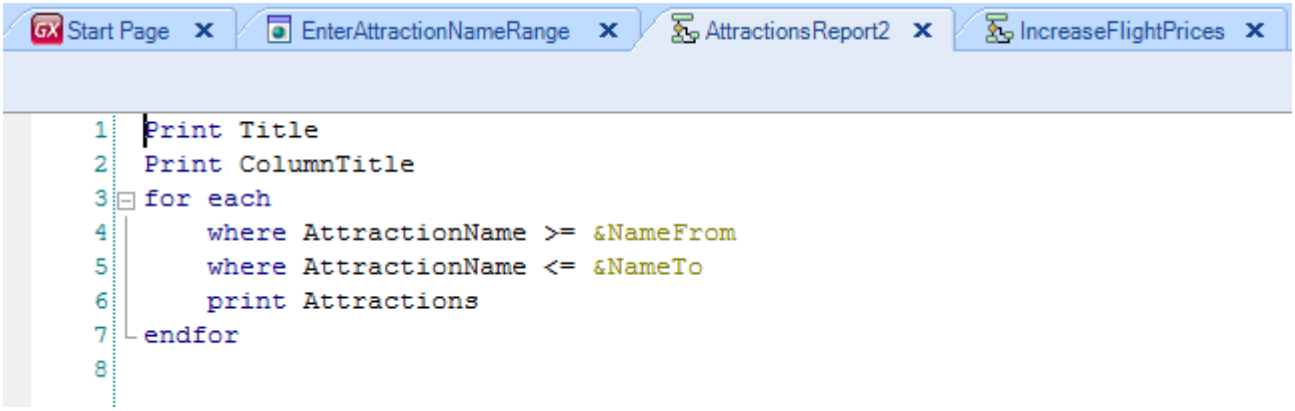


automáticamente el mismo actuará como filtro por igualdad en el objeto.

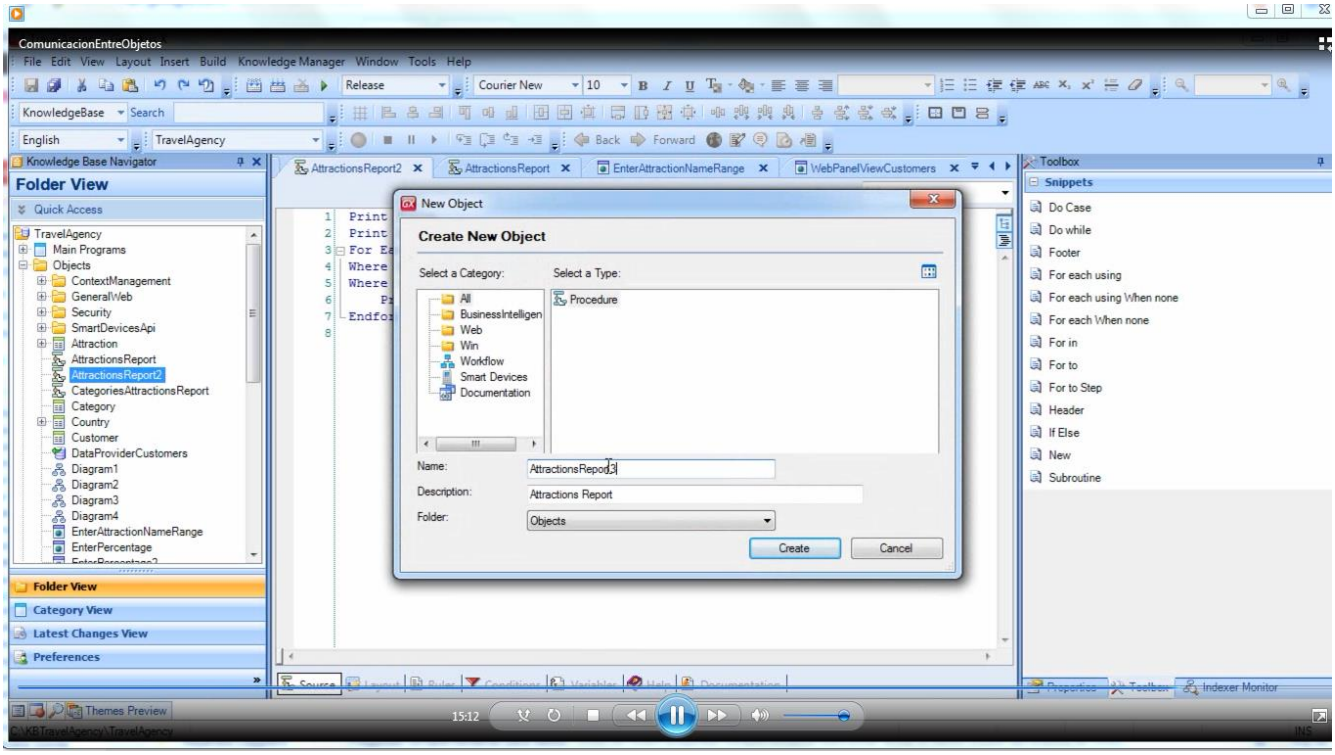
Veremos esto.



Vamos a grabar una copia de este procedimiento con otro nombre.



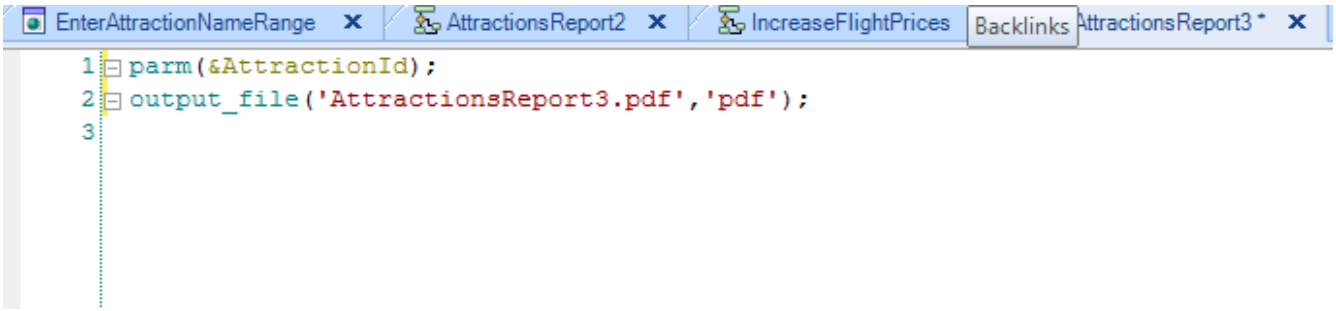
Le damos el nombre: AttractionsReport3.



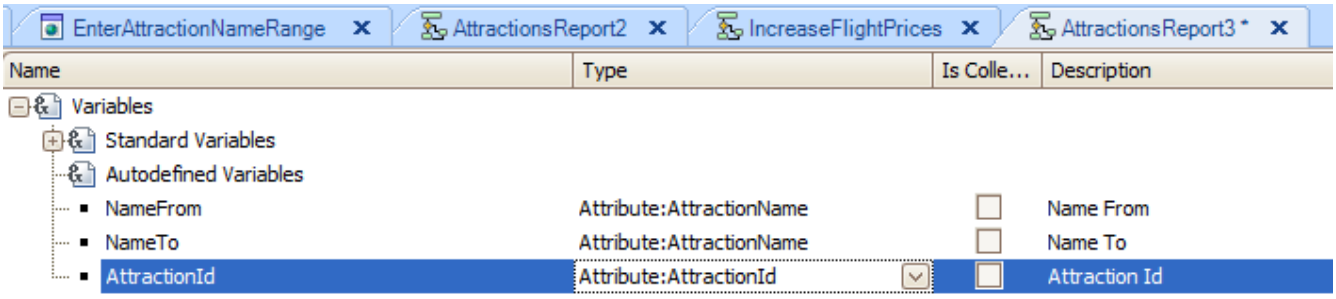
En este nuevo objeto, en vez de recibir 2 variables que contienen un valor inicial y final de nombres de atracciones… vamos a recibir una sola variable con un identificador de atracción, cuyos datos se deseen listar. Es decir que la idea aquí es listar los datos de una única atracción.

