

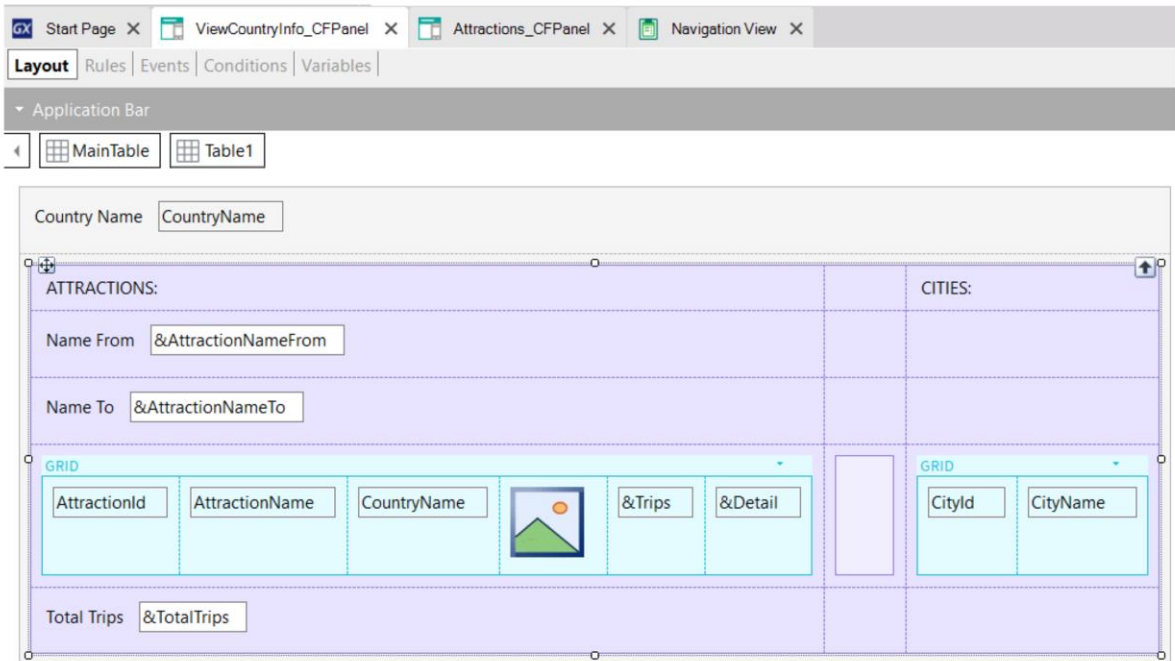
Web Screens with Customer-facing Focus

Using Multiple Grids

GeneXus™

We have seen how to use a grid in a panel object. We will now see the considerations we need to have if we add more than one grid to the panel.

Panel object with more than one grid



We will build a panel object that shows the attractions and cities of a country received by parameter.

To do so, we select Save As from the Attractions_CFPANEL panel and name it ViewCountryInfo_CFPANEL. Then we remove its main object setting, because the goal is to have it called from the list of attractions panel when the country's name is clicked on.

