

**Up-to-date
Development with**

GeneXus[™] 15

October 2017

Copyright 1988 - 2017 | GeneXus S.A.

All rights reserved. This document may not be reproduced by any means without the express consent of GeneXus S.A. The information contained herein is intended for personal use only.

Registered Trademarks

GeneXus is a registered trademark (®) in various countries and regions including, among others, Latin America, United States, Japan, the European Union and Uruguay. All other trademarks mentioned herein are the property of their respective owners.

Objective

GeneXus has grown and evolved substantially over time, offering new and interesting functionalities. Our objective is to make the most of everything it has to offer, and to do so we will avoid programming in the traditional way. Instead, we will adopt the new techniques available and meet our needs in an optimal manner (which is easier, once we get to know it).

Below are some exercises to complete using GeneXus 15, so as to use the solution options available to leverage its power and advantages.

The application we will work with allows managing the sale of tickets to various shows in different countries. It has a Web module and a small module for Smart Devices.

Getting Started

We will use **GeneXus 15 U6**, the **Chrome** browser and Android SDK emulator.

The icon associated with **GeneXus** is on your desktop.

First, start GeneXus and select:

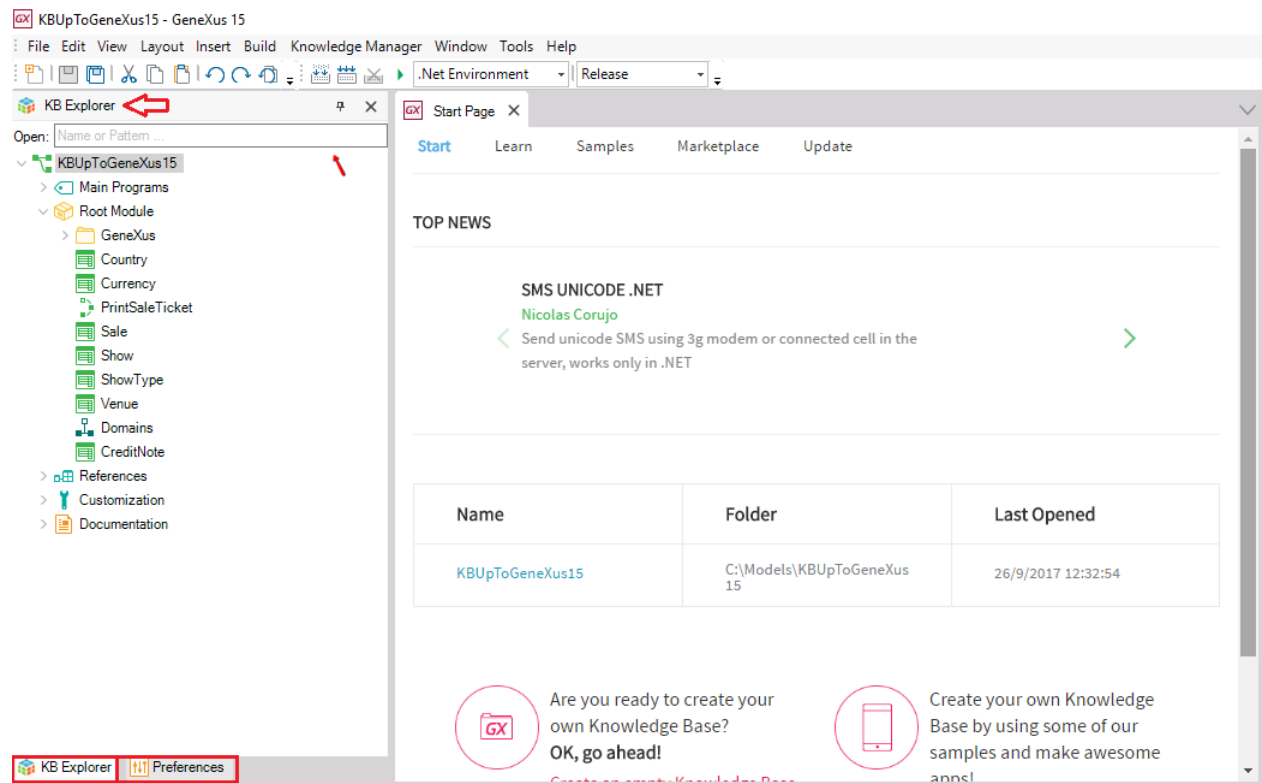
File / New / Knowledge Base from GeneXus Server

With this option we will create a local KB:

- Server: <http://samples.genexusserver.com/v15>
- KB name: **KBUpToDateGeneXus15**

Overview of the GeneXus 15 IDE

Look at the IDE. The section called **KB Explorer** replaces the Folder View section that was available in previous versions. Note that an object finder called “**Open:**” has been added. In addition to letting you quickly find the objects searched for, they can be opened from there.



