

Objeto Web Panel. Carregando dados e eventos

GeneXus™















Web Panel com grid e tabela base

Application Name

Country Id: (None) ▾

Attraction Name From:

Attraction Name To:

Attraction Name	Country Name	Attraction Photo	Trips
Christ the Redeemer	Brazil		2 
Eiffel Tower	France		2 
Forbidden City	China		0 
Great Wall	China		0 
Louvre Museum	France		1 
Matisse Museum	France		2 
Smithsonian Institute	United States		1 

Total Trips: 8

```

1  Event Load
2    &trips = count(TripDate)
3    &totalTrips = &totalTrips + &trips
4  -Endevent
5
6
7  Event Refresh
8    &totalTrips = 0
9  -Endevent
10
11
12 Event Start
13   &update.FromImage(UpdateIcon)
14 -Endevent
15
16
17 Event &update.Click
18   Attraction(TrnMode.Update, AttractionId)
19 -Endevent
20

```

Estávamos construindo nossa web panel WWAttractionsFromScratch. Havíamos visto como condicionar os dados que eram exibidos no grid e como ordená-los. O grid tinha tabela base.

Havíamos visto nesse caso como carregar uma variável do grid para cada linha com o evento Load. Havíamos utilizado o evento Refresh, assim como o Start, e também havíamos programado um evento no nível das linhas, para chamar a transação Attraction no modo update.

Além disso, tínhamos visto que ao ter um grid podíamos utilizar o evento Load do grid em vez do genérico, para nos antecipar à possibilidade futura de ter que inserir outro grid. E também que tínhamos um Refresh do grid. Neste momento, para não confundir, seguiremos com os genéricos.

Modificar uma linha de um grid

```

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 |   |
21 | -Endevent

```

Agora, adicionemos outra ação ao nível das linhas, mas uma que não chame outro objeto com interface, como foi o caso da invocação da transação Attraction.

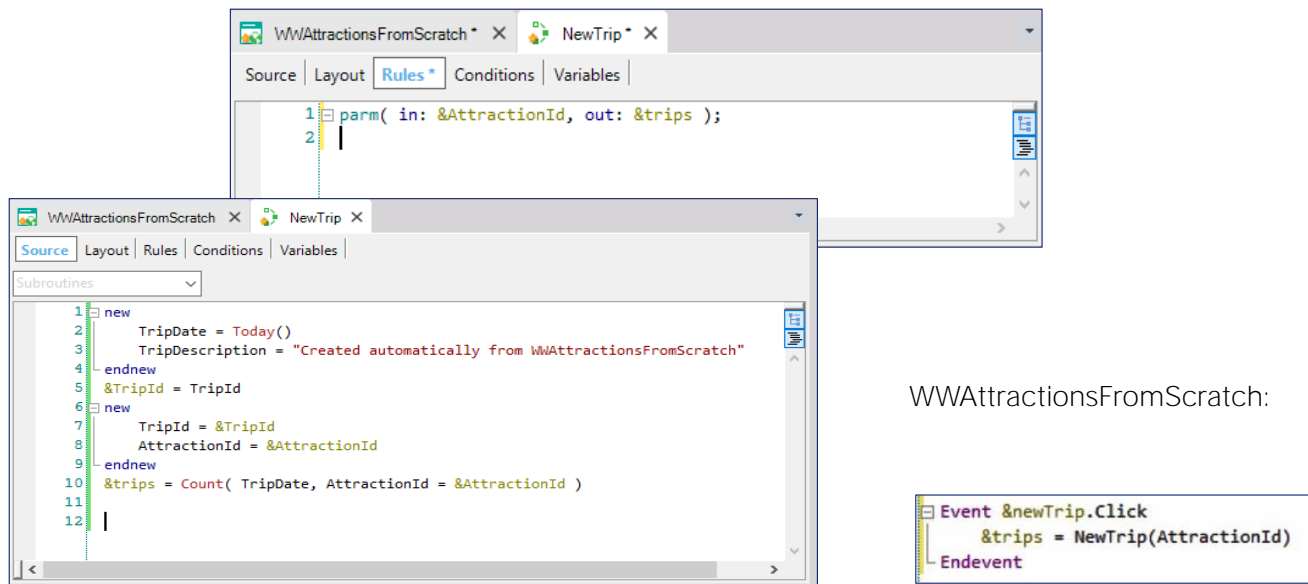
Por exemplo, imagine que queremos dar a possibilidade de que, a partir de uma linha (uma atração), se possa criar uma nova trip na base de dados, com essa atração.

Primeiramente adicionemos uma nova variável à grid, chamada newTrip, character de 10.

A mudamos para ReadOnly. Queremos que contenha o texto “**New Trip**”, por isso lhe atribuímos no evento Start, pois não variará de acordo com a linha. E programemos para essa variável o evento de click.

O que queremos fazer quando o usuário clicar sobre New Trip?