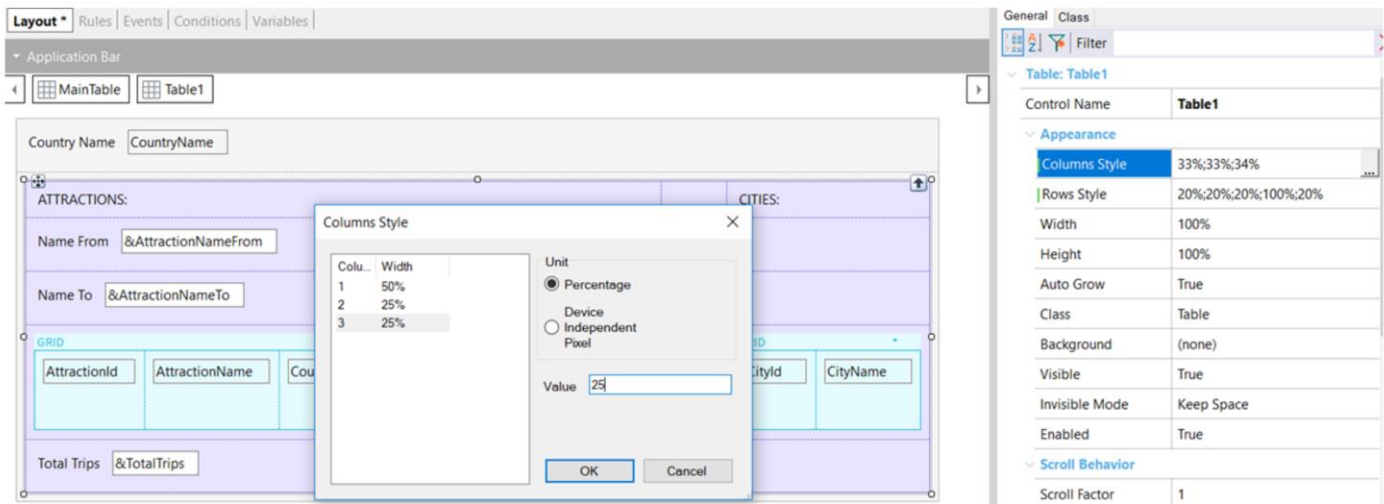
In the rules tab, we add a Parm rule with an input parameter, the CountryId attribute.

In the form, we add the CountryName attribute and delete the &CountryId variable of Dynamic combo type, since we will not choose a country but receive it by parameter.

To group all the data of the attractions and then the data of the cities, we insert a Table control from the toolbar and place the variables AttractionNameFrom, AttractionNameTo, the grid and the variable TotalTrips inside the table. Now to the right we insert another table as a separator and then we insert a grid to show the cities of the country, with the attributes CityId and CityName.

We add titles to the sections, placing a textblock ATTRACTIONS: above the attractions section and CITIES: above the cities section. We change the class to TextBlockTitle.

Changing the style of the rows and columns of a table



A few minutes ago, we inserted a table to separate the content of the attractions from the cities.

To define how we want the contents of a table to be seen, we must change the size of the rows or columns of the table. For example, we want the grids to have enough space in the corresponding row to be displayed correctly and to assign more space to the column of the table that shows the content of the attractions, because we have more things to show than for the cities.

For that we have the Columns Style and Rows Style properties of the board. First, we will change the columns that we know are 3, the column that contains the attractions' data, the column with the separating table, and the column that contains the cities' data.

If we select Table1 and click on the Column Style property, we see that by default each column takes up 33% of the available space.

The possible values to be assigned to the width of the columns are a percentage of the total table space or in Device Independent Pixels (DIPs). This measure allows us to assign units that are an abstraction of a pixel, which do not depend on the platform and will later be converted to real pixels when the application is executed. The number of pixels that each DIP will take up depends on the dimensions of the screen. This will allow us to scale to different screen sizes using standard sizes.

Let's assign 50% to the first column, 25% to the second column and 25% to the third column of the available space. This space is what remains after having assigned the columns with dips, i.e. the percentages are relative to the value that results from subtracting the fixed values (in DIPs) from the total width.

Changing the style of the rows and columns of a table

The screenshot displays the GeneXus IDE interface. On the left, a table design is visible with sections for 'Country Name', 'ATTRACTIONS', and 'CITIES'. A 'Rows Style' dialog box is open, showing a table with the following data:

Row	Height
1	pd
2	pd
3	pd
4	100%
5	pd

The dialog also shows 'Unit' options: Percentage, Device, Independent Pixel, and Platform Default. The 'Value' field is set to 20.

On the right, the 'Properties' pane for 'Table1' is shown. Under the 'Appearance' section, the 'Columns Style' is '50%;25%;25%' and the 'Rows Style' is '20%;20%;20%;100%;20%'.

Now we change the height of the rows. We click on the Rows Style property and see that by default all the rows have a 20% assigned, except the fourth row which is where the grids are located.

