

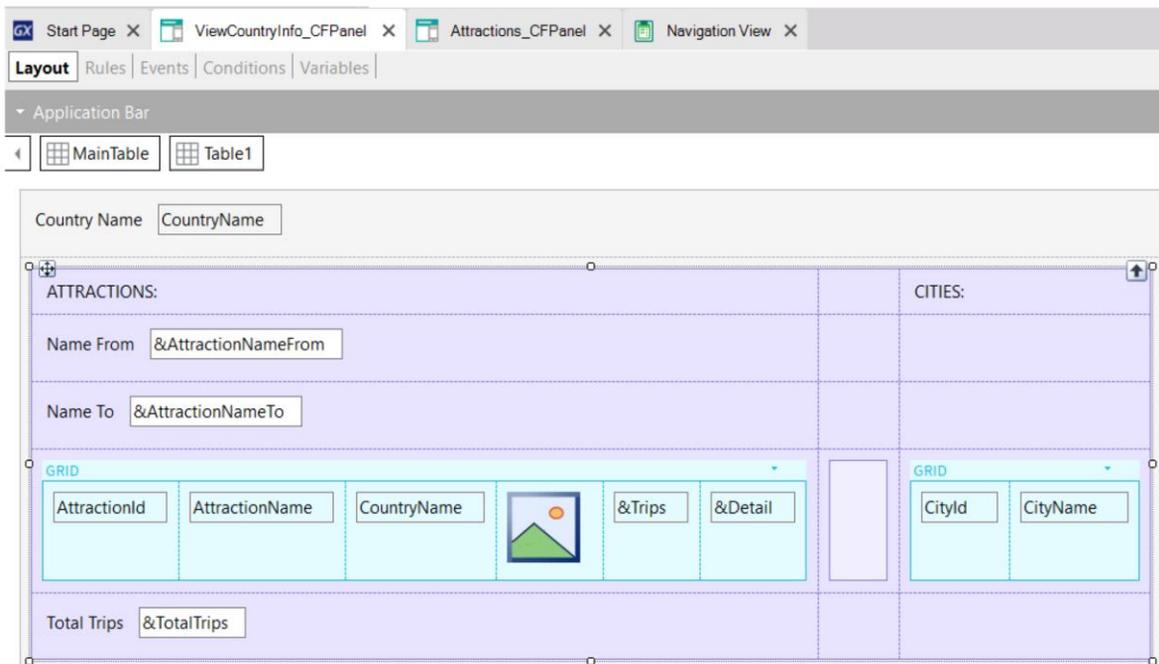
Telas web com foco em Customer-facing

Uso de múltiplos grids

GeneXus[™]

Vimos como usar um grid em um objeto panel. Veremos agora as considerações que precisamos ter se adicionamos mais de um grid ao panel.

Objeto panel com mais de um grid



Vamos construir um objeto panel que mostre as atrações e as cidades de um país recebido por parâmetro.

Para isso fazemos um Save As do panel Attractions_CFPanel e lhe colocamos como nome ViewCountryInfo_CFPanel. Em seguida removemos que seja objeto main, já que a ideia é que o mesmo seja chamado a partir do panel da lista de atrações quando se clica sobre o nome de um país.

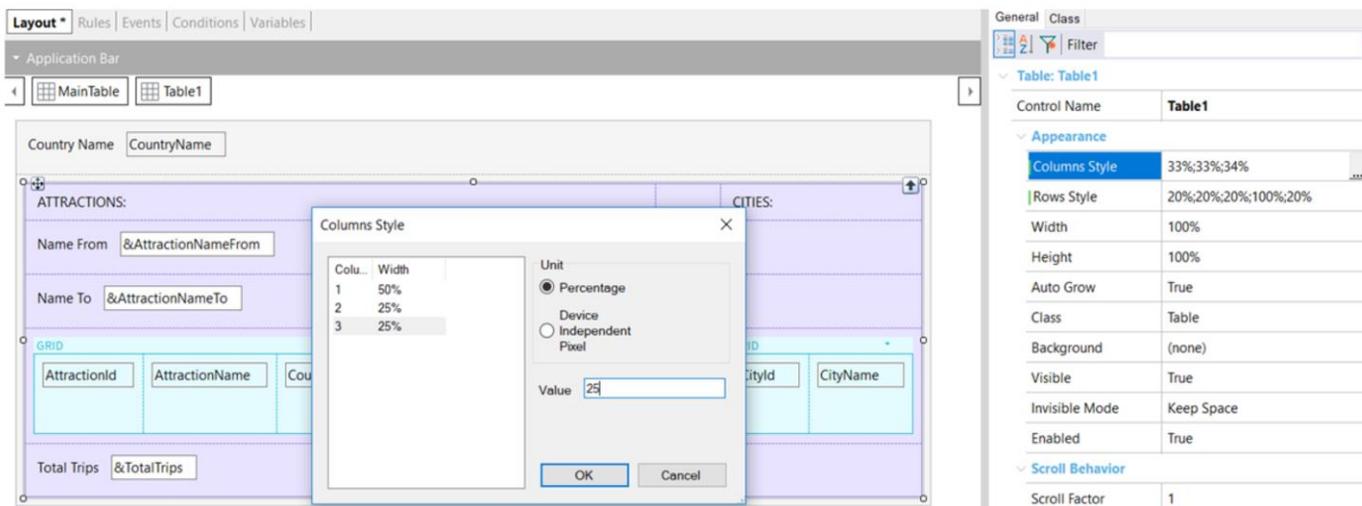
Agora vamos para a aba das regras e adicionamos uma regra Parm com um parâmetro de entrada, o atributo CountryId.

