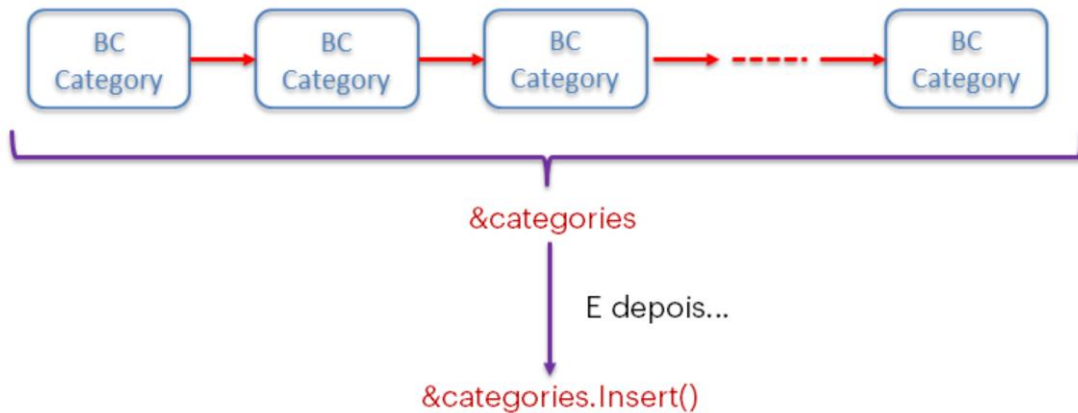


Popular com dados utilizando Business Component e  
Data Provider

*GeneXus™ 16*

**Objetivo: inicializar as tabelas de categorias e atrações com dados**

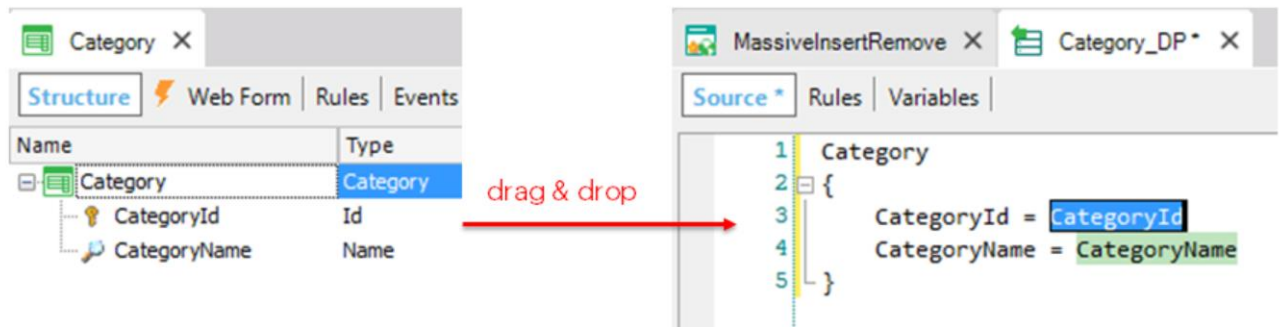
Precisamos:



Se pudéssemos criar uma variável coleção com itens do tipo Business Component de Category, carregada com as categorias que desejamos cadastrar no banco de dados, bastaria aplicarmos o método Insert() nessa variável coleção, pois, como mencionamos no vídeo anterior, isso nos permitirá fazer o Insert de todos os itens, ou seja, de todos os Business Components da coleção.

Para resolver nosso problema, basta declararmos essa coleção. Como fazemos isso?

Para obter a coleção, usamos também um data provider



Até agora sabemos que um Data Provider nos permite retornar dados estruturados, tanto simples como coleção. Em nosso caso, queremos devolver uma coleção de categorias, mas essas categorias não são tipos de dados estruturados (SDTs), mas sim Business Components.

Porém, um Business Component é exatamente igual a um SDT no que diz respeito à estrutura. Por isso, os Data providers também nos permitirão carregar e retornar Business Components, tanto simples como coleções.