We see that here we can assign the values using percentages, DIPS and Platform Default. This last value corresponds to: "Using the best value depending on the platform and the context;" that is, it varies between platforms and for the same platform, it also depends on the cell's content.

We are going to assign that value to all the rows except row 4 where the grids are, which we set to occupy the 100% that remains available.

Invocation of the detail panel

The screenshot shows the GeneXus IDE interface. At the top, there are three tabs: 'Start Page', 'ViewCountryInfo_CFPANEL *', and 'Attractions_CFPANEL'. Below the tabs is a menu bar with 'Layout', 'Rules', 'Events', 'Conditions', and 'Variables'. The 'Application Bar' is expanded to show 'MainTable'. The main workspace displays a form layout with the following elements:

- Country: &CountryId (dropdown)
- Name From: &AttractionNameFrom (text box)
- Name To: &AttractionNameTo (text box)
- GRID: A table with columns: AttractionId, AttractionName, CountryId, CountryName, &Trips, &Detail. The 'CountryName' column contains a landscape image icon.
- Total Trips: &TotalTrips (text box)

On the right side, the 'Events' list is visible, showing an event named 'Event CountryName.Tap' with the code 'ViewCountryInfo_CFPANEL(CountryId)' and 'Endevent'. A blue arrow points from the 'CountryName' column in the grid to this event.

If we right-click to see the navigation of the object we built, we see that the Output window shows an error and if we look at the error in the navigation list it says that the Load event cannot be programmed if we have multiple grids.

If we go to the events, we see that we still have the Load event programmed as we had it in the original object. Here we don't consider what we said before, that it would have been better to use the Load event of the grid and not the generic one to avoid that situation.

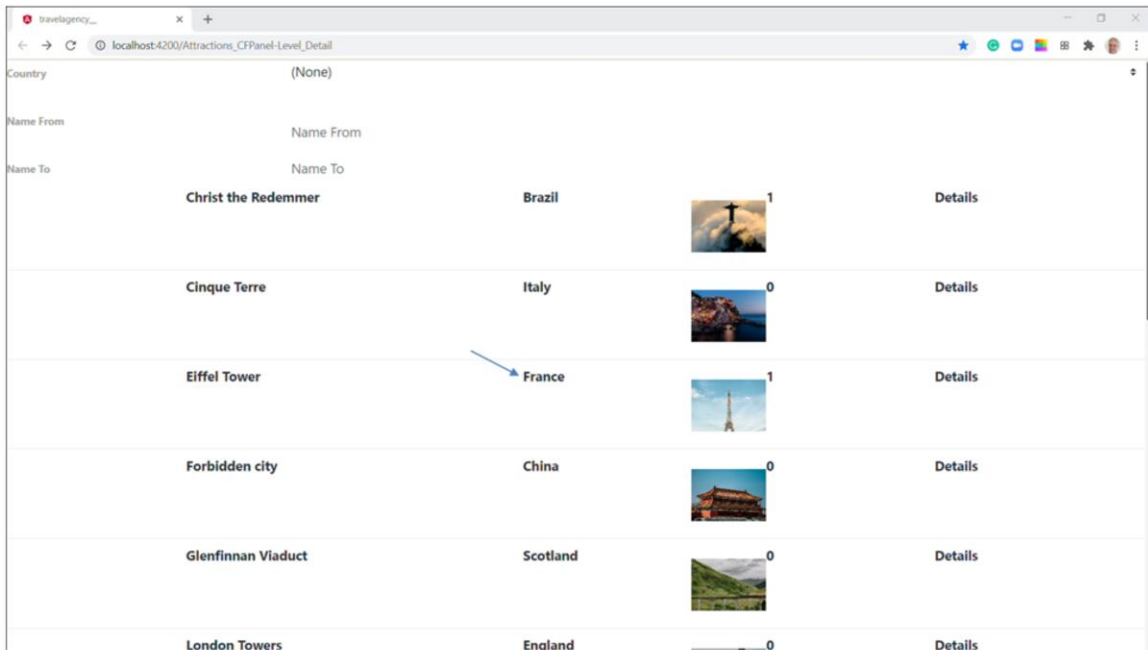
So we add Grid1 to the Load. If we select View Navigation, we see that the problem has been solved.







