

Interactive Screens
Web Panel Object

GeneXus® 16

Web Panel Object

- Implements a highly flexible web screen
- Example:

EnterAttractionsFilter X

Web Form Rules Events Conditions Variables

<No action group selected>

MainTable

Country Id &CountryId v

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

List Attractions By Country List Attractions By Name

Variables: input controls
(not readonly)

Web panels are the most versatile objects provided by GeneXus.

As we've seen in some examples, all web panels have a web form consisting of a web page that enables us to design and provide a variety of functionalities.

In the example we saw that the variables that we include in the web form are enabled for the user to assign values to them. This means that they are input controls; in other words, they are not read-only.

Example: variables in Web panel

The screenshot displays the GeneXus IDE interface for a web panel named 'EnterAttractionsFilter'. The design view on the left shows a form with a 'Country Id' dropdown menu, two text input fields for 'Attraction Name From' and 'Attraction Name To', and two buttons: 'List Attractions By Country' and 'List Attractions By Name'. The 'List Attractions By Country' button is highlighted with a red box. A red arrow points from the 'Country Id' dropdown to the 'Properties' window on the right. Another red arrow points from the 'List Attractions By Country' button to a code block in the 'Events' tab.

The 'Properties' window on the right shows the configuration for the 'Country Id' dropdown menu. The 'Attribute/Variable' is set to '&CountryId'. The 'Control Type' is 'Dynamic Combo Box'. The 'Data Source From' is 'Attributes'. The 'Item Values' are 'CountryId'. The 'Item Descriptions' are 'CountryName'. The 'Sort Descriptions' are 'True'. The 'Conditions' are empty. The 'Instantiated Attributes' are empty. The 'Empty Item' is 'False'. The 'Notify Context Change' is 'False'. A red arrow points from the 'CountryName' property to a database icon labeled 'Country Table'.

```
Event 'List Attractions By Country'
  AttractionsList(&CountryId)
Endevent
```

Specifically, this variable, of the dynamic combo type, expected the user to select one country from those loaded in the combo. After pressing the button “List Attractions By Country”, the associated event would be executed invoking the PDF file containing a list of attractions in that country.

Here, the database is accessed only to load the combo’s values.

Example: variables in Web panel

The screenshot displays the GeneXus IDE interface for configuring a web panel. The main window shows the 'EnterAttractionsFilter' web panel with the following elements:

- Country Id**: A dropdown menu with the value `&CountryId`.
- Attraction Name From**: A text input field with the value `&AttractionNameFrom`.
- Attraction Name To**: A text input field with the value `&AttractionNameTo`.
- List Attractions By Country**: A button.
- List Attractions By Name**: A button.

The 'List Attractions By Name' button is highlighted with a red box. Below it, the event 'List Attractions By Name' is defined with the following code:

```
Event 'List Attractions By Name'  
  AttractionsByName( &AttractionNameFrom, &AttractionNameTo )  
Endevent
```

The 'AttractionsByName' component is shown in the bottom right, with the 'Rules' tab selected. The rule is defined as:

```
1 param( in: &NameFrom, in: &NameTo );
```

Red arrows point from the `&NameFrom` and `&NameTo` variables in the rule to the `&AttractionNameFrom` and `&AttractionNameTo` variables in the event code, indicating the variable resolution.

In these other variables, the user would enter a range of attraction names so that when this other button is pressed the PDF list showing the attractions within the range received by parameter would be invoked.

Example:

The screenshot displays the GeneXus IDE interface for a web panel named 'AttractionsByName'. The top window shows the 'Layout' tab, which includes a title bar, a table with columns 'Id', 'Name', 'Country', and 'Photo', and a section titled 'Attractions List'. The bottom window shows the 'Source' tab with the following code:

```
1 param( in: &NameFrom, in: &NameTo );  
2  
3 Output_file('AttractionsList.pdf', 'pdf');  
4  
5  
6  
7
```

The code is highlighted with a red box, indicating the logic for filtering attractions by name. The text 'Can we implement this list in the previous web screen, which prompted the user for the filters?' is written to the right of the layout window.

Remember the layout. In the Source, we programmed the database query with the For Each, filtering by name.

But, why do we define these queries through PDF listings instead of doing it directly on the screen where the user is required to enter data for the filters?

Let's change the web panel so that it can query the DB

The screenshot shows the GeneXus IDE interface for a web form titled 'EnterAttractionsFilter'. The form is in 'Web Form' mode and contains several input fields: 'Country Id' with a dropdown menu, 'Attraction Name From', and 'Attraction Name To'. Below these fields, there are two buttons labeled '< Back' and 'Next >', which are crossed out with red lines. A red arrow points from the text 'Grid with attractions' to the area where the buttons were. To the right of the form, there is a database icon and a table structure for 'Attractions'.

Grid with attractions

Attractions
AttractionName
CountryName
AttractionPhoto

Why not add here a grid showing the desired attractions instead of the buttons?

The rows of the grid will be similar to the printblock that shows each attraction.

Web panels: interactive queries to the database

WWAttractionsFromScratch * X

Web Form * Rules Events Conditions Variables

<No action group selected>

MainTable Grid1

Country Id &CountryId

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

Id	Attraction Name	Country	Photo
AttractionId	AttractionName	CountryName	Photo

Attributes in web panel are **output** attributes: **readonly**



Apart from allowing the definition of variables to be used in actions programmed in buttons, web panels allow us to implement **interactive** queries to the database, which is in fact their main purpose.

“**Interactive**” implies that it is possible for the user to enter many different values – **in variables** – in the web page and then query the database for data matching the values entered, using them as filters, as we will see next.

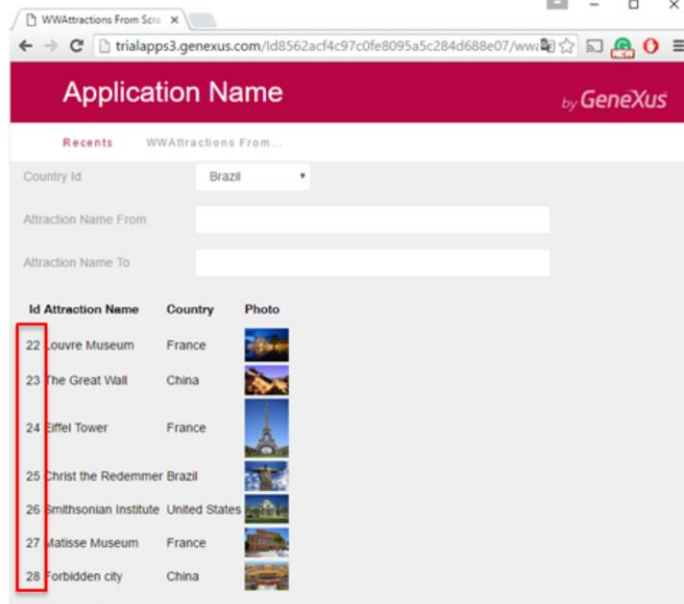
Let's now save this web panel with a different name. We will be implementing something similar to a Work With element, so we will call it WWAttractionsFromScratch.

