

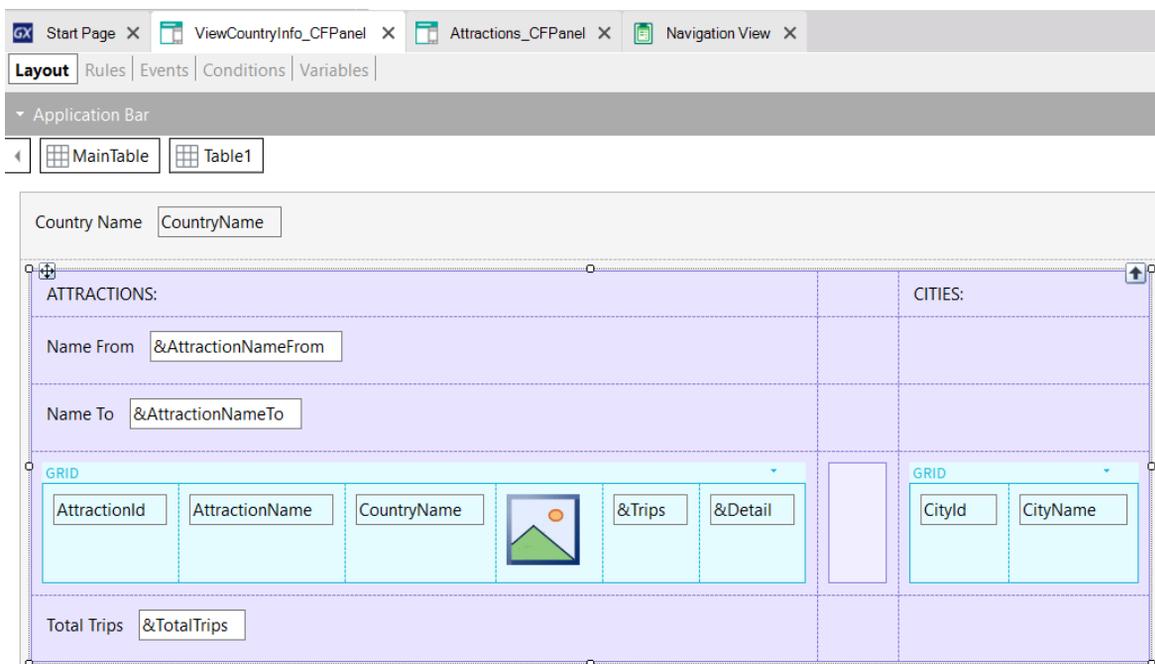
Pantallas web con foco en Customer-facing

Uso de múltiples grids

GeneXus™

Hemos visto como usar un grid en un objeto panel. Veremos ahora las consideraciones que tenemos que tener si agregamos al panel más de un grid.

Objeto panel con más de un grid



Vamos a construir un objeto panel que muestre las atracciones y las ciudades de un país recibido por parámetro.

Para eso hacemos un Save As del panel Attractions_CFPANEL y le ponemos de nombre ViewCountryInfo_CFPANEL. Luego le quitamos que sea objeto main, ya que la idea es que el mismo se llame desde el panel de la lista de atracciones cuando se haga clic sobre el nombre de un país.