No form, adicionamos o atributo CountryName e eliminamos a variável do tipo Dynamic combo &CountryId, já que não vamos escolher um país mas o recebemos por parâmetro.

Para agrupar todos os dados das atrações e, depois, os dados das cidades, inserimos um controle Table a partir da barra de ferramentas e colocamos as variáveis AttractionNameFrom, AttractionNameTo, o grid e a variável TotalTrips dentro da tabela. Agora, à sua direita inserimos outra tabela como separador e, depois, inserimos um grid para mostrar as cidades do país, com os atributos CityId e CityName.

Adicionamos títulos às seções, colocando um textblock ATTRACTIONS: acima da seção de atrações e CITIES: acima da seção de cidades. Mudamos a classe para TextBlockTitle.

Alterando o estilo das linhas e colunas de uma tabela



Para definir como queremos que o conteúdo de uma tabela seja visto, devemos mudar o tamanho das linhas ou das colunas da tabela. Por exemplo, queremos que os grids tenham espaço suficiente na linha correspondente para poderem ser exibidas corretamente e atribuir mais espaço à coluna da tabela que mostra o conteúdo das atrações, porque temos mais coisas para mostrar do que para as cidades.

Para isso contamos com as propriedades Columns Style e Rows Style da tabela. Primeiro vamos mudar as colunas que sabemos que são 3: a coluna que contém os dados das atrações, a coluna com a tabela separadora e a coluna que contém os dados das cidades.

Se selecionamos a Table1 e clicamos na propriedade Columns Style, vemos que, por padrão, cada coluna ocupa 33% do espaço disponível.

Vemos que os valores possíveis para serem atribuídos à largura das colunas são em porcentagem do espaço total da tabela ou em Device Independent Pixels (DIPs). Esta medida nos permite atribuir unidades que são uma abstração de um pixel, que não dependem da plataforma e que depois serão convertidos a pixels reais no momento em que a aplicação seja executada. A quantidade de pixels que cada DIP ocupará dependerá das dimensões da tela. Isto nos permitirá escalar diferentes tamanhos de tela usando tamanhos uniformes.

Vamos atribuir à primeira coluna 50%, à segunda 25% e à terceira outros 25% do espaço disponível. Este espaço é o que fica depois de ter atribuído as colunas com dips, ou seja, as porcentagens são relativas ao valor que resulta da subtração da largura total, os valores fixos (em dips).

Alterando o estilo das linhas e colunas de uma tabela

The screenshot shows the GeneXus IDE interface. On the left, a table design is visible with a 'GRID' section containing columns for 'AttractionId', 'AttractionName', and 'CityName'. 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 options for 'Unit' (Percentage, Device, Independent Pixel, Platform Default) and a 'Value' field set to 20. On the right, the 'Table1' properties panel is visible, showing the 'Rows Style' property set to '20%;20%;20%;100%;20%'.

Agora mudamos a altura das linhas. Clicamos na propriedade Rows Style e vemos que, por padrão, todas as linhas têm 20% atribuído, menos a quarta linha que é onde estão localizados os grids.

Vemos que aqui podemos atribuir os valores em porcentagem, em DIPS e em Platform Default. Este último valor difere de plataforma para plataforma e para uma mesma plataforma, também depende do conteúdo da célula.

Vamos atribuir esse valor a todas as linhas exceto à linha 4, onde estão os grids, que atribuímos que ocupem os 100% que fiquem disponíveis.