Uma nova excursão é adicionada a uma atração



Por exemplo, chamar um procedimento e passarmos o identificador da atração da linha e criar o trip com essa atração.

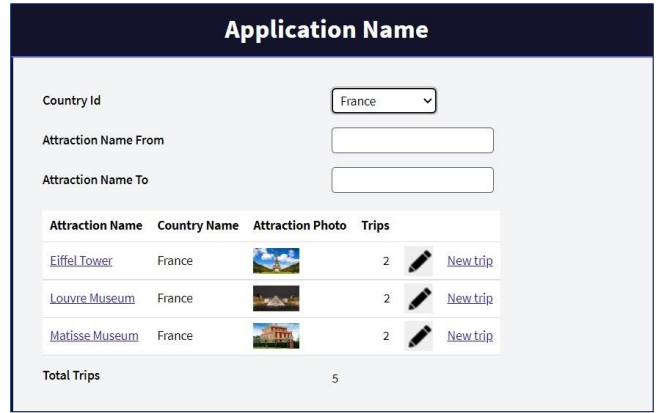
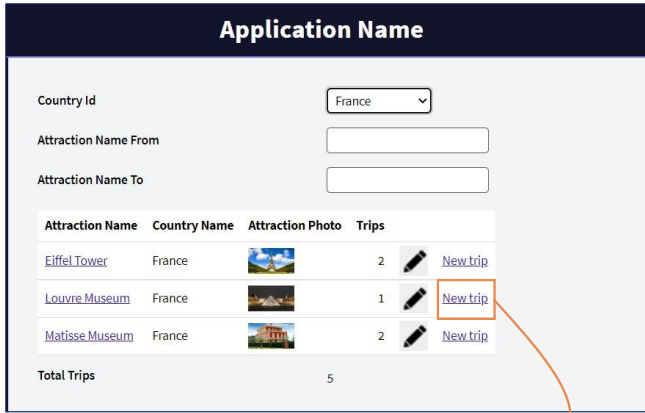
Observar que implementamos com o comando new para criar diretamente um registro na tabela Trip e um na tabela TripAttraction. Usamos essa solução para mostrar o uso desses comandos. Porém, a solução mais aconselhável seria utilizar o business component do Trip para inserir. Neste curso não foi ensinado como carregar um business component de dois níveis, mas é muito simples.

Vejam que logo depois de inserir o cabeçalho e a linha do Trip, calculamos a quantidade de trips nos quais esta atração pertence. Como a fórmula inline está sendo disparada sem estar posicionada em tabela alguma (está "solta" no código), temos que indicar a condição explícita de que queremos filtrar os trips da atração.

E esse valor é o valor devolvido a quem chama esse procedimento. Então, no evento click do controle (variável) &newTrip chamamos esse procedimento passando o AttractionId da linha que foi feito o clique, e como o parâmetro que devolve é o que necessitamos mostrar na coluna &Trips, atribuímo-os diretamente na variável.

Logicamente, quando o usuário clicar sobre New Trip deverá ver para essa linha, na coluna Trips, o valor que tinha antes do clique mais um, ou seja, deverá atualizar a linha.

Uma nova excursão é adicionada a uma atração



```
Event &newTrip.Click  
  &trips = NewTrip(AttractionId)  
Endevent
```

Apenas a linha foi atualizada. Mas o total não foi atualizado!

Executemos para testar.

Por exemplo, filtramos por França, e para a atração museu Louvre que até o momento não está em nenhum trip, clicamos sobre New Trip.

Visualizamos que automaticamente a linha foi atualizada:

Agora é mostrado a quantidade de trips do Louvre: 1.

Porém, o que não foi atualizado foi a conta do total, que deveria exibir 4, e, no entanto, exibe o valor anterior. Por que?

É que ao executar o evento click associado a variável &NewTrip, somente foi executado o seu código, e como dentro do mesmo atribuiu-se valor a uma variável do grid, a linha foi atualizada, apenas esta linha, sem executar nenhum outro evento. Nenhum, nem sequer o Load.

Comando Refresh

Alternativa 1

```

Event &newTrip.Click
  &trips = NewTrip( AttractionId )
  &totalTrips = &totalTrips + 1
Endevent

```

Alternativa 2

```

Event &newTrip.Click
  &trips = NewTrip( AttractionId )
  Refresh
Endevent

```

```

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

Event Refresh
  &totalTrips = 0
Endevent

```

Portanto, temos duas alternativas para manter atualizado o Total de Trips carregados no grid quando é produzido esse evento.

Uma possibilidade é a `&totalTrips` somar um, pois o procedimento somente adicionou um trip para essa atração.