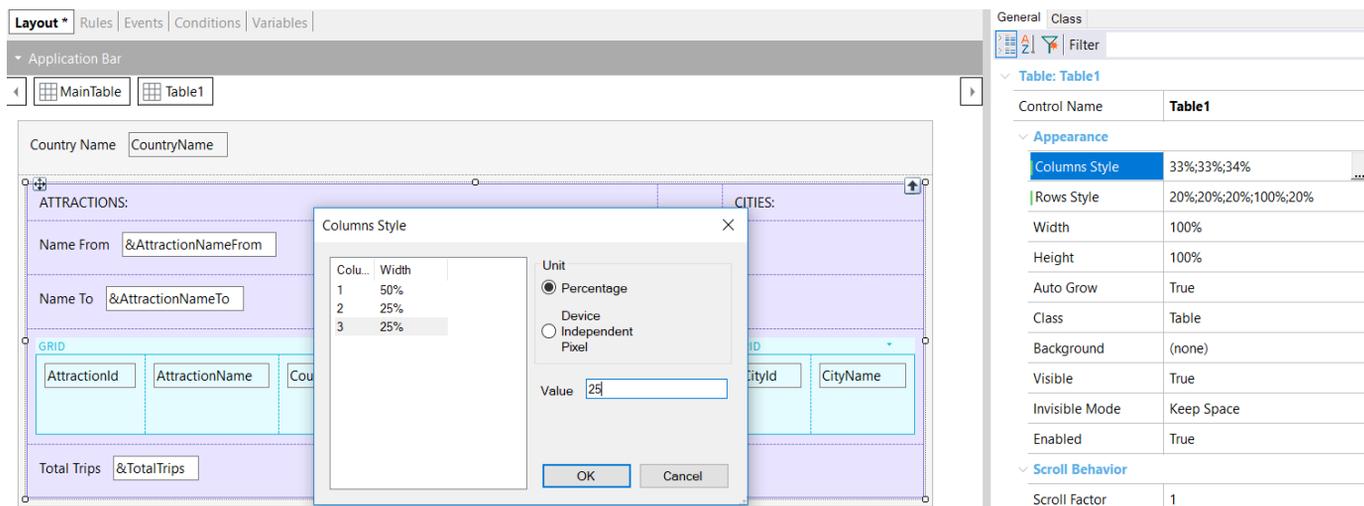
Ahora vamos a la solapa de las reglas y agregamos una regla Parm con un parámetro de entrada, el atributo CountryId.

En el form, agregamos al atributo CountryName y eliminamos la variable del tipo Dynamic combo &CountryId, ya que no vamos a elegir un país sino que lo recibimos por parámetro.

Para agrupar todos los datos de las atracciones y luego los datos de las ciudades, insertamos un control Table desde la barra de herramientas y colocamos a las variables AttractionNameFrom, AttractionNameTo, el grid y la variable TotalTrips dentro de la tabla. Ahora a su derecha insertamos otra tabla como separador y luego insertamos un grid para mostrar las ciudades del país, con los atributos CityId y CityName.

Agregamos títulos a las secciones, poniendo un textblock ATTRACTIONS: arriba de la sección de atracciones y CITIES: arriba de la sección de ciudades. Le cambiamos la clase a TextBlockTitle.

Cambiando el estilo de las filas y las columnas de una tabla



Hace unos minutos insertamos una tabla para separar el contenido de las atracciones de las ciudades.

Para definir cómo queremos que el contenido de una tabla sea visto, debemos cambiar el tamaño de las filas o de las columnas de la tabla. Por ejemplo queremos que los grids tengan espacio suficiente en la fila correspondiente para poder desplegarse correctamente y asignar más espacio a la columna de la tabla que muestra el contenido de las atracciones, porque tenemos más cosas que mostrar que para las ciudades.

Para eso contamos con las propiedades Columns Style y Rows Style de la tabla. Primero vamos a cambiar las columnas que sabemos que son 3, la columna que contiene los datos de las atracciones, la columna con la tabla separadora y la columna que contiene los datos de las ciudades.

Si seleccionamos la Table1 y hacemos clic sobre la propiedad Column Style, vemos que por defecto cada columna ocupa un 33% del espacio disponible.

Vemos que los valores posibles a ser asignados al ancho de las columnas, son en porcentaje del espacio total de la tabla o en Device Independent Pixels (DIPs). Esta medida nos permite asignar unidades que son una abstracción de un pixel, que no dependen de la plataforma y que luego serán convertidos a pixels reales en el momento de que la aplicación sea ejecutada. La cantidad de pixels que cada DIP ocupará dependerá de las dimensiones de la pantalla. Esto nos permitirá escalar a diferentes tamaños de pantalla usando tamaños uniformes.

Vamos a asignarle a la primer columna 50%, a la segunda 25% y a la tercera otro 25% del espacio disponible. Este espacio es el que queda después de haber asignado las columnas con dips, es decir que los porcentajes son relativos al valor que resulta de restar del ancho total, los valores fijos (en dips).

