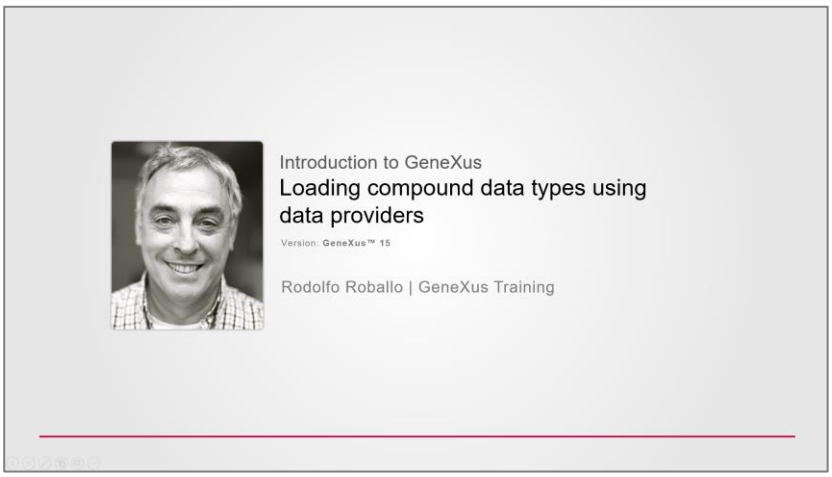
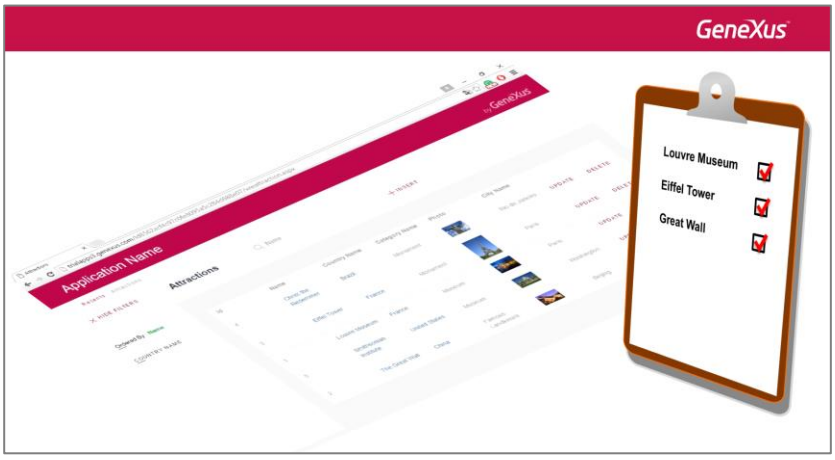
Cargando Tipos de Datos Compuestos (SDT) mediante Data Providers

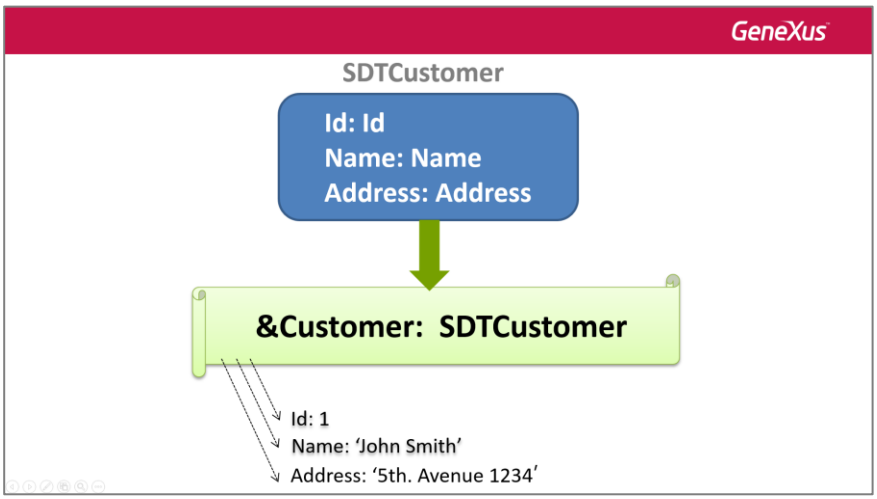


En más de una oportunidad necesitamos almacenar en memoria **una lista de elementos.** Por ejemplo, la agencia de viajes puede necesitar realizar operaciones con un grupo de clientes que cumplan cierto requisito...



