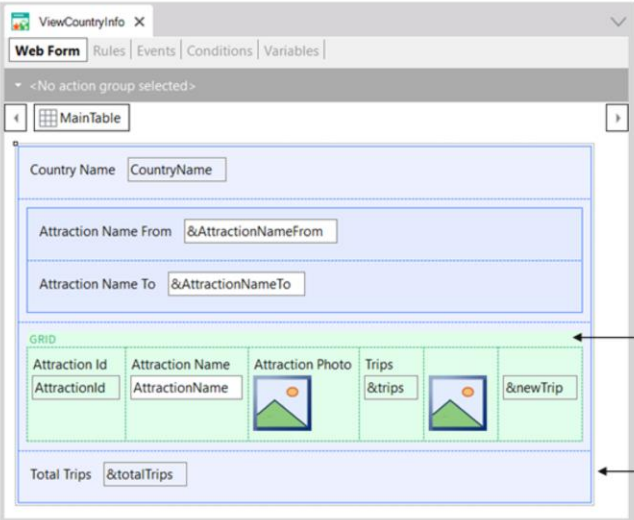


Telas web com foco em Back-Office

Objeto Web Panel. Múltiplos grids

GeneXus[™]

More than one grid



The screenshot shows a web form editor window titled 'ViewCountryInfo'. The form contains several input fields: 'Country Name' (CountryName), 'Attraction Name From' (&AttractionNameFrom), and 'Attraction Name To' (&AttractionNameTo). Below these is a green-shaded grid labeled 'GRID' with columns for 'Attraction Id' (AttractionId), 'Attraction Name' (AttractionName), 'Attraction Photo' (AttractionPhoto), 'Trips' (&trips), and a new trip button (&newTrip). At the bottom of the form is a 'Total Trips' field (&totalTrips). Arrows point from the text 'Grid1' to the grid and '+ Grid2' to the total trips field.

```

Event Grid1.Load
  &trips = Count(TripDate)
  &totalTrips = &totalTrips + &trips
Endevent

```

Dissemos algumas vezes que talvez tivesse sido melhor utilizar o evento Load do grid e não o genérico, que só serve no caso de um web panel sem grid ou com um grid. Usando o Load do grid, nos antecipamos ao futuro, para a possibilidade de inserir mais um grid.

New grid

The screenshot shows the GeneXus IDE interface. On the left, a web form is displayed with a 'Country Name' field, two 'Attraction Name' fields, a 'Total Trips' field, and a grid. The grid has columns for 'Attraction Id', 'Attraction Name', 'Attraction Photo', 'Trips', and '&newTrip'. On the right, the 'Properties' window for 'Grid: Grid2' is open, showing various settings and event handlers.

Properties Window - Grid: Grid2

Property	Value
Control Name	Grid2
Collection	
Base Tm	
Order	
Conditions	
Unique	
Save State	False
Data Selector	(none)
Appearance	
Class	Grid
Custom Render	
Empty Grid Text	
Auto Resize	True
Width	
Height	
Rows	0
Header	
Tooltip Text	
Layout	
Cell Padding	1
Cell Spacing	2
Behavior	
Sortable	True
Allow Drop	False

Event Handlers for Grid: Grid2

```

1 Event Load
2   &trips = Count(TripDate)
3   &totalTrips = &totalTrips + &trips
4 Endevent
5
6 Event Refresh
7   &totalTrips = 0
8 Endevent
9
10 Event Start
11   &update.FromImage(updateIcon)
12   &newTrip = "New trip"
13 Endevent
14
15 Event &update.Click
16   Attraction(trnMode.Update, AttractionId)
17 Endevent
18
19 Event &newTrip.Click
20   &trips = NewTrip(AttractionId)
21   Refresh
22 Endevent
23
24 Event AttractionName.Click
25   ViewAttractionFromScratch(AttractionId)
26 Endevent
  
```

Suponhamos que no web panel que mostra a informação de um país (seu nome e suas atrações turísticas), queremos adicionar também um grid com suas cidades. Antes de fazer isso, vejamos que sua lista de navegação indica a carga do -neste momento- único grid. Antes de adicionar o grid para as cidades, inserimos dentro de uma tabela tudo o que correspondia às atrações do país, para que fique toda essa informação junta.