Transactions created in the KB

Country

For each country, its unique identifier and name are recorded.

Currency

For each currency, its unique identifier (3 characters corresponding to the universal code) and name are recorded.

ShowType

For each type of show, its unique identifier and name (concert, theater play, circus, etc.) are recorded.

Venue

Every location that can host a show is recorded with a unique identifier, a name, a photo (of Image type), the country where it is located, and its corresponding geographic location (of Geography type).

Show

Each show is recorded with a unique identifier, a name, the type of show (concert, theater, play, circus, etc.), the date, the place where it will be held and a representative picture. Also, every show has a currency associated with it.

Shows have different sections enabled, each one with the total number of tickets available and a price. For example: balcony - 300 seats - \$500.

The sections corresponding to the same show are automatically numbered in sequence. The number of tickets sold and the tickets available in each section are always calculated in formula attributes.

Sale

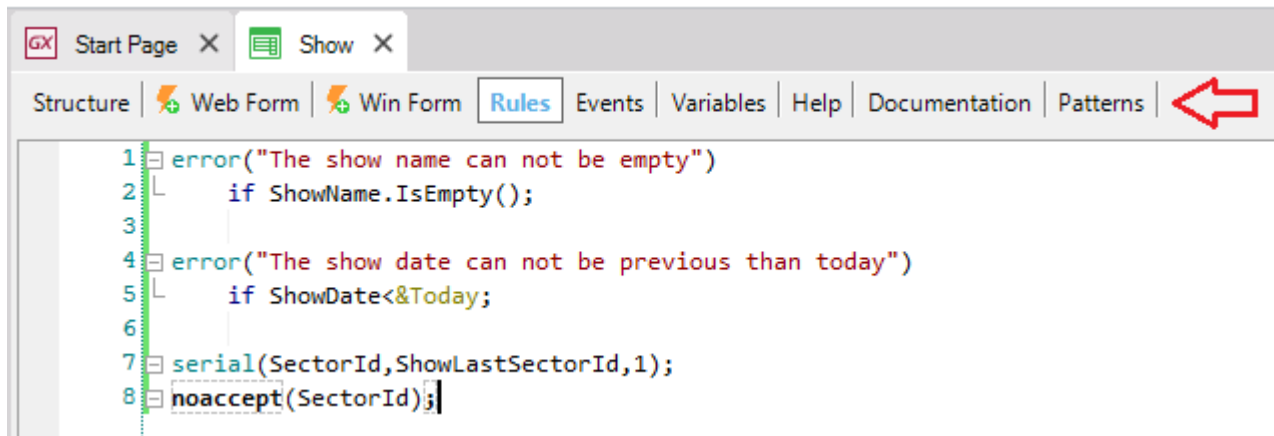
Every ticket sale is recorded with a unique identifier, the date of sale (automatically suggested by the system), the show's details and a purchased ticket. The amount of the sale is stored in the sale transaction. The details of each sale made are printed as an entrance ticket or pass for the show.

CreditNote

When a sale has to be returned/canceled, a credit note needs to be created. Every credit note has a unique identifier, a date, and a sale identifier.

Note:

Transactions have certain rules. Note that in GeneXus 15, the tabs to access the various sections of a transaction are placed above.



```
1 error("The show name can not be empty")
2   if ShowName.IsEmpty();
3
4 error("The show date can not be previous than today")
5   if ShowDate<&Today;
6
7 serial(SectorId,ShowLastSectorId,1);
8 noaccept(SectorId);
```

Take a few minutes to go over the rules defined in the various transactions of the Knowledge Base to familiarize yourself with the application.

Required Functionalities

1) Populate the physical tables indicated with the data provided below.

Currency

UYU, Uruguayan Peso
ARS, Argentine Peso
BRL, Brazilian Real
USD, US Dollar
EUR, Euro

Country

Uruguay
Argentina
Mexico
United States
Japan

ShowType

Concert
Musical Comedy
Circus
Opera
Ballet

Venue

1, Estadio Centenario, -56.152690 -34.894511, Uruguay, Image
2, Teatro Solis, -56.201040 -34.907680, Uruguay, Image
3, Sodre, -56.198283 -34.904363, Uruguay, Image
4, Luna Park, -58.368818 -34.602498, Argentina, Image
5, Auditorio Nacional, -99.194818 19.424795, Mexico, Image
6, Madison Square Garden, -73.993267 40.750525, United States, Image

Guide to populate tables.

For each transaction: Currency, Country, ShowType and Venue; their properties have to be set (they are grouped below the Data group):

- Data Provider = True
- Used To = Populate data

Because the Data Provider property is set to True in each one of the four transactions above, the following will be displayed in the KB Explorer:

- Below the Currency transaction, the Currency_DataProvider object created.
- Below the Country transaction; the Country_DataProvider object created.
- Below the ShowType transaction, the ShowType_DataProvider object created.
- Below the Venue transaction, the Venue_DataProvider object created.

Now you will only have to open each one of the Data Providers that were automatically created and complete them with the data to load in the corresponding physical tables:

Currency_DataProvider X

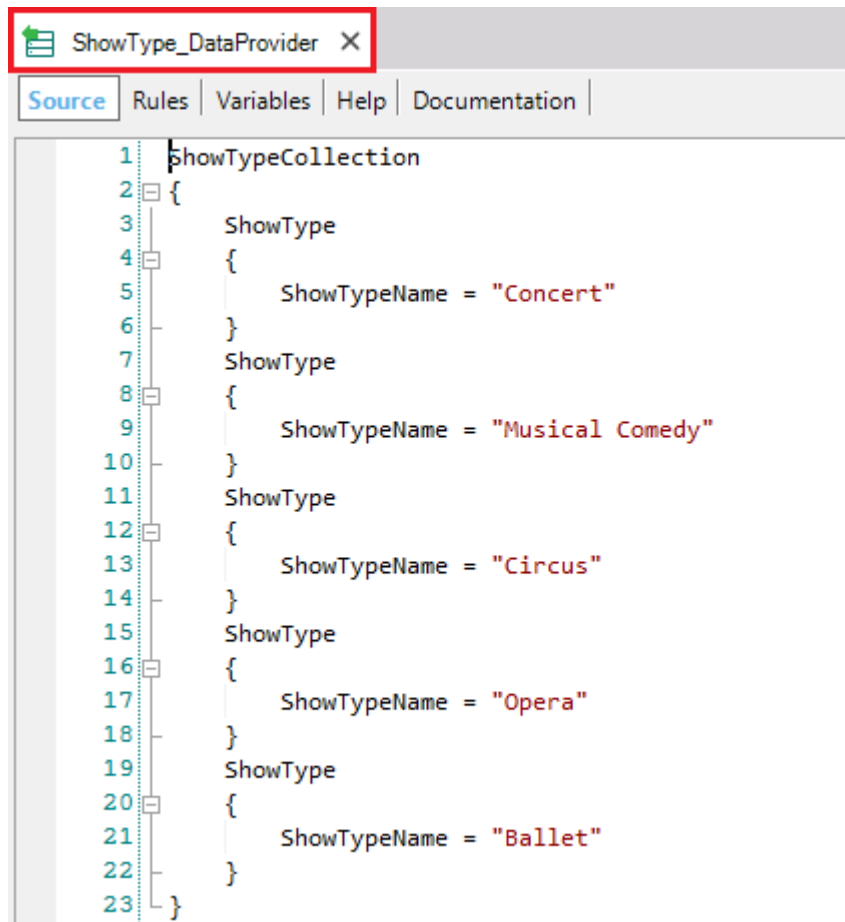
Source Rules Variables Help Documentation

```
1 CurrencyCollection
2 {
3     Currency
4     {
5         CurrencyCode = "UYU"
6         CurrencyName = "Uruguayan Peso"
7     }
8     Currency
9     {
10        CurrencyCode = "ARS"
11        CurrencyName = "Argentine Peso"
12    }
13    Currency
14    {
15        CurrencyCode = "BRL"
16        CurrencyName = "Brazilian Real"
17    }
18    Currency
19    {
20        CurrencyCode = "USD"
21        CurrencyName = "US Dollar"
22    }
23    Currency
24    {
25        CurrencyCode = "EUR"
26        CurrencyName = "Euro"
27    }
28 }
29 }
```



```
1 CountryCollection
2 {
3     Country
4     {
5         CountryName = "Uruguay"
6     }
7     Country
8     {
9         CountryName = "Argentina"
10    }
11    Country
12    {
13        CountryName = "Mexico"
14    }
15    Country
16    {
17        CountryName = "United States"
18    }
19    Country
20    {
21        CountryName = "Japan"
22    }
23 }
```

Note: It has been decided that countries will be autonumbered (CountryId has Autonumber=True), so it makes sense to define a unique index for the Country table by the CountryName attribute. Create this unique index to control that no country name can be repeated. Also, it avoids saving duplicated country names with different identifiers when running the population process several times.



```
1 ShowTypeCollection
2 {
3     ShowType
4     {
5         ShowTypeName = "Concert"
6     }
7     ShowType
8     {
9         ShowTypeName = "Musical Comedy"
10    }
11    ShowType
12    {
13        ShowTypeName = "Circus"
14    }
15    ShowType
16    {
17        ShowTypeName = "Opera"
18    }
19    ShowType
20    {
21        ShowTypeName = "Ballet"
22    }
23 }
```

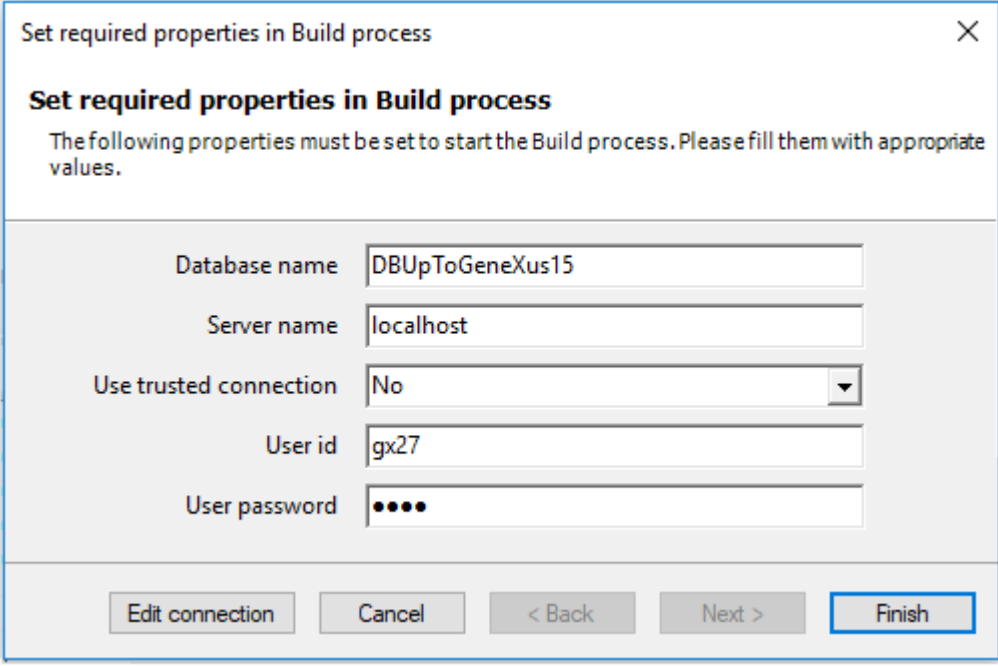
Note: It has been decided that show types will be autonumbered (ShowTypeId has Autonumber=True), so it makes sense to define a unique index for the ShowType table by the ShowTypeName attribute. Create this unique index to control that no show type name can be repeated. Also, it avoids saving duplicated show type names with different identifiers when running the population process several times.

```
1 VenueCollection
2 {
3     Venue
4     {
5         VenueId = 1
6         VenueName = "Estadio Centenario"
7         CountryId = Find(CountryId, CountryName="Uruguay")
8         VenueGeography = Geography.FromString("POINT(-56.152690 -34.894511)")
9         VenueImage = Centenario.Link()
10    }
11    Venue
12    {
13        VenueId = 2
14        VenueName = "Teatro Solis"
15        CountryId = Find(CountryId, CountryName="Uruguay")
16        VenueGeography = Geography.FromString("POINT(-56.201040 -34.907680)")
17        VenueImage = Solis.Link()
18    }
19    Venue
20    {
21        VenueId = 3
22        VenueName = "Sodre"
23        CountryId = Find(CountryId, CountryName="Uruguay")
24        VenueGeography = Geography.FromString("POINT(-56.198283 -34.904363)")
25        VenueImage = Sodre.Link()
26    }
27    Venue
28    {
29        VenueId = 4
30        VenueName = "Luna Park"
31        CountryId = Find(CountryId, CountryName="Argentina")
32        VenueGeography = Geography.FromString("POINT(-58.368818 -34.602498)")
33        VenueImage = LunaPark.Link()
34    }
35    Venue
36    {
37        VenueId = 5
38        VenueName = "Auditorio Nacional"
39        CountryId = Find(CountryId, CountryName="Mexico")
40        VenueGeography = Geography.FromString("POINT(-99.194818 19.424795)")
41        VenueImage = AuditorioNacional.Link()
42    }
43    Venue
44    {
45        VenueId = 6
46        VenueName = "Madison Square Garden"
47        CountryId = Find(CountryId, CountryName="United States")
48        VenueGeography = Geography.FromString("POINT(-73.993267 40.750725)")
49        VenueImage = MadisonSquare.Link()
50    }
51 }
```

Note: The images to be assigned to the Venuelmage attribute (Centenario, Solis, Sodre, LunaPark, AuditorioNacionalMexico, MadisonSquare) are included in the KB. To check this, select **View / Images** option in the main menu of the GeneXus IDE.

Press F5.

The image below shows the data to complete in order to create the local database:



Set required properties in Build process

Set required properties in Build process

The following properties must be set to start the Build process. Please fill them with appropriate values.

Database name	DBUpToGeneXus15
Server name	localhost
Use trusted connection	No
User id	gx27
User password	••••

Edit connection Cancel < Back Next > Finish

User ID: gx27

User password: gx27

- 2) The company's management requests that the application allows issuing a list as shown in the image. That is to say, a Cartesian product between Country and Currency to examine, choose and mark which countries accept which payment currencies.

Uruguay	Argentine Peso	<input type="checkbox"/>
Uruguay	Brazilian Real	<input type="checkbox"/>
Uruguay	Euro	<input type="checkbox"/>
Uruguay	US Dolar	<input type="checkbox"/>
Uruguay	Uruguayan Peso	<input type="checkbox"/>
Argentina	Argentine Peso	<input type="checkbox"/>
Argentina	Brazilian Real	<input type="checkbox"/>
Argentina	Euro	<input type="checkbox"/>
Argentina	US Dolar	<input type="checkbox"/>
Argentina	Uruguayan Peso	<input type="checkbox"/>
Mexico	Argentine Peso	<input type="checkbox"/>
Mexico	Brazilian Real	<input type="checkbox"/>
Mexico	Euro	<input type="checkbox"/>
Mexico	US Dolar	<input type="checkbox"/>
Mexico	Uruguayan Peso	<input type="checkbox"/>
United States	Argentine Peso	<input type="checkbox"/>
United States	Brazilian Real	<input type="checkbox"/>

United States	Euro	<input type="checkbox"/>
United States	US Dolar	<input type="checkbox"/>
United States	Uruguayan Peso	<input type="checkbox"/>
Japan	Argentine Peso	<input type="checkbox"/>
Japan	Brazilian Real	<input type="checkbox"/>
Japan	Euro	<input type="checkbox"/>
Japan	US Dolar	<input type="checkbox"/>
Japan	Uruguayan Peso	<input type="checkbox"/>

Note: Remember that reference can be made to more than one base transaction to solve this request efficiently.

- 3) It is requested that the application allows increasing the price of a certain section of a show. To this end, there has to be a page that allows selecting a show and a certain section enabled for this show; also, the user should be able to enter a percentage. There must be a button that enables the user to process the price increase of that section of the show, applying the increase percentage requested.

The screenshot shows a web application titled 'SHOW Magazine by GeneXus'. Below the title bar, there is a breadcrumb trail: 'Recents' > 'Show1' > 'Show1s' > 'Increase Prices'. The main form area contains three input fields: 'Show Id' with a dropdown menu showing 'Opera Phantom', 'Sector Id' with a dropdown menu showing 'Balcony', and 'Percentage' with a text input field containing '0'. Below these fields is a button labeled 'Increase Price'.

Notes:

- Note that the second Dynamic Combo Box requires that a condition be defined for it, because the sections to display must correspond to the Show previously selected.
- Remember that GeneXus 15 provides the GetByKey method to retrieve a certain record corresponding to a line (the entire structure must have been previously loaded in memory with Load). The section's price must be updated using this method:

Event 'Increase Price'

```
&Show.Load(&ShowId)
&OneSector=&Show.Sector.GetByKey(&SectorId)
&OneSector.SectorPrice = &OneSector.SectorPrice * (1 + &Percentage/100)
&Show.Save()
commit
```

Endevent

4) Dynamic Transactions

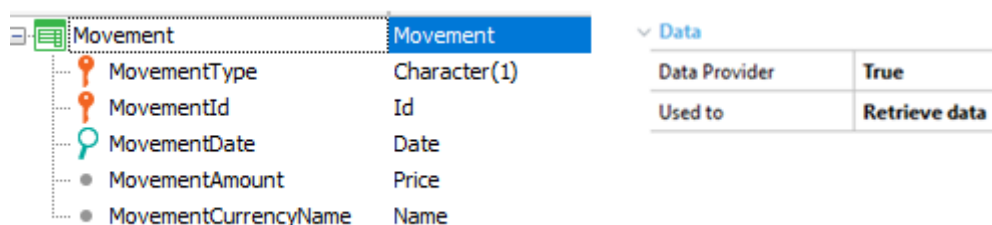
GeneXus 15 transactions have new properties (below the Data group) to be defined as Dynamic Transactions. A distinctive feature of Dynamic Transactions is that they will not cause the creation of physical tables associated with them. Instead, they generate views to retrieve the data at runtime from what we define in the Data Provider associated with the transaction. This allows us to solve many different query scenarios.

4.1) Defining a Dynamic Transaction: Movement

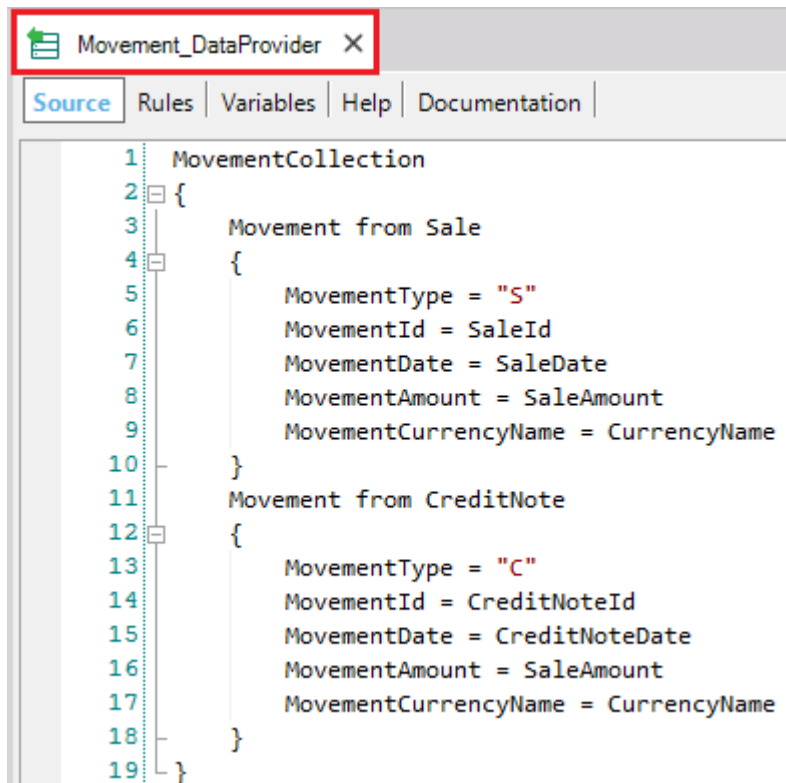
All Sales and all Credit Notes must be joined to count all movements and work with them.

Guide:

- Create a transaction called Movement with the attributes shown in the image below and set the necessary properties for it to be a dynamic transaction.



- Complete the Movement_DataProvider by loading the details of all sales and credit notes in the Movement attributes.



Web - Using the Movement dynamic transaction

Now we will simply use the attributes of the Movement dynamic transaction as we usually do.

- Create a list that shows the movements made today.
- Apply the WW for Web pattern to the Movement transaction.
- Include an action in the WW for Web pattern (outside the grid) that invokes the list.

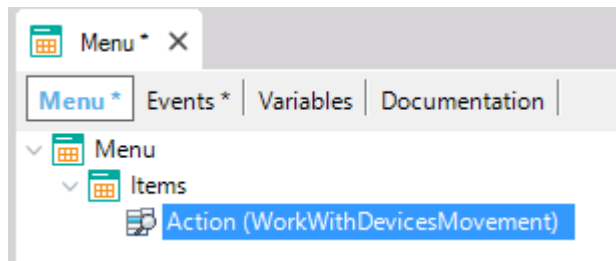
SD - Using the Movement dynamic transaction

- Apply the WW for Smart Devices pattern to the Movement transaction.
- In the grid of the List node, add the Movement attributes that are not included by default.

Note: For the attributes inside the grid not to have a label, set the **Label Position** property of each attribute included in the grid to **None**.

- Create a Menu for Smart Devices object (remember that the object called **Dashboard** in previous versions has been changed and is currently called: **Menu for Smart Devices**).

Below the **Items** node, add an **Action** that invokes the object → **WorkWithDevicesMovement**



Run the **Menu for Smart Devices** that you have defined.

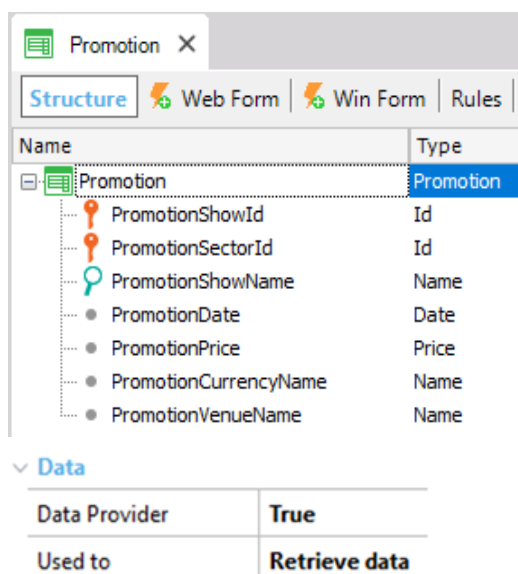
4.2) Defining a Dynamic Transaction: Promotion

The company creates promotions in this way: For those shows whose date is within 5 days or less, tickets will be sold with a 50% discount from the original price, in particular those sections that have 100 or more tickets available.

Create a dynamic transaction to model promotions according to the above description.

Guide:

- Create a transaction called Promotion with the attributes shown in the image below and set the necessary properties for it to be a dynamic transaction.



- Complete the Promotion_DataProvider by loading in the Promotion attributes the details of all the sections of shows included in a promotion when the query is made:

```

1 PromotionCollection
2 {
3     Promotion from Show.Sector
4         where (ShowDate-ServerDate()) <= 5 and (ShowDate-ServerDate()) >=0
5         where SectorAvailableQuantity >=100
6     {
7         PromotionShowId = ShowId
8         PromotionSectorId = SectorId
9         PromotionShowName = ShowName
10        PromotionDate = ShowDate
11        PromotionPrice = SectorPrice/2
12        PromotionCurrencyName = CurrencyName
13        PromotionVenueName = VenueName
14    }
15 }

```

Web - Using the Promotion dynamic transaction

Now we will simply use the attributes of the Promotion dynamic transaction as we usually do. In particular, create a web panel that shows the details of the current promotions.


50 %

Promotion Date	Promotion Show Id	Promotion Show Name	Promotion Sector Id	Sector Name	Promotion Venue Name	Promotion Price	Promotion Currency Name
10/03/17	1	Madonna in Concert	1	Orchestra Platinum	Madison Square Garden	100	US Dollar
10/03/17	1	Madonna in Concert	2	Orchestra Gold	Madison Square Garden	75	US Dollar
10/06/17	2	David Bisbal - Tour 2017	1	America Rosrtrum	Estadio Centenario	3500	Uruguayan Peso

Enjoy GeneXus 15 and the Course!