We remove the buttons and the associated events because they will no longer be necessary. Now, we insert a control of the grid type below the variables.

A screen opens up in order to select the attributes and/or variables that will be this grid's columns. Since we want to show the same we showed on the PDF listings, we select the AttractionId, AttractionName, AttractionPhoto and CountryName attributes and then press OK. We will then see that a grid has been created with these columns. We may change the column titles by editing the properties of each attribute comprised in the grid columns.

The **attributes** in a web panel form are, by default, **output** attributes. That is to say that they are **readonly**. This means that GeneXus understands that it must go to the database to search for their value and show it to the user.

Web panels: interactive queries to the database



Let's press F5 to run our new web panel, just as we have it so far.

We can see that all the attractions have been printed, with the data we indicated (ID, name, country and photo). We will also see that they have been ordered by AttractionId.

Grid with attributes is the equivalent to a For each command

The screenshot displays the GeneXus IDE interface with three main components:

- Web Form:** A form titled "AttractionsByForm" with a "Country Id" dropdown menu, two text input fields for "Attraction Name From" and "Attraction Name To", and a "GRID" control. The grid has four columns: "Id", "Attraction Name", "Country", and "Photo".
- Code Window:** A code editor showing a "For each Attraction" loop. The code is as follows:

```
1 print Title
2 print ColumnTitles
3 For each Attraction endfor
4   print Attraction
5 endfor
```
- Properties Window:** A properties panel for the "Grid: Grid1" control. The "Base Trn" property is highlighted in red and set to "Attraction".

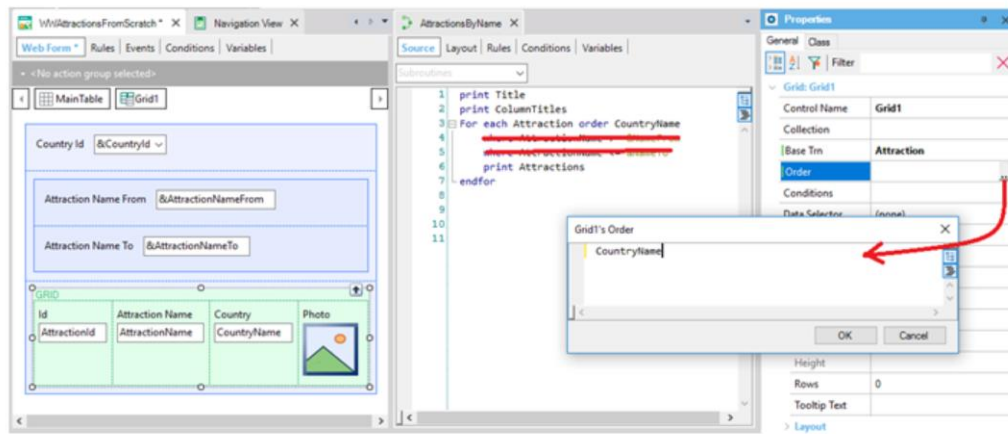
A red arrow points from the "Base Trn" property in the Properties window to the "Base transaction" text below the grid. Another red arrow points from the "For each Attraction" loop in the code window to the same "Base transaction" text.

Base transaction

The mere grid with these attributes drove GeneXus to understand that it should go to the database to navigate the Attraction table, and then access Country to bring the country of the attraction, as we did with the For Each command (without the order and where clauses).

We can see that one of the grid properties is called **Base Trn**. This property is similar to the base transaction of the For Each command. In fact, to make sure that the Attraction base table is selected for the grid, as we want, we should indicate the base transaction, just like for the For Each command.

Grid with attributes is the equivalent to a For each command



Order

In addition, note that an **Order** property is available for the grid. This property corresponds to the Order clause of the For Each command.

So, if we want to order by country name, as in the listing, in the Order property we have to write the CountryName attribute.

Grid with attributes is the equivalent to a For each command

Application Name

Recents WWAttractions From...

Country Id Brazil

Attraction Name From

Attraction Name To

Id Attraction Name	Country	Photo
25 Christ the Redeemer	Brazil	
23 The Great Wall	China	
28 Forbidden city	China	
22 Louvre Museum	France	
27 Matisse Museum	France	
24 Eiffel Tower	France	
26 Smithsonian Institute	United States	

Web Panel WWAttractionsFromScratch Navigation Report

Name WWAttractionsFromScratch Environment Default (C#)

Description WWAttractions From Scratch Spec. Version 15_0_1-106638

Form Class HTML

Program Name WWAttractionsFromScratch

Parameters

Warnings

spc0038 There is no index for order CountryName; poor performance may be noticed in grid 'Grid1'.

FILL &CountryId with CountryId, CountryName in

=Country (CountryId) INTO CountryId CountryName

Order CountryName

Event Load

Order: CountryName

No index

Navigation Start FirstRecord

filters: from: NotEndOfTable

Loop while: NotEndOfTable

Join location: Server

Attraction (AttractionId)

Country (CountryId)

Base table of the Grid

We press F5, and we will see that the grid is now ordered by country name.

Note that the navigation list of the web panel here indicates the navigation that has to be made to load the combo box of the &CountryId variable, which we have not used at all for the time being.

And, here, it indicates the navigation that will have to be made to load the grid. The list for this loading is identical to the list for a For Each command. It has selected the Attraction table, running it through CountryName - the attribute of the Order property. It will run through the entire table, and for each record in Attraction to be loaded, it will access the Country table to show the CountryName for the attraction.

Filter grid data

The screenshot shows the GeneXus IDE interface. The main window displays a form with the following fields:

- Country Id: &CountryId
- Attraction Name From: &AttractionNameFrom
- Attraction Name To: &AttractionNameTo

Below the form is a grid control named Grid1. The grid has the following columns:

- Id
- Attraction Name
- Country
- Photo

The Properties panel on the right shows the configuration for Grid1. The Conditions property is set to 'CountryId = &CountryId'. A red arrow points from the grid's photo column to the Conditions property in the Properties panel.

The Source tab shows the following code snippet:

```
1 print Title
2 print ColumnTitles
3 For each Attraction order CountryName
4   where CountryId = &CountryId
5   print Attractions
6 endfor
```

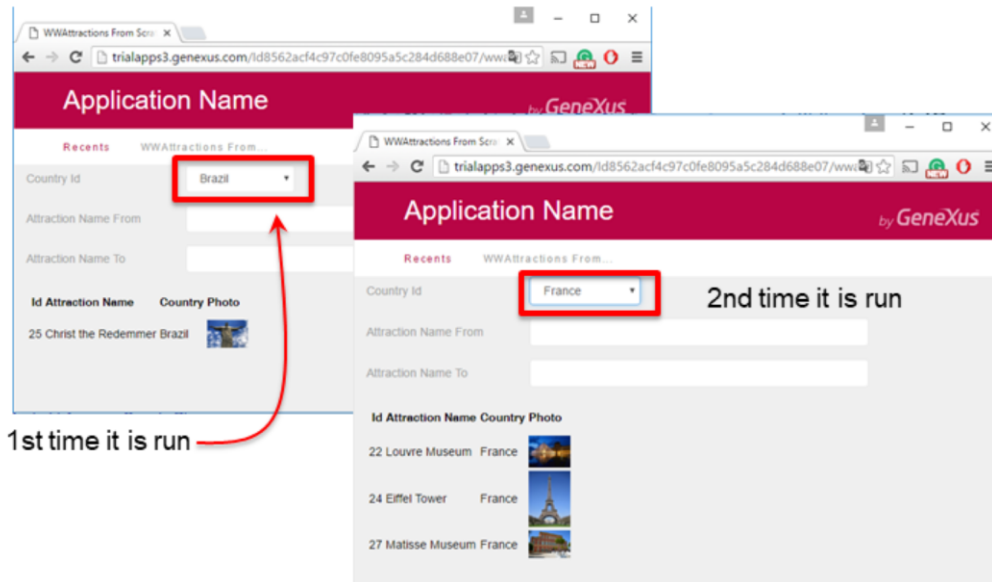
Filter conditions

F5

So far, we haven't done anything with variables. But we wanted to use them to filter the data displayed in the grid, by country and as well as by attraction name.