Então, arrastamos a transação Category para dentro do Source do Data Provider e vemos que GeneXus nos escreve a estrutura da transação. Vale notar que, à esquerda do sinal =, temos os elementos do Business Component, que estarão em memória. Eles tem o mesmo nome dos atributos, mas não são atributos. Já do lado direito, por padrão, aí sim GeneXus nos mostra os atributos da tabela de onde o Data Provider obterá os dados para carregar o BC que está em memória.

Para ficar do jeito que queremos, o Data Provider deverá devolver uma coleção deste Business Component, já que a tabela tem muitos registros.

Para que o DP nos retorne uma coleção, usamos a propriedade **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

Output

Infer Structure	No
Output	Category
Collection	True
Collection Name	CategoryCollection

Network

Se acessarmos as propriedades do Data Provider, vemos que a propriedade Output assumiu o valor do Business Component, mas a propriedade Collection não está True, como precisamos.

Quando setamos a propriedade Collection para True, aparece a nova propriedade Collection Name, que, por padrão, assume o nome do próprio Data Provider. Vamos alterar o seu valor para CategoryCollection.

Atribuímos novos valores especificados por nós...

The diagram illustrates the transformation of three separate `Category` objects into a single `CategoryCollection` object. On the left, three `Category` objects are listed, each with a `CategoryName` property: "Museum", "Monument", and "Tourist Site". These are separated by red equals signs from a single `CategoryCollection` object on the right. The `CategoryCollection` object contains a list of `Category` objects, each with the same `CategoryName` property values as the original objects.

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

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

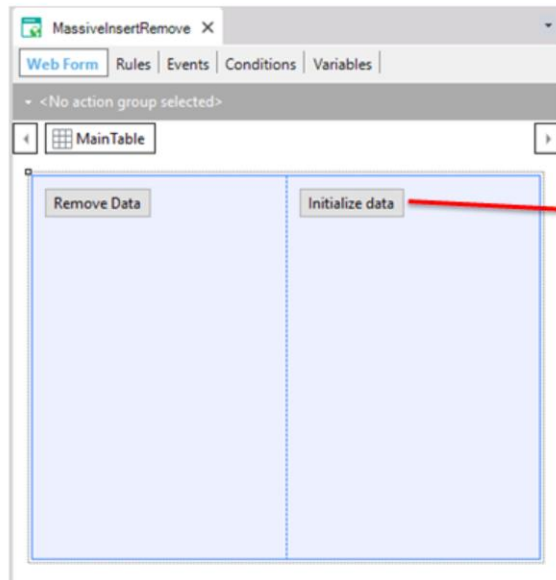
... e deixamos mais clara a codificação do data provider

Além disso, nós não queremos carregar esta coleção com informações do banco de dados, mas sim com novos valores, especificados por nós.

Portanto, um a um, vamos escrever os grupos associados aos itens da coleção. Como `CategoryId` é um atributo autonumérico, não precisamos definir valor para ele já que trata-se de uma inclusão de registro. Então, podemos apagar essa linha do código.

E, como o que queremos retornar é uma coleção chamada `CategoryCollection`, podemos encapsular todos os grupos `Category` dentro de um grupo `CategoryCollection`, que corresponde à coleção. Na verdade, isso não é obrigatório, porque quando colocamos a propriedade `Collection` como `True`, o Data Provider já sabe que devolverá uma coleção. Porém fazemos isso para deixarmos o código o mais claro possível.

## Chamamos o Data Provider de um webpanel



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

E aplicamos o método Insert() na coleção de categorias

Agora falta somente fazer a chamada deste Data Provider através de um evento associado a um botão do WebPanel e depois inserir os novos registros no banco de dados.