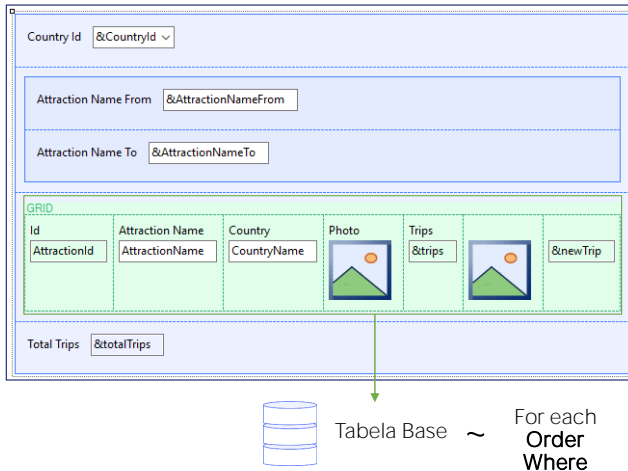
Mas se o procedimento for alterado e adicionado mais trips, teremos que lembrar de fazer a mudança de forma consistente nesse evento.

Uma melhor solução seria poder pedir ao web panel que volte a executar os eventos Refresh e Load. Para isso, contamos com o comando Refresh.

Ao fazer isso, a web panel voltará a ler base de dados para carregar o grid.

Resumo: Ordem de execução dos eventos

Web Panel com um grid (com atributos)



1ª vez:

Start

Refresh

Load (N vezes)

N vezes

User / Control Event

Executemos para testar.

Adicionemos um trip para o museu Matisse:

Podemos ver que agora tudo foi atualizado. É que voltou a ser executado os eventos Refresh e Load. Este último foi executado 7 vezes, uma vez para cada linha a ser carregada.

Façamos um repasse de tudo o que foi visto.

Para o caso de uma web panel com um grid com atributos, GeneXus entende que esse grid tem uma tabela base associada, isto é, uma tabela que deverá navegar para carregar as linhas do grid. Carregará uma linha para cada registro dessa tabela base. Para filtrar, temos a propriedade Conditions do grid, para ordenar, a propriedade Order. Um grid com tabela base é análogo a um for each.

Nas web panels são produzidos eventos do sistema que podemos programar código para que seja executado no momento em que for disparado. Aprendemos três desses eventos que são disparados sempre que é aberta uma web panel, isto é, na sua primeira execução:

O Start é disparado uma única vez. Nele podemos inicializar variáveis, por exemplo.

O Refresh é disparado antes carregar a informação da tela. Atrás deste evento será produzido o acesso a base de dados para trazer os dados da tabela base e

sua estendida.

Para cada registro da tabela base que está para ser carregado no grid, é produzido um evento Load. Portanto, aqui será o momento de programar todas as ações que desejamos que sejam executadas antes que a linha seja efetivamente carregada no grid. Os dados que são carregados no grid são exclusivamente os das colunas visíveis ou invisíveis que são incluídos nele.

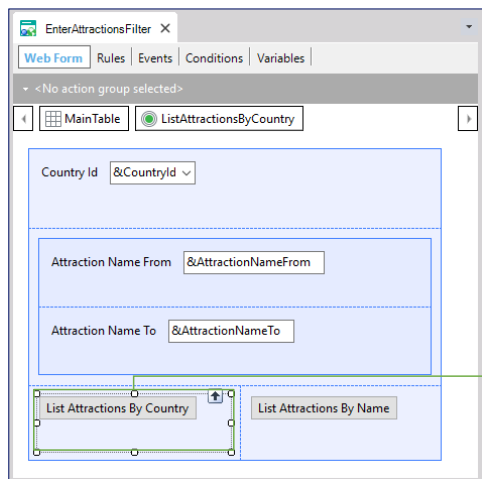
Depois disso, a web panel estará carregada com a informação que foi extraída da base de dados e se desconecta dela.

Agora é quando o usuário começa a interagir com a tela. Por exemplo, modifica o valor de uma das variáveis utilizadas para filtrar as informações do Grid. Automaticamente, é feito um Refresh pelo qual é novamente invocado o server para que execute desta vez o evento Refresh e o evento Load. No nosso caso, se o CountryId corresponde ao da França, então o Refresh será executado e em seguida o Load para cada atração da França. Observemos que a grande diferença com a primeira execução, é que aqui o Start não é executado novamente.

Mas os web panels permitem definir outros eventos, que também serão disparados após a primeira vez, ou seja, uma vez que o web panel já foi carregado, e sempre a partir da ação do usuário.

Por exemplo, o evento Click que programamos sobre esta imagem ou o Click sobre New Trip.