In AttractionsList we filtered by country. How do we indicate this filter for the grid? Through the **Conditions** property.

Filter grid data



Note that, by default, the combo box takes the value Brazil, and that the grid only shows an attraction in Brazil.

If we select France, we see that the screen is refreshed and the grid is loaded again, this time with the attractions in France.

Conditional filtering of grid data

The screenshot shows the GeneXus IDE interface with three main panels: Web Form, Source, and Properties.

Web Form: Displays a form with a 'CountryId' dropdown, 'Attraction Name From' and 'Attraction Name To' text boxes, and a 'GRID' table. The grid has columns: Id (AttractionId), Attraction Name (AttractionName), Country (CountryName), and Photo. A red arrow points from the 'CountryId' dropdown to the 'Empty Item' property in the Properties window.

Source: Shows the following code:

```
1 print Title
2 print ColumnTitles
3 For each Attraction order CC
4   where CountryId = &CountryId
5   print Attractions
6 endfor
```

Properties: The 'Filter' tab is active. The 'Control Info' section shows:

- Control Type: Dynamic Combo Box
- Data Source From: Attractions
- Item Values: CountryId
- Item Descriptions: CountryName
- Sort Descriptions: True
- Conditions: (empty)

The 'Instantiated Attributes' section shows:

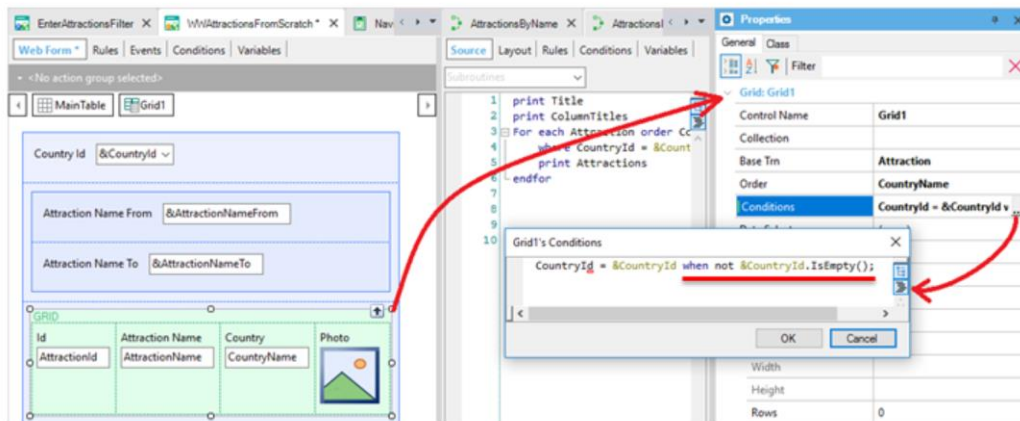
- Empty Item: True
- Empty Item Text: GX_EmptyItemText

A red box highlights the 'Empty Item' and 'Empty Item Text' properties. Below the Properties window, the text 'Value: "(None)"' is displayed.

We will probably want the combo to be first displayed without a selected value, with the attractions of all countries displayed.

To do this we must edit the combo box properties... and set the Empty Item property to True. This will add a "(none)" option to the combo. It will correspond to an empty value.

Conditional filtering of grid data

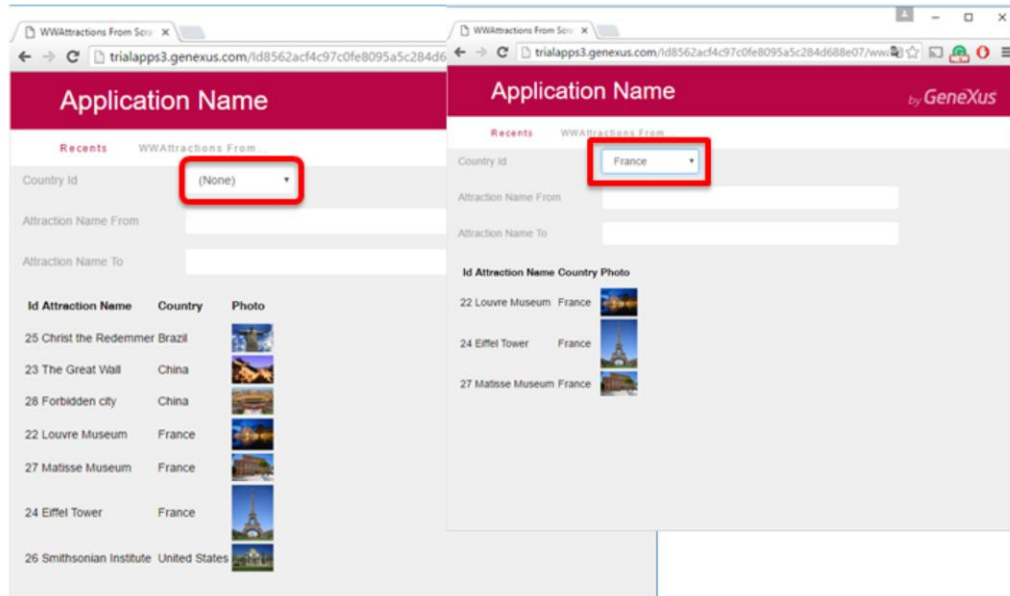


Conditional conditions

F5

So, we open the **Conditions** property and indicate that we want to have this condition applied only when the combo's value is not empty. When it is empty, the condition should not be applied.