Cambiando el estilo de las filas y las columnas de una tabla

The screenshot displays the GeneXus IDE interface. At the top, there are tabs for 'Layout *', 'Rules', 'Events', 'Conditions', and 'Variables'. Below this is the 'Application Bar' with 'MainTable' and 'Table1' buttons. The main workspace shows a table design with five rows. The first row is labeled 'Country Name' with a control 'CountryName'. The second row is labeled 'ATTRACTIONS:' and contains 'Name From' with control '&AttractionNameFrom'. The third row is labeled 'ATTRACTIONS:' and contains 'Name To' with control '&AttractionNameTo'. The fourth row is labeled 'GRID' and contains a table with columns 'AttractionId', 'AttractionName', and 'Cou'. The fifth row is labeled 'CITIES:' and contains 'Total Trips' with control '&TotalTrips'. A 'Rows Style' dialog box is open, showing a list of rows with their heights: Row 1 (pd), Row 2 (pd), Row 3 (pd), Row 4 (100%), and Row 5 (pd). The dialog also shows unit options: Percentage, Device, Independent Pixel, and Platform Default (selected). The value is set to 20. On the right, the 'Properties' panel for 'Table: Table1' shows the 'Appearance' section with 'Columns Style' set to '50%;25%;25%' and 'Rows Style' set to '20%;20%;20%;100%;20%'.

Ahora cambiamos el alto de las filas. Hacemos clic sobre la propiedad Rows Style y vemos que por defecto todas las filas tienen un 20% asignado, menos la cuarta fila que es donde están ubicadas las grillas.

Vemos que aquí podemos asignar los valores en porcentaje, en DIPS y en Platform Default. Este último valor corresponde a: "Using the best value depending on the platform and the context", es decir, que difiere de plataforma en plataforma y para una misma plataforma, también depende del contenido de la celda.

Vamos a asignarle ese valor a todas las filas excepto a la fila 4 donde están de las grillas, que le asignamos que ocupen el 100% que quede disponible.

Invocación al panel de detalle

The screenshot shows the GeneXus IDE interface with three open windows: 'Start Page', 'ViewCountryInfo_CFPanel*', and 'Attractions_CFPanel'. The 'Attractions_CFPanel' window is active, displaying a form layout. The form includes a 'Country' dropdown menu with the value '&CountryId', two text input fields labeled 'Name From' and 'Name To' with values '&AttractionNameFrom' and '&AttractionNameTo' respectively, a 'GRID' section, and a 'Total Trips' text input field with value '&TotalTrips'. The grid has columns for 'AttractionId', 'AttractionName', 'CountryId', 'CountryName', a landscape image icon, '&Trips', and '&Detail'. A blue arrow points from the 'CountryName' column to an event definition on the right side of the screen.

```

Event CountryName.Tap
  ViewCountryInfo_CFPanel(CountryId)
Endevent
  
```

Si damos botón derecho para ver la navegación del objeto que construimos, vemos que en la ventana de Output muestra un error y si vemos el error en el listado de navegación nos dice que el evento Load no puede ser programado si tenemos múltiples grids.

Si vamos a los eventos, vemos que todavía tenemos programado el evento Load como lo teníamos en el objeto original. Acá no damos cuenta lo que mencionamos antes, que hubiese sido mejor utilizar el evento Load del grid y no el genérico para evitar esa situación.

