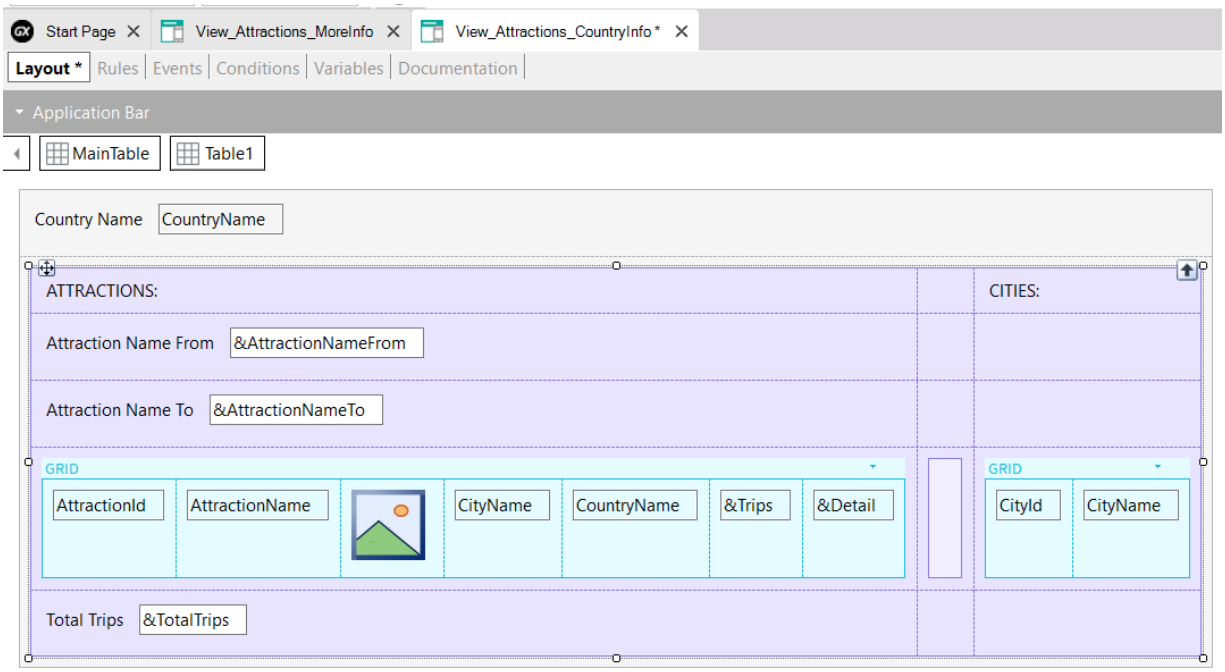
Panel con múltiples grids

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. Esto nos brindará los conocimientos necesarios para seguir avanzando en el desarrollo de nuestra aplicación para la agencia de viajes.

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 View_Attractions_MoreInfo y le ponemos de nombre View_Attractions_CountryInfo. Luego le quitamos que sea objeto main, ya que la idea es que el mismo se llame desde la grilla de atracciones cuando se haga clic sobre el nombre de un país.

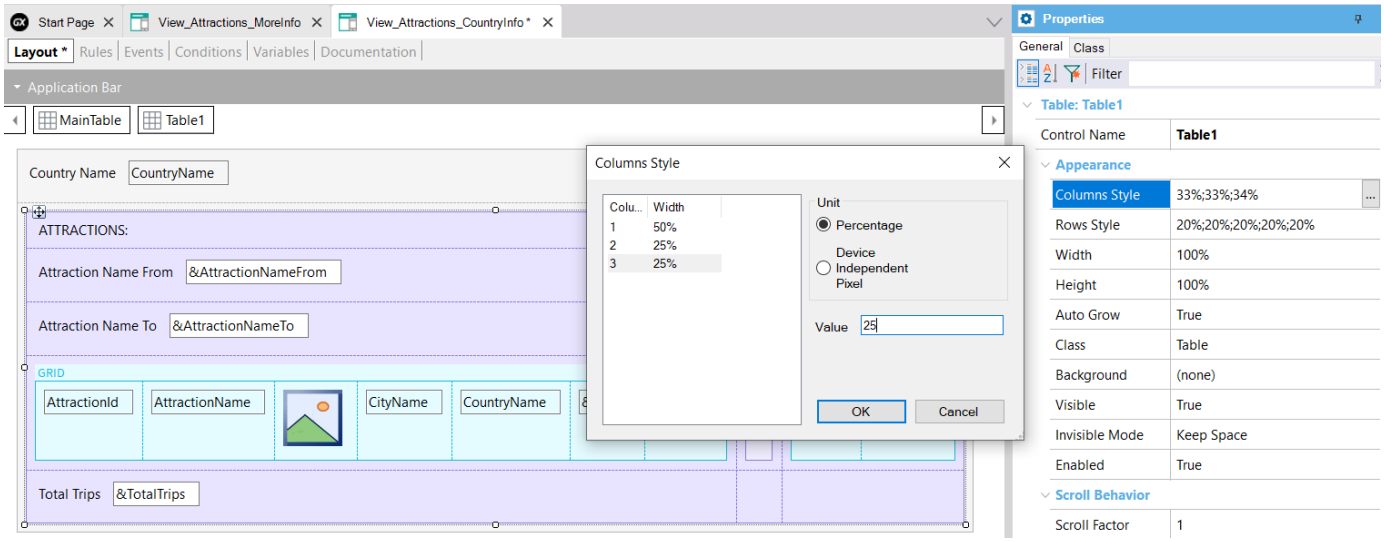
En el form, vamos a eliminar los títulos que teníamos, ya que mostraremos títulos por sección y en el grid vamos a agregar una variable &Detail, del tipo Character. Esta variable nos servirá más adelante para invocar a un panel que muestre el detalle de cada atracción.
En los eventos, eliminamos las líneas que asignaban el texto a los títulos.

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. Quitamos la variable de las conditions. Eliminamos el evento ControValueChanged asociado a la variable que quitamos, la cláusula where que la usaba y en la regla Parm cambiamos la variable &CountryId por el atributo CountryId

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.

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



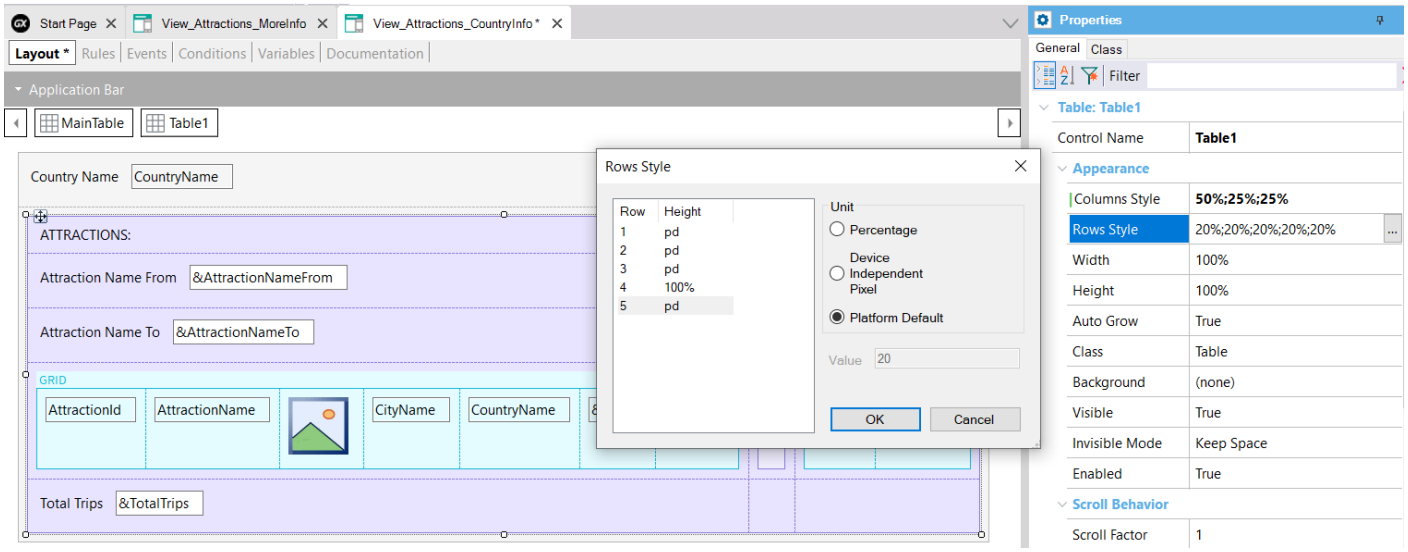
Para definir cómo queremos que el contenido de una tabla sea visto, debemos cambiar el tamaño de las filas o de las columnas de la tabla. Por ejemplo, queremos que los grids tengan espacio suficiente en la fila correspondiente para poder desplegarse correctamente y asignar más espacio a la columna de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

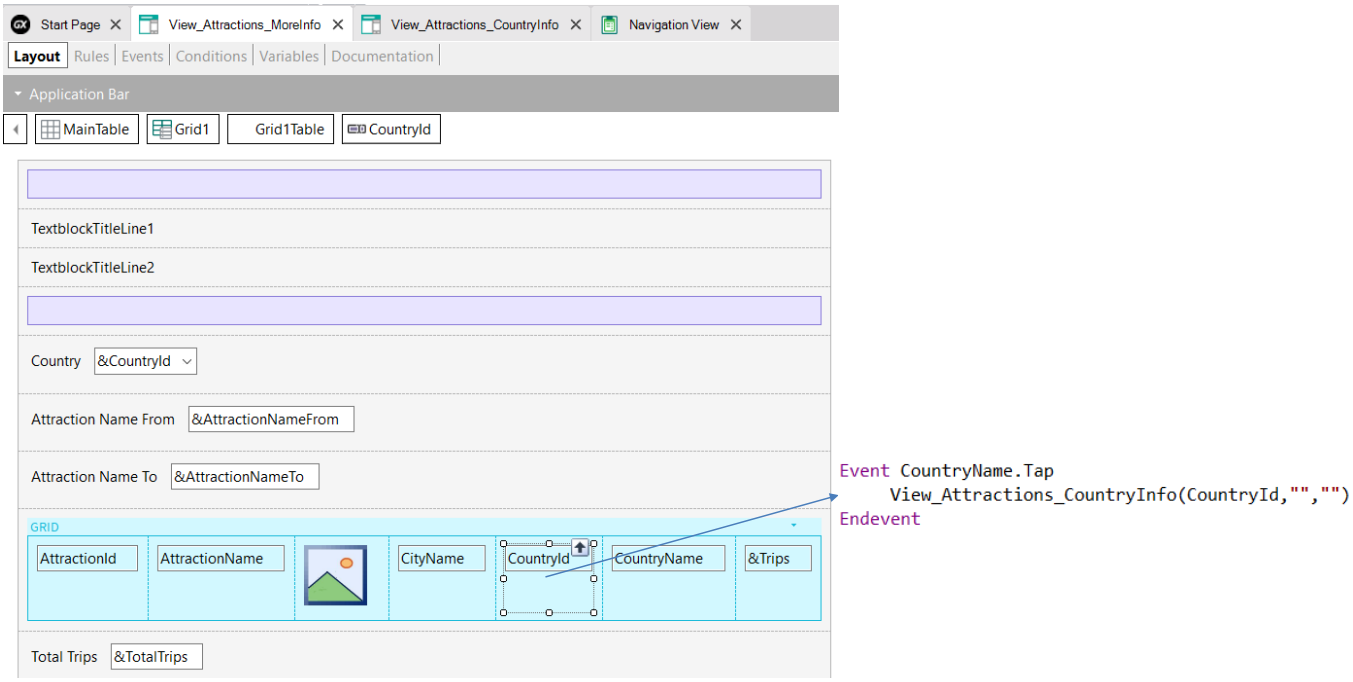


Ahora cambiamos el alto de las filas. Hacemos clic sobre la propiedad Rows Style y vemos que por defecto todas las filas tienen un 20% asignado, menos la cuarta fila que es donde están ubicadas las grillas.

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



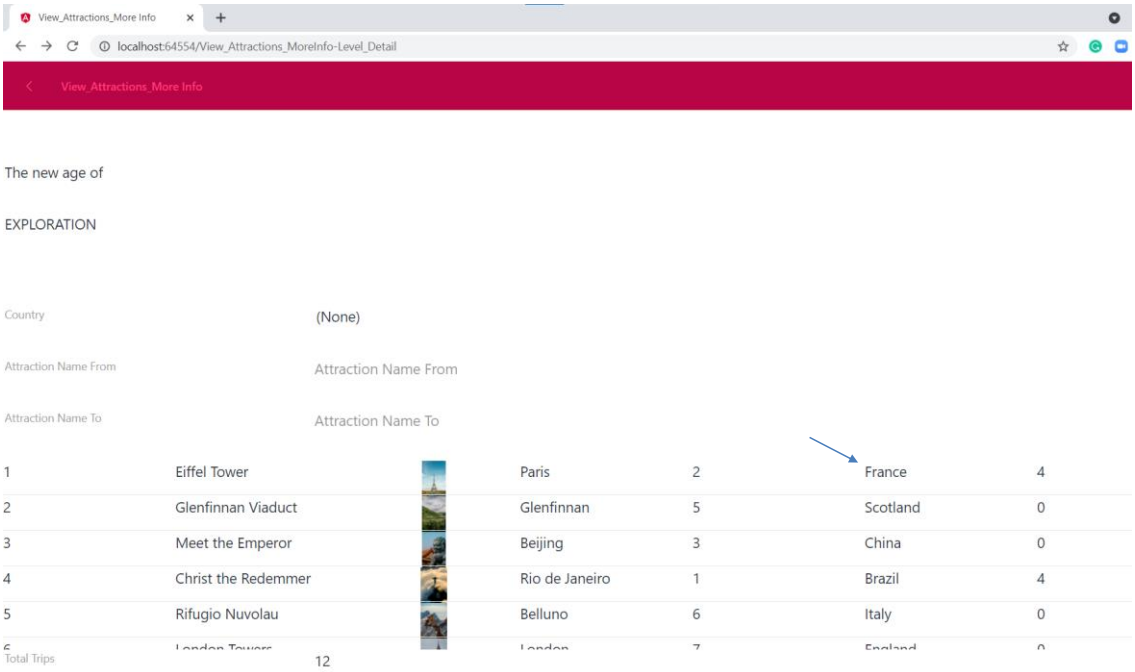
Si damos botón derecho para ver la navegación del objeto que construimos, vemos que en la ventana de Output muestra un error y si vemos el error en el listado de navegación nos dice que el evento Load no puede ser programado si tenemos múltiples grids.
Si vamos a los eventos, vemos que todavía tenemos 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 View_Attractions_MoreInfo 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 View_Attractions_CountryInfo, pasándole el CountryId como parámetro.

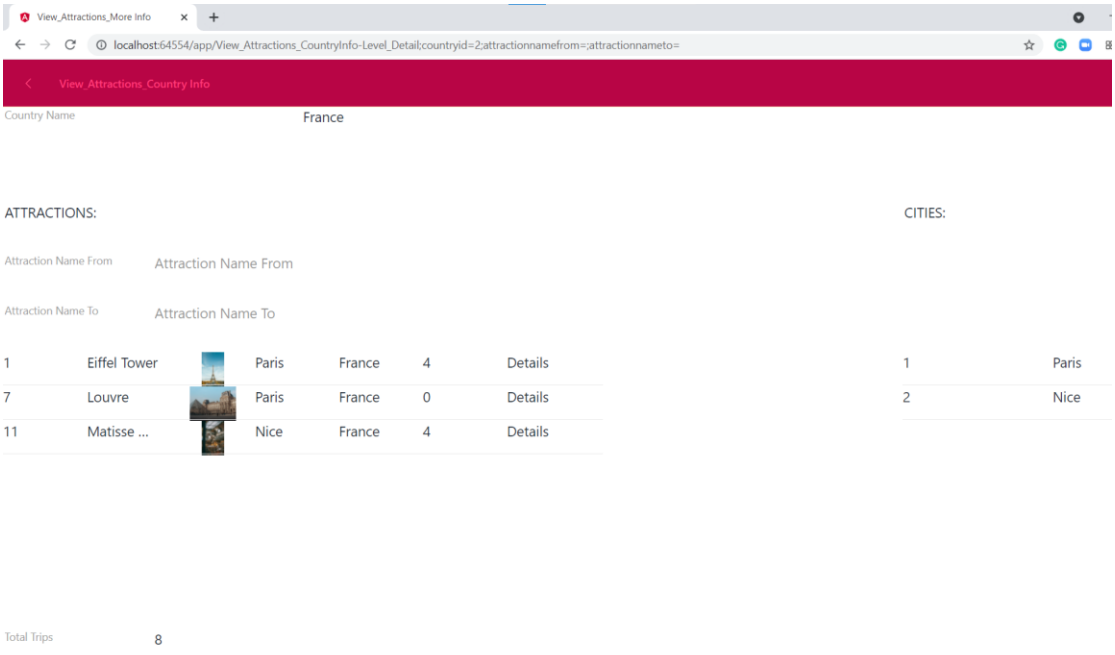
Ejecutemos para ver todo esto.

Ejemplo en ejecución



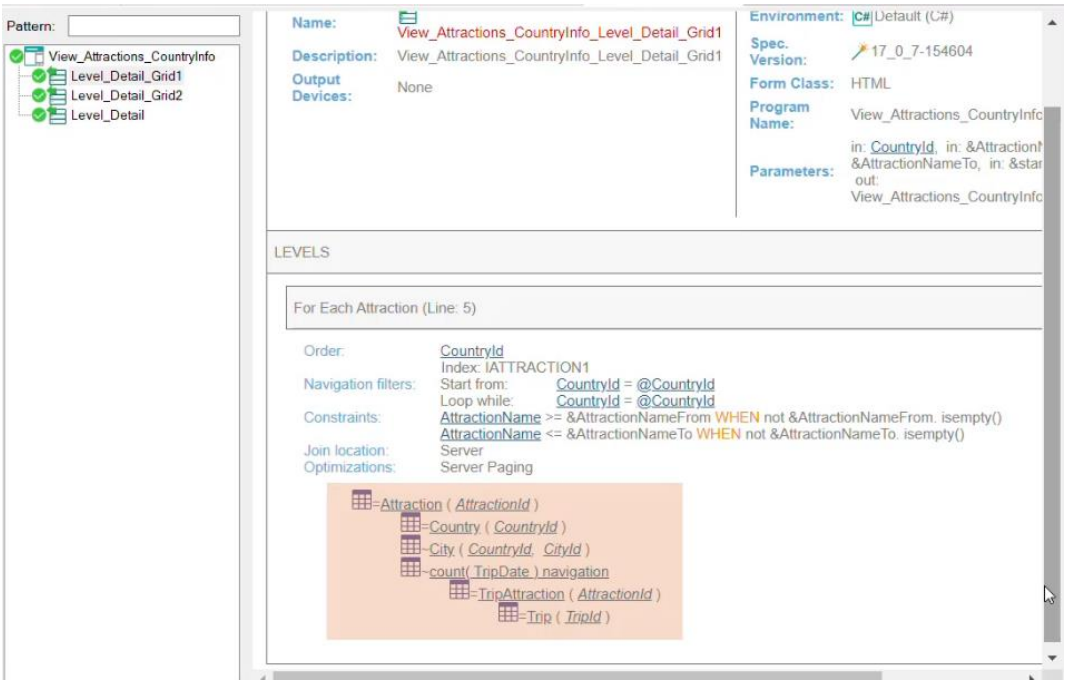
Si hacemos clic sobre France…

Ejemplo en ejecución



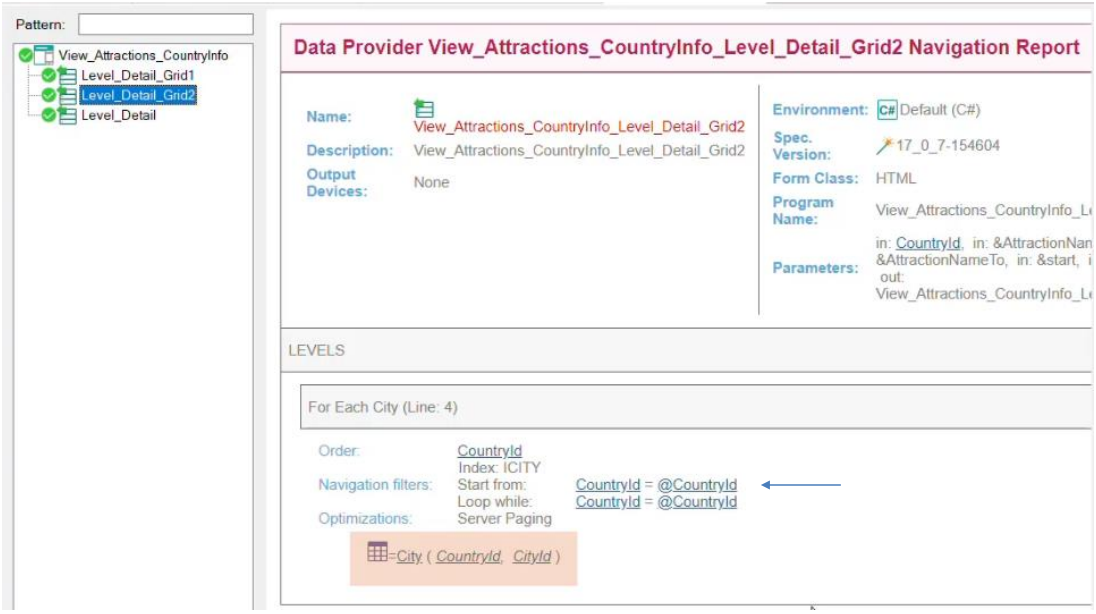
Se abre la información del país Francia, mostrándonos las atracciones con el total de viajes de cada atracción y el total de viajes a Francia y a la derecha las ciudades de ese país.

Listado de navegación del nuevo panel



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 View_Attractions_MoreInfo.

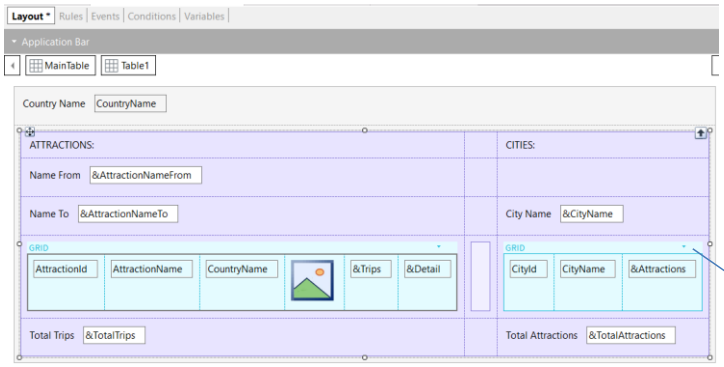
Listado de navegación del nuevo panel

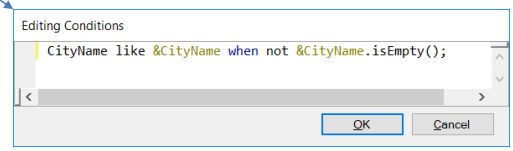


Si vamos al nodo correspondiente al Grid2, vemos que el grid está accediendo a la tabla City para mostrar las ciudades y que se está filtrando por CountryId, por el atributo que se recibe en la regla Parm.

Agregamos filtro por ciudad y totales por ciudad y país

```
Event Start
     &Detail = "Details"
Endevent

Event Grid1.Load
     &Trips = Count(TripDate)
Endevent

Event Grid2.Load
     &Attractions = Count(AttractionName)
Endevent

Event Grid1.Refresh
     &TotalTrips = 0
     For each Trip.Attraction
          Where AttractionName >= &AttractionNameFrom when not &AttractionNameFrom.IsEmpty()
          Where AttractionName <= &AttractionNameTo when not &AttractionNameTo.IsEmpty()
          &TotalTrips += 1
     Endfor
Endevent

Event Grid2.Refresh
     &TotalAttractions = 0
     For each Attraction
          &TotalAttractions += 1
     Endfor
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.

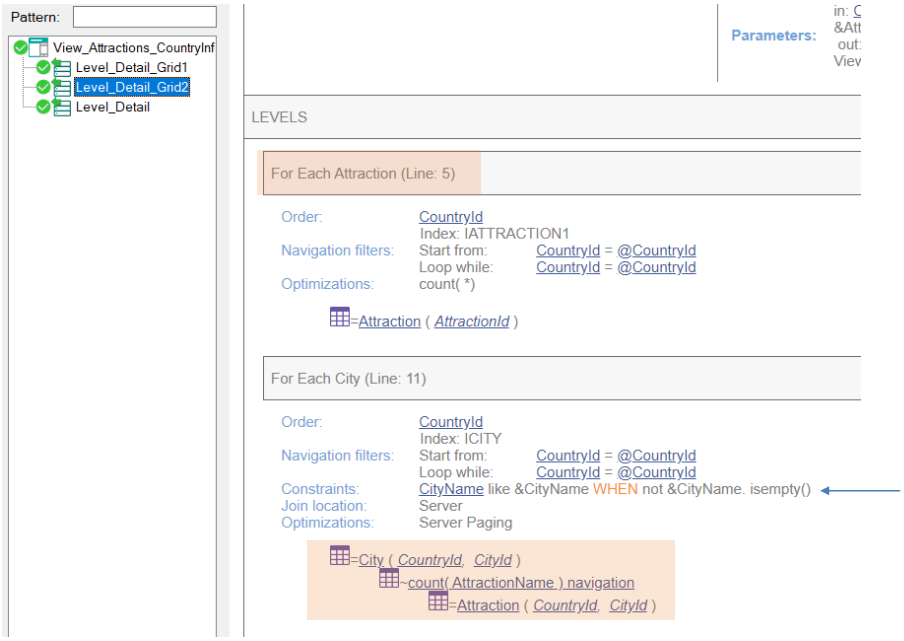
Antes que eso aprovechamos para asignar la propiedad Base Transaction en City. Ahora sí, hacemos clic en Conditions y escribimos la condición de filtro utilizando el operador like, ya que no filtraremos por rango de nombre, sino por un nombre parecido.

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 City, 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.

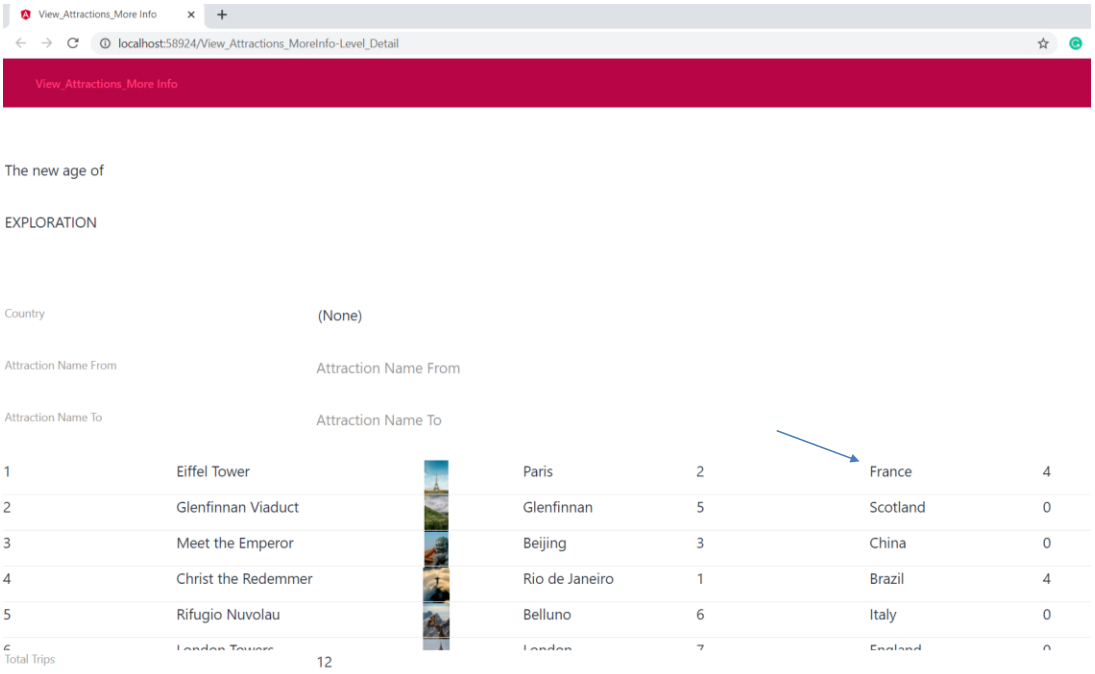
Listado de navegación del panel con los nuevos cambios



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 CityName.

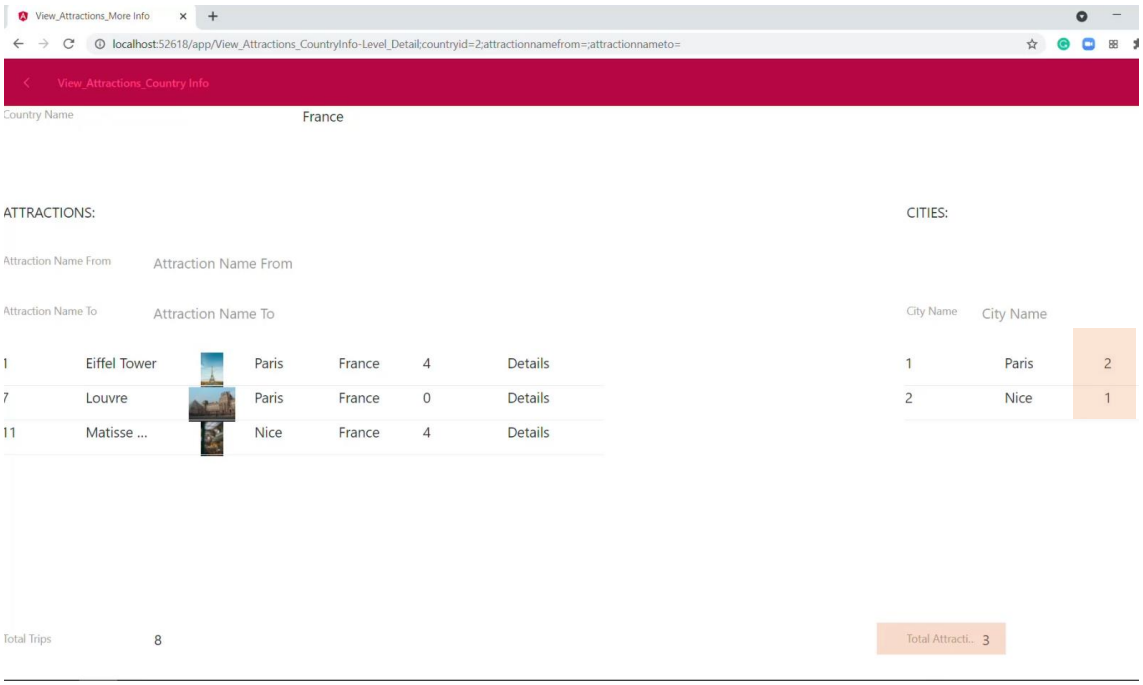
Viendo los totales por ciudad y país



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

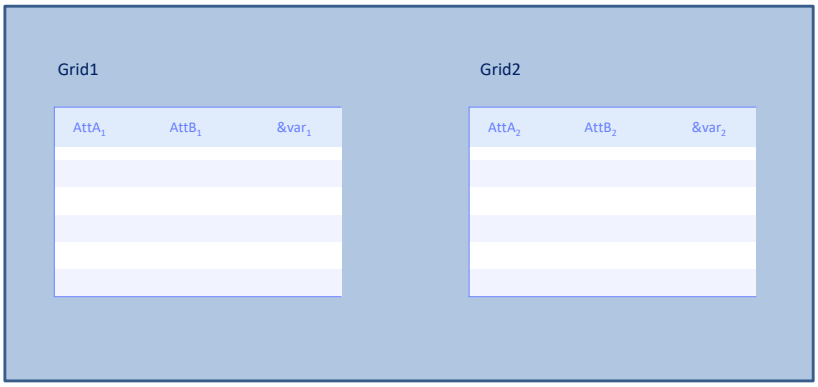
Hacemos clic en Francia nuevamente…

Viendo los totales por ciudad y país



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

Event execution order



Start

Refresh

Grid1.Refresh

Grid1.Load

Grid2.Refresh

Grid2.Load

Ya vimos que al tener más de un grid en el panel, debimos 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:

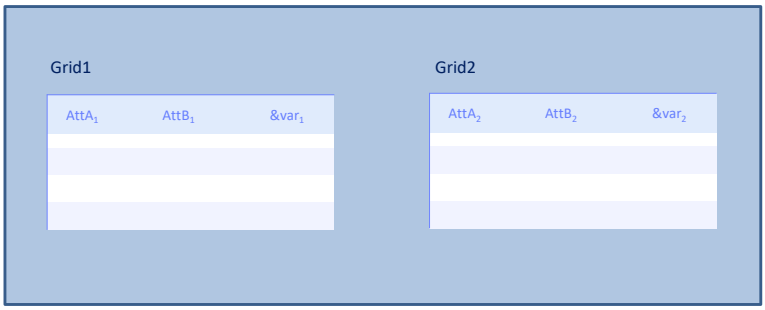
Primero el evento Start, por única vez (si es necesario ir al Servidor, como en el ejemplo).

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

Grid1.Load

Grid2.Refresh

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 View_Attractions_MoreInfo) 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.

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