## Agora vamos inicializar a tabela de atrações

The screenshot shows two panels in the GeneXus IDE. On the left, the 'Attraction' Business Component structure is displayed in the 'Structure' tab. It lists attributes: 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 shown in the 'Source' tab. It contains a list of attributes with their corresponding database field names: AttractionId = AttractionId, AttractionName = AttractionName, CountryId = CountryId, CityId = CityId, CategoryId = CategoryId, and AttractionPhoto = AttractionPhoto. A red arrow labeled 'drag & drop' points from the 'Attraction' structure to the 'Attraction\_DP' source.

Name	Type
Attraction	Attraction
AttractionId	Id
AttractionName	Name
CountryId	Id
CountryName	Name
CityId	Id
CityName	Name
CategoryId	Id
CategoryName	Name
AttractionPhoto	Image

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

Agora teríamos que inicializar a tabela de atrações. Da mesma forma que no exemplo anterior, vamos criar um novo Data Provider, que chamaremos de Attraction\_DP. Arrastamos a transação Attraction para o Source do Data Provider (a transação já deve estar definida como Business Component) e vemos que cada elemento do Business Component fica inicializado, por padrão, com o atributo correspondente da tabela.

Novamente, vemos que somente os atributos presentes fisicamente na tabela é que são considerados. Os atributos lógicos (inferidos e fórmulas) não aparecem na estrutura do Data Provider.

## Atribuímos os novos valores...



Como não queremos carregar atrações existentes (pois estamos criando esse Data Provider para inicializar a tabela com seus primeiros dados), eliminamos todos esses atributos, e informaremos estes valores manualmente. Como o Id é autonumérico, não precisamos informar um valor para esse elemento do Business Component, portanto excluimos essa linha código. As fotos das atrações serão incluídas depois, então, também excluimos essa linha.

Como aqui estamos informando os valores de CountryId, CityId e CategoryId, que são chaves estrangeiras na tabela de atrações, se algum dos valores informados não existir nas tabelas originais, quando tentarmos inserir os registros com o Business Component, serão disparados os controles de integridade referencial correspondentes e a inclusão falhará.

Para evitar informar valores que podem não existir, vamos utilizar a fórmula Find() para encontrar os identificadores corretos baseados nos nomes de País, Cidade ou Categoria.

Notemos que as fórmulas Find estão acessando o banco de dados somente para buscar os identificadores correspondentes aos nomes que queremos, mas o resto dos valores informados ao Business Component são fixos.



## Setamos a propriedade Collection como 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	

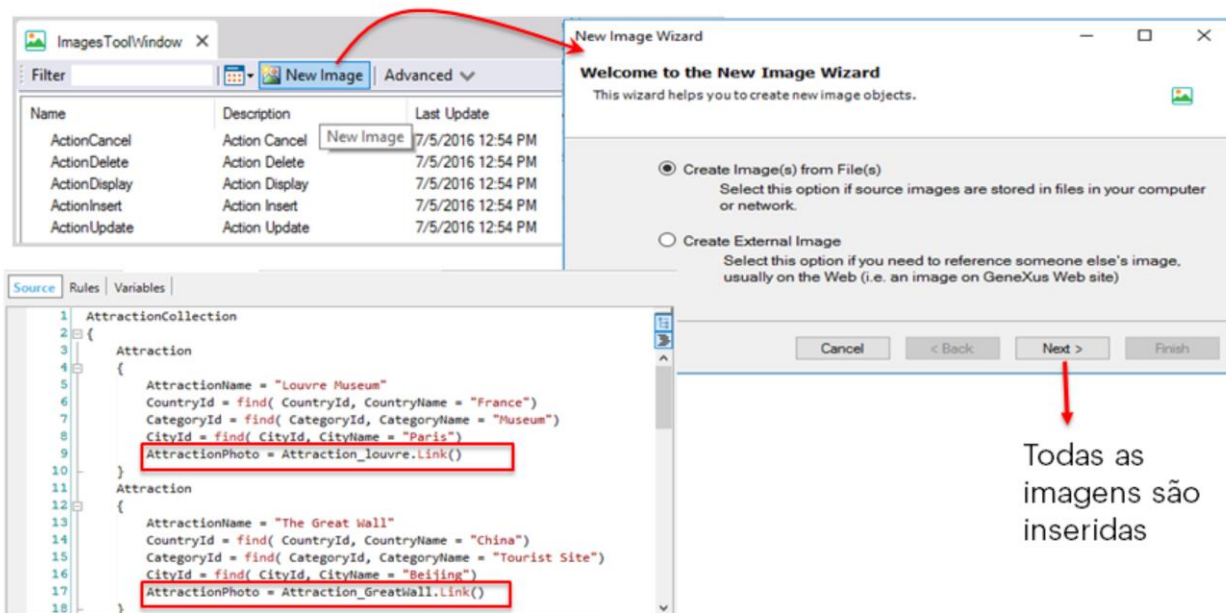
## AttractionCollection

```
{
  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")
  }
}
```