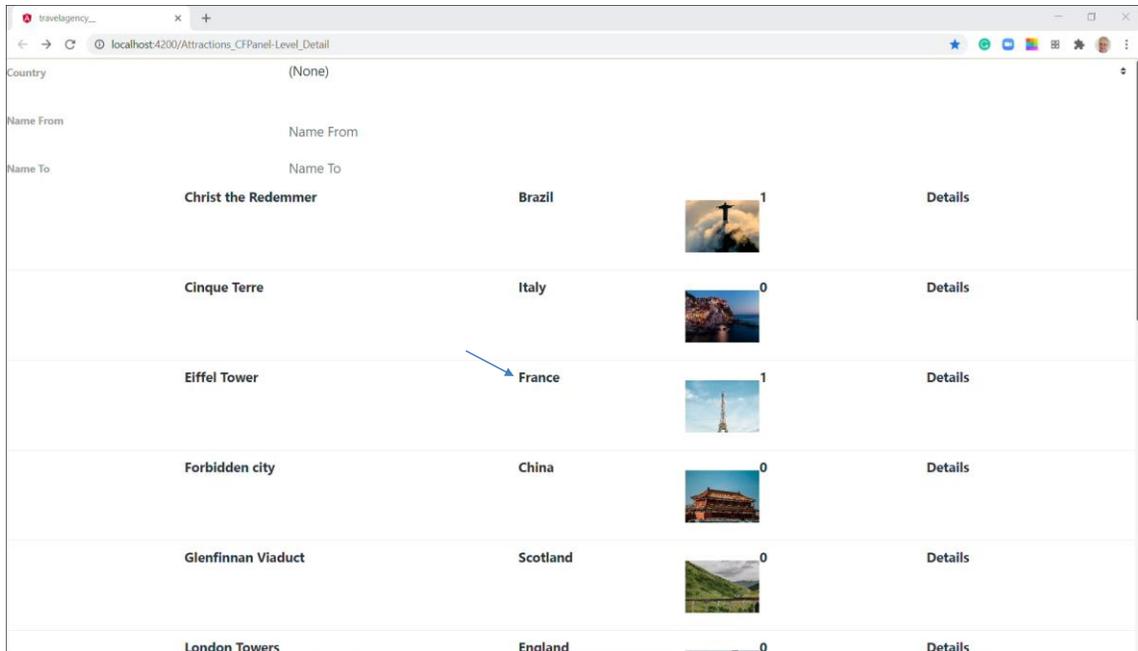
Así que agregamos Grid1 al Load. Si hacemos View Navigation, vemos que se solucionó el problema.

Antes de ejecutar, vamos al panel Attraction_CFPanel y agregamos al grid de atracciones el atributo CountryId que necesitamos para pasar al panel de información de un país, cuando hagamos clic sobre el nombre del país en el grid.

Luego agregamos al atributo CountryName un evento Tap, donde escribimos la invocación al panel ViewCountryInfo_CFPanel, pasándole el CountryId como parámetro.

Ejecutamos para ver todo esto.

Ejemplo en ejecución

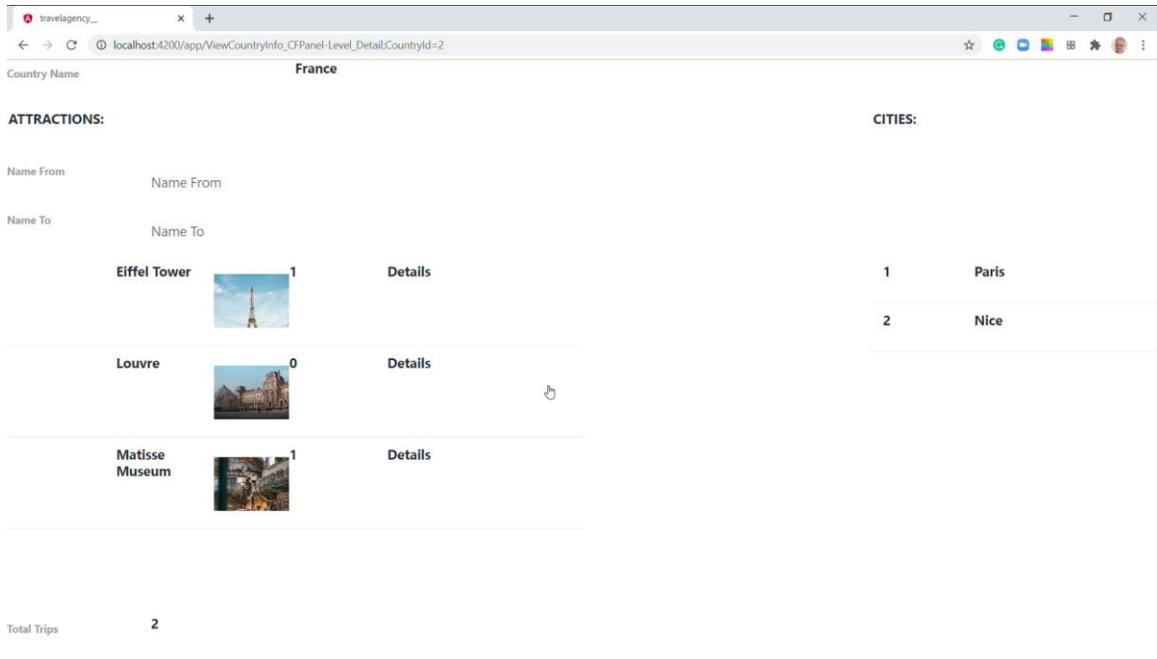


The screenshot shows a web browser window with the URL `localhost:4200/Attractions_CFPanel-Level_Detail`. The browser displays a table with the following data:

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

Si hacemos clic sobre France...