Resumo: Ordem de execução dos eventos



1ª vez:

Start

Refresh

Load (N vezes)

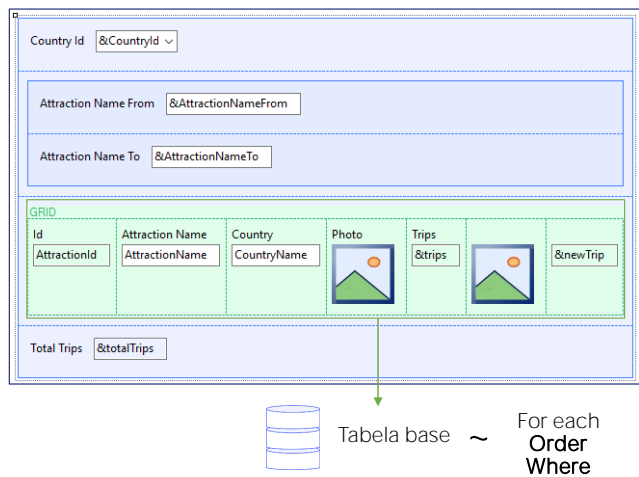
N vezes:

```
Event 'List Attractions By Country'  
  AttractionsList(&CountryId)  
  //AttractionsReport(&CountryId)  
Endevent
```

ou incluso na web panel que havíamos definido muitas aulas atrás, o evento de usuário que associamos ao botão, que colocamos esse nome.

Resumo: Ordem de execução dos eventos

Web Panel com um grid (com atributos)



1ª vez:

Start

Refresh

Load (N vezes)

N vezes:

User / Control Event

Comando Refresh

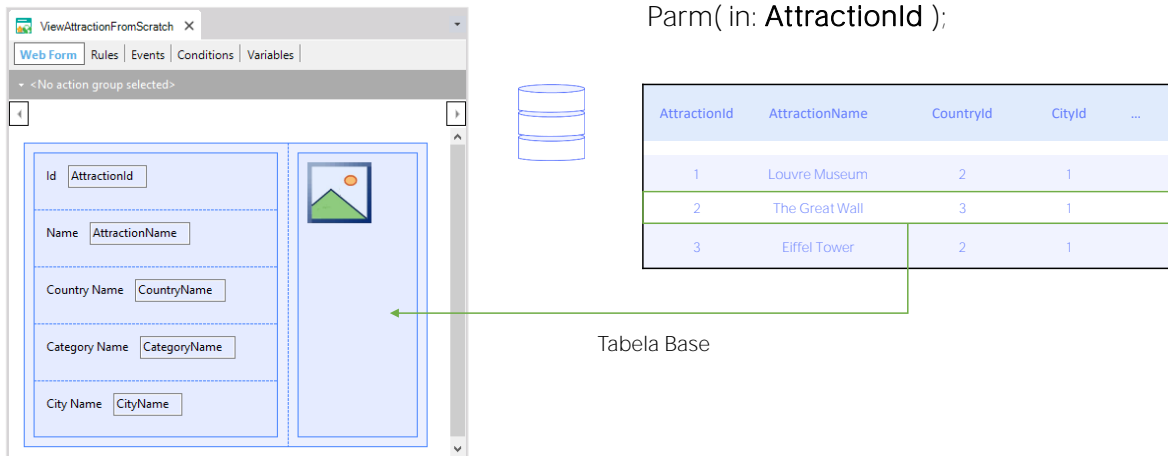
Esses eventos são conhecidos como eventos de usuário, ou eventos de controles (como o Click que vimos). Quando o usuário provoca um evento desses, apenas é executado o seu código, sem atualizar a tela. A única exceção é quando o evento é produzido no nível de uma linha do grid, como no caso que vimos de &newTrip, e dentro de seu código atribui-se valor a uma variável do mesmo grid, que em nosso caso era &Trips. Nesse caso, o valor da variável é atualizado na tela, para essa linha, para que seja exibida atualizada.

Se necessitamos que volte a ser executado o Refresh e volte a ser carregada as linhas no grid a partir da base de dados (por exemplo, para atualizar o total das linhas), então podemos escrever o comando Refresh no evento.

O comando Refresh provoca a execução dos eventos Refresh e Load.

Resumo e além Web Panel com tabela base e sem grid

Web Panel sem grid porém com atributos diretamente no form



Parm(in: **AttractionId**);

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

Tabela Base

Estudamos o caso de um web panel com uma grid com atributos. Mas, o que acontece se o que temos é um web panel sem grid, mas com atributos no form?

Suponhamos que queremos que, clicando sobre o nome da atração, no nosso web panel WWAttractionsFromScratch... seja chamado um web panel para mostrar todos os dados da atração.

Para isso, já implementamos este web panel... em cujo form inserimos os atributos de uma atração que nos interessa mostrar. Além disso, escrevemos uma regra parm, para receber um parâmetro. Vejamos que em vez de receber em uma variável, escolhemos receber no atributo AttractionId.

O que queremos é que, quando a partir do evento click de AttractionName em nosso outro web panel... Se chame a este Web panel, lhe passe o id da atração da linha da grid, automaticamente GeneXus vá à tabela de atrações, encontre a atração com esse id e para essa atração, mostre a informação dos atributos que foram colocados no form.