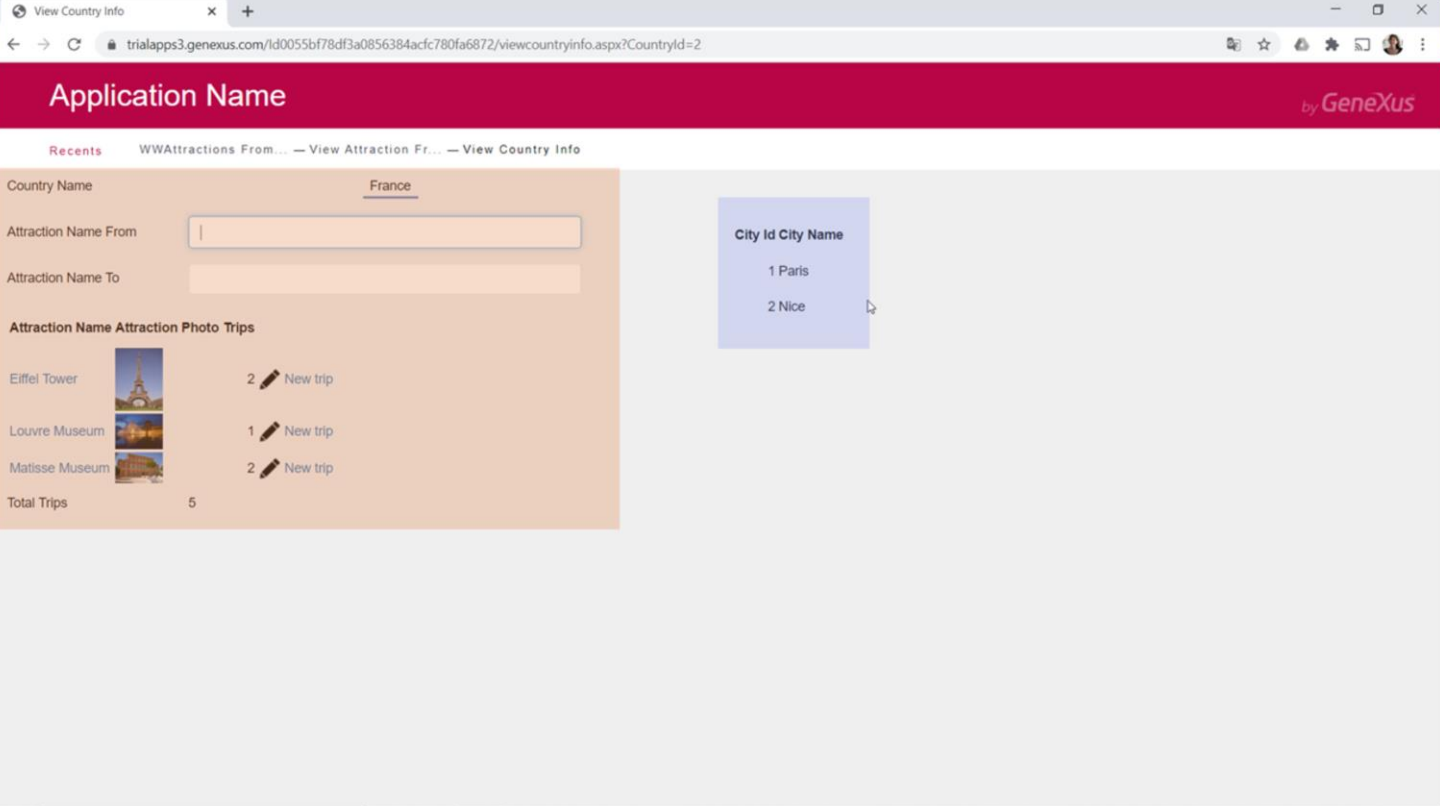
Depois, inserimos outra tabela para a informação das cidades. Ali dentro inserimos o novo grid, composto pelos atributos CityId e CityName.

Se observamos suas propriedades, vemos que o nomeou por padrão como Grid2.

Cada grid poderá ter ou não tabela base. Neste caso, ambos os grids têm atributos, assim ambos terão tabela base. Como se sabe a qual de ambos se aplica o código do evento Load genérico? De fato, se gravamos, vemos que a lista de navegação mostra um erro advertindo-o.

Está mostrando as navegações que deverá realizar para carregar cada grid, e até entendeu que a fórmula para calcular as trips deve pertencer à carga do Grid1, mas nos pede que determinemos isto. E é o que faremos.

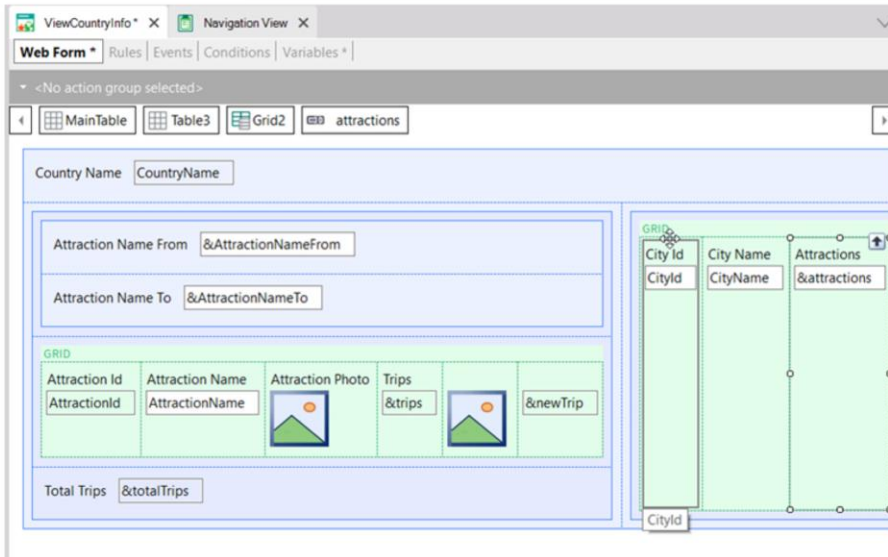
Agora, ao gravar o objeto, a lista de navegação já não mostra o erro.



Executemos.

Trata-se de duas navegações independentes, mas como as duas têm relação com o país recebido por parâmetro, em ambas se está filtrando por país.

New grid



```

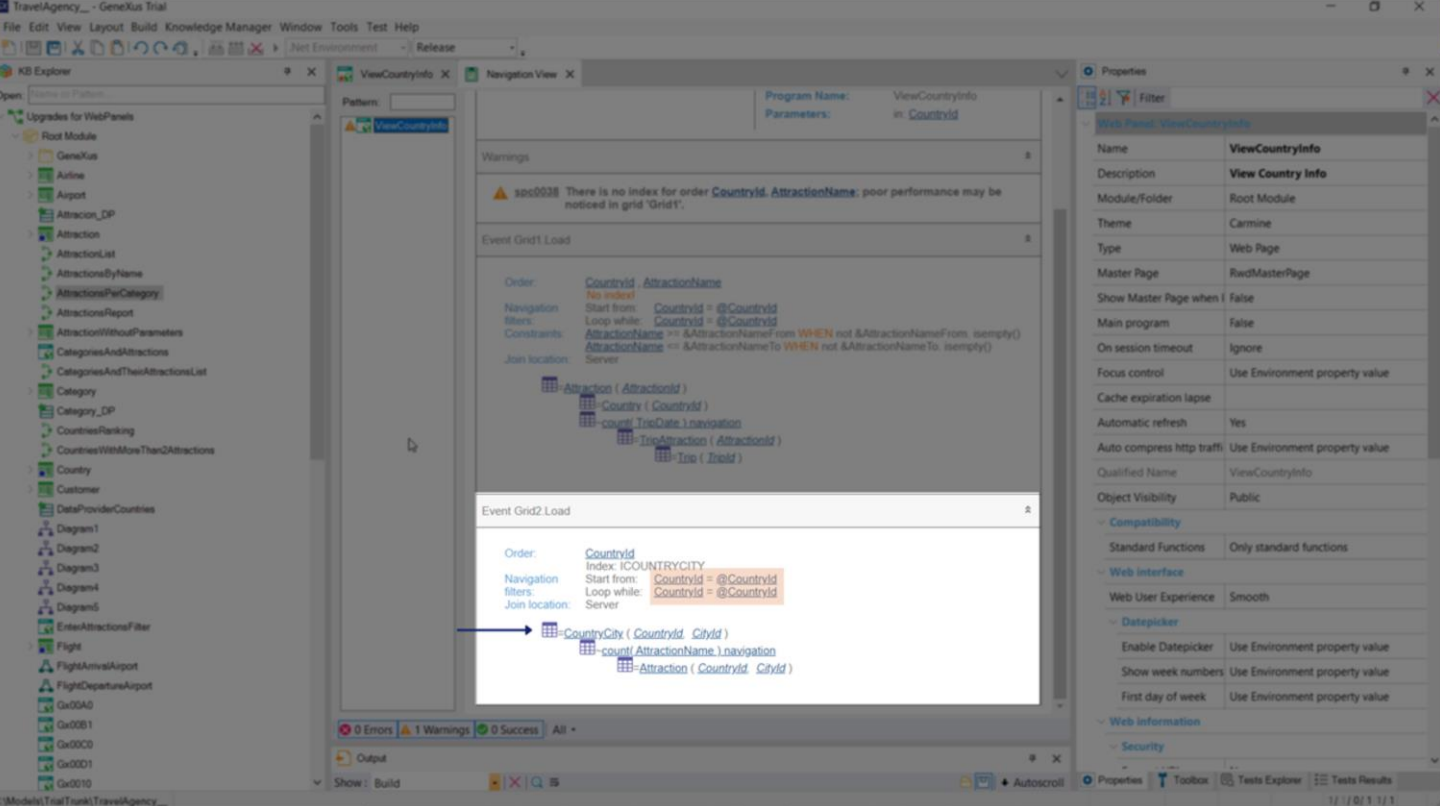
1 Event Grid1.Load
2     &trips = Count(TripDate)
3     &totalTrips = &totalTrips + &trips
4 Endevent
5
6 Event Grid2.Load
7     &attractions = Count(AttractionName)
8 endevent
9
10
11 Event Refresh
12     &totalTrips = 0
13 Endevent

```

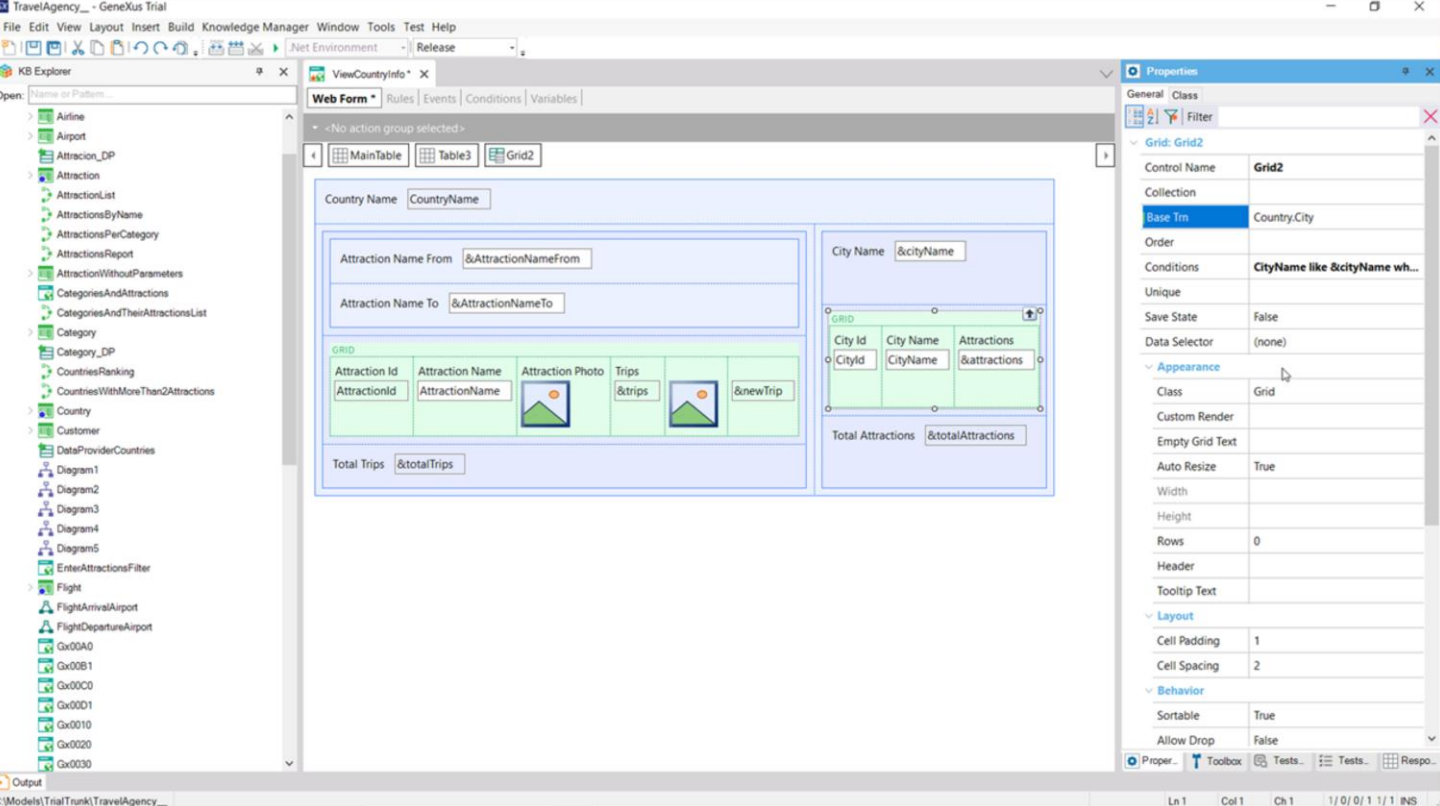
Se assim como para as atrações calculamos a quantidade de trips, para as cidades quiséssemos calcular a quantidade de atrações que cada uma tem... então, adicionamos uma variável &attractions ao grid, e a calculamos cada vez que se vai carregar uma linha, isto é, no evento Load do grid de nome Grid2.

Por que não é necessário condicionar esta fórmula para que conte só as atrações do país e cidade?

Executemos.



Enquanto gera, observemos a lista de navegação. Vemos que dentro do evento Load que será executado cada vez que se encontrar um registro da tabela das cidades que corresponda ao país recebido por parâmetro, dispara-se o cálculo da fórmula count sobre Attraction, filtrando por país e cidade.



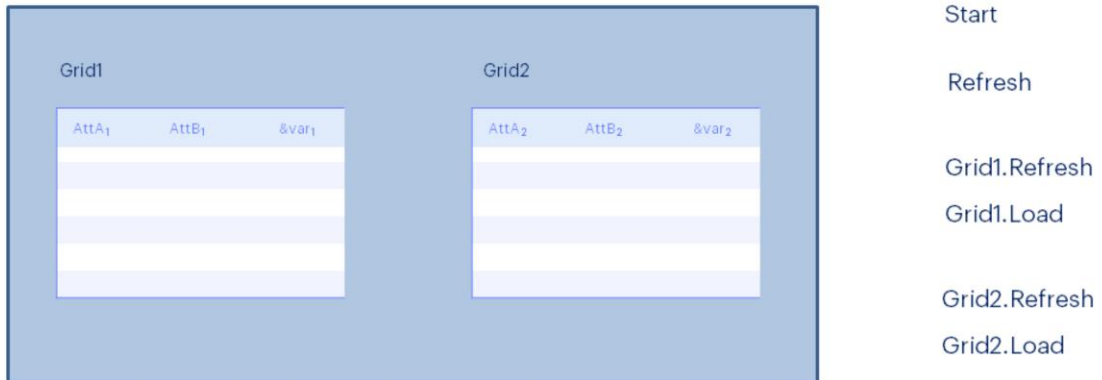
Para fazê-lo funcionalmente igual ao outro, podemos adicionar ao grid uma variável para filtrar as cidades mostradas e outra para mostrar o total de atrações de todas as cidades. Aqui já o fizemos.

Observemos que colocamos o filtro nas conditions do grid, utilizando o operador like. Não indicamos transação base e vimos que GeneXus a descobriu sozinho, mas nos convém fazê-lo.

Agora, tínhamos a inicialização da variável &totalTrips no evento Refresh genérico, e agora devemos inicializar também a variável &totalAttractions.

Mas temos, na verdade, três eventos Refresh: o genérico, que é o que temos por agora programado, e temos um Refresh de cada grid.

Event execution order



A ordem de execução dos eventos ao executar o web panel pela primeira vez será:

O evento Start

O evento Refresh genérico primeiro.

O Refresh do primeiro grid depois e, a seguir, se tiver tabela base, vai percorrer essa tabela filtrando os registros que correspondam, e executando o Load desse grid para cada um. Se não tem tabela base, então se executa o evento Load do grid apenas uma vez.

E depois, o mesmo com os eventos Refresh e Load do segundo grid.

Event execution order

The screenshot shows a user interface with two main sections. The left section contains a 'Country Name' filter, an 'Attraction Name From' to 'Attraction Name To' range filter, a 'GRID' table with columns for 'Attraction Id', 'Attraction Name', 'Attraction Photo', 'Trips', and '&newTrip', and a 'Total Trips' variable. The right section contains a 'City Name' filter, a 'GRID' table with columns for 'City Id', 'City Name', and 'Attractions', and a 'Total Attractions' variable.

```

Event Grid1.Refresh
  &totalTrips = 0
Endevent

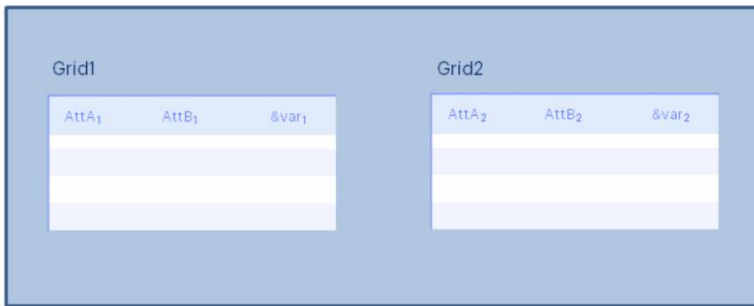
Event Grid2.Refresh
  &totalAttractions = 0
Endevent

```

No nosso exemplo, então, as variáveis `&totalTrips` e `&totalAttractions` deveriam inicializar-se no Refresh de cada grid, e não no genérico, porque depois, a ideia é que se mudarmos as variáveis de filtro de um grid, somente se atualize o que faz a esse grid, e não ao resto da tela.

Então, mudaríamos nossos eventos deste modo.

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
  
```

Claro, o aparecimento de mais grids faz com que o comando Refresh que tínhamos visto em outra aula programado em um evento de usuário, possa especializar-se.

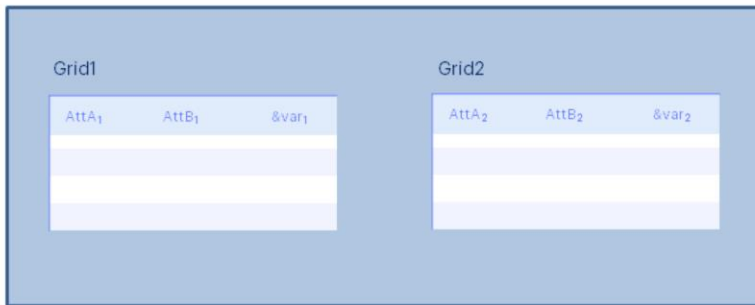
Por exemplo:

Temos o comando **Form.Refresh** que fará com que se atualize toda a página, executando-se Start, Refresh genérico, Refresh e Load de cada grid.

O comando **Refresh** genérico (o que tínhamos visto) faz com que se executem Refresh genérico, e Refresh e Load de cada grid (ou seja, tudo menos o Start).

E, agora, temos também o método Refresh de um grid, que fará com que se atualize só o grid, ou seja, que se executem o Refresh do grid e o Load do grid (uma vez ou n, dependendo de, se não tem ou tem tabela base).

Load command?



```

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

```

```

Event Grid1.Load
...
Load
endevent

```

```

Event 'User-event'
...
Grid1.Load()
endevent

```

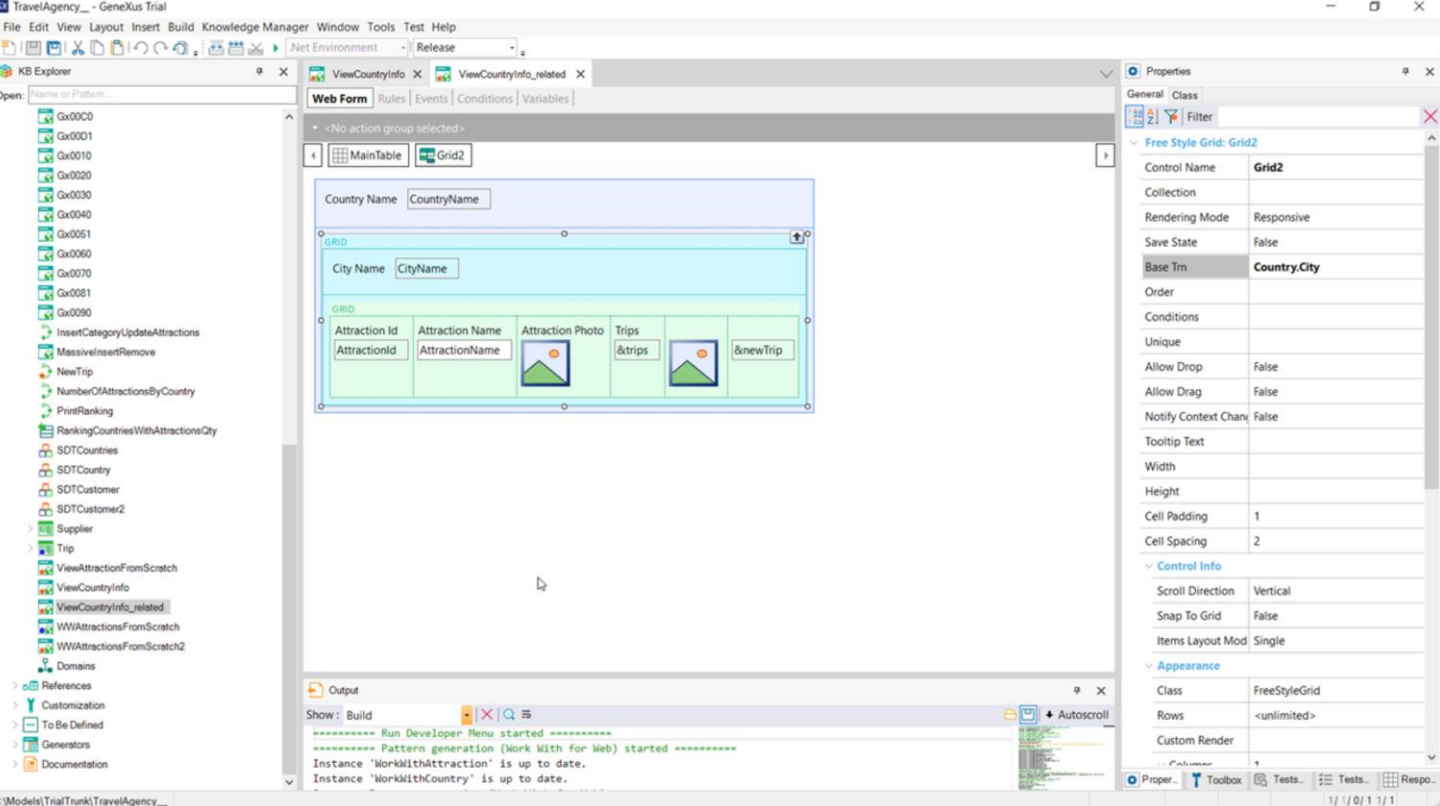
Para o caso do comando Load a coisa muda um pouco.

Quando houver mais de um grid, o **comando Load** só poderá ser escrito dentro do **evento Load** do grid de que se trata.

E se quiser carregar uma linha em um dos grids a partir de um evento de usuário, para isso terá que utilizar, necessariamente, o **método Load** do grid de que se trata.

Parallel or Nested Grids?

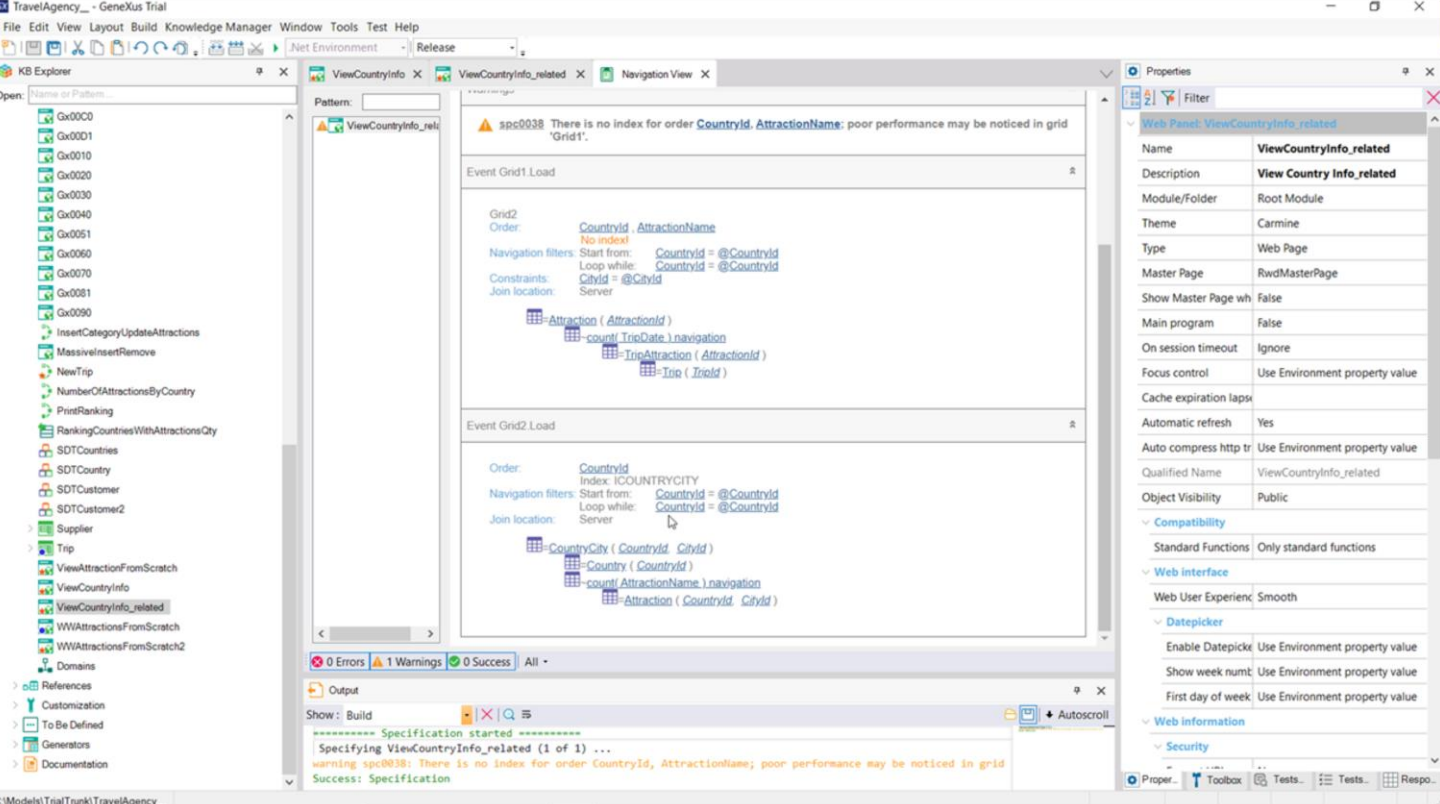
Aqui só vimos um exemplo de grids paralelos, mas os grids também podem aninhar-se, como os for eachs.



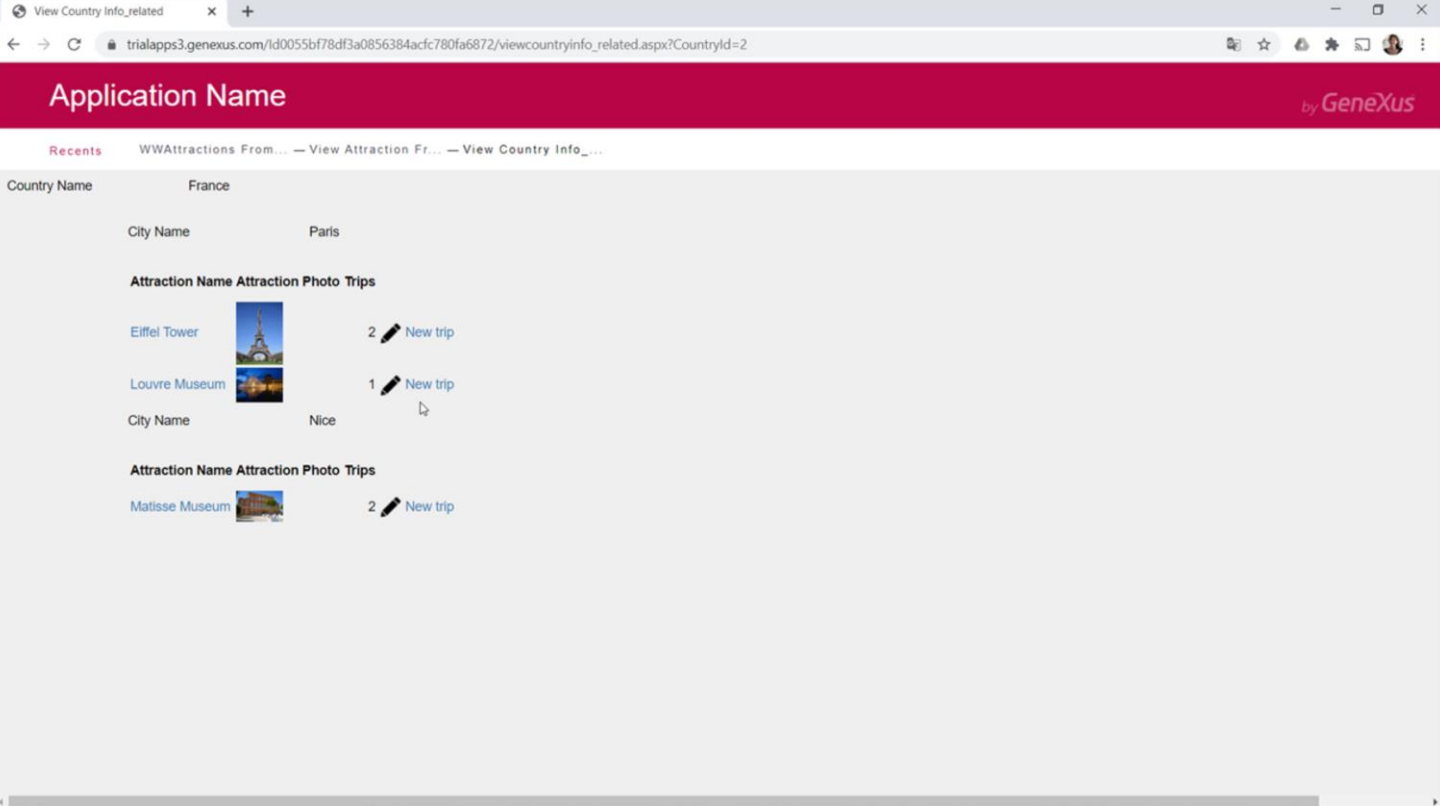
Por exemplo, se quiséssemos mostrar o país escolhido com a informação que vimos antes, mas de maneira relacionada.

Aqui o implementamos. Para que um grid possa conter outro, tem que ser um tipo especial de grid, de estilo livre e não tabular. Se chama Grid Freestyle.

Este grid vai percorrer a tabela CountryCity e para cada cidade encontrada, vai executar o Refresh e Load do segundo grid –o grid aninhado- que vai buscar as atrações desse país-cidade. Aqui podemos ver como a informação está relacionada.



Se observamos a lista de navegação, vemos que o Grid2 que era o de cidades –está percorrendo essa tabela, CountryCity, filtrando por país recebido por parâmetro- e então, no Load do Grid1 que é o que corresponde às atrações, está sendo percorrida a tabela de atrações filtrando pelo país e a cidade na qual se está posicionado em cada registro de CountryCity e por isso está aparecendo este arroba aqui por CityId.



Se executamos. Aqui o vemos. Paris e suas duas atrações. Nice e sua atração.

Parallel or Nested Grids?

Há muito mais para pesquisar sobre este tema. Aqui, nos contentamos com esta introdução.



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