Before running, we go to the Attraction_CFPANEL panel and add the CountryId attribute required to go to the information panel of a country, when we click on the name of the country in the grid.

Then we add to the CountryName attribute a Tap event, where we write the invocation to the ViewCountryInfo_CFPANEL panel, passing it the CountryId as parameter.

We run it to see all this.

Example at runtime






Country	(None)			
Name From	Name From			
Name To	Name To			
Christ the Redemmer	Brazil		1	Details
Cinque Terre	Italy		0	Details
Eiffel Tower	France		1	Details
Forbidden city	China		0	Details
Glenfinnan Viaduct	Scotland		0	Details
London Towers	England		0	Details

If we click on France...

Example at runtime

The screenshot shows a web browser window with the URL `localhost:4200/app/ViewCountryInfo_CFPannel-Level_Detail:CountryId=2`. The page displays information for the country France. At the top, the country name is "France". Below this, there are two main sections: "ATTRACTIONS:" and "CITIES:". The "ATTRACTIONS:" section lists three attractions: "Eiffel Tower" (1 trip), "Louvre" (0 trips), and "Matisse Museum" (1 trip). Each attraction has a small image and a "Details" link. The "CITIES:" section lists two cities: "Paris" (1 trip) and "Nice" (2 trips). At the bottom left, there is a "Total Trips" label with the value "2".

Country Name		France	
ATTRACTIONS:			
Name From	Name From		
Name To	Name To		
Eiffel Tower	 1	Details	
Louvre	 0	Details	
Matisse Museum	 1	Details	
Total Trips	2		

CITIES:	
1	Paris
2	Nice

The information of the country France is opened, showing us the attractions with the total of trips of each attraction and the global total of trips to France. To the right are the cities of that country.

Navigation list of the new panel

Pattern:

- ViewCountryInfo_CFPANEL
 - Level_Detail_Grid1
 - Level_Detail_Grid2
 - Level_Detail

Data Provider ViewCountryInfo_CFPANEL_Level_Detail_Grid2 Navigation Report

Name: ViewCountryInfo_CFPANEL_Level_Detail_Grid2

Description: ViewCountryInfo_CFPANEL_Level_Detail_Grid2

Output Devices: None

Environment: Default (C#)

Spec. Version: 17_0_0-144011

Form Class: HTML

Program Name: ViewCountryInfo_CFPANEL_Level_Detail_Grid2

Parameters: in: CountryId, in: &AttractionNameFrom, in: &AttractionNameTo, in: &CityName, in: &start, in: &count, in: &gxid, out: ViewCountryInfo_CFPANEL_Level_Detail_Grid2Sdt

LEVELS

For Each CountryCity (Line: 4)

Order: CountryId
Index: ICOUNTRYCITY

Navigation filters: Start from: CountryId = @CountryId
Loop while: CountryId = @CountryId

Optimizations: Server Paging

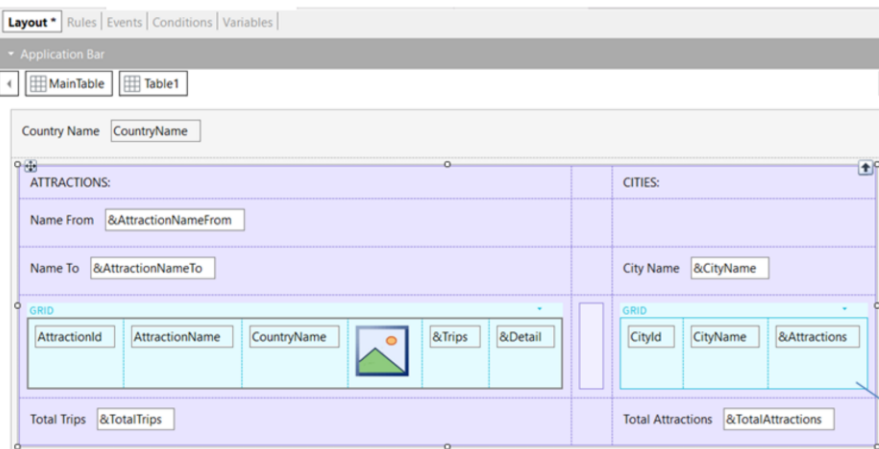
CountryCity (CountryId, CityId)