Ejemplo en ejecución



The screenshot shows a web browser window with the URL `localhost:4200/app/ViewCountryInfo_CFPannel-Level_Detail;CountryId=2`. The page content is as follows:

Country Name: **France**

ATTRACTIONS:

Name From	Name To	Image	Total Trips	Details
Eiffel Tower			1	Details
Louvre			0	Details
Matisse Museum			1	Details

Total Trips: **2**

CITIES:

1	Paris
2	Nice

Se abre la información del país Francia, mostrándonos las atracciones con el total de viajes de cada atracción y el total global de viajes a Francia y a la derecha las ciudades de ese país.

Listado de navegación del nuevo panel

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)

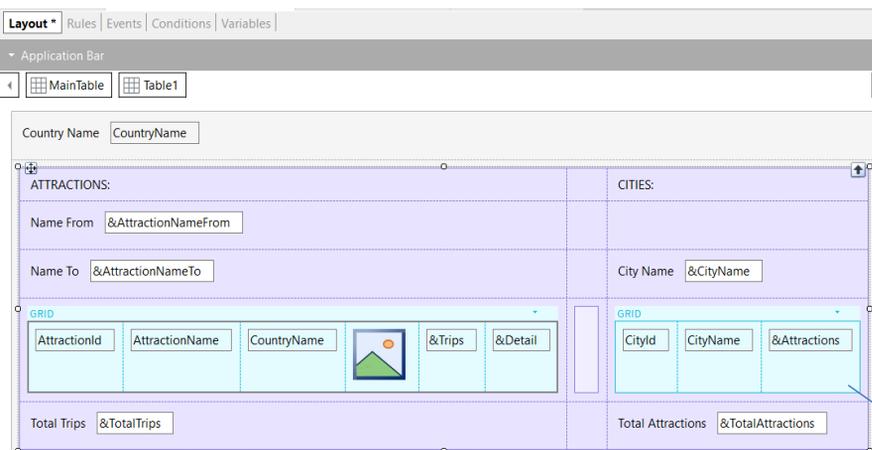
Order: CountryId
 Index: ICOUNTRYCITY
Navigation filters: Start from: CountryId = @CountryId
 Loop while: CountryId = @CountryId
Optimizations: Server Paging

CountryCity (CountryId, CityId)

Si vamos al listado de navegación, vemos que ahora hay una entrada Level_Detail para el Grid1 y otra para el Grid2. Para el Grid1 vemos el acceso a la tabla Attraction y la navegación de la fórmula que vimos cuando implementamos el panel Attractions_CFPANEL.

Si vamos al nodo correspondiente al Grid2, vemos que el grid está accediendo a la tabla CountryCity para mostrar las ciudades y que se está filtrando por CountryId, por el atributo que se recibe en la regla Parm.

Agregamos filtro por ciudad y totales por ciudad y país



```

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

```



En forma similar a los totales que mostramos para las atracciones y el filtro por nombre, vamos a mostrar la cantidad de atracciones que cada ciudad tiene, el total de atracciones del país y vamos a agregar un filtro por nombre de ciudad.

En la solapa Variables creamos una variable `&Attractions`, otra `&TotalAttractions` y otra `&CityName`. Ahora agregamos la `&Attractions` al grid ajustando su propiedad Label Position en None. Luego agregamos a `&TotalAttractions` debajo del grid y a la variable de filtro `&CityName` arriba del grid de ciudades. Vamos a agregar al grid la condición necesaria para el filtro.

Antes que eso aprovechamos para asignar la propiedad Base Transaction en `Country.City`. Ahora sí, hacemos clic en Conditions y escribimos la condición de filtro utilizando el operador `like`, ya que no filtraremos por rango de nombre, sino por un nombre parecido al que coloquemos en el filtro.

