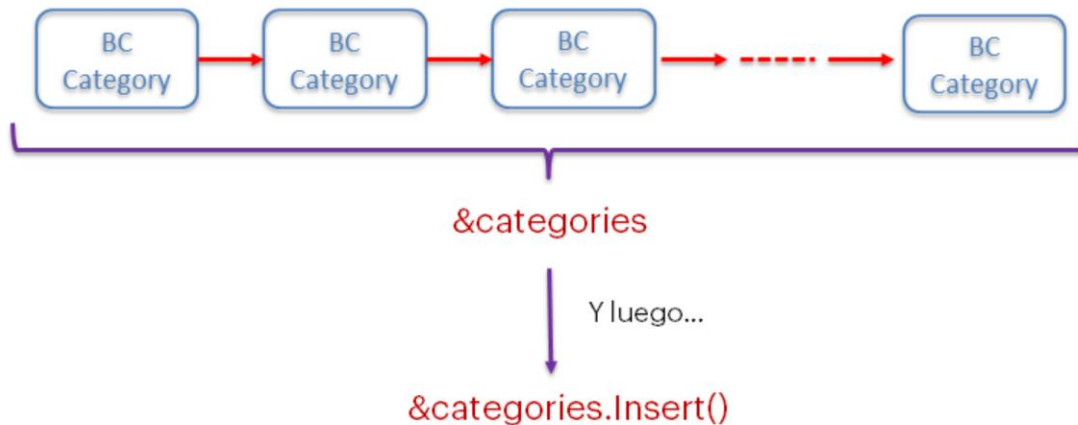


Poblar con datos utilizando Business Component y Data Provider

GeneXus™ 16

Objetivo: inicializar las tablas de categorías y atracciones con datos

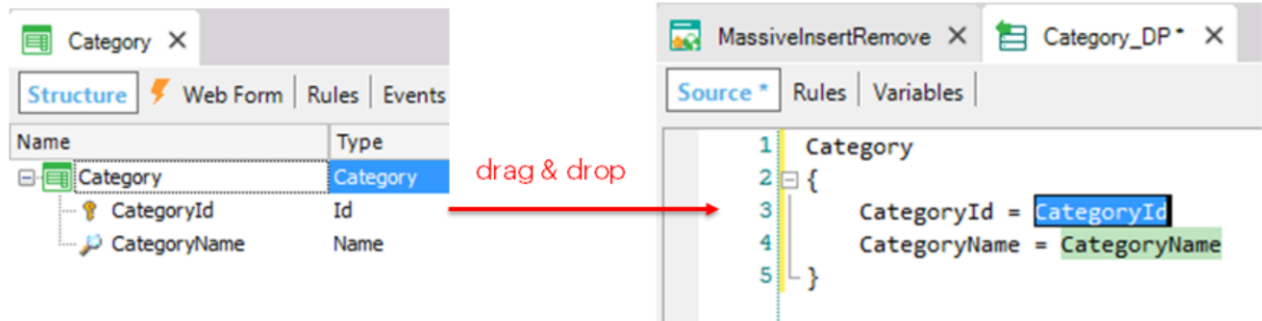
Necesitamos:



Si lográramos obtener una variable colección de ítems del tipo business component de Category, cargada con las categorías que deseamos ingresar a la base de datos, nos alcanzará con aplicar el método Insert() a esa variable colección, pues como mencionamos en el video anterior, esto permitirá hacer el Insert de todos los ítems, es decir, de todos los business components de la colección.

Nuestro problema se reduce, entonces, a obtener esa colección. ¿Cómo hacemos?

Para obtener la colección, usamos también un data provider



Hasta ahora sabíamos que un data provider nos permitía devolver datos estructurados, tanto simples como colección. En nuestro caso, queremos devolver una colección de categorías, pero estas categorías no son tipos de datos estructurados sino business components.

Sin embargo, un business component es exactamente igual que un SDT en lo que hace a su estructura. Por lo que los data providers también nos permitirán cargar y devolver business components, tanto simples como colecciones.

Arrastramos la transacción Category dentro del Source del data provider y vemos que nos escribe la estructura de la transacción. Notemos que a la izquierda tenemos los elementos del business component, que estarán en memoria, y que tienen el mismo nombre que los atributos, pero no lo son. Mientras que a la derecha, por defecto se están mostrando, esta vez sí, los atributos de la tabla correspondiente, desde donde el data provider obtendrá los datos para cargar el BC que está en memoria.

Si esto es lo que quisiéramos, el data provider debería devolver una colección de este business component, dado que la tabla tiene muchos registros.