If we go to the navigation list, we see that now there is a Level_Detail entry for Grid1 and another one for Grid2.

For Grid1 we see the access to the Attraction table and the navigation of the formula that we saw when we implemented the Attractions_CFPANEL panel.

If we select the node corresponding to Grid2, we see that the grid is accessing the CountryCity table to show the cities, and that the filter being used is CountryId, by the attribute received in the Parm rule.

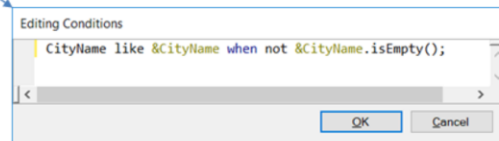
We add a filter by city and totals by city and country



```

1 | Event Grid1.Load
2 |     &Trips = Count(TripDate)
3 | Endevent
4 |
5 | Event Grid2.Load
6 |     &Attractions = Count(AttractionName)
7 | Endevent
8 |
9 | Event Grid1.Refresh
10 |     &TotalTrips = 0
11 |     For Each Trip.Attraction
12 |         &TotalTrips += 1
13 |     Endfor
14 | Endevent
15 |
16 | Event Grid2.Refresh
17 |     &TotalAttractions = 0
18 |     For Each Attraction
19 |         &TotalAttractions += 1
20 |     Endfor
21 | Endevent

```



In a similar way to the totals we show for the attractions and the filter by name, we will show the number of attractions of each city, and the total number of attractions in the country, and we will add a filter by city name.

In the Variables tab, we create an `&Attractions` variable, another one called `&TotalAttractions` and a third one called `&CityName`. Now we add the `&Attractions` variable to the grid by setting its Label Position property to None. Then we add `&TotalAttractions` below the grid and the filter variable `&CityName` above the city grid. Let's add to the grid the necessary condition for the filter.

Before that we take the opportunity to assign the Base Transaction property in `Country.City`. Here we click on Conditions and write the filter condition using the like operator, since we will not filter by name range, but by a name similar to the one we put in the filter.

Then we add the `Grid2.Load` event where we load the `&attractions` variable with a Count formula with the `AttractionName` attribute. Since the base table of the grid is `CountryCity`, the formula will only count the attractions of the city corresponding to each line.

To calculate the total number of attractions, for the same reasons as when we calculated the total number of trips, we wrote the `Grid2.Regresh` event, in which we programmed a For Each command on the Attractions table to count the attractions, which will be filtered by the attribute of the country identifier received by parameter.

We calculate it every time a line is going to be loaded; that is, in the Grid2 Load event.

Navigation list of the panel with the new changes

ViewCountryInfo_CFPANEL

- Level_Detail_Grid1
- Level_Detail_Grid2
- Level_Detail

For Each Attraction (Line: 5)

Order: CountryId
Index: IATTRACTION1

Navigation filters: Start from: CountryId = @CountryId
Loop while: CountryId = @CountryId

Optimizations: count(*)

=Attraction (AttractionId)

For Each CountryCity (Line: 11)

Order: CountryId
Index: ICOUNTRYCITY

Navigation filters: Start from: CountryId = @CountryId
Loop while: CountryId = @CountryId

Constraints: CityName like &CityName WHEN not &CityName.isempty() ←

Join location: Server

Optimizations: Server Paging

=CountryCity (CountryId, CityId)