Luego agregamos el evento `Grid2.Load` donde cargamos la variable `&attractions` con una fórmula `Count` con el atributo `AttractionName`. Como la tabla base del grid es `CountryCity`, la fórmula contará solamente las atracciones de la ciudad correspondiente a cada renglón.

Para calcular el total de atracciones, por idénticas razones a como lo hicimos cuando calculamos el total de viajes, escribimos el evento `Grid2.Regresh`, en la que programamos un `For Each` sobre la tabla de

Atracciones para contar las atracciones, que quedarán filtradas por el atributo del identificador de país recibido por parámetro.

, y la calculamos cada vez que se va a cargar una línea, es decir, en el evento Load del Grid2.

Listado de navegación del panel con los nuevos cambios

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)

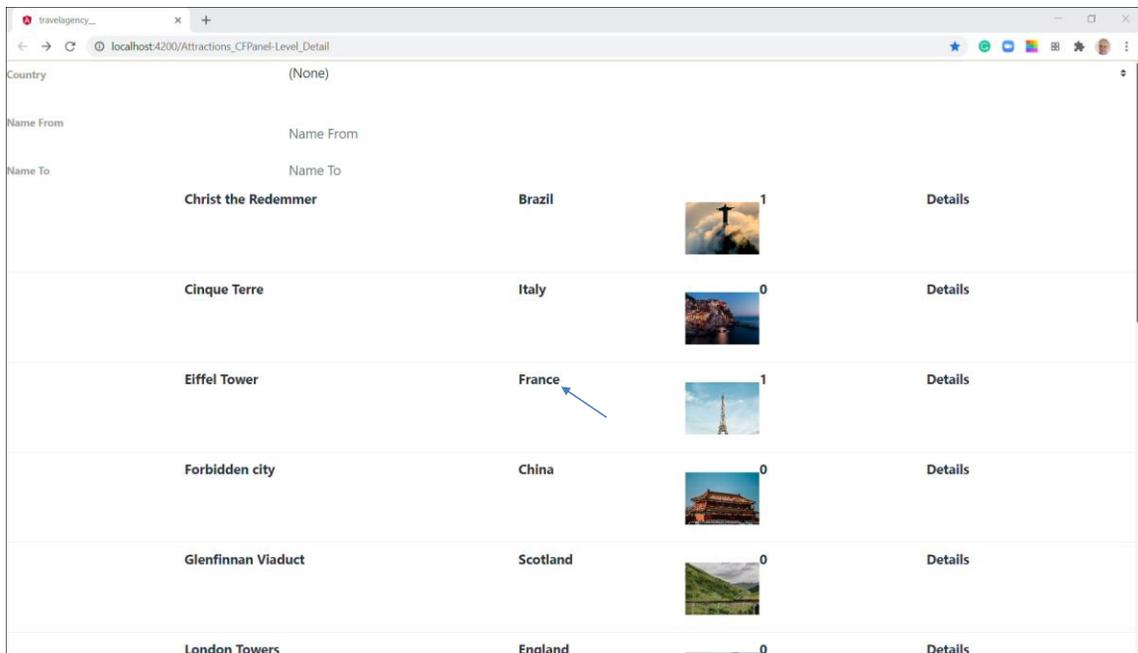
-count(AttractionName) navigation

-Attraction (CountryId, CityId)

Si vemos ahora el listado de navegación del panel, en el nodo Level_Detail_Grid2 vemos la navegación del For Each a la tabla Attraction y más abajo la navegación de la fórmula Count sobre la tabla Attraction.

Y aquí vemos el filtro que agregamos por CountryName.

Viendo los totales por ciudad y país



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

Ejecutemos nuestro objeto main, Attractions_CFPANEL, para ver lo que hicimos.

Hacemos clic en Francia nuevamente...

Viendo los totales por ciudad y país

The screenshot shows a web application interface for a travel agency. The main heading is "Country Name: France". Below this, there are two 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 columns: "City Name" and "City Name". The table contains two rows: "1 Paris 2" and "2 Nice 1". At the bottom left, there is a "Total Trips" label with the value "2". At the bottom right, there is a "Total Attract..." label with the value "3".