Para que el DP nos devuelva una colección, usamos la propiedad Collection

Data Provider: Category_DP

Name	Category_DP
Description	Category_DP
Expose as Web Service	False
Module/Folder	Root Module
Qualified Name	Category_DP
Object Visibility	Public
v Output	
Infer Structure	No
Output	Category
Collection	True
Collection Name	CategoryCollection
> Network	

Si vamos a las propiedades, vemos que la propiedad Output asumió el valor del business component, pero la propiedad collection no está en True, como necesitamos.

Así que la cambiamos y nos aparece la nueva propiedad Collection name, que por defecto asume el nombre del data provider. Lo cambiamos por CategoryCollection.

Asignamos valores nuevos especificados por nosotros...

The diagram illustrates the transformation of three separate `Category` objects into a single `CategoryCollection` object. On the left, three `Category` objects are shown, each with a `CategoryName` property set to "Museum", "Monument", and "Tourist Site" respectively. These are separated by a red equals sign. On the right, a `CategoryCollection` object is shown, which contains three `Category` objects, each with the same `CategoryName` values as the objects on the left. This represents a more structured and clear way to represent a collection of data.

```
Category
{
  CategoryName = "Museum"
}
Category
{
  CategoryName = "Monument"
}
Category
{
  CategoryName = "Tourist Site"
}

=

CategoryCollection
{
  Category
  {
    CategoryName = "Museum"
  }
  Category
  {
    CategoryName = "Monument"
  }
  Category
  {
    CategoryName = "Tourist Site"
  }
}
```

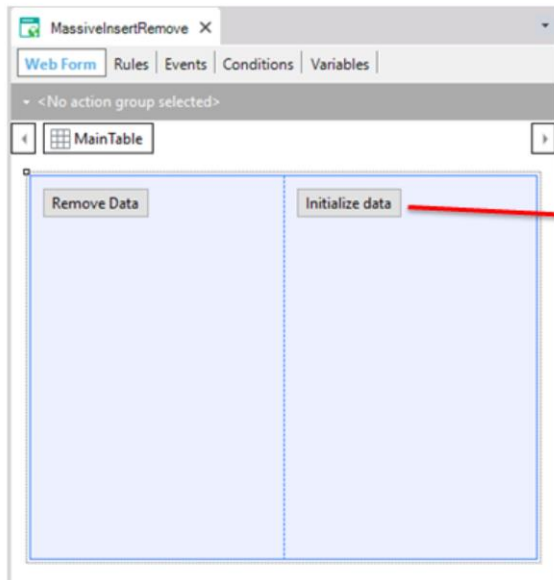
... y clarificamos la notación del data provider

Y además, nosotros no queremos cargar esta colección con datos de la base de datos, sino que queremos asignarles valores nuevos, especificados por nosotros.

Por lo tanto, uno a uno, escribiremos grupos asociados a los ítems de la colección. Como `CategoryId` es un atributo autonumerado, no necesitamos asignarle valor cuando queramos insertar un registro, que será lo que haremos luego, así que directamente borramos esa asignación.

Y como lo que queremos devolver es una colección de nombre `CategoryCollection`, aunque no sea necesario --porque al ponerle a la propiedad `Collection` en valor `True`, el data provider ya sabe que devolverá una colección, a la que llamará de esa manera--, para clarificar el código podemos explicitar lo que GeneXus ya interpreta: encerrando todos los grupos `Category` dentro del grupo `CategoryCollection`, que corresponde a la colección.

Invocamos al Data Provider desde un webpanel



```
Event 'Initialize data'  
  &Categories = Category_DP()  
  &Categories.Insert()  
  Commit  
Endevent
```

Y aplicamos el método Insert() a la colección de categorías

Ahora solamente nos faltará invocar a este data provider desde el evento asociado al botón del webpanel y luego insertar en la base de datos.

Ahora vamos a inicializar la tabla de atracciones


The screenshot shows the GeneXus IDE interface. On the left, the 'Attraction' business component is open, displaying its structure with fields: AttractionId (Id), AttractionName (Name), CountryId (Id), CountryName (Name), CityId (Id), CityName (Name), CategoryId (Id), CategoryName (Name), and AttractionPhoto (Image). On the right, the 'Attraction_DP' data provider is open, showing its source code. A red arrow labeled 'drag & drop' points from the 'AttractionId' field in the business component to the 'AttractionId' assignment in the data provider source code.

```
1 Attraction
2 {
3   AttractionId = AttractionId
4   AttractionName = AttractionName
5   CountryId = CountryId
6   CityId = CityId
7   CategoryId = CategoryId
8   AttractionPhoto = AttractionPhoto
9 }
```

Ahora tendríamos que inicializar la tabla de atracciones. Análogamente crearemos un data provider, Attraction_DP. Arrastraremos la Transacción (de la que ya habíamos obtenido el business component) y vemos que cada elemento del business component queda inicializado por defecto con el atributo correspondiente de la tabla.