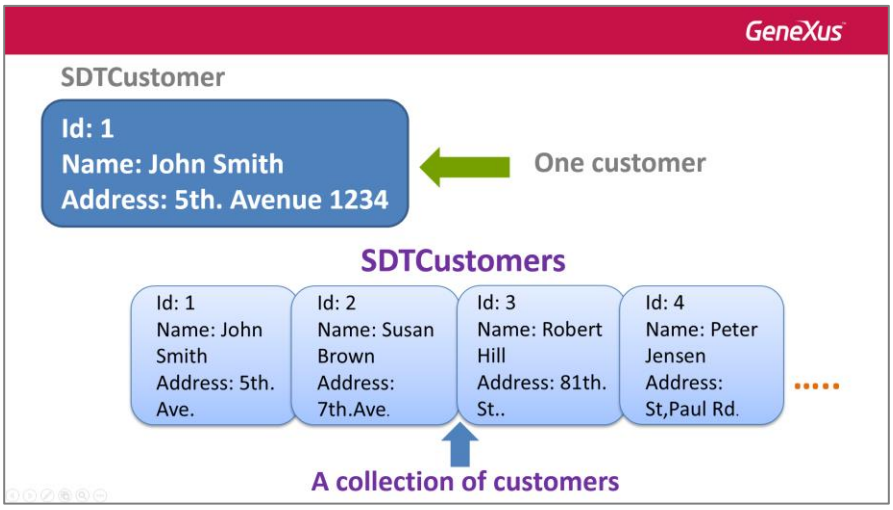
O nos pueden solicitar procesar información de algunos datos específicos de un conjunto de atracciones turísticas y eso nos puede implicar tener que cargar estas listas temporalmente en memoria.

Para resolver este tipo de requisito, es necesario crear una estructura en memoria **capaz de almacenar una colección de elementos**.

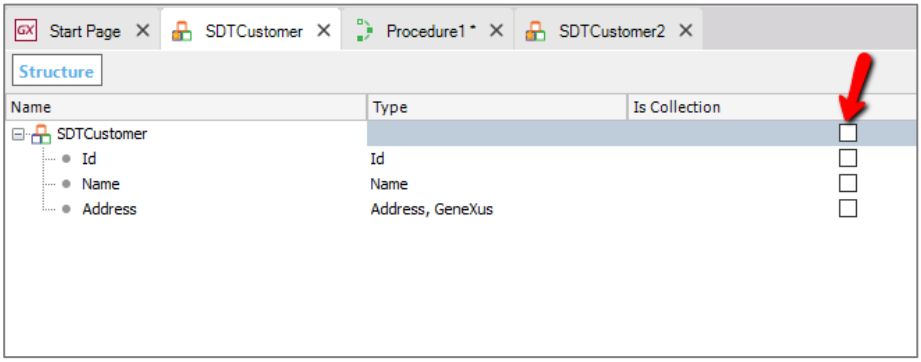


Ya hemos visto que **los tipos de datos estructurados** nos permiten definir estructuras que almacenan varios datos correspondientes a **una entidad,** como en este ejemplo SDTCustomer que nos permite almacenar la información de un cliente.

Con esta estructura, estamos almacenando en memoria el identificador, el nombre y la dirección de **un** cliente.



Para almacenar datos de varios clientes, vimos que es necesario definir el tipo de datos estructurado y luego marcar que se trata de una colección.



Vayamos ahora a GeneXus para implementar un nuevo requerimiento solicitado por la agencia de viajes, relacionado con lo que acabamos de ver.

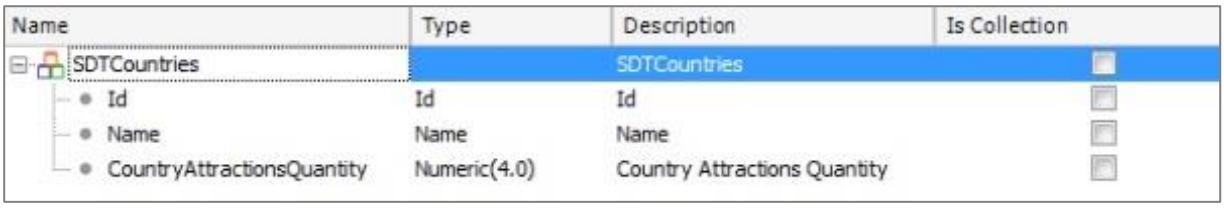
La Agencia de viajes desea ofrecer a sus clientes un ranking de países según la cantidad de atracciones turísticas que ofrecen.

Es decir que debemos mostrar todos los países, ordenados de mayor a menor, en lo que se refiere a la cantidad de atracciones que poseen.

Para poder resolver este requerimiento, necesitamos primero cargar en alguna estructura a todos los países almacenados en la base de datos, cada uno con su correspondiente cantidad de atracciones turísticas. Y luego, tendremos que ordenarlos de mayor a menor por dicha cantidad, para finalmente mostrarlos, ya sea en un web panel o en un listado.