Conditional filtering of grid data

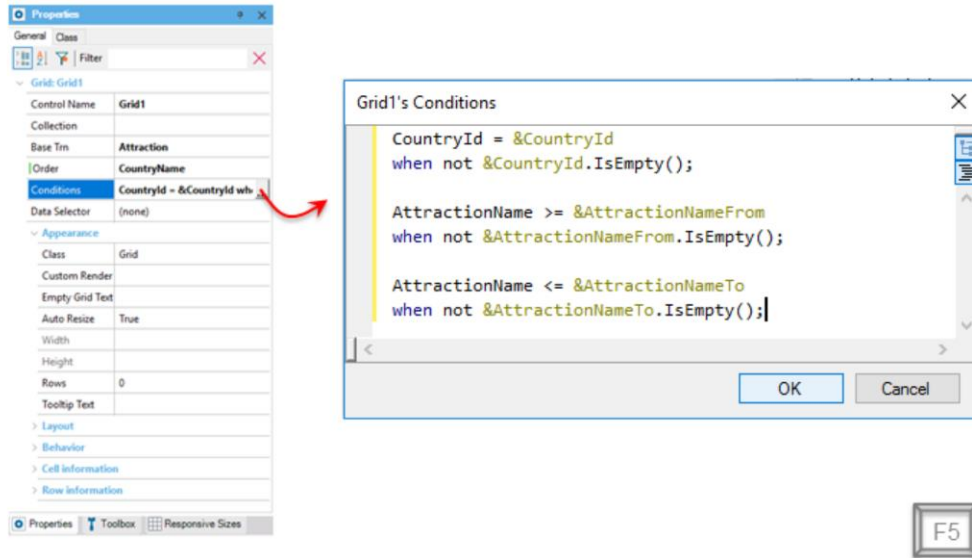


We run it...

...and see how the (None) value is displayed in the combo. In addition, in this case there is no filtering for the attractions, and all of them are displayed.

If now we choose, France, for example, since the variable's value is not empty, the filter is applied and the attractions in France are displayed.

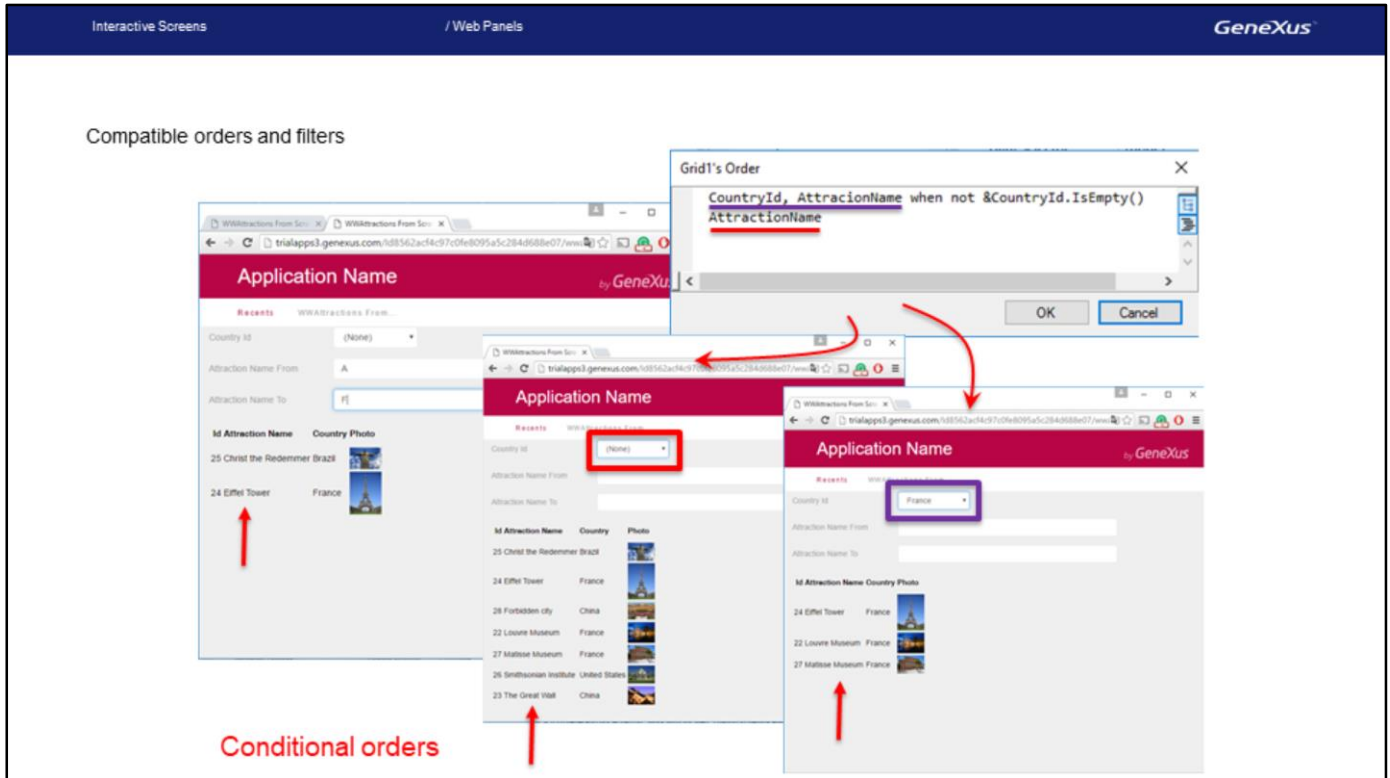
Filtering grid data by multiple conditions



We would also have to add the filters by attraction name that we want added to the other filter. So, if in the listing we filtered in the For Each command using these two Where clauses... we will add them to the grid as conditions.

AttractionName greater than or equal to the value of the &AttractionNameFrom variable of the form, which may be entered by the user. Once again, if the user doesn't enter a value in the variable, we will not want this filter to be applied. So, we use the When clause. This clause may also be used in the Where clause of the For Each command, in a completely analogous way.

We add the other filter.



We run again. And now we choose to see the attractions between A and F.

For the case when the user selects a country, we may instruct the web panel to then sort the information by CountryId, and inside CountryId by AttractionName, or otherwise, by AttractionName. This is meant to optimize the search of table records.

To do it, we edit the Order property of the grid and write the order first, conditioned to the selection, by the user, of a country in the combo. In this case, the data will be filtered by that country, and also the attractions will be listed in alphabetical order for that country. If the user left the combo empty, with the "(none)" value, then the following order –by AttractionName– will be selected.

We will not go into further detail here. The purpose of mentioning this was just to show that we can also condition the way in which we want to have information ordered. It works exactly like in a For Each command.

We run it...Here, it ordered by AttractionName. And if we select France, it will order by France's ID, and inside it, by AttractionName.

In sum, attractions will always be shown in alphabetical order.

If, within France's attractions, we want to see those between the letters A and F, we will see that the grid will load filtering by the three conditions we had added.

In summary

The screenshot shows a web panel design on the left and the Properties window for 'Grid: Grid1' on the right. The web panel includes a 'CountryId' dropdown, two text input fields for 'Attraction Name From' and 'Attraction Name To', and a 'GRID' control. The 'GRID' control has columns for 'Id', 'Attraction Name', 'Country', and 'Photo'. Below the grid, a database icon is shown with the text '≈ For each Extended table'. The Properties window for 'Grid: Grid1' shows the following properties:

Property	Value
Control Name	Grid1
Collection	
Base Trn	Attraction
Order	CountryId, Attraction...
Conditions	CountryId = &Country...
Data Selector	(none)

Red annotations highlight the 'Base Trn' property as the 'Base table' and the 'Order' and 'Conditions' properties as '≈ For each Order Where'.

We implemented a web panel where, in addition to including some variables for which the user enters values, we also inserted a Grid control with attributes.