Invocação ao painel de detalhe

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 main workspace displays a form with the following elements:

- A dropdown menu for 'Country' with the value '&CountryId'.
- Text input fields for 'Name From' (value: '&AttractionNameFrom') and 'Name To' (value: '&AttractionNameTo').
- A grid with columns: 'AttractionId', 'AttractionName', 'CountryId', 'CountryName', a landscape image icon, '&Trips', and '&Detail'.
- A 'Total Trips' field with the value '&TotalTrips'.

An arrow points from the 'CountryName' column of the grid to a code snippet in the right-hand pane:

```

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

Se clicamos com o botão direito para ver a navegação do objeto que construímos, vemos que na janela de Output aparece um erro, e se vemos o erro na lista de navegação, nos diz que o evento Load não pode ser programado se temos múltiplos grids.

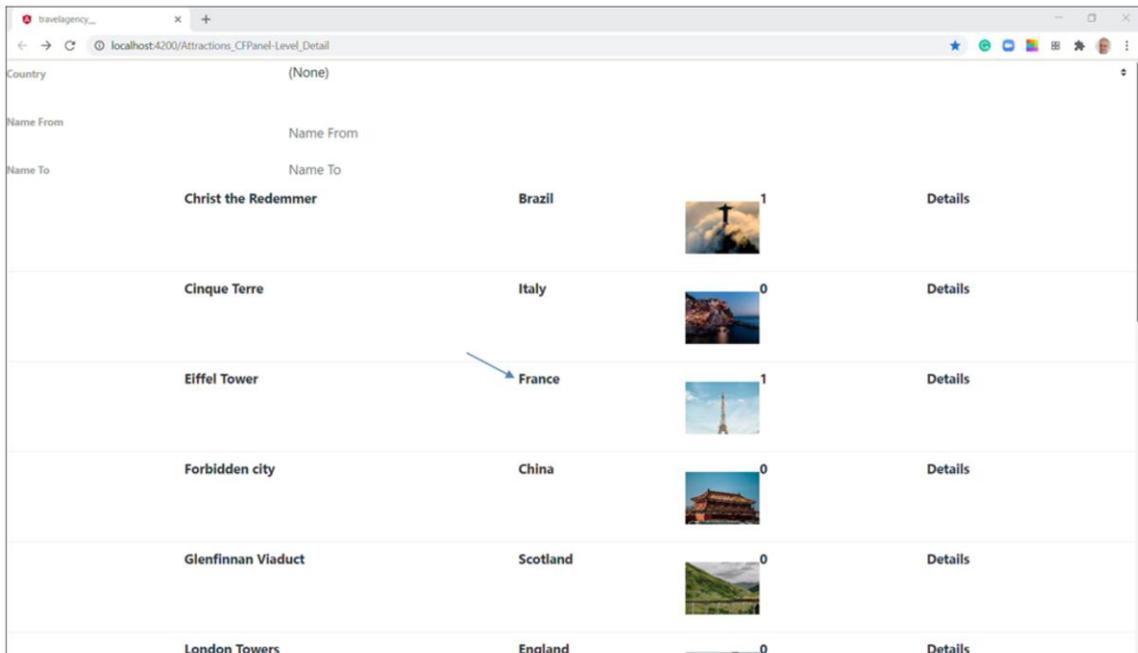
Se vamos aos eventos, vemos que ainda temos programado o evento Load como o tínhamos no objeto original. Aqui não damos conta do que mencionamos antes, que teria sido melhor utilizar o evento Load do grid, e não o genérico, para evitar essa situação.

Então, adicionamos Grid1 ao Load. Se fazemos View Navigation, vemos que o problema foi solucionado.

Antes de executar, vamos ao panel Attraction_CFPanel e adicionamos ao grid de atrações o atributo CountryId, que precisamos para passar ao panel de informações de um país quando clicamos no nome do país no grid. Depois, adicionamos ao atributo CountryIName um evento Tap, onde escrevemos a invocação ao painel ViewCountryInfo_CFPanel, passando-lhe o CountryId como parâmetro.