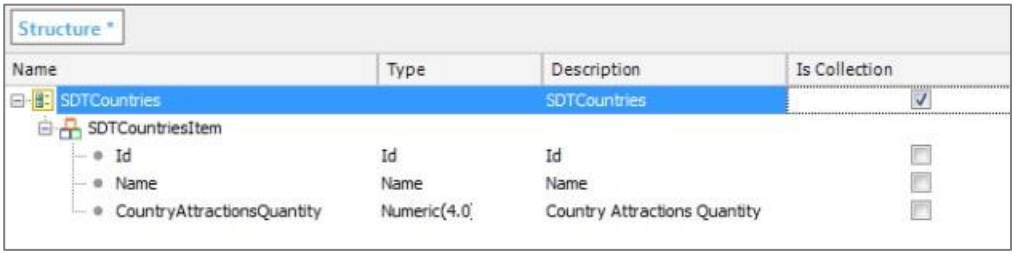
Comencemos entonces creando un nuevo objeto de tipo: Structured data type y le damos el nombre: SDTCountries.

De cada país necesitamos el identificador, el nombre y la cantidad de atracciones turísticas registradas para el mismo.



Con esta definición estamos representando que se guardará en memoria la información asociada a un solo país. Pero nosotros necesitamos diseñar un ranking de países, por lo tanto necesitamos almacenar en memoria muchos países.

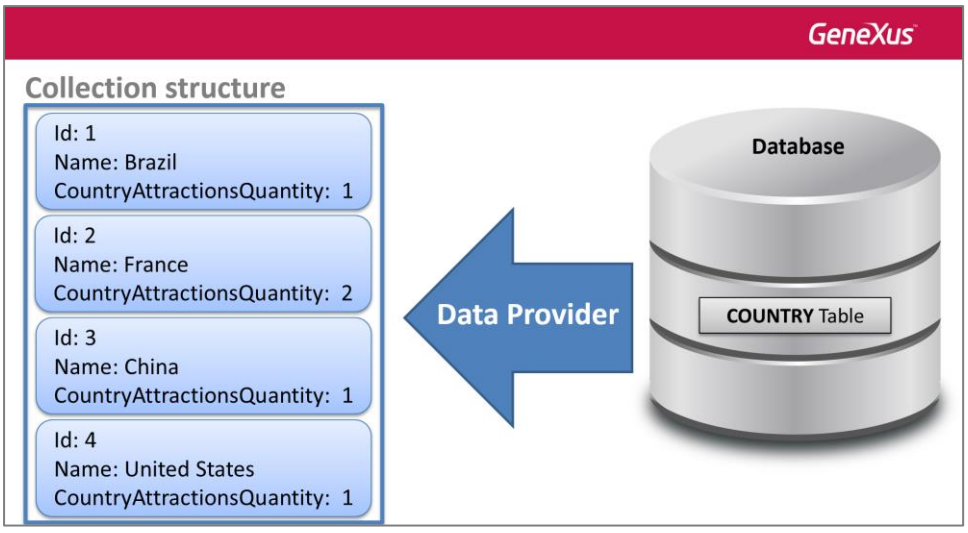
Así que marcamos la casilla **Is collection** para obtener una estructura que permita almacenar la información de muchos países.



Observemos que ahora esta estructura tiene los mismos miembros que definimos al comienzo, pero agrupados en una subestructura denominada SDTCountriesItem.

Esta subestructura se creó automáticamente cuando marcamos que se trata de una colección. Cada ítem almacenará los datos de un país, y **el conjunto de ítems** conforman la colección.

Para cargar los datos de la misma, vamos a utilizar un **objeto Data Provider**.



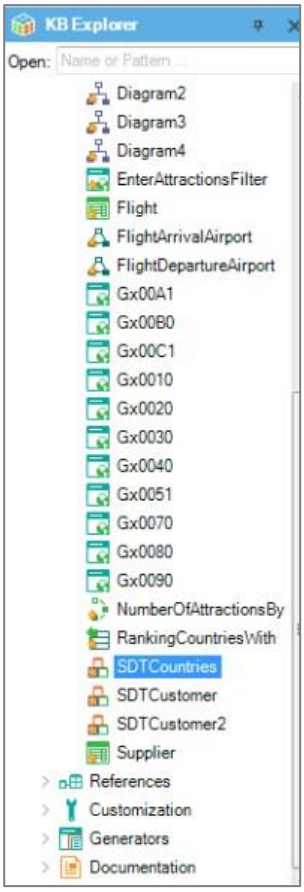
El Data Provider nos permite **cargar una estructura de datos**, por ejemplo a partir de datos de la base de datos, y nos devuelve la estructura cargada.

Vamos a GeneXus para crear un objeto Data Provider. Elegimos el tipo Data Provider y ponemos como nombre: RankingCountriesWithAttractionsQty.

Vemos que GeneXus nos posiciona en la sección Source del Data Provider.

Aquí es donde vamos a declarar cómo queremos que se carguen los datos en la colección que devolverá el Data Provider. Observemos qué fácil resulta declarar la carga.

Vamos a la ventana del Knowledge Explorer:



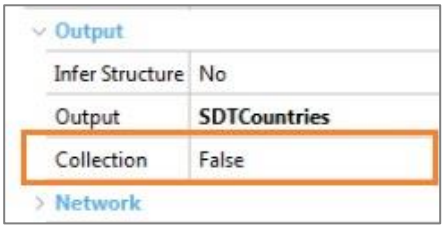
…ubicamos al tipo de datos estructurado SDTCountries y lo arrastramos hacia el Source del Data Provider.

Vemos que GeneXus escribió automáticamente varias líneas de texto. Estas líneas de texto definen en forma declarativa la estructura de datos que será llenada por el Data Provider y de dónde se obtendrán los datos para cargarla.

Si observamos ahora las propiedades del DataProvider, observamos que GeneXus asignó el nombre de la colección SDTCountries a la propiedad Output.

Esto significa que el DataProvider devolverá una colección del tipo de datos estructurado SDTCountries, cargada con datos.

Como el SDTCountries ya es una colección, no es necesario configurar la propiedad **Collection** del Data Provider con valor True.



Esto lo haríamos si quisiéramos que el Data Provider nos devuelva una colección a partir de un tipo de datos

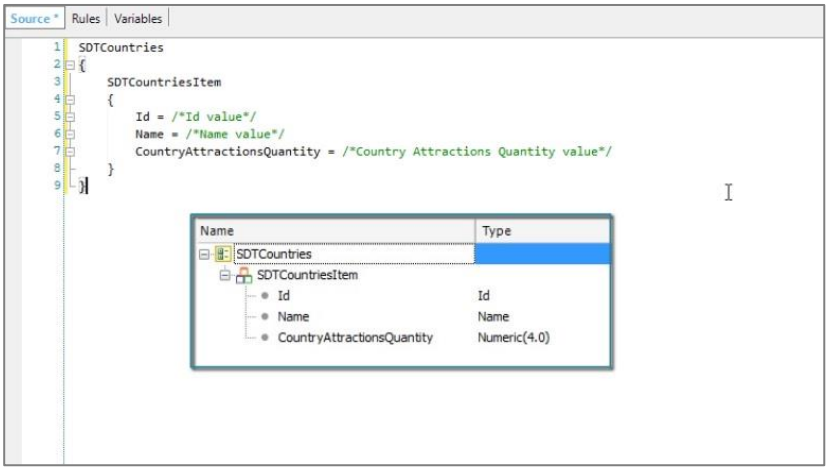
estructurado simple. Lo veremos luego.

Estudiemos ahora qué fue lo que escribió GeneXus en el source…

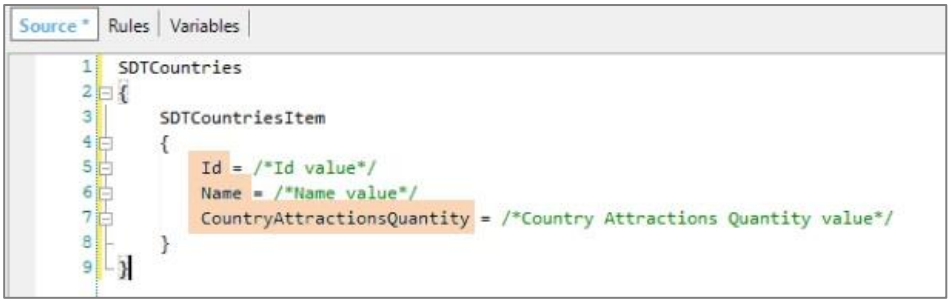
Reconocemos el nombre del tipo de datos estructurado SDTCountries que es una colección:

Y después entre llaves está la subestructura del ítem de la colección:

Comparemos esto con la estructura del SDT:



Vemos que GeneXus representó en forma de texto la estructura del SDTCountries y nos dejó listos los miembros Id, Name y CountryAttractionsQuantity de la subestructura SDTCountriesItem para cargarles su valor, colocándolos a la izquierda de los signos de igual.