Em suma, um web panel que não possui nenhuma grid, mas que possui atributos no form, também terá uma tabela base. Como GeneXus a determina, já que agora não temos uma transação base para indicar? Não veremos neste curso, mas é análogo ao caso de um for each em que não

se indica a Transação Base.

Mas neste caso, uma vez que não temos uma grid, teremos que carregar unicamente UM registro dessa tabela base e sua estendida. Onde indicamos o filtro que permita devolver esse registro?

Em nosso caso, foi recebendo no atributo AttractionId. Lá estamos especificando o filtro automático, para que saiba com qual de todos os registros da tabela base deve permanecer.

Resumo e além Web Panel com tabela base e sem grid

Web Panel sem grid porém com atributos diretamente no form

ViewAttractionFromScratch X
Web Form | Rules | Events | Conditions | Variables
<No action group selected>

Id

Name

Country Name

Category Name

City Name



Tabela Base

Parm(in: &AttractionId);


ViewAttractionFromScratch* X
Web Form | Rules | Events | Conditions* | Variables*
1 AttractionId = &AttractionId;
2

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

Se precisarmos estabelecer outro tipo de condição, ou se, por exemplo, tivéssemos recebido o parâmetro em variável, no lugar de atributo, teríamos que ir na aba de Conditions para estabelecer o filtro.

Evento Load no Web Panel com tabela base e sem grid

Application Name

Attraction Name	Eiffel Tower	
Country Name	France	
Category Name	Monument	
City Name	Paris	

```

1 | Event Load
2 |   &trips = Count(TripDate)
3 | Endevent
  
```

Event Load

Order: **AttractionId**
 Index: IATTRACTION

Navigation filters: Start from: **AttractionId = @AttractionId**
 Loop while: **AttractionId = @AttractionId**
 Join location: Server

Attraction (AttractionId)
Country (CountryId)
CountryCity (CountryId, CityId)
Category (CategoryId)
count(TripDate).navigation
TriAttraction (AttractionId)
Trip (TripId)

Web Form

Actions: Update Delete

MainTable AttributesTable trips

Name:

Country Name:

Category Name:

City Name:

Trips:

Se quiséssemos também mostrar aqui a quantidade de trips nas quais se encontra a atração, necessitamos da variável &trips mas, desta vez, onde a carregamos? No evento Load! Também! Testemos.

Vale o mesmo que se houvesse um grid. Vejamos o que informa a lista de navegação. O evento Load tem associada a tabela Attraction, que é a tabela base. Mas em vez de percorrê-la toda, somente fica com este registro.

Evento Load no Web Panel com tabela base e sem grid

```
1 parm(in:&AttractionId);
```

Event Load

Order: AttractionId
Index: IATTRACTION

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

- [-] Attraction (*AttractionId*)
 - [-] Country (*CountryId*)
 - [-] CountryCity (*CountryId*, *CityId*)
 - [-] Category (*CategoryId*)
 - [-] count (*TripDate*) navigation
 - [-] TripAttraction (*AttractionId*)
 - [-] Trip (*TripId*)

Name:

Country Id:

Country Name:

Category Name:

City Name:

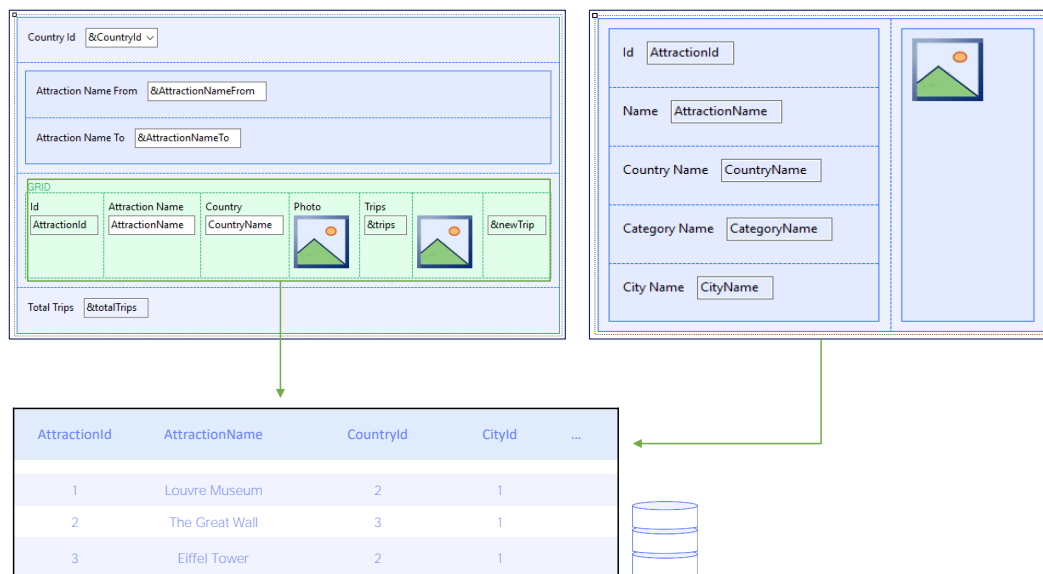
Trips:

Por exemplo, o que teria acontecido se não tivéssemos recebido no atributo, mas em variável, mas tivéssemos omitido especificar o filtro nas conditions? Os valores de qual atração carregará na tela? Vejamos a lista de navegação antes de executar. Vai percorrer igualmente toda a tabela base e a fazer um Load para cada registro, mas como não há um grid, estará sobrescrevendo para cada registro em que se posicione, os atributos apresentados na tela, e o que vamos ver finalmente, será o último registro da leitura, que como é realizada por chave primária, corresponde à última atração adicionada. Vejamos.

Forbidden city era a última atração adicionada.

Se agora colocamos a condição de filtro, e vemos a lista de navegação vemos que passará outra vez a filtrar ficando com uma única atração. Executemos.

Web panels com tabela base



Temos visto até aqui duas web panels que tem tabela base

- a primeira com um grid, onde a tabela base do grid é a tabela base da web panel, ou seja, a tabela na qual a web panel automaticamente escolhe para navegar e trazer a informação a ser carregada na tela.
- o segundo sem grid, mas com atributos, onde a tabela base da web panel é encontrada a partir desses atributos.




Mas temos também o caso misto, que é quando temos um web panel com grid com atributos, mas onde também há atributos soltos no que chamamos parte fixa do panel, ou seja, fora de todo grid.

Web panels com tabela base

Country Name

Attraction Name From

Attraction Name To

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

Total Trips

Event Load

Order: CountryId, AttractionName
No index

Navigation filters: Start from: CountryId = @CountryId
Loop while: CountryId = @CountryId

Constraints: AttractionName >= &AttractionNameFrom WHEN not &AttractionNameFrom.isempty()
AttractionName <= &AttractionNameTo WHEN not &AttractionNameTo.isempty()

Join location: Server

- Attraction (AttractionId)
 - Country (CountryId)
 - count (TripDate) .navigation
 - TripAttraction (AttractionId)
 - Trip (TripId)

WWAttractionsFromScratch X ViewAttractionFromScratch X

Web Form | Rules | **Events** | Conditions | Variables

CountryName.Click

```

1 Event Load
2   &trips = Count(TripDate)
3 -Endevent
4
5
6 Event CountryName.Click
7   ViewCountryInfo(CountryId)
8 -Endevent
9
10
11 Event Start
12   CountryId.Visible = False
13 -Endevent
  
```

ViewCountryInfo X

Web Form | **Rules** | Events | Conditions | Variables

```

1 parm( in: CountryId );
2
  
```

Por exemplo, suponhamos que ao clicar no nome do país da atração turística queremos visualizar a informação geral do país, e todas as suas atrações turísticas. É parecido com o panel que fizemos, mas onde aqui não estamos permitindo filtrar por país, apenas por nome de atração.

Se observarmos como o implementamos, a partir do panel do View da atração programamos o evento click sobre o nome de país (observe que tivemos que colocar o atributo CountryId invisível para poder fazer esta invocação. Por quê?) e ali chamamos este novo Web Panel, que criamos salvando o que tínhamos com este outro nome. Como vemos, este novo web panel é muito parecido, exceto que recebe um parâmetro, o identificador de país. Em vez de ter a variável CountryId, tem o atributo CountryName. A ordem do grid, que no primeiro era condicional, aqui é fixa por CountryId, AttractionName porque o país virá instanciado, por parâmetro. E as condições para carregar o grid, que antes incluíam uma para filtrar por país, já não o fazem. É que ao recebê-lo no atributo, atuará como um filtro automático.

Se observamos a lista de navegação, vemos que segue indicando que o evento Load, para carregar a informação a ser mostrada na tela, segue percorrendo a mesma tabela Attraction.

É que pudemos tirar o atributo do grid e colocá-lo na parte fixa, porque o país sempre vai ser o mesmo, para todas as atrações, já que o estávamos recebendo por parâmetro no atributo. Poderíamos tirá-lo do grid, onde não terá mais sentido.

Web panels sem tabela base

EnterAttractionsFilter X

Web Form Rules Events Conditions Variables

<No action group selected>

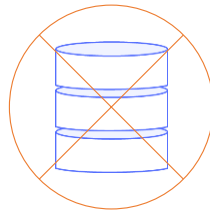
MainTable

Country Id &CountryId

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

List Attractions By Country List Attractions By Name



Agora veremos o caso dos web panel sem tabela base, ou seja, web panel que não possuem consulta ao banco de dados programada automaticamente.

O caso mais óbvio é quando o banco de dados não é consultado de forma alguma, como foi o caso do nosso painel da web inicial, que apenas pedia dados ao usuário e chamava outros objetos.