Vamos a la regla parm, quitamos estas 2 variables y escribimos &AttractionId.

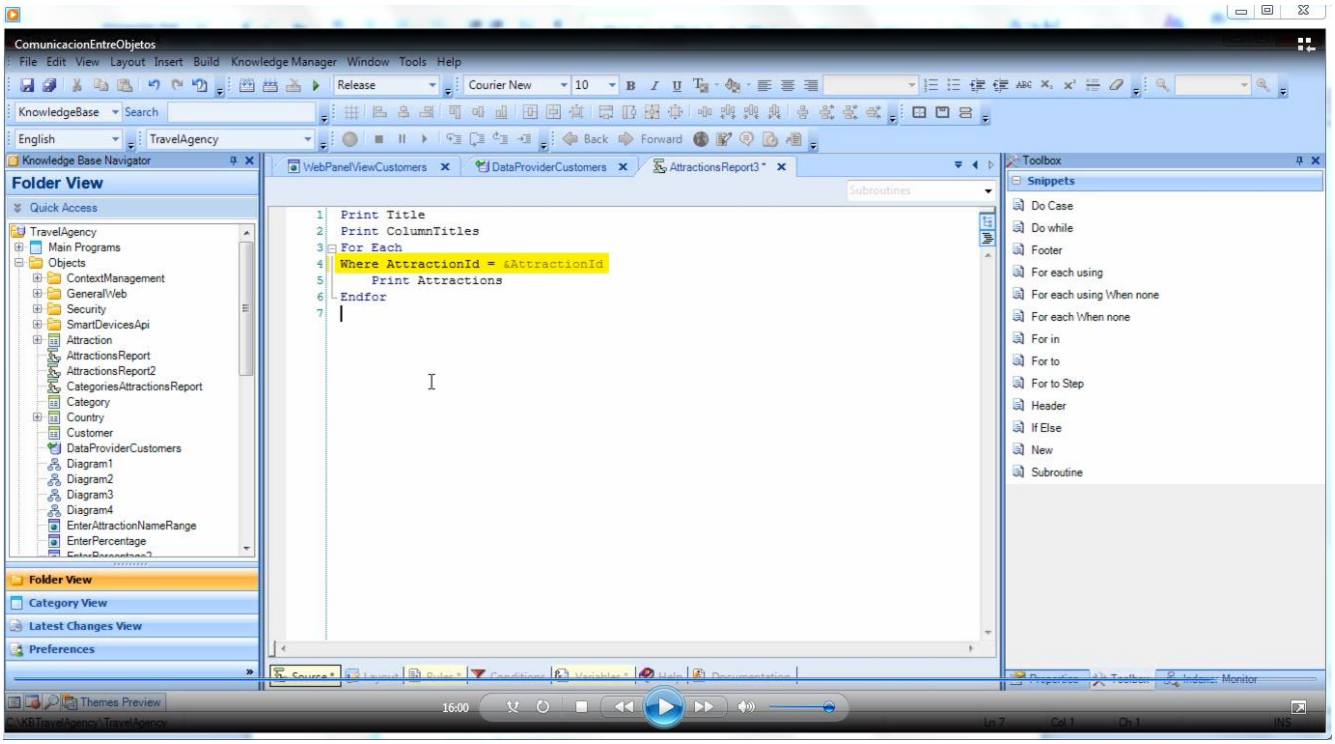
Ya que estamos, modificamos el nombre del archivo pdf a: AttractionsReport3.



Ahora vamos a la sección de variables y definimos la variable &AttractionId.