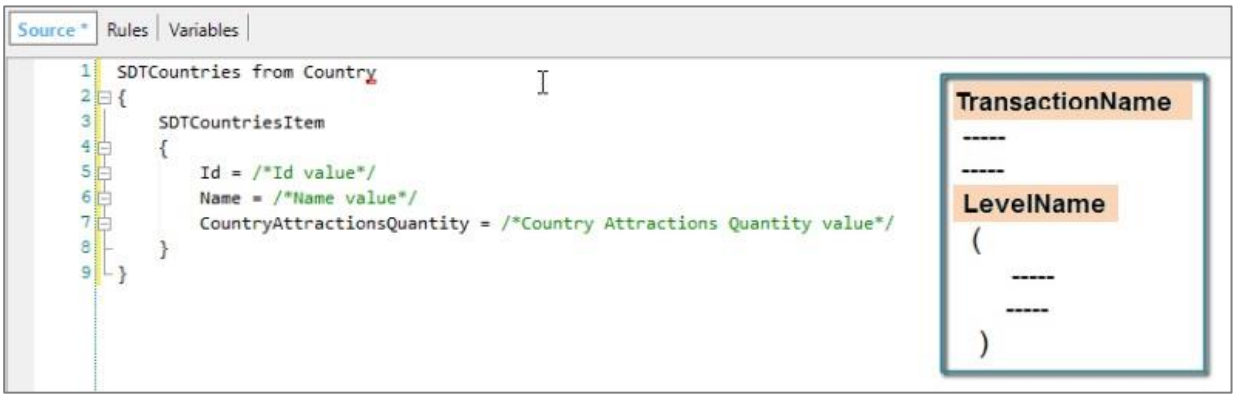


Como vamos a cargar esta colección a partir del contenido de la tabla asociada a la transacción Country, debemos indicarle al Data Provider que deberá recorrer dicha tabla y del lado derecho de los signos de igual, mencionar los atributos que asignaremos a cada miembro del SDTCountries.

Para indicar qué transacción base queremos recorrer, utilizamos la cláusula **From** y al lado ponemos el nombre de la transacción cuya tabla base es la tabla a recorrer.

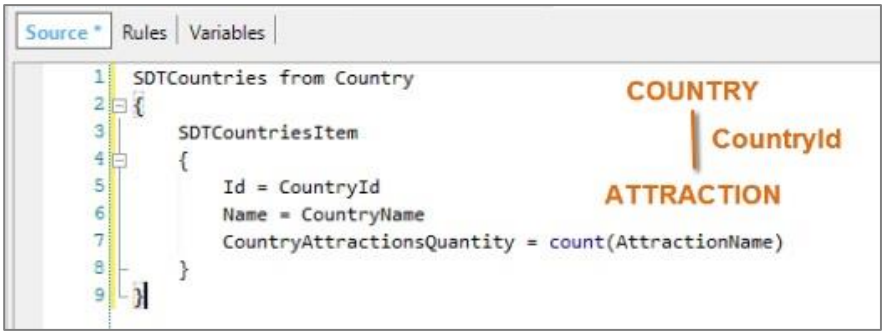
En nuestro caso escribimos: From Country

Si la transacción tuviera más de un nivel, para especificar un nivel determinado asociado a cierta tabla base que queremos navegar, tendríamos que escribir: **from** el **nombre de la transacción, punto, y el nombre del nivel**.



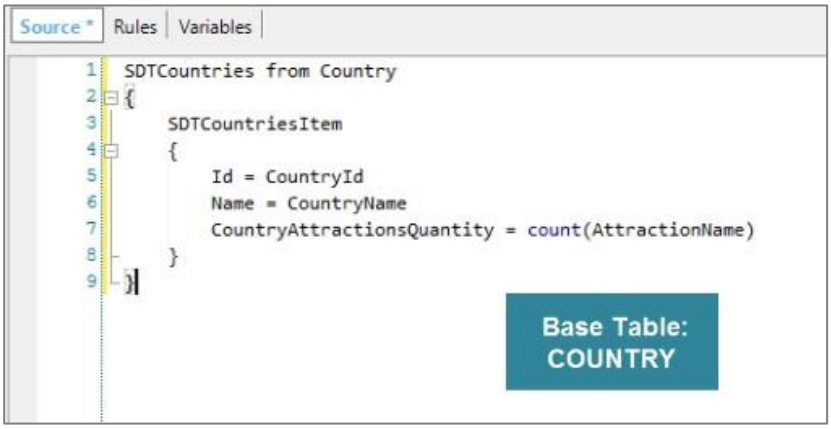
Ahora vamos a indicar que al elemento Id lo queremos cargar con el valor del atributo CountryId, al elemento Name lo queremos cargar con el valor de CountryName, y al miembro CountryAttractionsQuantity lo queremos cargar con la cantidad de atracciones turísticas que tiene cada país, así que asignamos a este miembro: el resultado de la fórmula Count(AttractionName).

Repasemos un concepto ya estudiado…. y es que esta fórmula inline definida, navegará la tabla ATTRACTION, por el atributo indicado dentro del paréntesis. Y además, como hay un atributo en común en las tablas navegadas por el Data Provider, que es CountryId, la fórmula contará las atracciones **del país** navegado por el Data Provider cada vez.



De modo que lo que hemos hecho simplemente ha sido: **declarar una transacción base a ser navegada por el Data Provider,** y **para cada registro accedido, hemos indicado los valores que deseamos asignar a un ítem nuevo en la colección de países.**

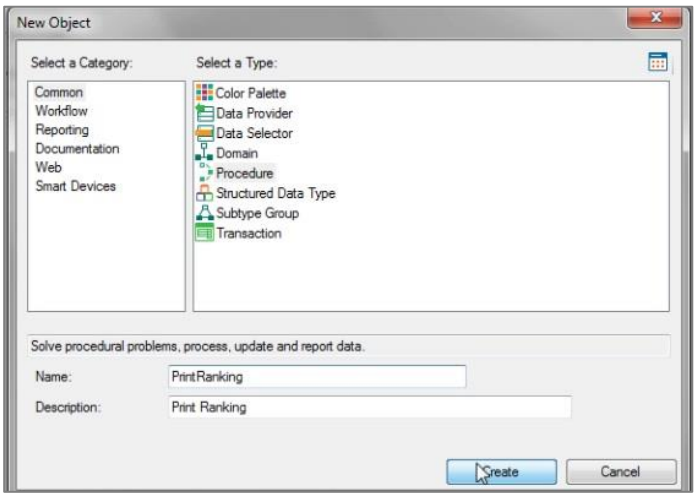
Este grupo del Data provider es análogo a un for each.
Y dado que recorre la tabla COUNTRY, solemos decir **que la tabla base de este grupo del Data Provider es COUNTRY:**



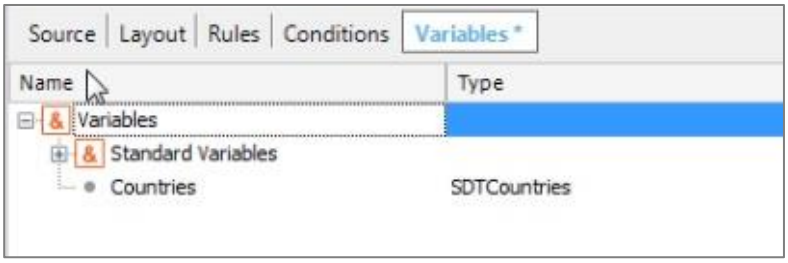
El resultado final será que habrán quedado almacenados en la colección en memoria, los datos de todos los países de la base de datos, c/u de ellos con su cantidad de atracciones.

Salvamos la definición de este Data Provider, y vamos a crear ahora un objeto Procedimiento para visualizar el contenido de la colección de países.

A este procedimiento le ponemos como nombre “PrintRanking”.

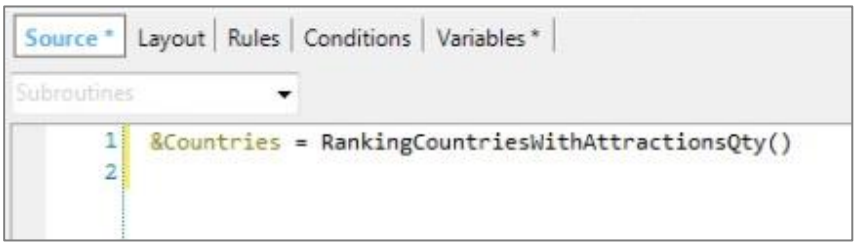


Vamos ahora a la sección variables de este procedimiento y definimos una variable &Countries de tipo de dato SDTCountries…



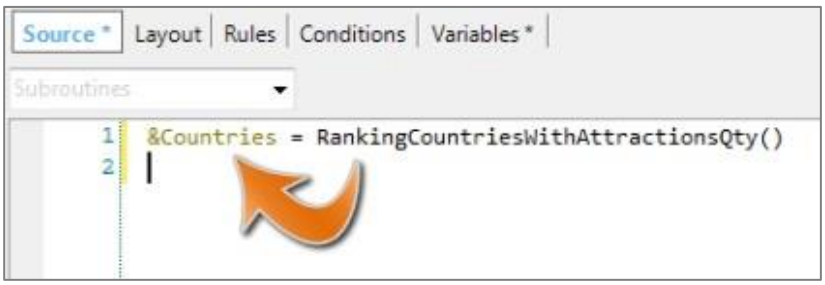
Ahora en el Source escribimos &, y elegimos nuestra variable Countries.
Y vamos a cargar a esta variable de tipo colección, asignándole el resultado que devuelva el Data Provider que acabamos de crear.

Así que escribimos el signo de igual y desde la ventana Knowledge Base Explorer, ubicamos al Data Provider y lo arrastramos. Completamos esta instrucción agregando 2 paréntesis:



Con esta instrucción que hemos escrito, estamos **invocando** al Data Provider y éste retornará una colección de países, que quedará cargada en la variable &Countries.

Observar que si el Data Provider requiriera que se le enviase información para poder trabajar con ella (por ejemplo, un rango de identificadores de países, para poder devolver solamente la colección de países dentro de ese rango), deberá agregarse regla Parm al Data Provider para declarar esos parámetros.



Ahora recordemos cuál era el requerimiento exacto de la Agencia de Viajes. Era visualizar un ranking de todos los países ordenados de mayor a menor según la cantidad de atracciones que tienen registradas.

Por lo tanto, nos está faltando ordenar la colección que obtuvimos cargada.

Es decir, ordenar los ítems de la colección de países, antes de ser mostrada, por orden de mayor a menor según la cantidad de atracciones que tienen registradas.

Para resolver esto contamos con el método **Sort**.

La sintaxis es la siguiente:

La variable &Countries, punto, Sort y entre paréntesis y comillas el nombre del miembro del SDT por el cual queremos ordenar. En nuestro caso CountryAttractionsQuantity:

Pero de esta forma la colección de países quedará ordenada de menor a mayor por la cantidad de atracciones y nosotros necesitamos que se ordene de mayor a menor, ya que queremos implementar un ranking.