Executamos para ver tudo isto.

Exemplo em execução



The screenshot shows a web browser window with a table of attractions. The table has columns for Name From, Name To, Country, a small image, a count, and a Details link. A blue arrow points to the 'France' entry in the 'Country' column.

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

Se clicamos em France...

Exemplo em execução

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

Country Name **France**

ATTRACTIONS:

Name From	Name To	Count	Details
Eiffel Tower		1	Details
Louvre		0	Details
Matisse Museum		1	Details

CITIES:

1	Paris
2	Nice

Total Trips **2**

Abre-se a informação do país França, mostrando-nos as atrações com o total de viagens de cada atração e o total global de viagens para França e, à direita, as cidades desse país.

Lista de navegação do novo 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

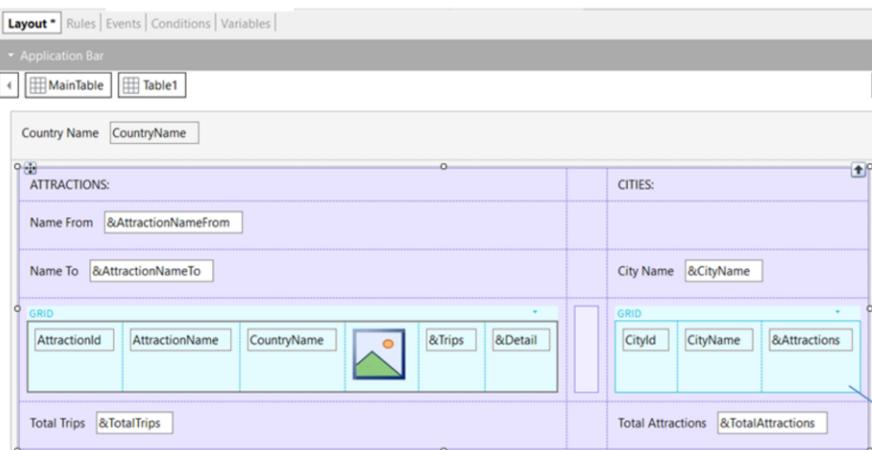
Optimizations: Server Paging

CountryCity (CountryId, CityId)

Se vamos à lista de navegação, vemos que agora há uma entrada Level_Detail para o Grid1 e outra para o Grid2. Para o Grid1 vemos o acesso à tabela Attraction e a navegação da fórmula que vimos quando implementamos o panel Attractions_CFPanel.

Se vamos ao nó correspondente ao Grid2, vemos que o grid está acessando a tabela CountryCity para mostrar as cidades e que está filtrando por CountryId, pelo atributo que se recebe na regra Parm.

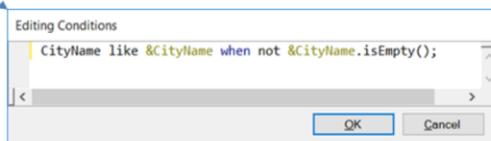
Adicionamos filtro por cidade e totais por cidade e país