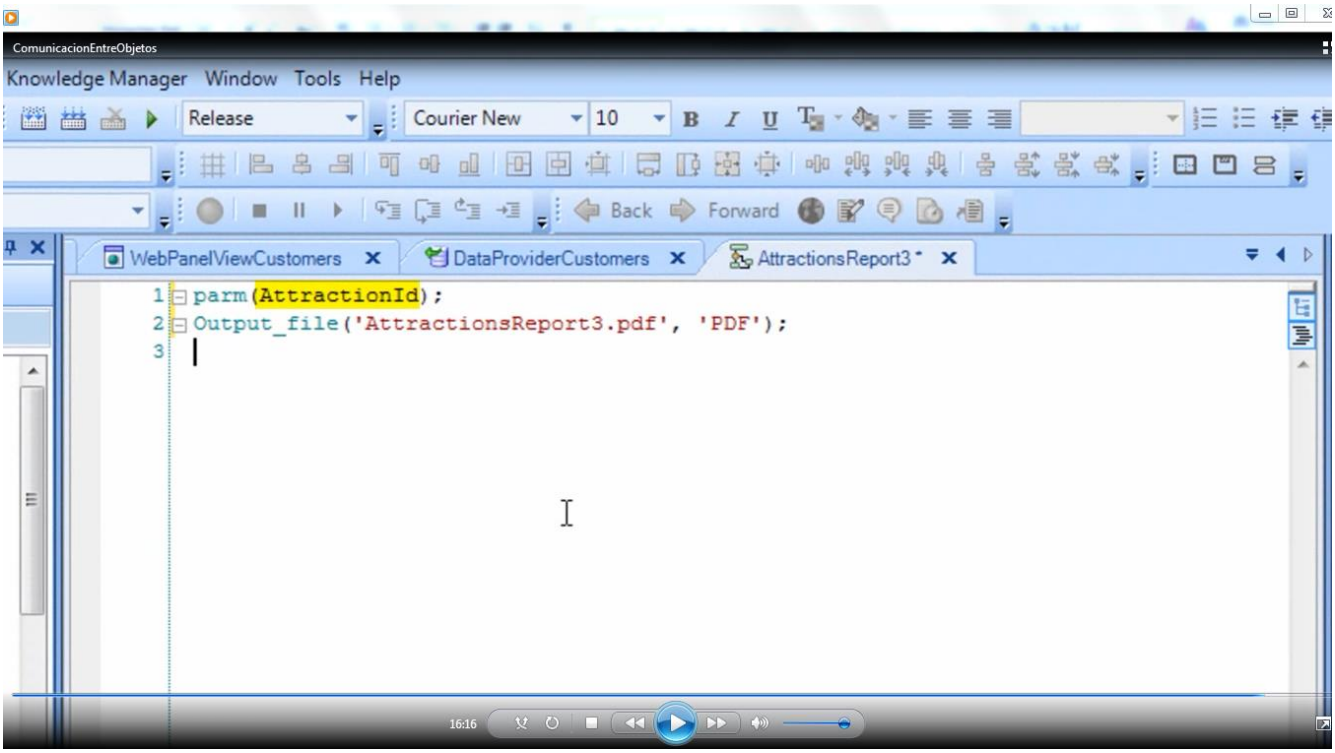
Y vamos a modificar en el source los filtros del For Each



Es claro que con esta definición, estamos listando los datos de una atracción recibida por parámetro.