Web panels sem tabela base

The screenshot shows a web panel design with the following components:

- Form:**
 - Country Id:
 - Attraction Name From:
 - Attraction Name To:
- GRID:**

Attraction Id	Attraction Name	Country	Photo	Trips	
&AttractionId	&AttractionName	&CountryName		&trips	
- Total Trips:**
- Annotation:** An arrow points from the **Total Trips** field to the text "Somente variáveis!" (Only variables!).

On the right, the event script is shown:

```

Event Refresh
  &totalTrips = 0
Endevent

Event Start
  &Update.FromImage(updateIcon)
  &newTrip = "New Trip"
Endevent

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

Event &newTrip.Click
  &trips = NewTrip( &AttractionId )
  Refresh
Endevent

Event &AttractionName.Click
  ViewAttractionFromScratch( &AttractionId )
Endevent
  
```

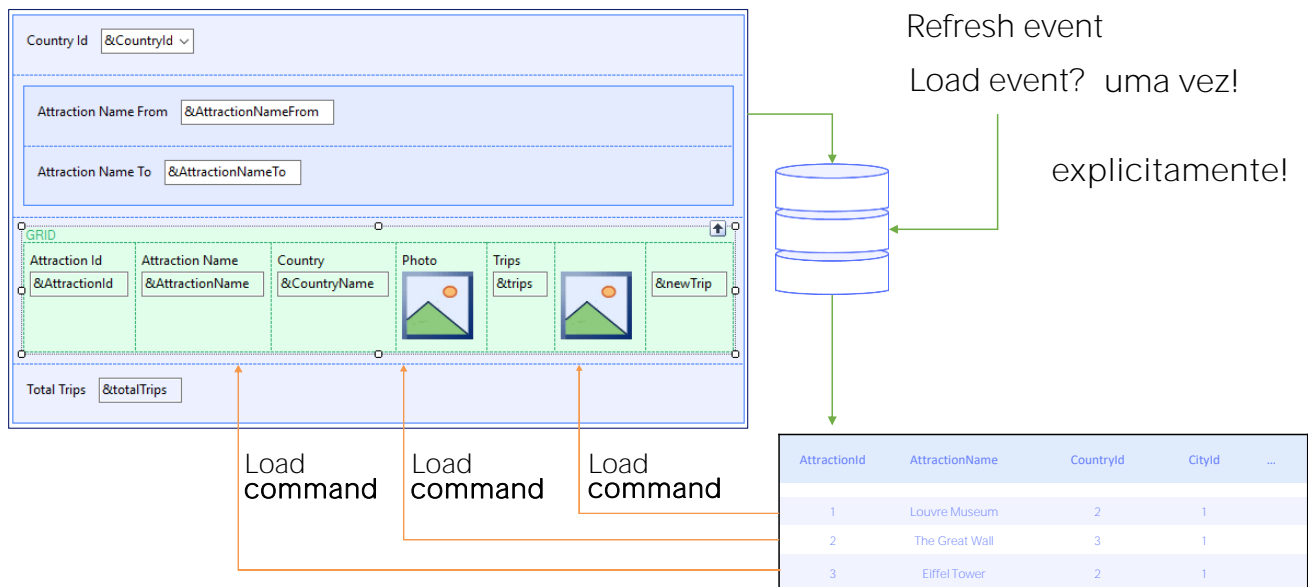
Mas também podemos ter um web panel que sim consulte a base de dados, só que essa consulta e carga na tela estão completamente nas mãos do desenvolvedor.

Os web panels que implementamos com tabela base poderiam ter sido implementados desta outra maneira.

Vamos ver o caso do web panel que mostra as atrações na grade. Mas agora, em vez de ter atributos na grid, temos variáveis.

Assim, nos eventos, temos que mudar as invocações que tínhamos, onde passávamos o atributo AttracionId, pela variável &AttractionId.

Web panels sem tabela base



Agora bem, se agora só temos variáveis, como GeneXus sabe que tem que navegar na tabela Attraction e sua estendida para carregar uma linha da grid por registro?

Não sabe. A carga dessa grid não será automática, como no caso em que colocamos atributos.

Ou seja, o evento Load não será executado quando já se tenha ido navegar uma tabela, para cada registro no qual se está posicionado em um determinado momento. É que não se está posicionado em lugar algum!

No entanto, o evento Load será disparado, sim. Só que o fará apenas uma vez, depois do Refresh e sem estar na base de dados em absoluto.

É nesse disparo, nessa execução, na qual teremos que programar a carga da grid manualmente.

Ou seja, teremos que solicitar explicitamente que se acesse a base de dados (no nosso caso, indo à tabela Attraction) e dizer-lhe cada vez que carregue cada linha.