The attributes correspond to information in the database, so GeneXus understands that it has to search for it. A grid with attributes is like a For Each, so, we have the **Base Trn** property, as in the case of a For Each, to specify the level of the transaction whose associated table we want to run through. This table is called **grid base table**. When we don't specify it, as it may also be the case with a For Each command, then GeneXus will infer it based on the attributes used. However, we will not be considering such case at this point.

All the grid attributes will have to belong, as in a For Each command, to the extended table of that base table. Just like in a For Each we order data with the Order clause and filter the data to be returned by the query with one or several Where clauses, in order to do the same with the grid we have, respectively, the Order and Conditions properties.

Grid loading

< No action group selected >


MainTable Grid1

Country Id &CountryId ▾

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

Id	Attraction Name	Country	Photo
AttractionId	AttractionName	CountryName	

In PDF procedure

For each
Main_Code { Print Attractions
endfor

Attractions			
Id	AttractionName	CountryName	AttractionPhoto
AttractionId	AttractionName	CountryName	AttractionPhoto

In a For Each command we program what we want to do with each record that meets the conditions inside its body.

For example, in the list of tourist attractions, the print Attraction command sends for print, in the output, what in this case would be the grid line, but in the case of the grid, we need not indicate it because it is done automatically.

Grid loading: Load event

< No action group selected >


MainTable Grid1

Country Id &CountryId

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

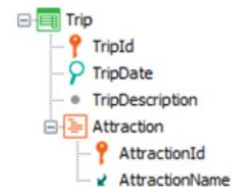
Id	Attraction Name	Country	Photo
AttractionId	AttractionName	CountryName	
			

In PDF procedure

For each
Main_Code
 endfor

&trips = Count(TripDate)
 Print Attractions

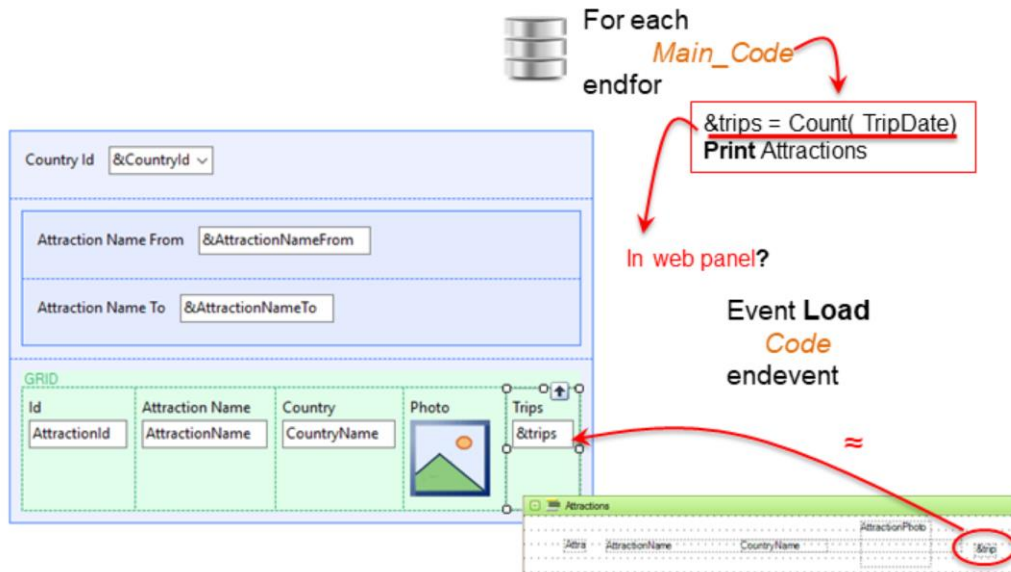
Attractions			
Attr	AttractionName	CountryName	&trips



But let's suppose, for example, that we have a Trip transaction that records the trips offered by the travel agency. In a very simplified way, suppose that for every trip we only record the date on which it will take place, and its description; and then the tourist attractions that will be visited during that trip are recorded. Let's also suppose that in the list of tourist attractions we also want to see the number of trips associated with each attraction. To do so, we only needed to define a numeric variable &trips, and inside the body of the For Each command (that is to say, when the For Each command is positioned on the record of its base table about to be processed) assign it the result of counting the trips of that attraction, to then include that variable in the print block.

Grid loading: Load event

In PDF procedure



To do the same in the web panel, we right-click on the grid and select Insert Attribute/Variable. We create the &trips variables, with type Numeric(4,0), and we move it to the position where we want inside the grid. This corresponds to having inserted the variable in the print block. But, where do we indicate how the calculation is done? In the case of the For Each command, we did inside its body. And where do we do it in this case?

To do this we have the system's **Load** event.

Grid loading: Load Event



```
For each  
Main_Code  
endfor
```

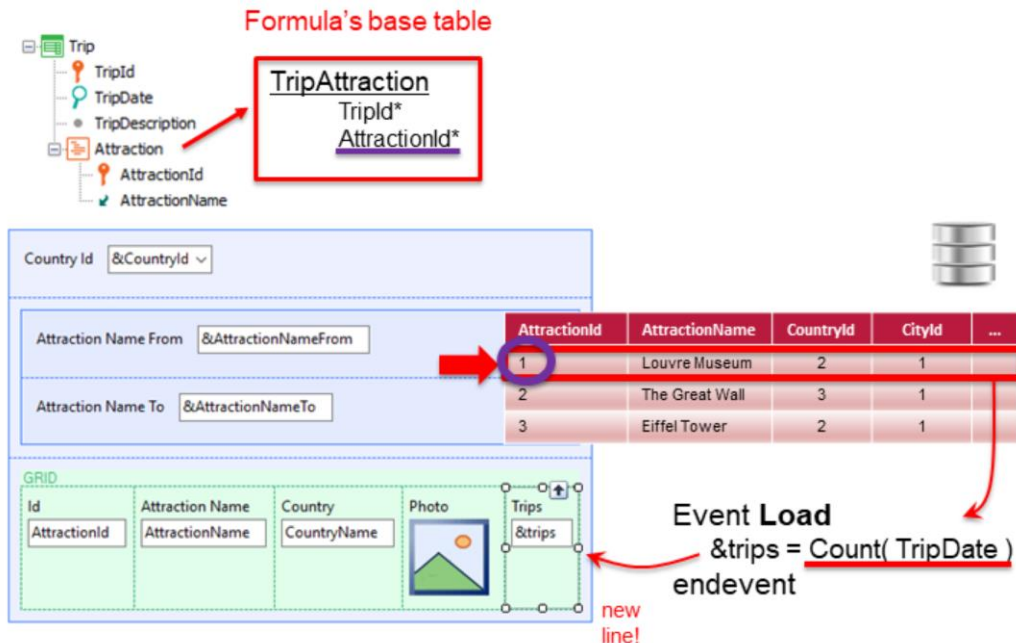
```
&trips = Count( TripDate)  
Print Attractions
```



Inside it we program what we want executed when we're positioned on a record of the grid base table, right before the corresponding line is loaded in the grid.

In this case, there is where we would assign a value to the &Trips variable.