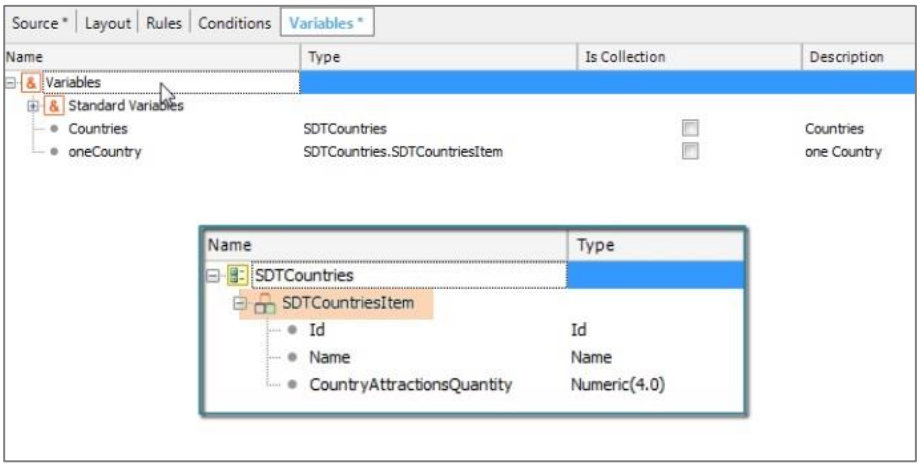
Así que para indicar el orden inverso, dentro de las comillas agregamos paréntesis rectos.

Ahora lo único que nos está faltando es recorrer la colección devuelta por el Data Provider e imprimir para cada ítem de la misma, los datos almacenados.

Para recorrer una colección almacenada en memoria, contamos con el comando **For elemento in Colección**.

Vamos ahora a definir una variable &oneCountry para cargar en ella cada elemento que vayamos iterando de la colección.

El tipo de dato será SDTCountries.SDTCountiresItem, que es como se escribe el tipo de datos correspondiente a cada ítem de la colección

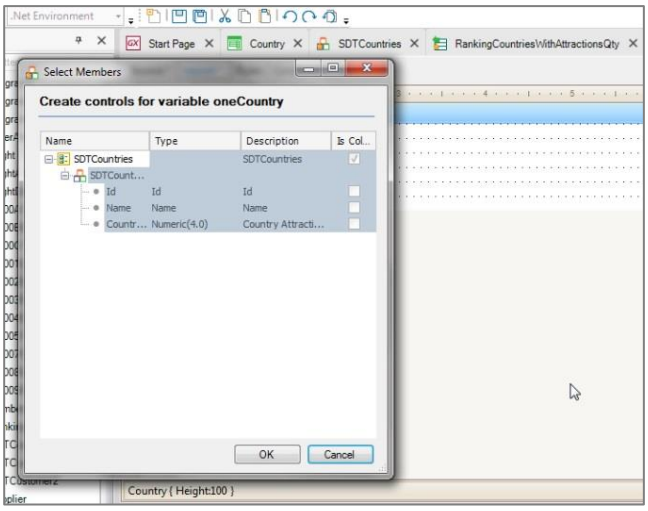


Ahora volvamos al source para completar el código que recorre la colección:

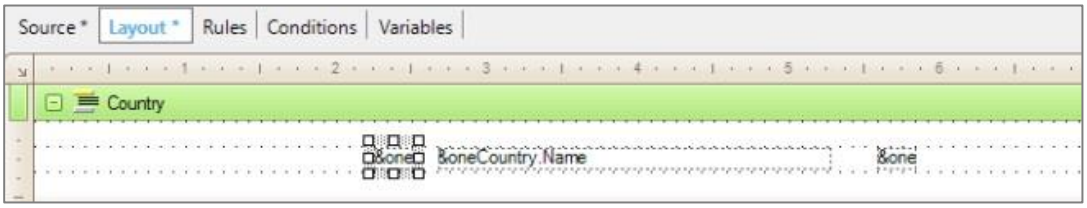
Escribimos: For &oneCountry in &Countries… print el printblock de nombre Country… y cerramos con EndFor.

Vamos ahora a la sección “layout” y, nombramos a este printblock como“Country”

Seleccionamos: Insert / Variable… elegimos &oneCountry…. y vemos que nos ofrece insertar variables para cada elemento o miembro del ítem.