Otra vez, vemos que solamente se toman en cuenta los atributos presentes físicamente en la tabla, no los que son inferidos en la transacción o fórmulas

Asignamos los valores nuevos...



```
Attraction
{
  AttractionName = "Louvre Museum"
  CountryId = 2
  CityId = 1
  CategoryId = 1
}
Attraction
{
  AttractionName = "The Great Wall"
  CountryId = 3
  CityId = 1
  CategoryId = 2
}
Attraction
{
  AttractionName = "Eiffel Tower"
  CountryId = 2
  CityId = 1
  CategoryId = 2
}
```

```
Attraction
{
  AttractionName = "Louvre Museum"
  CountryId = Find(CountryId, CountryName="France")
  CityId = Find(CityId, CityName="Paris")
  CategoryId = Find(CategoryId, CategoryName="Museum")
}
Attraction
{
  AttractionName = "The Great Wall"
  CountryId = Find(CountryId, CountryName="China")
  CityId = Find(CityId, CityName="Beijing")
  CategoryId = Find(CategoryId, CategoryName="Monument")
}
Attraction
{
  AttractionName = "Eiffel Tower"
  CountryId = Find(CountryId, CountryName="France")
  CityId = Find(CityId, CityName="Paris")
  CategoryId = Find(CategoryId, CategoryName="Monument")
}
```

...usando el método find() para evitar errores

Como no nos interesa cargar atracciones existentes (pues de hecho ejecutaremos este data provider para cargar los primeros datos), eliminamos todos estos atributos, e ingresaremos a mano estos valores. Además, como el Id es autonumber, tampoco necesitamos asignar valor para ese elemento del business component. Las fotos de las atracciones las asignaremos después, así que también quitamos este atributo.

Como aquí estamos asignando los valores de CountryId, CityId y CategoryId de memoria, quizás no existan en las tablas correspondientes. Si alguno de los valores no existiera, al momento de intentar insertar los registros con el business component, se dispararán los controles de integridad referencial correspondientes y la inserción fallará.

Para evitar asignar valores que pueden no existir, vamos a utilizar la fórmula Find() para encontrar los identificadores correctos a partir de los nombres del país, ciudad o categoría.

Notemos que las fórmulas Find están accediendo a la base de datos solamente a buscar los identificadores correspondientes a los nombres que usamos, pero el resto de los valores asignados al business component son fijos.

Asignamos la propiedad Collection en True...

Data Provider: Attraction_DP

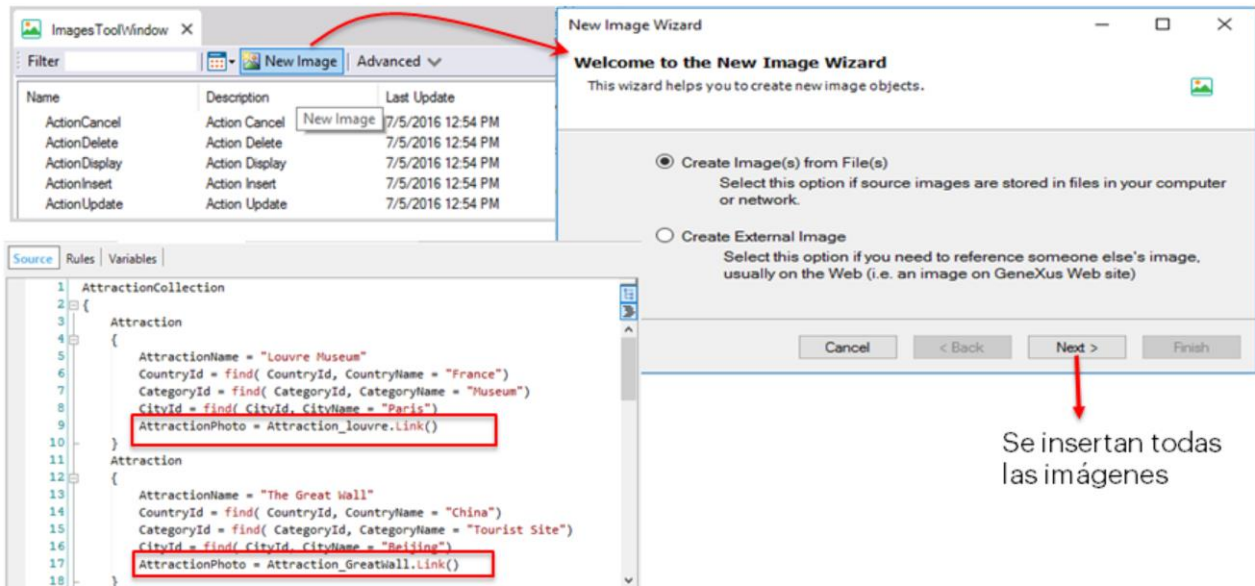
Name	Attraction_DP
Description	Attraction_DP
Expose as Web Service	False
Module/Folder	Root Module
Qualified Name	Attraction_DP
Object Visibility	Public
Output	
Infer Structure	No
Output	Attraction
Collection	True
Collection Name	AttractionCollection
Network	

```
AttractionCollection
{
  Attraction
  {
    AttractionName = "Louvre Museum"
    CountryId = Find(CountryId, CountryName="France")
    CityId = Find(CityId, CityName="Paris")
    CategoryId = Find(CategoryId, CategoryName="Museum")
  }
  Attraction
  {
    AttractionName = "The Great Wall"
    CountryId = Find(CountryId, CountryName="China")
    CityId = Find(CityId, CityName="Beijing")
    CategoryId = Find(CategoryId, CategoryName="Monument")
  }
  Attraction
  {
    AttractionName = "Eiffel Tower"
    CountryId = Find(CountryId, CountryName="France")
    CityId = Find(CityId, CityName="Paris")
    CategoryId = Find(CategoryId, CategoryName="Monument")
  }
}
```