The Load event will be automatically executed for every record of the grid base table that meets the filtering conditions, immediately prior to the addition of the line to the grid.



That's why, when its code is executed, we know we're working with a record from the base table and its extended table; and this inline formula will not count all the trips. Instead, it will count only those from the TripAttraction table that correspond to this AttractionId, that of the Attraction record that we're about to load in the grid.

Note that even though the TripDate attribute belongs in the Trip table, GeneXus will not choose the Trip table as the formula's base table, but the TripAttraction table. We will not go into this here, but TripDate is in the extended table of TripAttraction, table which has a relationship to the Attraction table. GeneXus looked for a table which allowed relating the data.



DEMO

The screenshot shows a web application interface with a red header bar containing 'Application Name' and 'by GeneXus'. Below the header, there's a 'Recents' section with a 'Trip' dropdown. The main content area features a table with columns: 'Id', 'Attraction Name', 'Country', 'Photo', and 'Trips'. The table lists several attractions with their respective trip counts. A red box highlights the 'Trips' column, and a red arrow points from the text 'Variable outside the grid to total?' to the 'Total Trips: 6' summary at the bottom.

Id	Attraction Name	Country	Photo	Trips
25	Christ the Redemmer	Brazil		2
24	Eiffel Tower	France		2
28	Forbidden city	China		0
22	Louvre Museum	France		0
27	Matisse Museum	France		1
26	Smithsonian Institute	United States		1
23	The Great Wall	China		0

Total Trips: 6



Let's implement it in GeneXus. We've already created the Trip transaction. Now we go to the events section of the Web Panel. In this combo box, we're offered predefined events; that is to say, system events that are generated at specific moments, and for which we may program code.

Among them is the Load event. Here we will program what we want executed every time we're positioned on a record from the Attraction table, before loading the line in the grid.

We run it...

Now we want to add the sum of trips in which the attractions displayed on the grid are included. That is to say, the sum of the values in this column:

Variable outside the grid to calculate the total

The screenshot shows a web panel with a form and a grid. The form includes a 'Country Id' dropdown, 'Attraction Name From' and 'Attraction Name To' text boxes, and a 'GRID' section. The grid has columns for 'Id', 'Attraction Name', 'Country', 'Photo', and 'trips'. Below the grid is a 'Total Trips' label and a text box for '&totalTrips'. To the right, a data table shows the first three rows of the grid. A red arrow points from the first row of the data table to the first row of the grid. Another red arrow points from the 'trips' column of the first row of the data table to the '&trips' variable. Below the data table, an 'Event Load' script is shown:

```
Event Load
&trips = count( TripDate )
&totalTrips = &totalTrips + &trips
Endevent
```

 To the right of the script, two variables are shown: '&trips' with a value of 2 and '&totalTrips' with a value of 2.

AttractionId	AttractionName	CountryId	CityId	...
1	Louvre Museum	2	1	
2	The Great Wall	3	1	
3	Eiffel Tower	2	1	

Event Load

```
&trips = count( TripDate )
&totalTrips = &totalTrips + &trips
Endevent
```

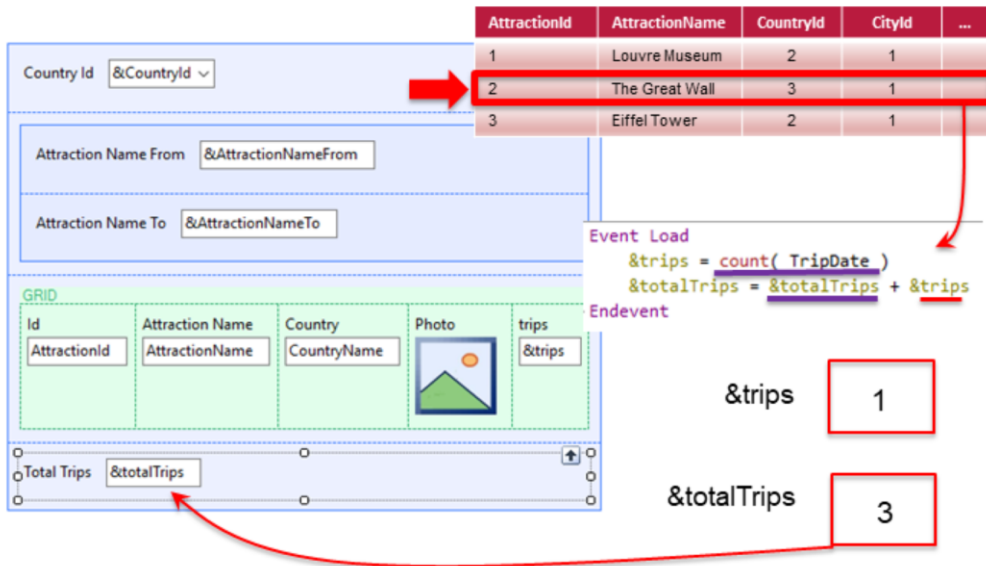
&trips 2

&totalTrips 2

An efficient way of calculating the value that the variable will have to display is... every time a line is to be loaded in the grid, we should add the value of the &Trips variable of that line, to the value calculated so far in &totalTrips.

This means that, in the Load event, after calculating the value of &trips, we assign to &totalTrips the value that it contains so far plus the value of the &trips variable.

Variable outside the grid to calculate the total



For the second line to be loaded, the value of &trips is calculated and &totalTrips will contain the previous value, to which the value of the &trips variable will be added for this second line, and so on.

Once the last line has been loaded, the &totalTrips variable will have the desired value.

Interactive Screens / Web Panels GeneXus

Refresh

The screenshot shows a web panel titled 'Application Name' by GeneXus. It features a 'Recents' section with a 'Country Id' dropdown set to 'France'. Below this is a table with columns: Id, Attraction Name, Country, Photo, and Trips. The table lists several attractions, including the Eiffel Tower, Christ the Redeemer, Louvre Museum, and Matisse Museum. A 'Total Trips' field at the bottom of the table shows the value 9. A red box highlights this field and the event code below it. The event code includes an 'Event Load' section that calculates the total trips and an 'Event Refresh' section that resets the total trips to 0. A red text box asks 'Why does it show 9 instead of 3?'.

Id	Attraction Name	Country	Photo	Trips
24	Eiffel Tower	France		2
25	Christ the Redeemer	Brazil		2
22	Louvre Museum	France		0
27	Matisse Museum	France		1
28	Forbidden city	China		0
26	Smithsonian Institute	United States		1
23	The Great Wall	China		0

Event Load

```
&trips = count( TripDate )
&totalTrips = &totalTrips + &trips
- Endevent
```

Event Refresh

```
&totalTrips = 0
- Endevent
```

Why does it show 9 instead of 3?

Let's run it.

We can see two things: first, the sum is being made correctly; second: because it is a variable, it is an input method where the user may change the value, and this doesn't make sense. So, the first thing to do is set it as Readonly.

To do so, we click on the variable in the form and, among its properties, we change its Readonly variable by setting it to True.

We may find it odd that &trips, which is also a variable, is shown as Readonly even when we did nothing in that sense. When no events have been programmed at the line level and when they are not run through by code, grid variables will always be Readonly. We will be seeing this in detail further ahead.