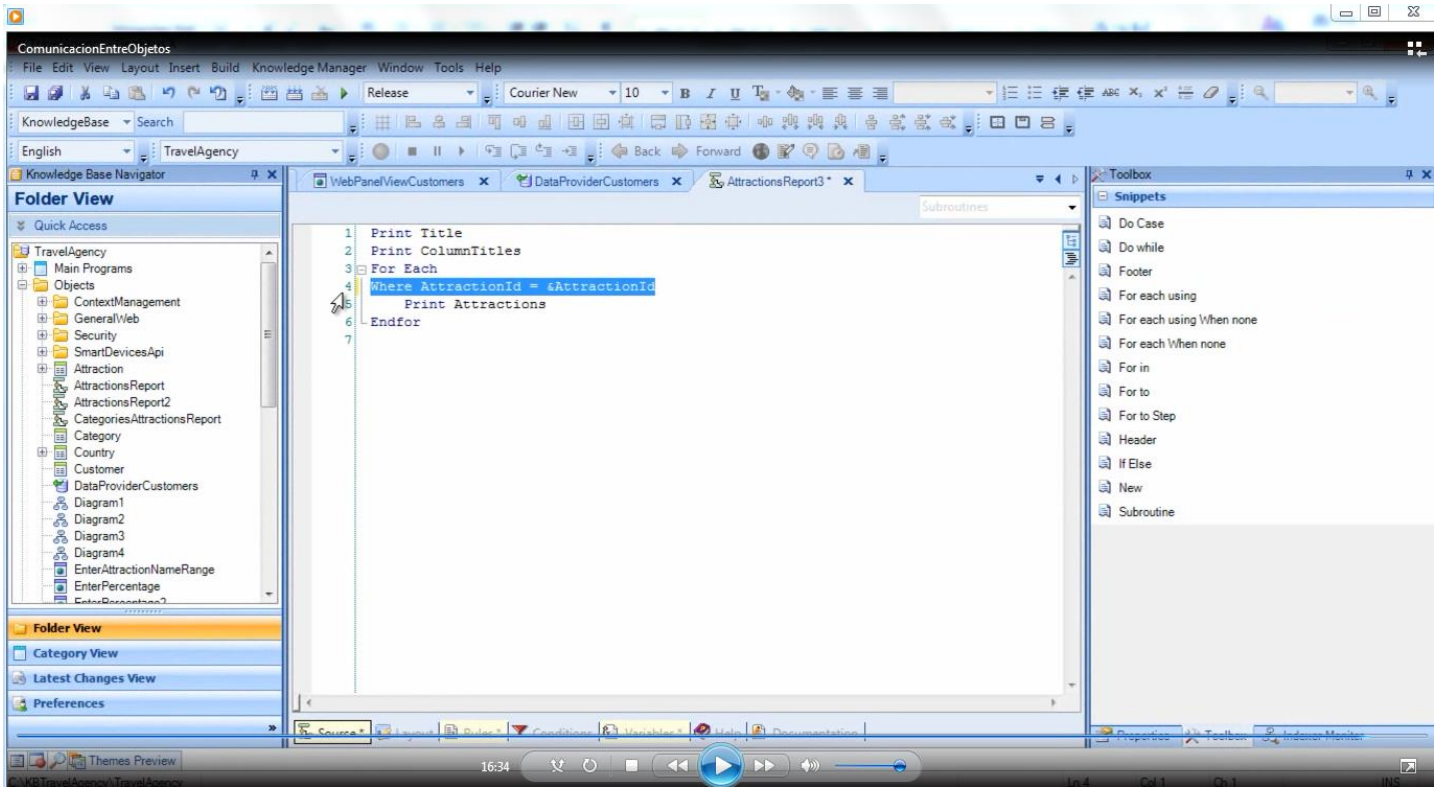
Ahora vamos a ver qué sucede si en vez de recibir al identificador de atracción en una variable, lo recibimos en su nombre de atributo.

Cuando recibimos el valor en un atributo en la regla,



GeneXus filtra por igualdad, es decir, solamente se va a acceder a los registros que tengan ese valor de identificador de atracción, en todos los accesos a la base de datos que se hacen en el source del objeto.

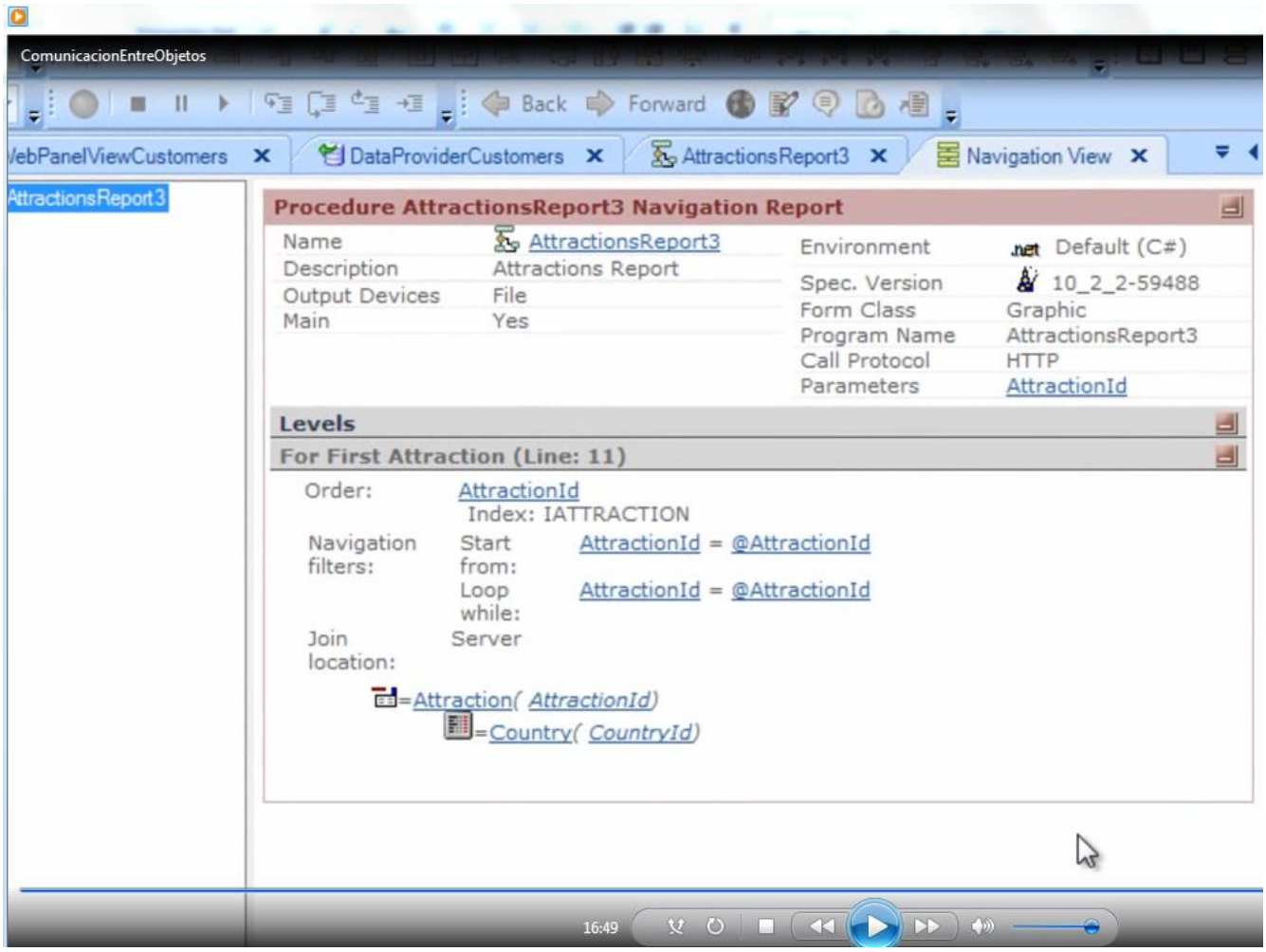
Por lo tanto, no es necesario escribir este where



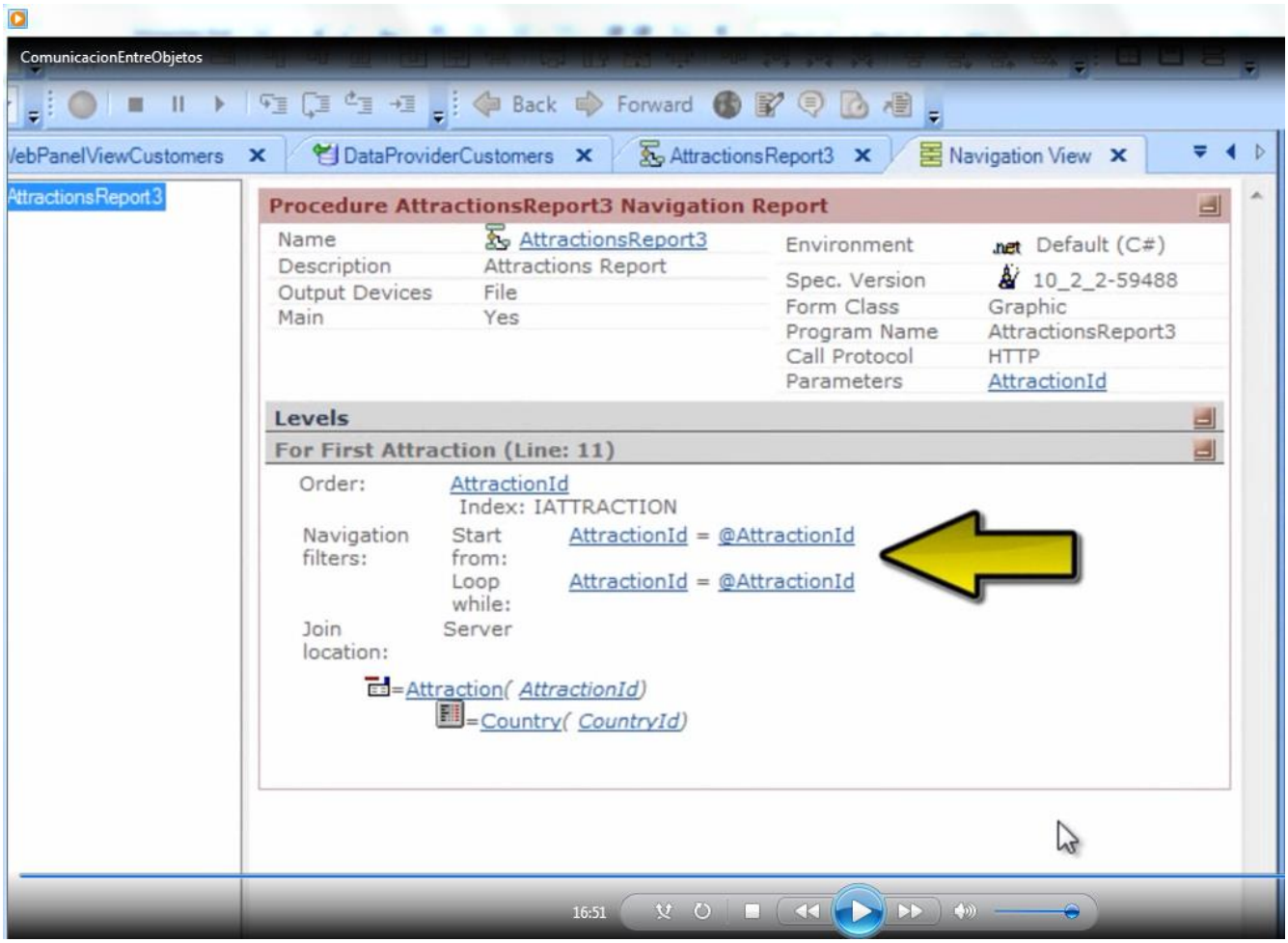
ya que el efecto de filtrar por dicho atributo ya se logra al recibir en el atributo en la regla parm.

Eliminamos entonces esta línea…

Si pedimos ver la navegación de este objeto



vemos que se está realizando el filtro



aunque no esté escrito el where. Si lo escribimos, no nos da un error, pero es innecesario escribirlo.

Hemos visto dos maneras de filtrar por igualdad la información accedida. Si nuestro objetivo no es utilizar un valor recibido para filtrar por igualdad, entonces la única solución posible es recibir los valores en variables y utilizarlas libremente.