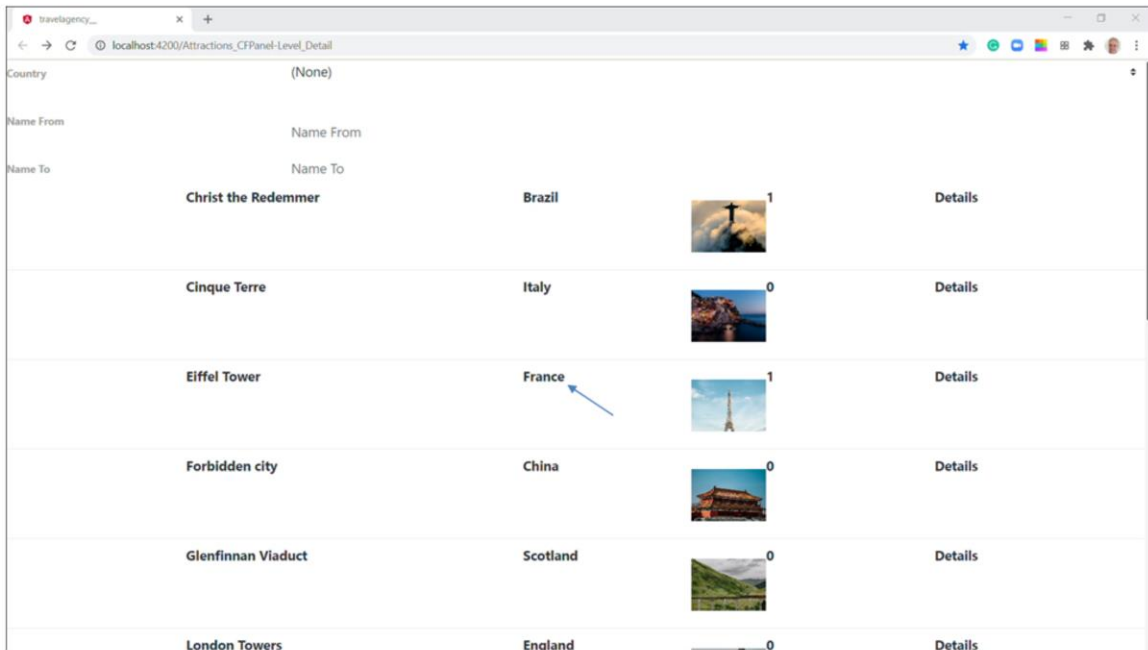
=count(AttractionName) navigation







=Attraction (CountryId, CityId)

If we now see the panel navigation list, in the node Level_Detail_Grid2 we see the navigation of the For Each command to the Attraction table, and below it the navigation of the Count formula on the Attraction table.

And here we see the filter we added by CountryName.

Viewing totals by city and country



Country	(None)			
Name From	Name From			
Name To	Name To			
Christ the Redemmer	Brazil		1	Details
Cinque Terre	Italy		0	Details
Eiffel Tower	France		1	Details
Forbidden city	China		0	Details
Glenfinnan Viaduct	Scotland		0	Details
London Towers	England		0	Details

Let's run our main object, Attractions_CFPANEL, to see what we did.

We click on France again...

Viewing totals by city and country

The screenshot shows a web application interface for a travel agency. The main content is for the country of France. It is divided into two main sections: 'ATTRACTIONS:' and 'CITIES:'. The 'ATTRACTIONS:' section lists three items: 'Eiffel Tower' (1), 'Louvre' (0), and 'Matisse Museum' (1). Each item has a small image and a 'Details' link. The 'CITIES:' section is a table with two rows: Paris (2) and Nice (1). At the bottom, there are two summary boxes: 'Total Trips: 2' and 'Total Attract...: 3'.