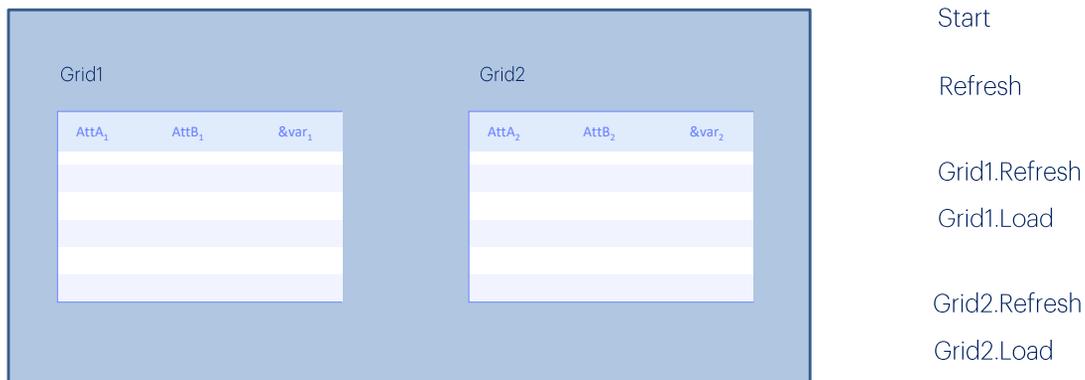
City Name	City Name	
1	Paris	2
2	Nice	1

Total Trips: 2

Total Attract...: 3

y ahora podemos ver el total de atracciones de cada ciudad de Francia y el total de atracciones del país Francia.

Event execution order



Ya vimos que al tener más de un grid en el panel, debemos usar el evento Refresh y Load de cada grid.

Pero al haber agregado estos eventos, nos surge la pregunta de cuál sería el orden de disparo de estos eventos con relación a los eventos propios del objeto panel.

Al ejecutarse el objeto panel por primera vez, el orden de disparo de ejecución de los eventos será:

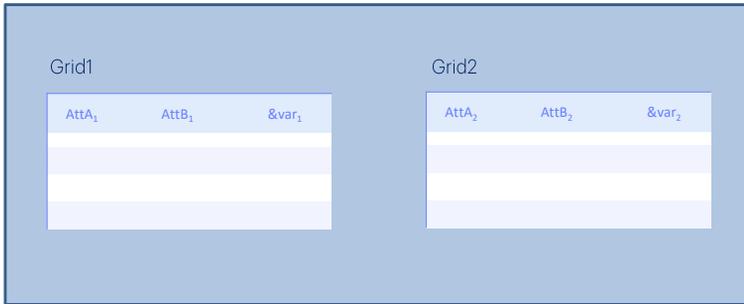
Primero el evento Start (por única vez).

Luego el evento Refresh genérico, es decir el evento Refresh propio del panel.

Luego el Refresh del primer grid y luego si éste tiene tabla base se ejecutará el evento Load del grid tantas veces como registros se recuperen de la base de datos, filtrando los registros que correspondan. Si no tiene tabla base, entonces se ejecuta el evento Load del grid por única vez y si es un grid basado en un SDT no se ejecuta el evento Load.

Y luego lo mismo con los eventos Refresh y Load del segundo 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
  
```

Al haber más de un grid, el comando Refresh también debe especializarse para indicar a qué grid se quiere refrescar.

El comando **Refresh** genérico (el que habíamos visto cuando lo usamos en el panel Attractions_CFPANEL) provoca que se ejecuten el Refresh genérico, y el Refresh y Load de cada grid (es decir, todo menos el Start).

Y ahora tenemos también el método Refresh de un grid, que hará que se refresque solo ese grid, es decir que se ejecuten el Refresh del grid y el Load del grid (n veces, una vez, o ninguna), dependiendo si el grid tiene tabla base, no tiene o es un grid de una variable SDT colección, respectivamente.

En este video vimos como podemos trabajar con múltiples grids en un objeto Panel, en este caso con grids paralelos y las consideraciones que tenemos que tener en la invocación de los eventos de cada grid. En este curso no abordaremos al tema grids anidados en un objeto panel.

GeneXus™

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