Let's also see what happens if, for example, we filter by France

Instead of showing a total of 3, it shows 9, which is the sum of the value displayed before –6– plus the 3 that it should be displaying now. Why did this happen?

After changing one of the filter variables, the web panel loaded the grid again, meaning that it queried the Attraction table in the database again, and it executed the Load event again for every record that met the filters criteria. The problem is that the &totalTrips variable should have been reset, returning it to zero, before the start of the grid load.

Where do we do this? In the **Refresh event**.

Note that the order in which the events are written is not important. Here we will only indicate the code that will be run when each one of them is triggered.

Refresh Event

- System event: It occurs every time the web panel is running the Load event for each line to be loaded).

```

Event Load
  &trips = count( TripDate )
  &totalTrips = &totalTrips + &trips
Endevent

Event Refresh
  &totalTrips = 0
Endevent

```

Application Name by GeneXus

Recents WWAttractions From...

Country Id (None)

Attraction Name From

Attraction Name To

Start (only the first time)

Refresh (once)

Load (7 times)

Id	Attraction Name	Country	Photo	Trips
25	Christ the Redemmer	Brazil		2
24	Eiffel Tower	France		2
28	Forbidden city	China		0
22	Louvre Museum	France		0
27	Matisse Museum	France		1
26	Smithsonian Institute	United States		1
23	The Great Wall	China		0

Total Trips 6

Let's now press F5.

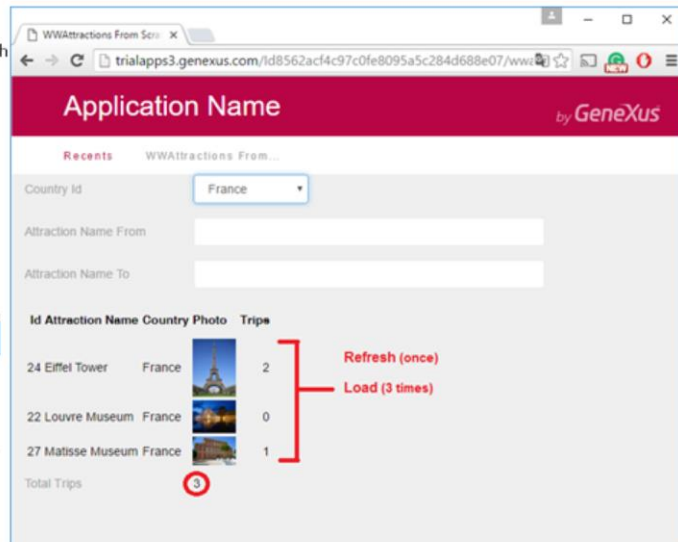
We will see that the total number of trips is now Readonly. To run this web panel for the first time, three events were triggered in sequence: the **Start event**, which is run only when the web panel is opened for the first time, the **Refresh event**, which set the variable to zero, and the **Load event**, as many times as lines were to be loaded in the grid. In this case, they were 7.

Refresh Event

- Changing the value of the variable involved in th

```
Event Load
  &trips = count( TripDate )
  &totalTrips = &totalTrips + &trips
Endevent

Event Refresh
  &totalTrips = 0
Endevent
```



Now, if we filter by a country, such as France, we see that the number of trips is correctly calculated.

Changing the value of a variable that affects the conditions that must be met by the records to be loaded in the grid causes the Refresh event to be triggered again (and therefore, the `&totalTrips` variable is reset to zero), and the database is accessed to filter and load the records in the grid again. Therefore, the **Load** event is triggered again for every attraction in France that is to be loaded.

Interactive Screens
/ Web Panels
GeneXus

Work With Attractions element of the Pattern

The screenshot shows a web application titled 'Attractions' with a table of data. The table has columns: Id, Name, Country Name, Category Name, Photo, and City Name. The data rows include attractions like 'Christ the Redemmer', 'Eiffel Tower', 'Forbidden city', 'Louvre Museum', 'Matisse Museum', 'Smithsonian Institute', and 'The Great Wall'. Annotations include a red box around the '+ INSERT' button, a red box around the 'UPDATE' and 'DELETE' buttons for the first row, and a red circle around the 'Id' column header. A red arrow points from the 'Attraction' transaction icon to the code block below the table.

```

parm(in:&Mode, in:&AttractionId);
AttractionId = &AttractionId if not &AttractionId.IsEmpty();

```

TrnMode Domain: Enum Values Insert, Insert, INS; Update, Update, UPD; Delete, Delete, DLT; Display, Display, DSP

Actions over data:
Insert, Update, Delete
an attraction

If we look at the web panel that we have implemented so far, we can see that it now looks like the object created by the WorkWith Pattern applied to the Attraction transaction.

Obviously, this object was a web panel.

What's most interesting is that, in addition to allowing us to filter the grid data, the WorkWith includes the option to run actions over the data. For example, updating an attraction's data or deleting the attraction or adding a new attraction.

To this end, the pattern inserted two controls at the grid line level, and another one outside. In any of these three cases, the action associated with each control consists in calling the Attraction transaction, sending it, as parameter, the **mode** in which the transaction must be opened if it is called to update, delete or insert data. In the first two cases, since Update or Delete will correspond to events of one line, the ID of the line attraction will be sent as a second parameter to the transaction, in order to update the data of **that** attraction, or in order to delete **that** attraction. For the Insert control outside the grid, 0 will be sent as a second parameter, because the attractions in Insert are autonumbered.

This is why the pattern modified the Attraction transaction by adding the Parm rule, among other things. As we can see, it receives two variables: the &Mode variable is a standard variable in transactions, 3 Character, which accepts one of four values indicated in the TrnMode enumerated domain: Insert (INS), Update (UPD), Delete (DLT), and Display (DSP)

Upon receiving one of these four values, the transaction will know the mode in which it should be opened.

In addition, it will receive, as a second parameter, the attraction ID, in the &AttractionId variable, for updating, deleting or displaying.

Update action at the grid lines level

Country Id

Attraction Name From

Attraction Name To

GRID

Id	Attraction Name	Country	Photo	trips	Update
AttractionId	AttractionName	CountryName		&trips	

Total Trips

updateIcon X

Images

New Image

Image

updateIcon24.png
(Default)
Size: 24x24 px

We add the image to the KB

&Update of Image type

Event Start
&Update.FromImage(updateIcon)
Endevent

Event &Update.Click
Attraction(TrnMode.Update, AttractionId)
Endevent

In our web panel, we will implement one of these actions over the transactions. For example, Update. The idea is to show an example of actions over the data.

We will have to insert a control in the grid. In the case of the pattern, a character variable called update is inserted. It is assigned the "UPDATE" text that we see in runtime. But we will choose to insert an image, which we must insert in the KB to start with. We will call it updateIcon.