y corregimos la notación para aclarar que se devuelve una colección...

Al igual que hicimos con el data provider de las categorías, debemos poner la propiedad Collection en True, ya que vamos a devolver muchas atracciones y también ajustaremos la notación en el source encerrando los grupos dentro del grupo AttractionCollection, para indicar que es una colección de atracciones.

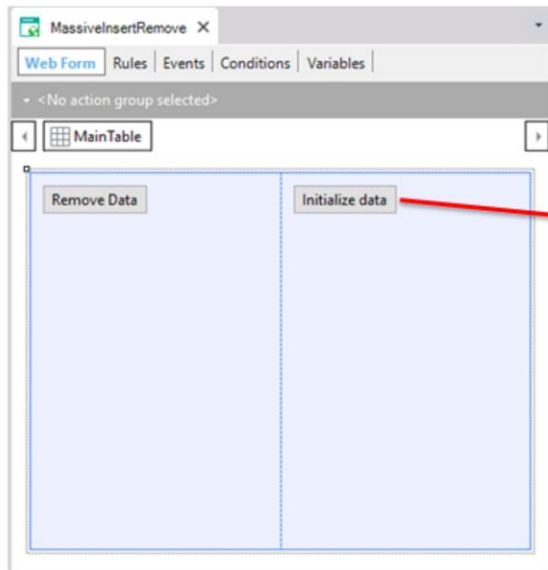
Para cargar las imágenes de las atracciones...



Para cargar también las fotos de las atracciones, una posibilidad es insertarlas primeramente como objetos imagen en la KB...

Y luego para cada grupo del Data Provider, simplemente asignarle a AttractionPhoto el nombre de la imagen punto link.

Agregamos la invocación al DP en el evento del webpanel



```
Event 'Initialize data'  
  &Categories = Category_DP()  
  &Categories.Insert()  
  Commit  
  
  &Attractions = Attraction_DP()  
  &Attractions.Insert()  
  Commit  
Endevent
```

Ahora solamente falta invocar al data provider para que devuelva la colección cargada.

Observemos que para poder insertar las atracciones, primero tenemos que tener creadas las categorías, por lo que el orden es el que usamos en el código del evento.

Resumen

Data Provider

simple / collection

SDT

/

BC

```
&collectionVar = DataProvider()  
&collectionVar. Insert()  
                Delete()
```



Aquí vimos cómo un data provider no solamente permite cargar una estructura con datos de la base de datos, sino también a partir de datos fijos, pero también podrá hacerlo a través de otras fuentes externas, como podrá estudiar en cursos más avanzados.

Además vimos que un data provider permite cargar la estructura de un business component (y no sólo de un SDT), tanto simple como colección.

Por último vimos que en caso de que la estructura sea del tipo colección podremos aplicar métodos que afectan a todos los items de la colección, en una sola operación, como por ejemplo insert() y delete().



Videos

training.genexus.com

Documentation

wiki.genexus.com

Certifications

training.genexus.com/certifications