City Name	City Name	
1	Paris	2
2	Nice	1

Country Name: **France**

ATTRACTIONS:

Name From: Name From

Name To: Name To

Eiffel Tower 1 **Details**

Louvre 0 **Details**

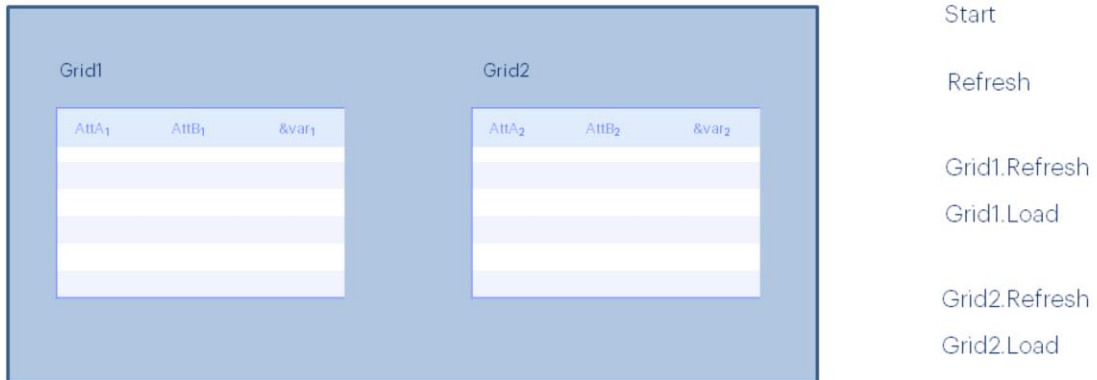
Matisse Museum 1 **Details**

Total Trips: 2

Total Attract... 3

... and now we can see the total of attractions of each city of France and the total of attractions of the country France.

Event execution order



We saw that because we had more than one grid in the panel, we had to use the Refresh and Load event of each grid.

But after adding these events, we wonder what would be the triggering order of these events in relation to the events of the panel object.

When the panel object is executed for the first time, the triggering order of event execution will be:

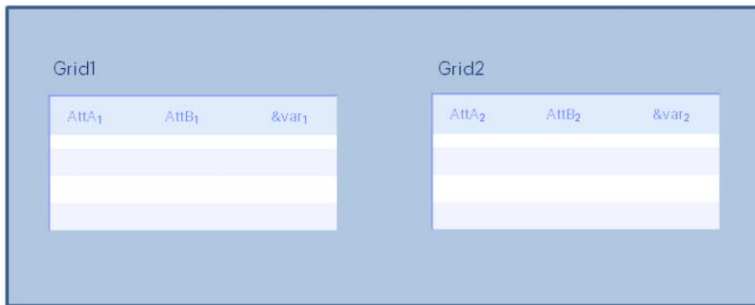
First, the Start event (only once).

Then the generic Refresh event; i.e. the panel's own Refresh event.

Finally, the Refresh of the first grid. If it has a base table, the Load event of the grid will be executed as many times as records are retrieved from the database, filtering the corresponding records. If it doesn't have a base table, the Load event of the grid is executed only once, and if it is a grid based on an SDT, the Load event is not executed.

And then the same with the Refresh and Load events of the second grid.

What do you want to refresh?



```

Start
Refresh
Grid1.Refresh      Grid2.Refresh
Grid1.Load         Grid2.Load
  
```

```

Event 'User-event'
...
  Form.Refresh
endevent
  
```

```

Event 'User-event'
...
  Refresh
endevent
  
```

```

Event 'User-event'
...
  Grid1.Refresh()
endevent
  
```

Since there is more than one grid, the Refresh command must also be specialized to indicate which grid you want to refresh.

The generic **Refresh** command (the one we had seen when we used it in the Attractions_CFPANEL panel) causes the generic Refresh to be executed, as well as the Refresh and Load event of each grid (that is, everything but Start).

And now we also have the Refresh method of a grid, which will refresh only the grid; that is to say, run the grid Refresh and Load (n times, once, or never), depending on whether the grid has a base table or not, or if it's a grid of a collection SDT variable, respectively.

In this video we saw how we can work with multiple grids in a Panel object, in this case with parallel grids and the considerations we need to have when invoking the events of each grid. In this course, we will not address the topic of nested grids in a panel object.

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications