

Telas interativas
Objeto Web Panel

GeneXus® 16

Objeto Web Panel

- Implementa uma tela web extremamente flexível
- Exemplo:

The screenshot shows the GeneXus Web Panel designer interface. The title bar is 'EnterAttractionsFilter'. The 'Web Form' tab is selected and highlighted with a red box. Below the tabs, there's a status bar indicating '<No action group selected>'. A 'MainTable' is visible. The form contains three input fields: 'Country Id' with a dropdown arrow, 'Attraction Name From' with a text input, and 'Attraction Name To' with a text input. Each field is preceded by a variable symbol (&). A red bracket on the right side of the form groups these three fields. Below the form, there are two buttons: 'List Attractions By Country' and 'List Attractions By Name'.

Variáveis: controles de entrada
(não readonly)

O web panel é o objeto mais flexível que GeneXus fornece.

Como já vimos em alguns exemplos que mostramos, todo web panel oferece um web form, que é uma página web que nos permite desenhar e oferecer funcionalidades variadas.

Neste exemplo, tínhamos visto que, ao incluir variáveis no web form, estas ficavam habilitadas para que o usuário inserisse algum valor. Ou seja, eram controles de entrada, ou, em outras palavras, não readonly.

Exemplo: variáveis no Web panel

The screenshot displays the GeneXus IDE interface for a web panel named 'EnterAttractionsFilter'. The design view shows a form with a 'Country Id' dropdown, two text input fields for 'Attraction Name From' and 'Attraction Name To', and two buttons: 'List Attractions By Country' and 'List Attractions By Name'. The 'List Attractions By Country' button is highlighted with a red box, and a red arrow points from it to the 'Country Id' dropdown. Another red arrow points from the 'Country Id' dropdown to the 'Item Values' property in the Properties window. The Properties window shows the 'Country Id' dropdown is a 'Dynamic Combo Box' with 'CountryId' as the 'Item Values' and 'CountryName' as the 'Item Descriptions'. A red arrow points from 'CountryName' to a database icon labeled 'Tabela Country'.

Properties Window:

Attribute/Variable: &CountryId	
Control Name	&CountryId
Attribute	&CountryId
Label Position	Left
Label Caption	Country Id
ReadOnly	False
Return On Click	False
Appearance	
Control Info	
Control Type	Dynamic Combo Box
Data Source From	Attributes
Item Values	CountryId
Item Descriptions	CountryName
Sort Descriptions	True
Conditions	
Instantiated Attributes	
Empty Item	False
Notify Context Change	False

Event 'List Attractions By Country':

```

AttractionsList(&CountryId)
Endevent
  
```

Em particular, esta variável, do tipo combo dinâmico, esperava que o usuário escolhesse um país daqueles carregados no combo e pressionando o botão "List Attractions By Country", se executava o evento, invocando o pdf que listava as atrações desse país.

Aqui se acessa a base de dados apenas para carregar os valores do combo.

Exemplo: variáveis no Web panel

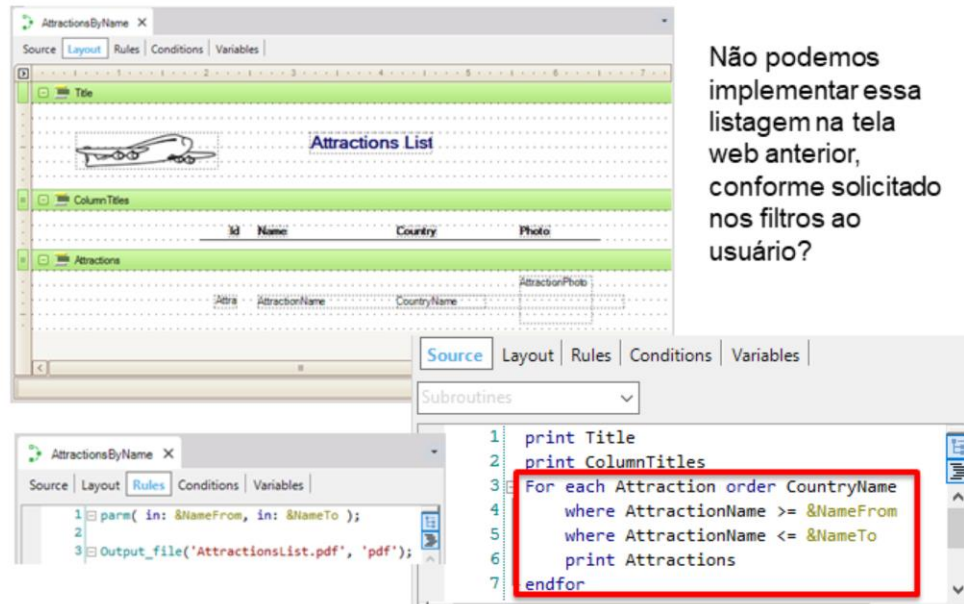
The screenshot displays the GeneXus IDE interface. On the left, a web panel titled 'EnterAttractionsFilter' is shown in design view. It contains a 'Country Id' dropdown menu, two text input fields labeled 'Attraction Name From' and 'Attraction Name To' (highlighted with a red rectangle), and two buttons: 'List Attractions By Country' and 'List Attractions By Name' (also highlighted with a red rectangle). The 'List Attractions By Name' button is connected to an event handler. On the right, the 'AttractionsByName' object is shown in 'Rules' view. The code for the 'List Attractions By Name' event is as follows:

```
1 parm( in: &NameFrom, in: &NameTo );  
  
Event 'List Attractions By Name'  
AttractionsByName( &AttractionNameFrom, &AttractionNameTo )  
Endevent
```

Red arrows point from the variables `&NameFrom` and `&NameTo` in the code to the corresponding input fields in the web panel design.

E nessas outras variáveis, o usuário inseria um intervalo de nomes de atração, de modo que, ao pressionar este outro botão, era invocada a lista pdf que mostrava as atrações dentro dessa faixa recebida por parâmetro.

Exemplo:



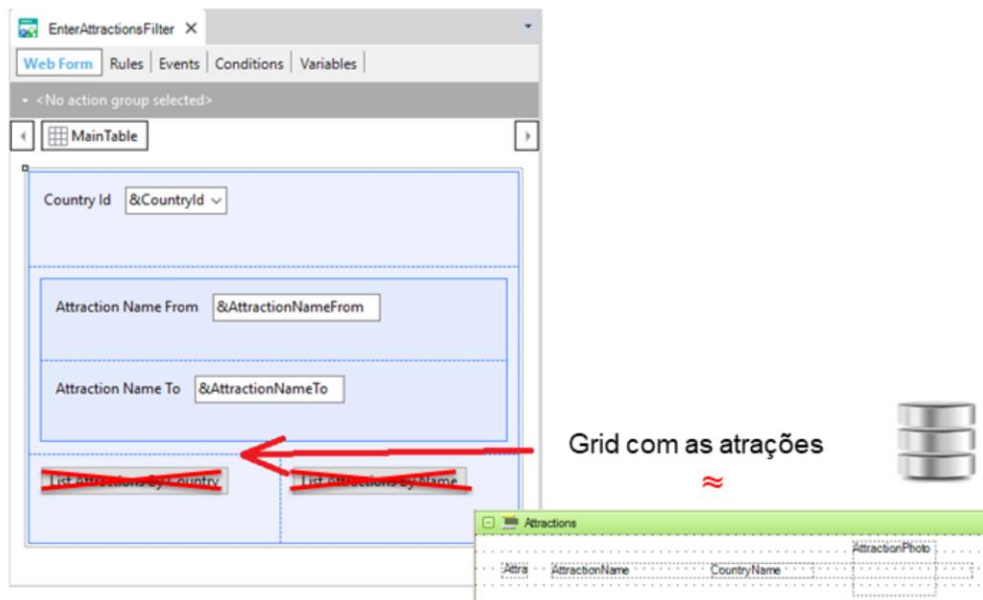
Não podemos implementar essa listagem na tela web anterior, conforme solicitado nos filtros ao usuário?

```
1 print Title
2 print ColumnTitles
3 For each Attraction order CountryName
4   where AttractionName >= &NameFrom
5   where AttractionName <= &NameTo
6   print Attractions
7 endfor
```

Lembremos do Layout. No Source, programamos a consulta à base de dados com o for each, filtrando por nome.

Mas por que definir essas consultas através de listas pdf e não diretamente na própria tela onde pedimos ao usuário os dados dos filtros?

Modifiquemos o painel web para que possa consultar o BD



Por que não colocar aqui, em vez dos botões, uma grade que mostre as atrações desejadas?

As linhas da grade serão semelhantes ao printblock que imprime cada atração.

Web panels: consultas interativas à base de dados

Atributos no web panel são de saída: readonly



Os web panels, além de nos permitir definir variáveis a serem usadas em ações programadas em botões, nos permitem -e é seu objetivo fundamental- implementar consultas **interativas** à base de dados.

O termo "**interativas**" refere-se ao fato de que o usuário pode inserir na página do web panel uma e outra vez valores diferentes –**em variáveis**– e depois consultar dados da base de dados que correspondem aos valores inseridos, usando-os como filtros, como veremos agora.

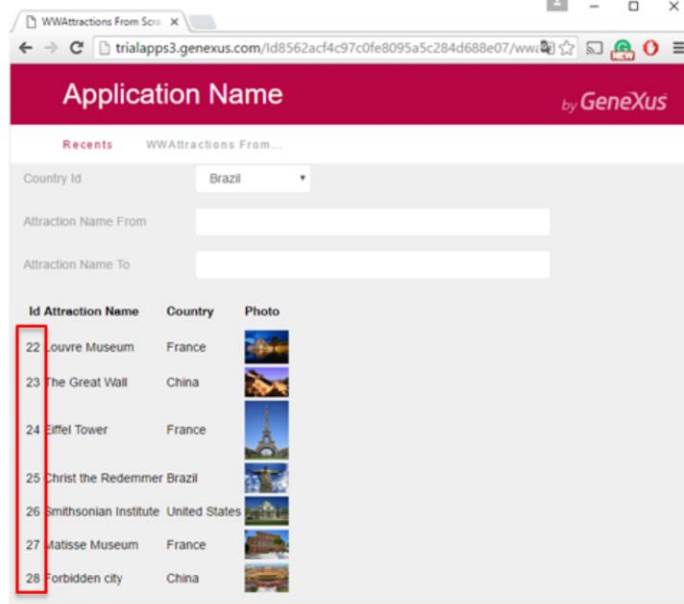
Vamos gravar este web panel com outro nome. Como o que vamos implementar será semelhante a um work with, vamos chamá-lo de WWAttractionsFromScratch.

Removemos os botões, que já não serão mais necessários, bem como os eventos associados. Agora, inserimos abaixo das variáveis um controle do tipo grade (**grid** em inglês).

Uma tela é aberta para escolher os atributos e/ou variáveis que serão as colunas desta grid. Como o que queremos mostrar é o mesmo que mostrávamos nas listas pdf, escolhemos os atributos AttractionId, AttractionName, AttractionPhoto e CountryName. Podemos alterar os títulos de cada coluna, editando as propriedades de cada um dos atributos que compõem essas colunas da grid.

Os **atributos** no form de um web panel são, por padrão, **de saída**. Ou seja, são **readonly**. Isso significa que GeneXus interpreta que deve ir à base de dados para buscar seu valor para mostrar ao usuário.

Web panels: consultas interativas à base de dados



Pressionemos F5 para executar o nosso novo web panel como o temos até agora.

Vemos que foram impressas todas as atrações, com os dados que indicamos (o id, nome, país e foto). Também vemos que saíram ordenadas por AttractionId.

Grid com atributos, equivale a um for each

The screenshot illustrates the GeneXus IDE interface. On the left, a web form is shown with a grid named 'Grid1'. The grid has columns for 'Id', 'Attraction Name', 'Country', and 'Photo'. The 'Country' column is linked to a dropdown menu labeled '&CountryId'. The 'Attraction Name' column is linked to a text box labeled '&AttractionNameFrom'. The 'Photo' column is linked to a text box labeled '&AttractionNameTo'. In the center, a code window shows a 'for each' loop over 'Attraction' with the following code:

```

1 print Title
2 print ColumnTitles
3 For each Attraction endfor
4   print Attractions
5   print Attractions
6   print Attractions
7 endfor

```

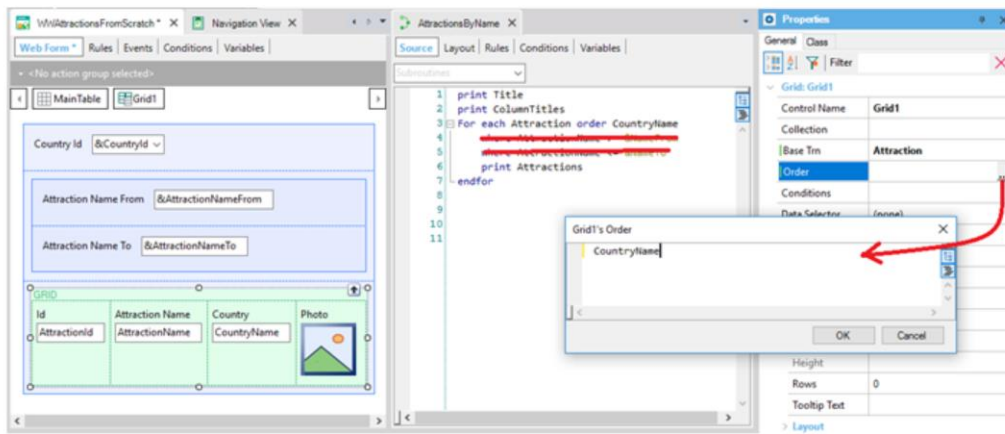
A red box highlights the 'for each' loop, and a red arrow points from it to the 'Grid1' control in the web form. On the right, the 'Properties' window for 'Grid1' is shown. The 'Collection' property is set to 'Attraction'. The 'Base Trn' property is highlighted with a red box, and the 'Attraction' table is selected in the dropdown menu.

Transação base

Apenas colocando uma grid com esses atributos, GeneXus entendeu que devia ir à base de dados para navegar na tabela Attraction, acessando Country para trazer o país da atração, assim como fazíamos com o comando for each (por enquanto sem as cláusulas order e where).

Se observamos as propriedades da grid, vemos que de fato possui uma de nome **Base Trn**, que é análoga à transação base do for each. Na verdade, para nos certificarmos de que para a grid se escolha a tabela base Attraction, que é o que queremos, é uma boa prática indicar a transação base, assim como para o for each.

Grid com atributos, equivale a um for each



Order

Por outro lado, note que temos uma propriedade **Order** para a grid. Esta propriedade corresponde à cláusula **Order** do **for each**.

Assim, se quiséssemos ordenar pelo nome do país, como na lista, na propriedade **Order** nós colocaríamos o atributo **CountryName**.

Grid com atributos, equivale a um foreach

Application Name

Recents WWAttractions From...

Country Id: Brazil

Attraction Name From:

Attraction Name To:

Id	Attraction Name	Country	Photo
25	Christ the Redeemer	Brazil	
23	The Great Wall	China	
28	Forbidden city	China	
22	Louvre Museum	France	
27	Matisse Museum	France	
24	Eiffel Tower	France	
26	Smithsonian Institute	United States	

Web Panel WWAttractionsFromScratch Navigation Report

Name: WWAttractionsFromScratch
Description: WWAttractions From Scratch
Environment: Default (C#)
Spec: 15_0_1-106638
Form Class: HTML
Program Name: WWAttractionsFromScratch
Parameters:

Warnings:

spc0038 There is no index for order CountryName; poor performance may be noticed in grid 'Grid1'.

FILL &CountryId with CountryId, CountryName in

=Country (CountryId) INTO CountryId CountryName
Order CountryName

Event Load

Order: CountryName
! No index

Navigation Start: FirstRecord
filters: from: NotEndOfTable
Loop while: Server

Join location: =Attraction (AttractionId) → **Base table of the Grid**
=Country (CountryId)

Navegação para carregar o combo dinâmico

Pressionamos F5. E vemos que agora a grid é classificada por nome do país.

Vejamos a lista de navegação do web panel.

Aqui está indicando a navegação que precisa realizar para carregar o combo box da variável &CountryId, que por enquanto não estamos usando para nada.

E aqui está indicando a navegação que terá que executar para carregar (Load) a grid. Vemos que o que lista para esta carga é idêntico ao que lista para um for each. Vemos que escolheu a tabela Attraction, percorrendo-a por CountryName, o atributo da propriedade Order. Percorrerá toda a tabela. E para cada registro de Attraction a ser carregado, acessará a tabela Country para mostrar o CountryName da atração.

Filtrar os dados do grid

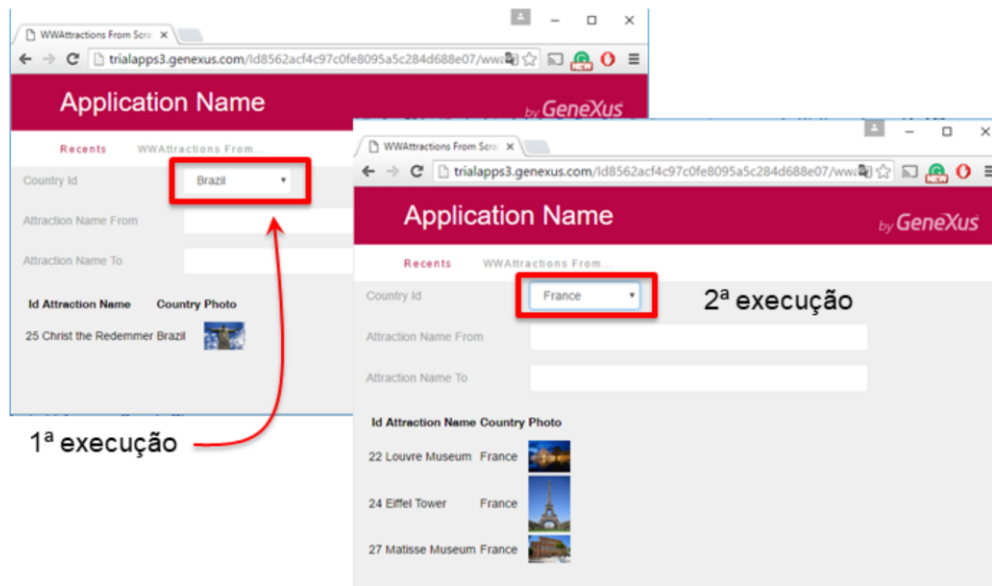
The screenshot shows the GeneXus IDE interface. On the left, a web panel is displayed with a form containing a dropdown for 'Country Id' and two text boxes for 'Attraction Name From' and 'Attraction Name To'. Below the form is a grid with columns for 'Id', 'Attraction Name', 'Country', and 'Photo'. A red arrow points from the 'Conditions' property of the grid in the Properties panel to the grid itself. The Properties panel on the right shows the 'Grid: Grid1' properties. The 'Conditions' property is highlighted with a red box and contains the text 'CountryId = &CountryId'. The 'Control Name' property is also highlighted with a red circle and contains the text 'Grid1'. The 'Base Trm' property is set to 'Attraction' and the 'Order' property is set to 'CountryName'. The 'Data Selector' property is set to '(none)'. The 'Appearance' section shows the 'Class' as 'Grid' and 'Auto Resize' as 'True'. The 'Rows' property is set to '0'. The 'Layout' section is partially visible at the bottom.

Condições de filtro

F5

Até agora, não fizemos nada com as variáveis. Mas queríamos usá-las para filtrar os dados mostrados na grid, tanto por país quanto por nome da atração. Em `AttractionsList` filtrávamos por país. Como indicamos esse filtro para a grid? Através da propriedade **Conditions**.

Filtrar os dados do grid



Observemos que, por padrão, o combo assume o valor Brasil, e na grid só vemos a atração do Brasil.

Se escolhemos França, vemos que a tela se atualiza, recarregando a grid, agora com as atrações da França.

Filtrar os dados do grid com condições

The screenshot shows the GeneXus IDE interface. On the left, a web form is being designed with a filter section containing a 'CountryId' dropdown and a grid section. The grid has columns for 'Id', 'Attraction Name', 'Country', and 'Photo'. In the center, a code window shows a script for filtering attractions by country. On the right, the 'Properties' window is open for a 'Dynamic Combo Box' control. A red arrow points from the 'CountryId' dropdown in the web form to the 'Empty Item' property in the 'Dynamic Combo Box' settings, which is set to 'True'. Below the arrow, the text 'Valor: "(None)"' is displayed.

Properties window - Dynamic Combo Box:

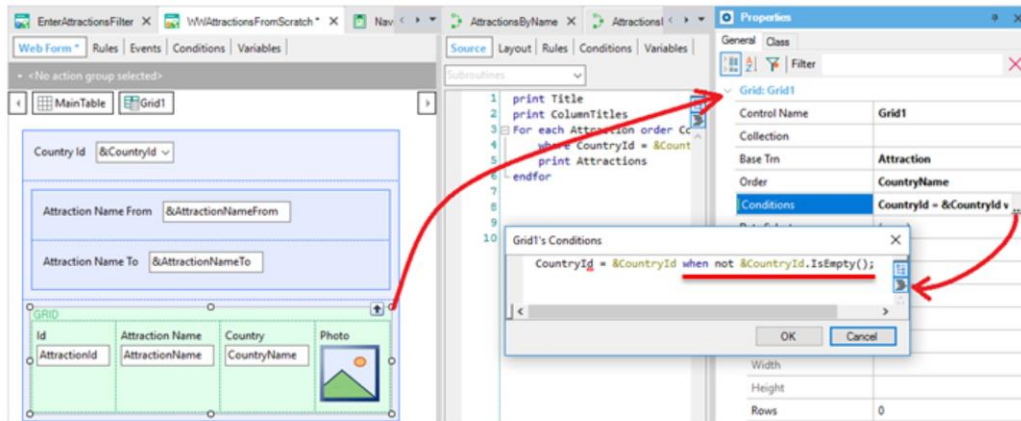
Property	Value
Label Caption	Country Id
ReadOnly	False
Return On Click	False
Class	Attribute
Invite Message	
Auto Resize	True
Width	4chr
Tooltip Text	
Control Type	Dynamic Combo Box
Data Source From	Attributes
Item Values	CountryId
Item Descriptions	CountryName
Sort Descriptions	True
Conditions	
Empty Item	True
Empty Item Text	GX_EmptyItemText

Valor: "(None)"

Provavelmente queremos que o combo apareça sem nenhum valor escolhido na primeira vez e, que nesse caso, sejam exibidas as atrações de todos os países.

Para fazer isso, editemos as propriedades do dynamic combo box... e passemos a **True** a de nome Empty Item. Isto fará com que se adicione uma opção "(None)" ao combo. Corresponderá ao valor empty, isto é, vazio, zero.

Filtrar os dados do grid com condições

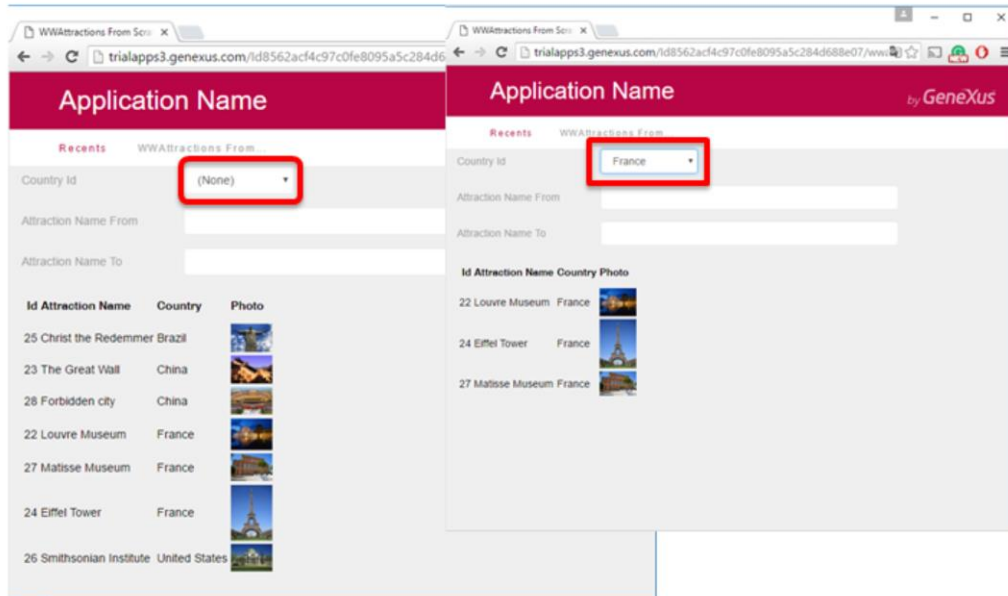


Condições aplicadas



Em seguida, vamos à propriedade **Conditions** e especificamos que queremos que essa condição seja aplicada somente quando o combo não possui o valor vazio. Quando for, que não se aplique.

Filtrar os dados do grid com condições

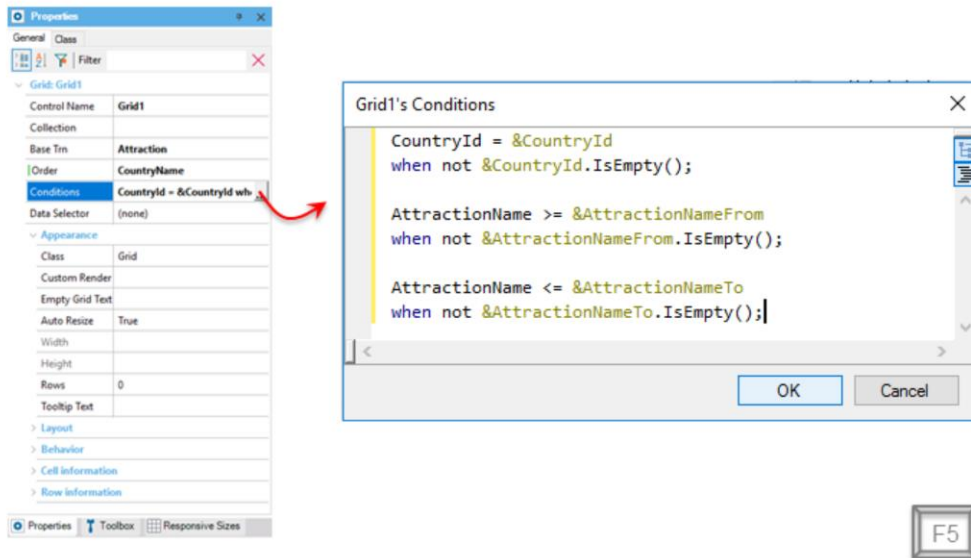


Vamos executar ...

Vemos como aparece o valor (None) no combo, e também como neste caso não estão sendo filtradas as atrações. Todas são mostradas

Se agora escolhermos, por exemplo, França, uma vez que a variável não possui o valor vazio, sim o filtro é aplicado, e são mostradas as atrações da França.

Filtrar os dados do grid com condições

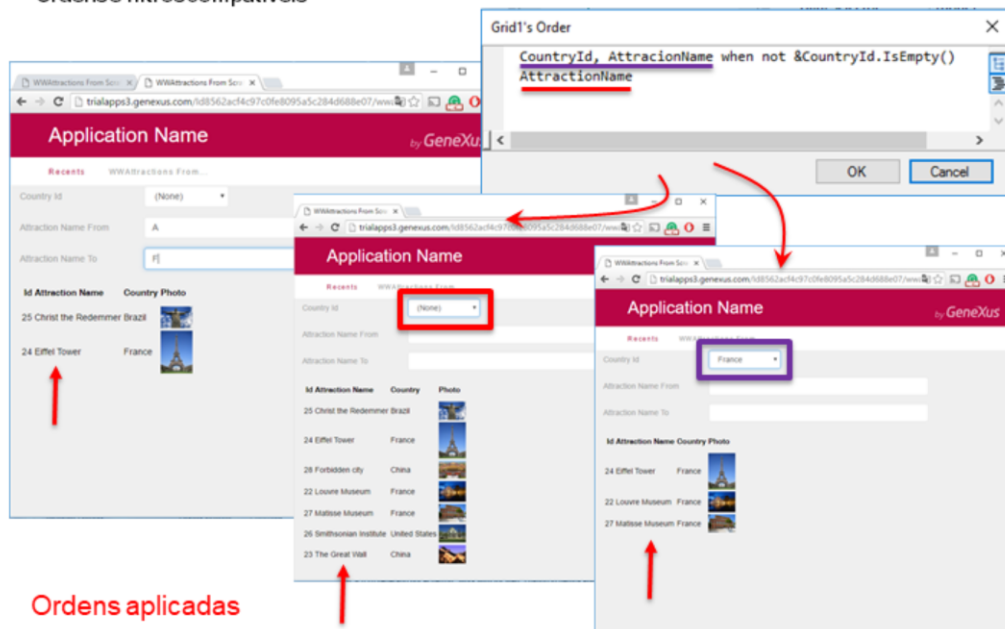


Também teríamos que adicionar os filtros pelo nome da atração, que queremos que se somem ao outro filtro. Então... se na lista filtrávamos no for each com estas duas cláusulas where... na grid vamos adicioná-las como condições.

AttractionName maior ou igual ao valor da variável &AttractionNameFrom do form, que o usuário irá inserir, se desejado. Novamente, se o usuário não inserir valor na variável, não queremos que este filtro se aplique. Então usamos a cláusula when. Esta cláusula também pode ser usada na cláusula where do for each, de maneira completamente análoga.

E adicionamos o outro filtro.

Ordense filtros compatíveis



Executemos novamente. E escolhemos ver as atrações entre A e F.

Podemos pedir ao web panel que, se o usuário escolher um país, então ordene a informação por CountryId e dentro do CountryId por AttractionName e, se não, a ordene por AttractionName. Isto pensando em otimizar a busca dos registros da tabela.

Para fazer isso, editamos a propriedade Order da grid e escrevemos primeiro a ordem, condicionando a que o usuário tenha escolhido um país no combo. Nesse caso, será filtrado por esse país, mas, além disso, as atrações serão listadas alfabeticamente para esse país. E se o usuário deixou o combo com o valor "(none)", isto é, vazio, então será escolhida a próxima ordem, que é por AttractionName.

Não nos prenderemos nisto aqui. O deixamos apontado apenas para mostrar que também é possível condicionar a forma como se quer ordenar a informação. Isso é idêntico em um for each.

Executemos... Aqui ordenou por AttractionName. E se escolhermos França, ordenará pelo Id da França e dentro dele por AttractionName.

Em suma, sempre vemos as atrações ordenadas alfabeticamente.

Se dentro das atrações da França, queremos aquelas que estão entre A e F, veremos que a grid se carregará filtrando pelas três condições que tínhamos escrito.

Resumindo

The screenshot shows a web panel design on the left and the Properties window for 'Grid1' on the right. The web panel has a form with fields for 'Country Id', 'Attraction Name From', and 'Attraction Name To'. Below the form is a grid with columns: 'Id', 'Attraction Name', 'Country', and 'Photo'. Red arrows point from the grid columns to a database icon labeled 'Tabela estendida' and from the 'Base Trn' property to a label 'Tabela base'. The Properties window for 'Grid1' shows the following properties:

Property	Value
Control Name	Grid1
Collection	
Base Trn	Attraction
Order	CountryId, Attraction...
Conditions	CountryId = &Country...
Data Selector	(none)

Annotations on the right side of the Properties window:

- Red arrow pointing to 'Base Trn': Tabela base
- Red text: ≈ For each Order Where

Annotation below the grid:

- Red arrow pointing to the database icon: ≈ For each Tabela estendida

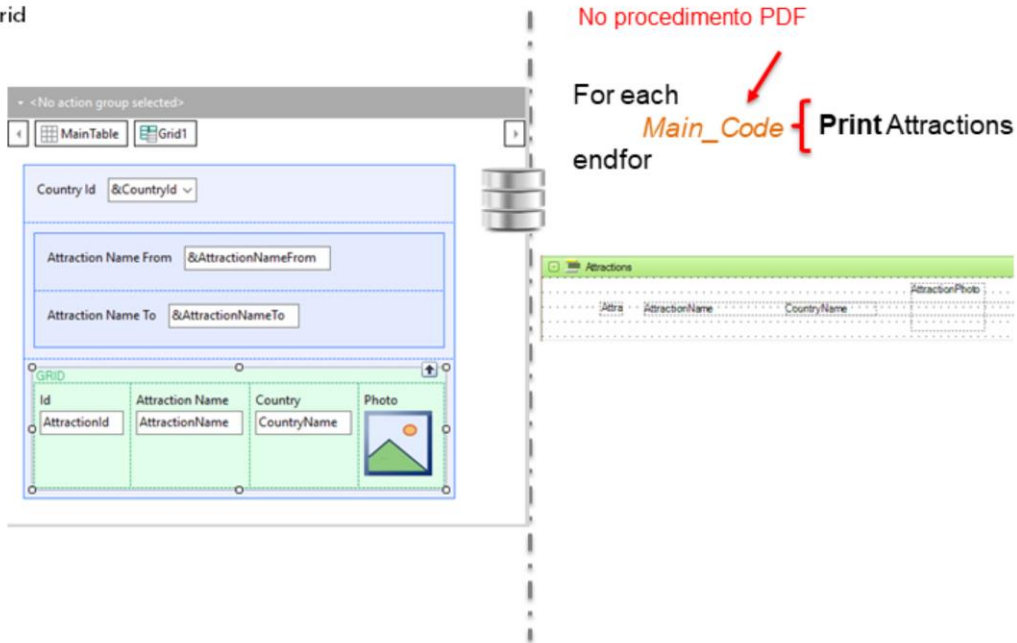
Implementamos um web panel no qual incluímos algumas variáveis para que o usuário insira o valor, e no qual inserimos um controle Grid com atributos.

Os atributos correspondem a informações da base de dados, razão pela qual GeneXus entende que deve ir buscar essa informação. Uma grid com atributos é como um for each.. É por isso que temos a propriedade **Base Trn**, como a do for each, para especificar o nível da transação cuja tabela associada queremos percorrer. Chamamos essa tabela, **tabela base da grid**. Se não a especificarmos, como também acontece com um for each, GeneXus a infere com base nos atributos que são usados. Embora este caso não veremos.

Todos os atributos do grid deverão pertencer, como no caso de um for each, à tabela estendida dessa tabela base.

Como em um for each, ordenamos a informação com a cláusula Order e filtramos os dados a serem devolvidos pela consulta com uma ou várias cláusulas where, para fazer o mesmo com a grid temos as propriedades Order e Conditions, respectivamente.

Carga do grid



Agora bem, em um for each programamos o que queremos que se faça com cada registro que cumpra as condições, dentro de seu corpo.

Por exemplo, na lista de atrações turísticas, o comando comando print Attraction enviava para imprimir na saída o que em nosso caso seria a linha da grid. No caso da grid não precisa especificá-lo. Isso é feito automaticamente.

Carga do grid: evento Load

<No action group selected>

MainTable Grd1

Country Id &CountryId

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

Id	Attraction Name	Country	Photo
AttractionId	AttractionName	CountryName	

No procedimento PDF

For each
Main_Code
endfor

&trips = Count(TripDate)
Print Attractions

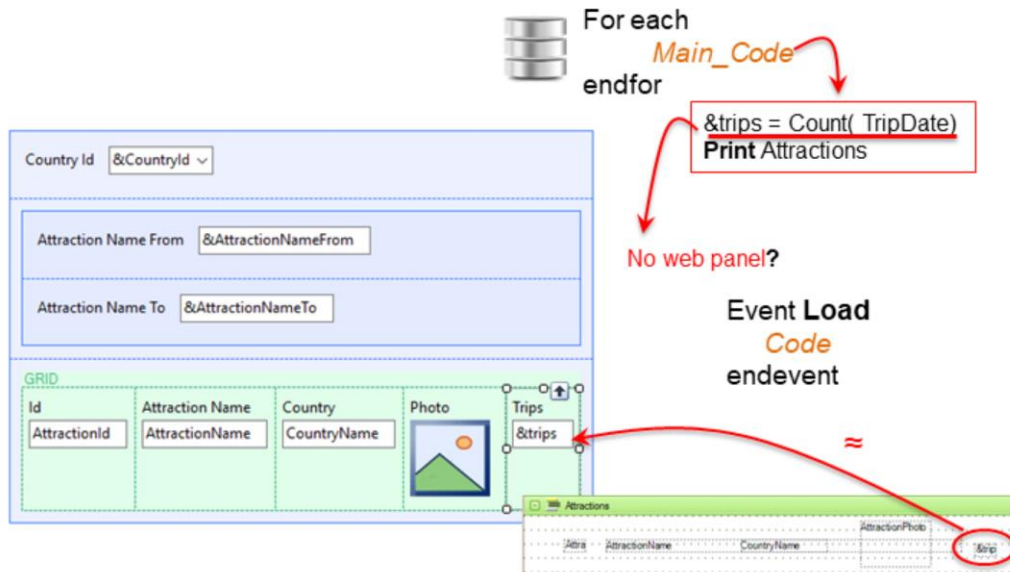
Attractions			
Attr	AttractionName	CountryName	AttractionPhoto
			&trips



Mas, por exemplo, vamos imaginar que temos uma transação Trip que registra as excursões oferecidas pela agência de viagens. De uma forma muito simplificada, imaginemos que de uma excursão só se registra a data na qual se realizará e a descrição e, em seguida, se registram as atrações turísticas que serão visitadas por essa excursão. Ok, então agora suponhamos que na lista de atrações turísticas queremos ver também a quantidade de trips que tem associadas cada atração. Para fazer isso, definíamos uma variável &trips, numérica, e a atribuímos dentro do corpo do for each (ou seja, quando o for each está posicionado no registro de sua tabela base que está por processar) o resultado da contagem das trips dessa atração. E colocar essa variável no print block.

Carga do grid: evento Load

No procedimento PDF



Para fazer o mesmo no web panel clicamos o botão direito sobre a grid e Insert Attribute/Variable. Criamos a variável &trips Numeric(4.0), e a movemos para ocupar a posição que nos interessa dentro da grid. Isso corresponde a ter inserido a variável no printblock. Mas onde vamos dizer-lhe como calculá-la? No for each é dentro de seu corpo. Aqui?

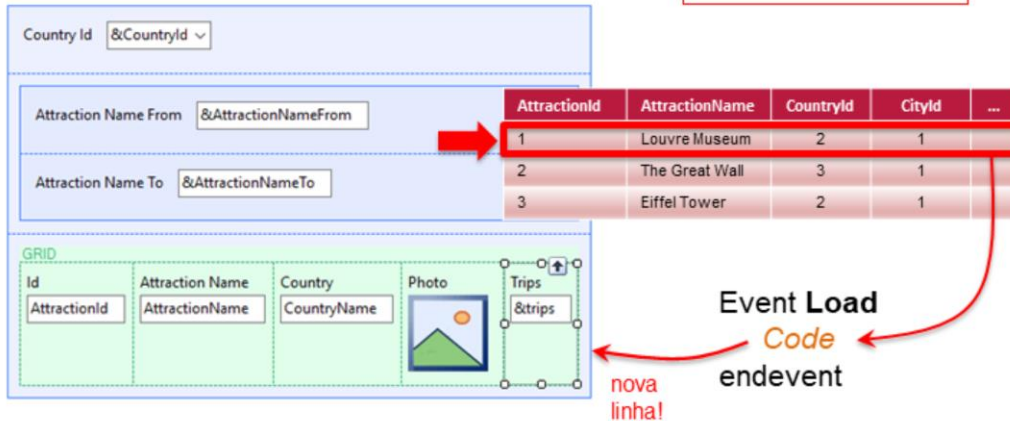
Contamos com o evento **Load** do sistema. Lá dentro vamos programar o que queremos executar quando se está posicionado em um registro da tabela base da grid, imediatamente antes de que a linha correspondente seja carregada na grid.

Carga do grid: evento Load



For each
Main_Code
endfor

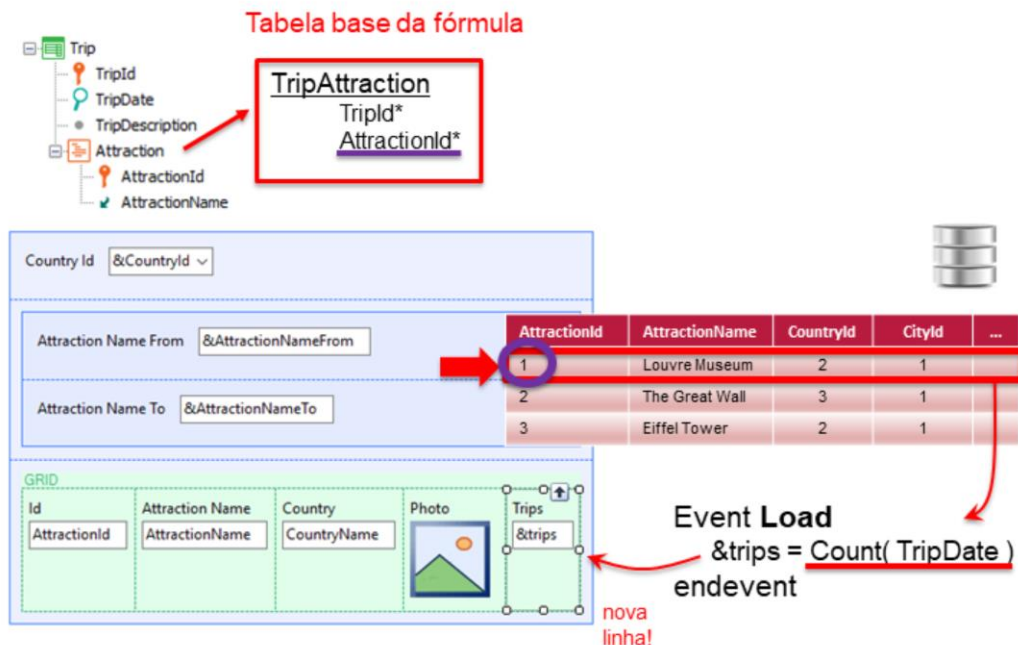
&trips = Count(TripDate)
Print Attractions



Lá dentro vamos programar o que queremos executar quando se está posicionado em um registro da tabela base da grid, imediatamente antes de que a linha correspondente seja carregada na grid.

No nosso caso, lá é onde atribuímos valor para a variável &Trips.

O evento Load será executado automaticamente para cada registro da tabela base da grid que cumpre com as condições de filtro, imediatamente antes da linha ser adicionada à grid.



É por essa razão que ao executar seu código se sabe que estamos trabalhando com um registro da tabela base e sua estendida, e esta fórmula inline não contará todas as trips, mas apenas aquelas da tabela TripAttraction que correspondam a esse AttractionId, a do registro de Attraction que estamos prestes a carregar na grid.

Observe-se que, embora o atributo TripDate seja da tabela Trip, GeneXus não escolherá a tabela Trip como tabela base da fórmula, mas a tabela TripAttraction. Nós não nos aprofundaremos nisso aqui, mas TripDate está na estendida de TripAttraction, tabela que tem relação com a de Attraction. GeneXus buscou uma tabela base que permitiu relacionar os dados.

DEMO



WWAttractions From Sc...

trialapps3.genexus.com/id8562acf4c97c0fe8095a5c284d688e07/www...

Application Name

by GeneXus

Recents Trip — WWAttractions From...

Country Id (None)

Attraction Name From

Attraction Name To

Id	Attraction Name	Country	Photo	Trips
25	Christ the Redemmer	Brazil		2
24	Eiffel Tower	France		2
28	Forbidden city	China		0
22	Louvre Museum	France		0
27	Matisse Museum	France		1
26	Smithsonian Institute	United States		1
23	The Great Wall	China		0

Variável fora do grid para totalizar?

Variável fora do grid
para totalizar?



Total Trips: 6

Implementemos em GeneXus. Já temos a transação Trip criada. Vamos para a seção de eventos do web panel. Neste combo, nos são oferecidos os eventos predefinidos, ou seja, eventos do sistema que ocorrem em momentos específicos e para os quais podemos programar o código.

Entre eles, vemos o evento Load. Aqui, dentro, programaremos o que queremos que se execute cada vez que se esteja posicionado em um registro da tabela Attraction, antes de carregar a linha na grid.

Vamos executar ...

Agora, queremos adicionar a soma das excursões nas quais as atrações mostradas na grid em cada oportunidade estejam incluídas. Ou seja, a soma dos valores desta coluna.

Variável fora do grid para totalizar

The screenshot illustrates a web panel layout for managing attractions. It includes a search section with filters for Country Id, Attraction Name From, and Attraction Name To. Below these is a grid displaying attraction data. A red box highlights the first row of the grid, and a red arrow points from this row to a code snippet. The code snippet shows an 'Event Load' event where the value of &trips is calculated as the count of TripDate, and then &totalTrips is updated by adding the value of &trips. Below the code, two variables are shown: &trips with a value of 2 and &totalTrips with a value of 2.

AttractionId	AttractionName	CountryId	CityId	...
1	Louvre Museum	2	1	
2	The Great Wall	3	1	
3	Eiffel Tower	2	1	

```
Event Load
  &trips = count( TripDate )
  &totalTrips = &totalTrips + &trips
Endevent
```

&trips 2

&totalTrips 2

Uma maneira eficiente de calcular o valor que a variável deve exibir é ... cada vez que se vá carregar uma na grid, some o valor da variável &Trips dessa linha ao valor calculado até o momento em &totalTrips.

Ou seja, no evento Load, depois de calcular o valor de &trips, a &totalTrips atribuir o valor que contém até o momento mais o valor da variável &trips.

Variável fora do grid para totalizar

Country Id

Attraction Name From

Attraction Name To

AttractionId	AttractionName	CountryId	CityId	...
1	Louvre Museum	2	1	
2	The Great Wall	3	1	
3	Eiffel Tower	2	1	

GRID

Id	Attraction Name	Country	Photo	trips
AttractionId	AttractionName	CountryName		&trips

Total Trips

```
Event Load
&trips = count( TripDate )
&totalTrips = &totalTrips + &trips
Endevent
```

&trips 1

&totalTrips 3

Para a segunda linha a ser carregada, se calcula o valor de &trips e &totalTrips conterá o valor anterior, ao que será adicionado o da variável &trips para esta segunda linha, e assim por diante.

Após a última linha ser carregada, a variável &totalTrips terá o valor desejado.

Telas Interativas / Web Panels
GeneXus

Refresh

Por que é
mostrado 9 ao
invés de 3?

Executemos para testar.

Vemos duas coisas: primeiro, que está somando corretamente; segundo: que por tratar-se de uma variável, é de entrada, de modo que o usuário pode modificar o valor, o que não faz sentido. Então, antes de tudo, vamos configurá-la como Readonly.

Para isso, nos posicionamos sobre a variável no form e entre suas propriedades, modificamos a Readonly passando-a para True.

Também pode chamar nossa atenção que &trips também é uma variável, está mostrando-se como readonly, e ainda não fizemos nada para isso. As variáveis nas grids, quando nenhum evento foi programado no nível das linhas, nem são percorridas pelo código, serão readonly. Voltaremos a isto mais tarde.

Mas também, vejamos o que acontece se filtramos, por exemplo, por França.

Em vez de nos mostrar um total de 3, nos está mostrando 9, que é a soma do valor que nos mostrava antes, 6, mais os 3 que agora deveria estar mostrando. Por que isso aconteceu?

Ao modificar uma das variáveis de filtro, o web panel recarregou a grid. Ou seja, ele voltou a consultar a tabela Attraction da base de dados, e novamente executou o evento **Load** para cada registro que satisfazia os filtros. O problema é que a variável &totalTrips deveria ter sido reiniciada, isto é, retornado para zero, antes de iniciar a carga da grid.

Onde fazemos isso? No **evento Refresh**.

A ordem em que os eventos são escritos não tem a menor importância. Aqui, apenas se indica o código que será executado quando cada um deles for produzido.

Evento Refresh

- Evento do sistema: é disparado sempre que se carrega o web panel, antes de ir ao BD buscar informação para carregar o grid (imediatamente antes do evento **Load** ser executado a cada linha a ser carregada)

```

Event Load
  &trips = count( TripDate )
  &totalTrips = &totalTrips + &trips
~Endevent

Event Refresh
  &totalTrips = 0
~Endevent

```

The screenshot shows a web application titled "Application Name by GeneXus". It features a search interface with "Country Id" set to "(None)" and input fields for "Attraction Name From" and "Attraction Name To". Below the search fields is a table of attractions. A red bracket on the right side of the table indicates the sequence of events: "Start (only the first time)", "Refresh (once)", and "Load (7 times)". The "Total Trips" at the bottom of the table is circled in red and shows the value "6".

Id	Attraction Name	Country	Photo	Trips
25	Christ the Redemmer	Brazil		2
24	Eiffel Tower	France		2
28	Forbidden city	China		0
22	Louvre Museum	France		0
27	Matisse Museum	France		1
26	Smithsonian Institute	United States		1
23	The Great Wall	China		0
Total Trips				6

Pressionemos F5.

Podemos ver que agora o total de trips aparece Readonly. Ao executar este web panel pela primeira vez, três eventos foram disparados de forma consecutiva: o **evento Start**, que é executado somente quando o web panel é aberto, a primeira vez, o **evento Refresh**, que colocou a variável em zero, e o **evento Load**, tantas vezes quanto as linhas foram carregadas na grid. Nesse caso, foram 7.

Evento Refresh

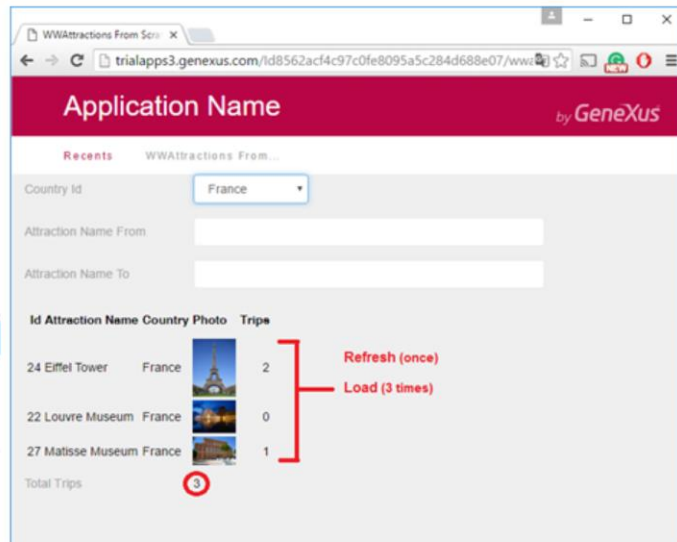
- Ao mudar o valor de uma variável referenciada nas Conditions é disparado novamente **Refresh** e **Load** (mas não **Start**)

```

Event Load
    &trips = count( TripDate )
    &totalTrips = &totalTrips + &trips
Endevent

Event Refresh
    &totalTrips = 0
Endevent

```



Agora, se optamos por filtrar por um país, por exemplo, França, vemos que se está calculando corretamente o número de trips.

Ao alterar o valor de uma variável que tem efeitos sobre as condições que os registros devem atender para serem carregados na grid, o evento **Refresh** é disparado novamente (e, portanto, a variável **&totalTrips** é reiniciada para zero) e a ida à base de dados para voltar a filtrar e carregar os registros na grid. Portanto, o **Load** é disparado novamente para cada atração da França a ser carregada.

Work With Attractionsdo Pattern

Ações sobre os dados: Inserir, Atualizar, Eliminar uma atração

Attraction

```

parm(in:&Mode, in:&AttractionId);
AttractionId = &AttractionId if not &AttractionId.IsEmpty();

```

Domínio TrnMode: Enum Values

Insert, Insert, INS	Update, Update, UPD	Delete, Delete, DLT	Display, Display, DSP
---------------------	---------------------	---------------------	-----------------------

Agora, se olharmos para o web panel que implementamos até agora, podemos apreciar que é semelhante ao objeto que criou o Pattern Work with aplicado à transação Attraction.

Este objeto era, evidentemente, um web panel.

O mais interessante é que além de permitir filtrar os dados da grid, o Work With oferece executar ações sobre os dados. Por exemplo, poder atualizar os dados de uma atração, ou eliminar a atração, bem como inserir uma nova.

Para isso, o pattern inseriu dois controles no nível das linhas da grid, e um fora. Em qualquer um dos três casos, a ação associada a cada controle consiste em chamar a transação Attraction, enviando-lhe como parâmetro o **modo** como a transação deve ser aberta, ou seja, se a chama para atualizar, excluir ou inserir. Nos dois primeiros, Update ou Delete, como corresponderão a eventos de uma linha, teremos o id da atração da linha, que será enviado como o segundo parâmetro à transação, de modo a atualizar os dados **dessa** atração, ou eliminar **essa** atração. No caso de Insert, controle fora da grid, será enviado 0 como o segundo parâmetro, uma vez que as atrações em Insert se autonumeram.

É por isso que o pattern modificou a transação Attraction, adicionando, entre outras coisas, a regra Parm que, como vemos, recebe duas variáveis: a variável &Mode é uma variável padrão em transações, Character de 3, que aceita um dos quatro valores, especificados no domínio enumerado TrnMode: Insert (INS), Update (UPD), Delete (DLT) e Display (DSP).

Recebendo um destes quatro valores, a transação saberá em qual modo é solicitado que se abra.

E, por outro lado, receberá como segundo parâmetro o id da atração, na variável &AttractionId, para quando quiser fazer update, delete ou display.

Ação de Update no nível das linhas do grid

Country Id

Attraction Name From

Attraction Name To

Id	Attraction Name	Country	Photo	trips	Update
AttractionId	AttractionName	CountryName		&trips	

Total Trips

Event Start
 &Update.FromImage(updateIcon)
 Endevent

Event &Update.Click
 Attraction(TrnMode.Update, AttractionId)
 Endevent

updateIcon X

Images

New Image

Image

updateIcon24.png
 (Default)
 Size:24x24 px

Inserimos imagem na KB

&Update tipo Image

No nosso web panel, implementaremos uma destas ações sobre as atrações. Por exemplo, a de Update. A intenção é mostrar um exemplo de ações sobre os dados.

Teremos que inserir um controle na grid. No caso do pattern, é inserida uma variável character, que foi chamada update e à qual se atribui o texto "UPDATE" que vemos em execução. Mas nós vamos optar por inserir uma imagem, que devemos primeiro inserir na KB.

A chamamos de updateIcon.

Agora vamos ao web panel e arrastamos da toolbox o controle Attribute/Variable para a última coluna da grid, definimos a nova variável como &Update, mas não como character, mas como Image.

Removemos o título para que não apareça como um título de coluna, e nos resta carregar essa variável com a imagem que acabamos de inserir na KB. Onde fazemos isso?

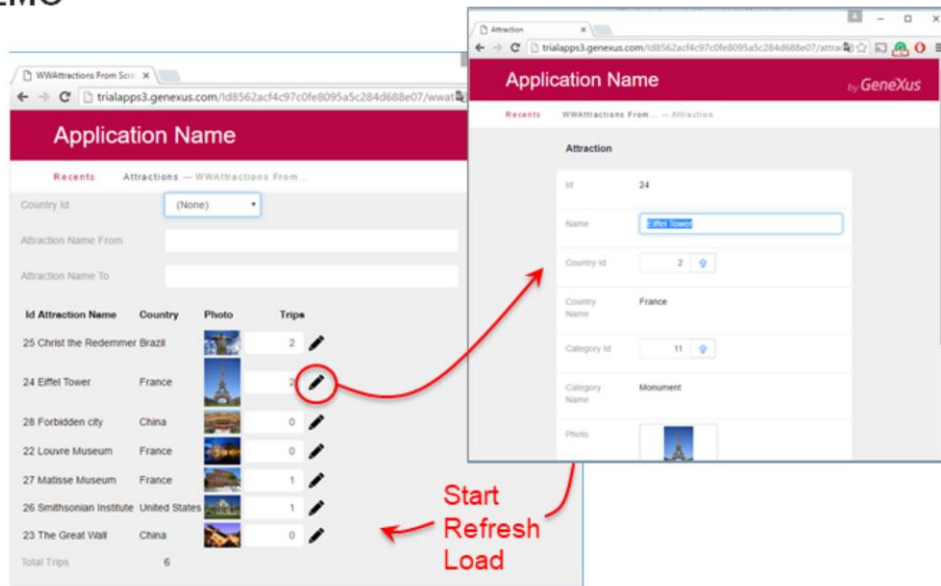
Se a imagem varia de acordo com a linha da grid, faríamos isso no evento Load, mas a imagem será a mesma para cada linha e ela nunca mudará, então uma boa opção é fazê-lo no **evento Start**, que será executado apenas uma vez, quando se abra o web panel, e não mais.

Agora, nos resta associar um evento a essa imagem, de modo que quando o usuário clique nela, se produza esse evento e se execute seu código, no qual invocaremos a transação Attraction.

Existem várias alternativas para fazer isto. Uma delas é usar o mesmo clique do controle variável &Update.

Desta forma, quando o usuário clicar na imagem para uma linha, se executará o código que escrevemos dentro deste evento. O que queremos fazer, nesse caso, é invocar a transação Attraction, passando-lhe o modo Update, ou seja, o valor Update do domínio enumerado TrnMode que vimos anteriormente, e o valor de AttractionId correspondente à linha da grid onde se fez o clique .

DEMO



Executemos para testar.

Primeiro notamos que a coluna da variável &Trips agora aparece como editável. Não tínhamos a definido como ReadOnly explicitamente porque ao executar vimos que já estava. Como dissemos antes, as variáveis de grid em princípio são colocadas como readOnly, a menos que se defina algum evento no nível das linhas, como foi nosso caso, ou outras exceções que não veremos agora. A configuremos como ReadOnly para a próxima execução.

Selecionemos, por exemplo, o país França. E agora vamos clicar na imagem de update para a Torre Eiffel. Vemos como a transação é chamada em update. Vamos modificar algo... por exemplo, passamos de maiúscula para minúscula a letra T de Tower.

Confirmamos... e como o pattern work with, embora não o vemos, adicionou à transação um comando Return, retorno, para voltar a quem a chamou, é que se retorna para o web panel. Esse comando Return é como fazer uma invocação para o web panel pela primeira vez, portanto, neste se executará o evento Start, seguido pelo Refresh e do Load tantas vezes quanto registros serão carregados.

Quando não é possível não incluir coluna na grid?

The screenshot shows the GeneXus IDE interface. At the top, there are tabs for 'Start Page', 'WebForm', 'Rules', 'Events', 'Conditions', and 'Variables'. The 'WebForm' tab is active, showing a form with fields for 'Country Id', 'Attraction Name From', 'Attraction Name To', and a grid. The grid has columns for 'Id', 'Attraction Name', 'Country', 'Photo', 'Trips', and 'Trips'. A red arrow points from the 'Visible' property in the properties window to the 'AttractionId' column in the grid. A blue arrow points from the 'AttractionId' parameter in the event code to the 'Visible' property in the properties window.

Event &Update.Click
Attraction(TrnMode.Update, AttractionId)
Endevent

Tem que estar na grid!
Mas se não a queremos ver, podemos ocultá-la

Visible: False

O que aconteceria se não tivéssemos colocado o atributo `AttractionId` na grid? Ao clicar na imagem para atualizar, qual `AttractionId` teria sido enviado como parâmetro para a transação? Não teria esse valor para enviar.

Como vamos usá-lo em um evento no nível das linhas, que será disparado após as linhas terem sido carregadas, não podemos remover `AttractionId` da grid. É que aqui já não se está mais na base de dados. A grid armazenou, quando carregada com o `Load`, todos os valores de suas colunas e nada mais. Um evento posterior funcionará somente sobre os dados carregados na grid. Então, o que podemos fazer se não queremos ver essa coluna na grid, é ocultá-la. Ela permanecerá presente, mas invisível. Para isso, usamos a propriedade `Visible`.



Vídeos

training.genexus.com

Documentação

wiki.genexus.com

Certificações

training.genexus.com/certifications