e melhoramos a codificação para deixar claro que retornará uma coleção...

Assim como fizemos com o Data Provider das categorias, devemos alterar sua propriedade Collection para True, já que vamos retornar muitas atrações. Também ajustaremos o Source, encapsulando os grupos dentro do grupo AttractionCollection, para indicar que é uma coleção de atrações.

## Para carregar as imagens das atrações...

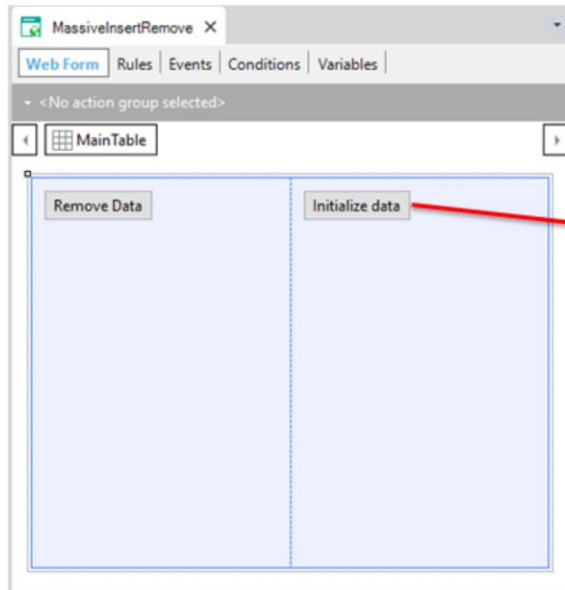


Todas as  
imagens são  
inseridas

Para carregar também as fotos das atrações, uma possibilidade é Inclui-las, primeiramente como objetos do tipo Image na KB...

Depois, para cada grupo do Data Provider, simplesmente podemos atribuir a AttractionPhoto, o nome da imagen da KB seguido de .link()

## Acrescentamos a chamada do DP num evento do webpanel



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

Agora só falta fazer a chamada do Data Provider para que retorne a coleção carregada.

Observemos que, para poder incluir as atrações, primeiro temos que ter as categorias já criadas. Por isso é que escrevemos o código do evento nessa ordem.

## Resumo

## Data Provider

simple / collection

SDT

/

BC

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



Aqui vimos como um Data Provider não somente nos permite carregar uma estrutura com informações do banco de dados, como também a partir de dados fixos. Também permite fazer essa carga através de outras fontes externas, como poderá estudar em cursos mais avançados.

Além disso, vimos que um Data Provider permite carregar a estrutura de um Business Component (e não somente de um SDT), tanto simples como coleção.

Por último, vimos que, nos casos em que a estrutura é do tipo coleção, podemos aplicar métodos que afetam a todos os itens da coleção, em uma única operação, como por exemplo insert() e delete().



Videos

[training.genexus.com](https://training.genexus.com)

Documentation

[wiki.genexus.com](https://wiki.genexus.com)

Certifications

[training.genexus.com/certifications](https://training.genexus.com/certifications)