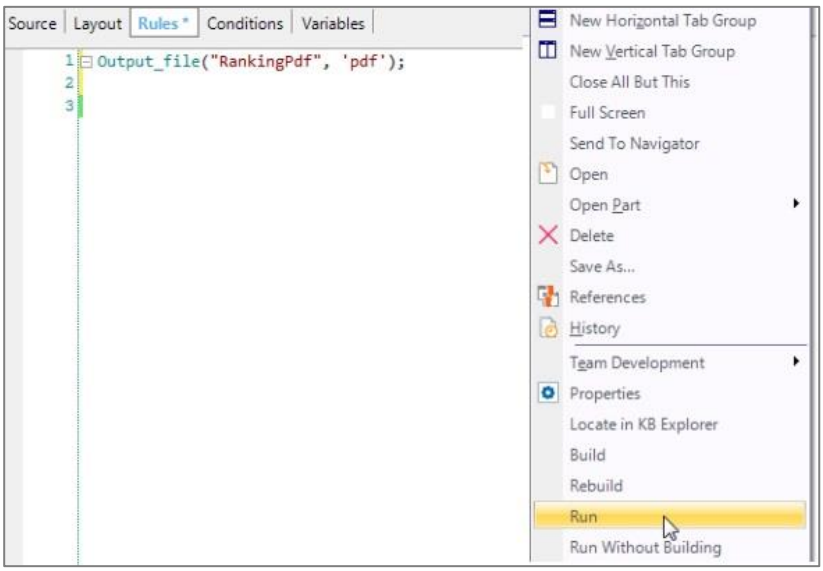
Ubicamos los controles en el printblock,



y ahora solamente nos resta definir las propiedades necesarias para que se imprima el listado con formato PDF.

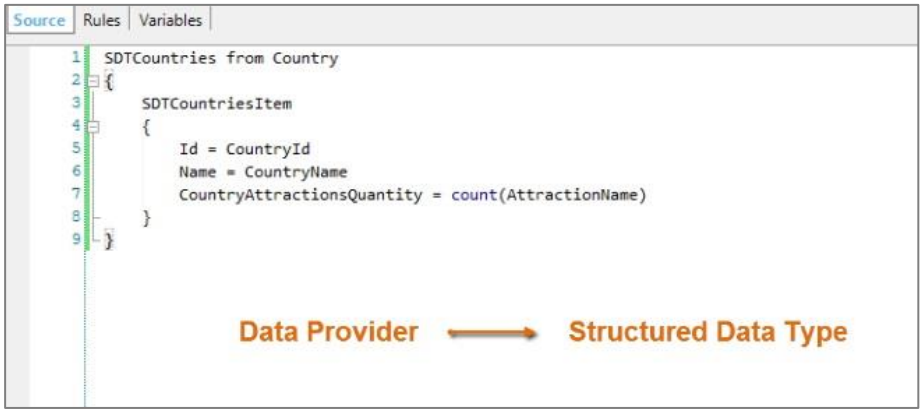
Vamos entonces a las propiedades del procedimiento, marcamos “Main program” con el valor True, “Call protocol” como “HTTP”, por último insertamos la regla Output_File y completamos poniendo el nombre del archivo del listado que será RankingPdf, y el formato pdf.

Ahora sí el desarrollo de lo solicitado está completo. Seleccionemos **Run** sobre el procedimiento para ver el ranking en ejecución:



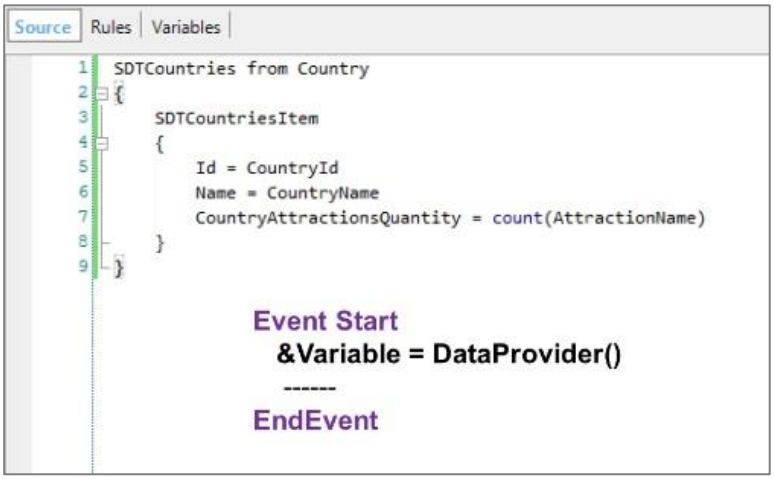
Aquí vemos el listado PDF con todos los países que estaban almacenados en la base de datos, cada uno con su cantidad de atracciones, y ordenados en forma descendente por cantidad de atracciones, tal cual fue solicitado.

De esta forma hemos visto la potencia y la simplicidad de los Data Providers para cargar datos en una estructura de datos en memoria, en particular del tipo colección, y cómo GeneXus resuelve todo lo necesario para llevarlo a cabo.



Además, los Data Providers aceptan opcionalmente la cláusula where para filtrar, como el comando For each…

y también pueden ser invocados desde otros objetos GeneXus, como veremos más adelante en otros ejemplos de uso.



Para finalizar, actualizamos los cambios en GeneXus Server.