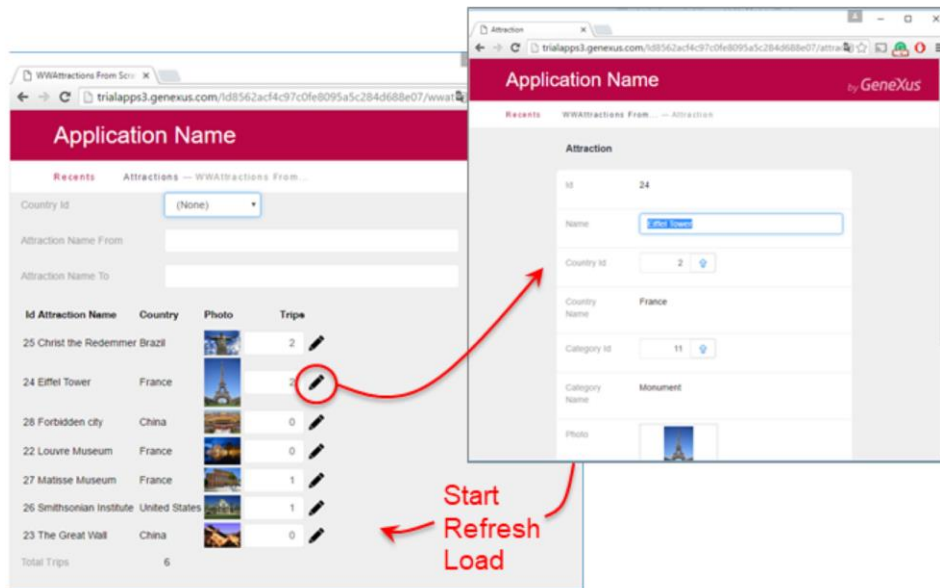
Now we go to the web panel and drag the Attribute/Variable control from the toolbox to the last grid column, and define the new variable as &Update, of the Image type instead of character. We remove the title so that it isn't displayed as a column title and then we need to load that variable with the image we've just inserted in the KB. Where do we do this?

If the image changed according to the grid line, we would then do it in the Load event. But since the image will never vary and will remain the same for every line, then a good option is to do it in the **Start event**, which will be run only once, when the web panel is opened.

Now we need to associate an event with that image, so that when the user clicks on it, the event is triggered and its code is run, where we will invoke the Attraction transaction.

There are several options available to do this. One of them is to use the click event of the &Update variable control.

This will cause that, when the user clicks on the image for a line, the code that we type inside this event will be executed. In such case what we will want to do is to invoke the Attraction transaction. We will pass the Update mode, that is to say, the Update value of the TrnMode enumerated domain we saw before, and the AttractionId value corresponding to the grid line where the click was made.



Let's run it.

First, we see that the column of the &Trips variable is now editable. We had not defined it as Readonly explicitly because we had noticed, upon the execution, that it was already done. As we said, grid variables are first added as readonly, except when an event is defined at the line level, as in this case, or under other circumstances that we will not discuss now. Let's set it as Readonly for the next execution.

We now select a country, like France. And now we click on the update image for the Eiffel Tower. In update we see the name of the transaction. Now we modify something... for example, we can change the T in Tower from uppercase to lowercase.

We now confirm... and since the work with pattern, which we have not seen, has added a Return command to the transaction, to return to the caller, we will return to the web panel. This Return command is similar to invoking the web panel for the first time, so the Start event will be executed in it, followed by the Refresh and Load events as many times as the number of records that will be loaded.

When is it not possible to include a column in the grid?

The screenshot shows the GeneXus IDE with a web form titled 'Attraction'. The form includes a 'Country Id' dropdown, 'Attraction Name From' and 'Attraction Name To' text boxes, and a grid. The grid has columns for 'Id', 'Attraction Name', 'Country', 'Photo', and 'Trips'. The 'Id' column is highlighted with a red arrow pointing to the 'AttractionId' column. Below the grid is a 'Total Trips' label with a value of 5. An event handler is defined for the grid's 'Click' event, with the code: `Attraction(TrnMode.Update, AttractionId)`. The 'AttractionId' parameter is underlined in the code. A blue arrow points from this parameter to the 'Visible' property in the Properties window, which is set to 'False'.

Event &Update.Click
Attraction(TrnMode.Update, AttractionId)
Endevent

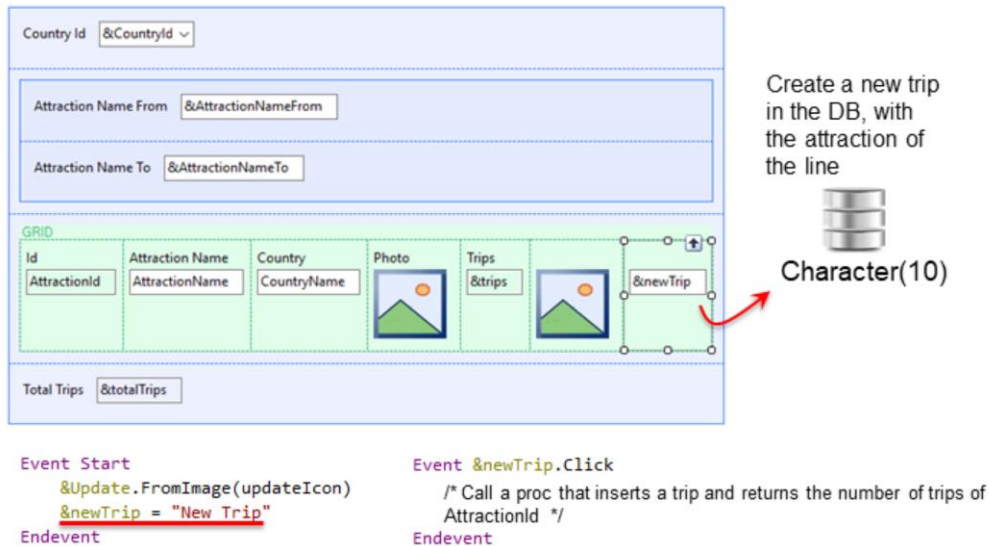
It must be in the grid! If we don't want to see it, we can hide it.

Visible: False

What would happen if we hadn't added the `AttractionId` attribute in the grid? By clicking on the image to update, what `AttractionId` would have been sent as a parameter to the transaction? It would not have that value to send.

Since it will be used in an event at the line level, which will be triggered after the lines have been loaded, we cannot remove `AttractionId` from the grid, because here, we are no longer in the database. The grid has saved, upon the loading with the Load event, all its column values and nothing else. A later event will work only on the data loaded in the grid. So, if we don't want to see this column in the grid, we can hide it. It will continue to be present, though hidden. To this end, we use the **Visible property with its value set to False**.

Another action at the lines level: refreshing the line



Let's now add another action at the line level. But this action will not call another object with interface as it happened in the case invoking the Attraction transaction.

We can imagine that, for instance, we will enable the possibility to create, from a line (an attraction), a new trip in the database, with that attraction.

First we add a new variable –called newTrip and 10 character- to the grid.

Let's change it to ReadOnly. We want it to contain the "New Trip" text, so we assign it in the Start event because there will be no variations by line. And let's program the Click event for that variable.

What do we want to do when the user clicks on New Trip?



Videos

training.genexus.com

Documentation

wiki.genexus.com

Certifications

training.genexus.com/certifications