```

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

```



De forma similar aos totais que mostramos para as atrações e o filtro por nome, vamos mostrar a quantidade de atrações que cada cidade tem, o total de atrações do país e vamos adicionar um filtro por nome de cidade.

Na aba Variables criamos uma variável &Attractions, outra &TotalAttractions e outra &CityName.

Agora adicionamos a &Attractions ao grid ajustando sua propriedade Label Position em None. Em seguida, adicionamos &TotalAttractions abaixo do grid e a variável de filtro &CityName acima do grid de cidades. Vamos adicionar ao grid a condição necessária para o filtro.

Antes disso, aproveitamos para atribuir a propriedade Base Transaction em Country.City. Agora sim, clicamos em Conditions e escrevemos a condição de filtro utilizando o operador like, já que não filtraremos por faixa de nome, mas por um nome parecido.

Em seguida, adicionamos o evento Grid2.Load onde carregamos a variável &attractions com uma fórmula Count com o atributo AttractionName. Como a tabela base do grid é CountryCity, a fórmula contará somente as atrações da cidade correspondente a cada linha.

Para calcular o total de atrações, pelas mesmas razões que fizemos quando calculamos o total de viagens, escrevemos o evento Grid2.Refresh, no qual programamos um For Each sobre a tabela de Atrações para contar as atrações, que ficarão filtradas pelo atributo do identificador de país recebido por parâmetro.

Lista de navegação do painel com as novas alterações

The screenshot displays the GeneXus IDE interface. On the left, a navigation tree shows the project structure: ViewCountryInfo_CFPANEL, Level_Detail_Grid1, Level_Detail_Grid2, and Level_Detail. The main workspace shows two 'For Each' blocks:

For Each Attraction (Line: 5)

- Order: CountryId
- Index: IATTRACTION1
- Navigation filters: Start from: CountryId = @CountryId; Loop while: CountryId = @CountryId
- Optimizations: count(*)
- Table: 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
- Table: CountryCity (CountryId, CityId)
- Table: count(AttractionName) navigation
- Table: Attraction (CountryId, CityId)

Se vemos agora a lista de navegação do painel, no nó Level_Detail_Grid2 vemos a navegação do For Each para a tabela Attraction e, mais abaixo, a navegação da fórmula Count sobre a tabela Attraction.

E aqui vemos o filtro que adicionamos por CountryName.

Vendo os totais por cidade e 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

Executemos nosso objeto main, Attractions_CFPANEL, para ver o que fizemos.

Clicamos novamente em França...

Vendo os totais por cidade e país

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 columns: 'City Name' and 'City Name'. The table has 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 summary box labeled 'Total Attract...' with the value '3'.

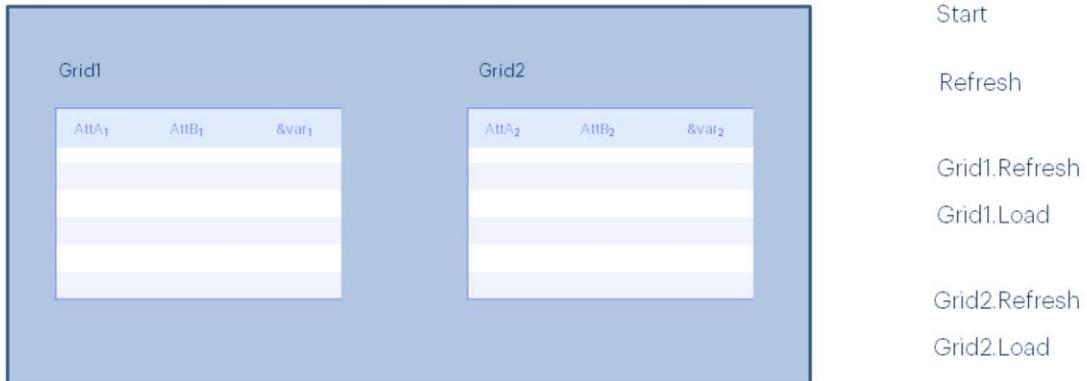
City Name	City Name	
1	Paris	2
2	Nice	1

Total Trips: 2

Total Attract...: 3

e agora podemos ver o total de atrações de cada cidade da França e o total de atrações do país França.

Event execution order



Já vimos que ao ter mais de um grid no panel, devemos usar o evento Refresh e Load de cada grid.

Mas ao ter adicionado estes eventos, surge-nos a questão de qual seria a ordem de disparo destes eventos em relação aos eventos próprios do objeto panel.

Ao executar o objeto panel pela primeira vez, a ordem de disparo de execução dos eventos será:

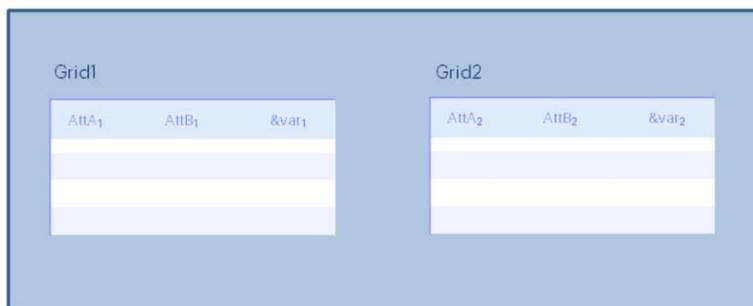
Primeiro o evento Start (apenas uma vez).

Em seguida, o evento Refresh genérico, ou seja, o evento Refresh próprio do panel.

Depois o Refresh do primeiro grid e depois, se este tem tabela base, será executado o evento Load do grid tantas vezes quanto registros se recuperem da base de dados, filtrando os registros que correspondam. Se não tem tabela base, então é executado o evento Load do grid apenas uma vez e, se for um grid baseado em um SDT, não executa o evento Load.

E depois, o mesmo com os eventos Refresh e Load do 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
  
```

Havendo mais de um grid, o comando Refresh também deve ser especializado para indicar qual grid quer atualizar.

O comando **Refresh** genérico (o que tínhamos visto quando o usamos no painel Attractions_CFPANEL) faz com que se executem o Refresh genérico, e o 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 esse grid, ou seja, que se executem o Refresh do grid e o Load do grid (n vezes, uma vez, ou nenhuma), dependendo se o grid tem tabela base, não tem ou é um grid de uma variável SDT coleção, respectivamente.

Neste vídeo vimos como podemos trabalhar com múltiplos grids em um objeto Panel, neste caso com grids paralelos e as considerações que temos que ter na invocação dos eventos de cada grid. Neste curso não abordaremos o tema grids aninhados em um objeto panel.

GeneXus™

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