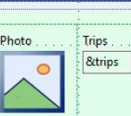
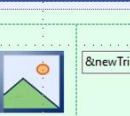
Para dizer-lhe para inserir uma nova linha na grid, com os valores que nesse momento possuam as variáveis correspondentes às colunas, se conta com o comando Load. O comando, não o evento. Sempre que dentro do evento Load, se encontre um comando Load, uma linha será inserida na grid.

Web panels sem tabela base

```

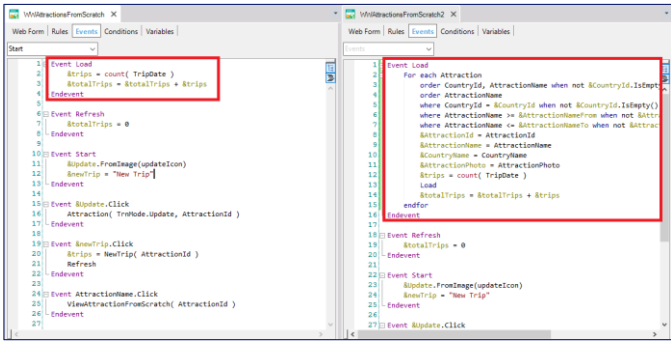
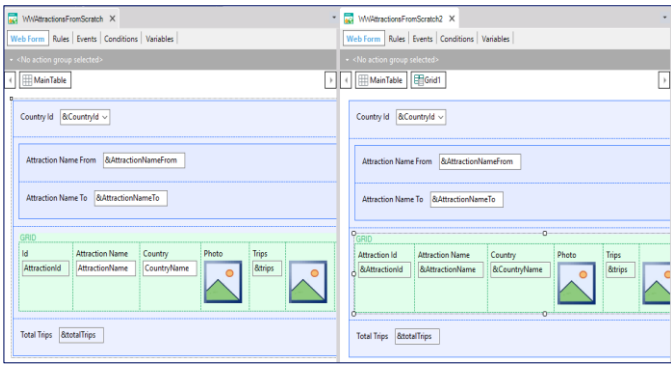
Event Load
For each Attraction
  order CountryId, AttractionName when not &CountryId.IsEmpty()
  order AttractionName
  where CountryId = &CountryId when not &CountryId.IsEmpty()
  where AttractionName >= &AttractionNameFrom when not &AttractionNameFrom.IsEmpty()
  where AttractionName <= &AttractionNameTo when not &AttractionNameTo.IsEmpty()
  &AttractionId = AttractionId
  &AttractionName = AttractionName
  &CountryName = CountryName
  &AttractionPhoto = AttractionPhoto
  &trips = count( TripDate )
  Load
  &totalTrips = &totalTrips + &trips
endfor
Endevent

```

Attraction Id	Attraction Name	Country	Photo	Trips	
&AttractionId	&AttractionName	&CountryName			&newTrip

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

O que teremos que fazer é programar, então, dentro do evento Load, o acesso à tabela Attraction com um for each. Nele especificaremos as cláusulas order e nas cláusulas where as condições que os dados devem atender. E para cada registro que atenda esses filtros, carregaremos as variáveis da grid com os valores dessa atração. E quando temos todas as variáveis carregadas, então escrevemos o comando Load, de modo que uma linha seja adicionada na grid com esses valores.



Navigation Report:



Control Name	Grid1
Collection	
Base Tm	
Order	
Conditions	

Grid sem tabela base

Aqui temos essa web panel já implementada. Fizemos um Save as da web panel que tínhamos com atributos, e mudamos por variáveis. E também mudamos os eventos, tal como estudamos recentemente.

Testemos o que foi feito para ver como em execução não notamos diferença alguma entre a web panel com tabela base e a web panel sem tabela base.

Filtremos, por exemplo, por França.

Agora, se observarmos o relatório de navegação da web panel sem tabela base:

Podemos ver que aparece o for each no Load indicando a navegação.

Uma observação: para dizer que o grid e o web panel não têm tabela base, não basta que só haja variáveis na tela e nenhum atributo. Não deverá haver atributos no Order do grid, nem nas conditions, claro, não deverá haver Transação base indicada, nem atributos fora de for eachs nos eventos, como aqui.

Evento Load: Mais de um grid

The screenshot shows a GeneXus 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 "GRID" with columns: "Attraction Id" (AttractionId), "Attraction Name" (AttractionName), "Attraction Photo" (with a photo icon), "Trips" (&trips), and "&newTrip" (with another photo icon). At the bottom of the form is a "Total Trips" field (&totalTrips). An arrow labeled "Grid1" points to the grid, and another arrow labeled "+ Grid2" points to the total trips field. To the right, a code block shows the event logic for Grid1.Load:

```
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, à possibilidade de inserir mais um grid. Sobre isto também voltaremos em outro vídeo.

Vamos subir tudo o que foi feito para o GeneXus Server

*GeneXus*TM

training.genexus.com
wiki.genexus.com