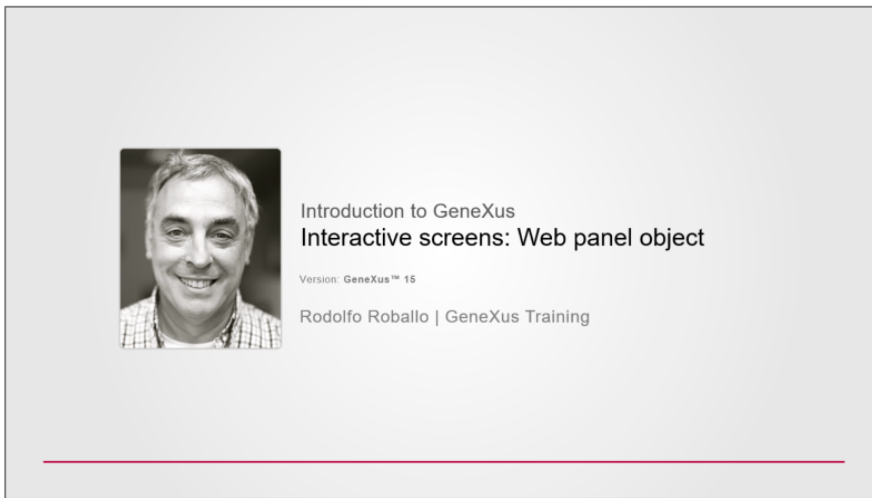


Telas interativas no ambiente web: objeto Web Panel



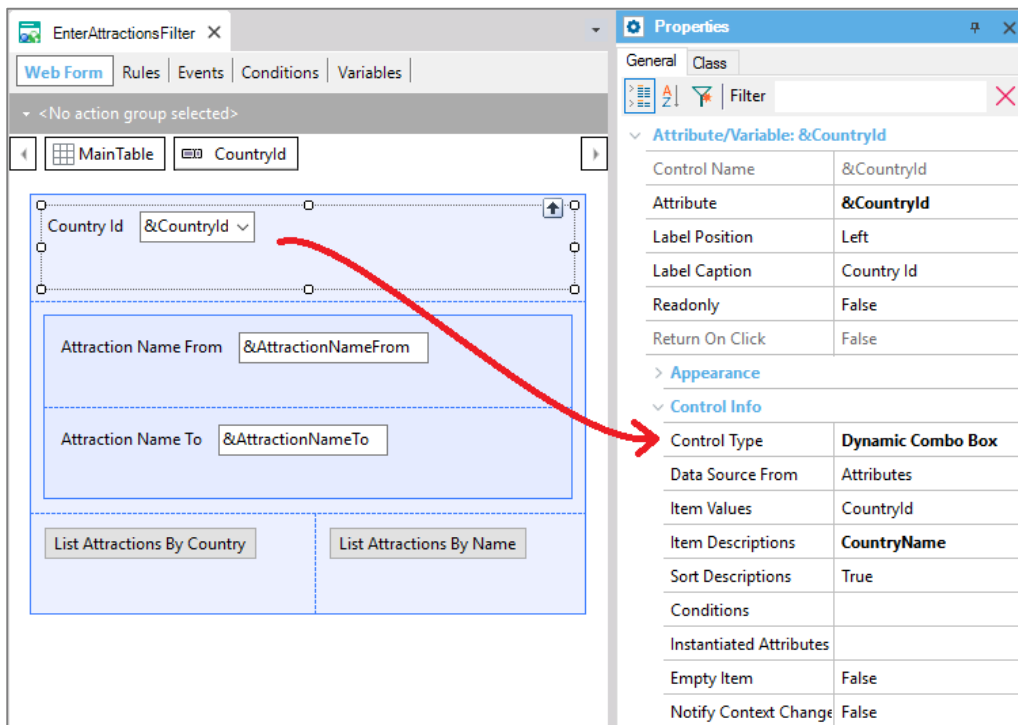
A web panel é o objeto mais flexível que GeneXus oferece.

Como já temos visto em alguns exemplos que temos mostrado, toda web panel oferece um web form, que é uma página web que nos permite “desenhar” e disponibilizar várias funcionalidades.

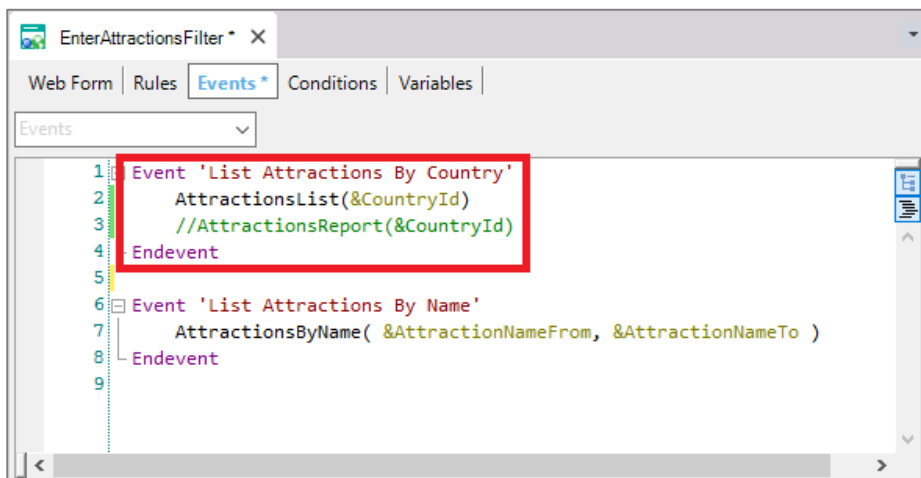
A screenshot of the GeneXus Web Form designer. The window title is 'EnterAttractionsFilter'. It has tabs for 'Web Form' (highlighted with a red box), 'Rules', 'Events', 'Conditions', and 'Variables'. Below the tabs is a dropdown menu showing '<No action group selected>'. There is a 'MainTable' tab with a grid icon. The main area contains a form with three sections: a top section with 'Country Id' and a dropdown menu '&CountryId'; a middle section with 'Attraction Name From' and a text input '&AttractionNameFrom'; and a bottom section with 'Attraction Name To' and a text input '&AttractionNameTo'. At the bottom of the form are two buttons: 'List Attractions By Country' and 'List Attractions By Name'.

Nesse exemplo havíamos visto que o fato de incluir variáveis no web form tornam as variáveis habilitadas para que o usuário entre com algum valor. Eram controles de entrada, ou também conhecidos como não readonly.

Essa variável em particular, do tipo combo dinâmico

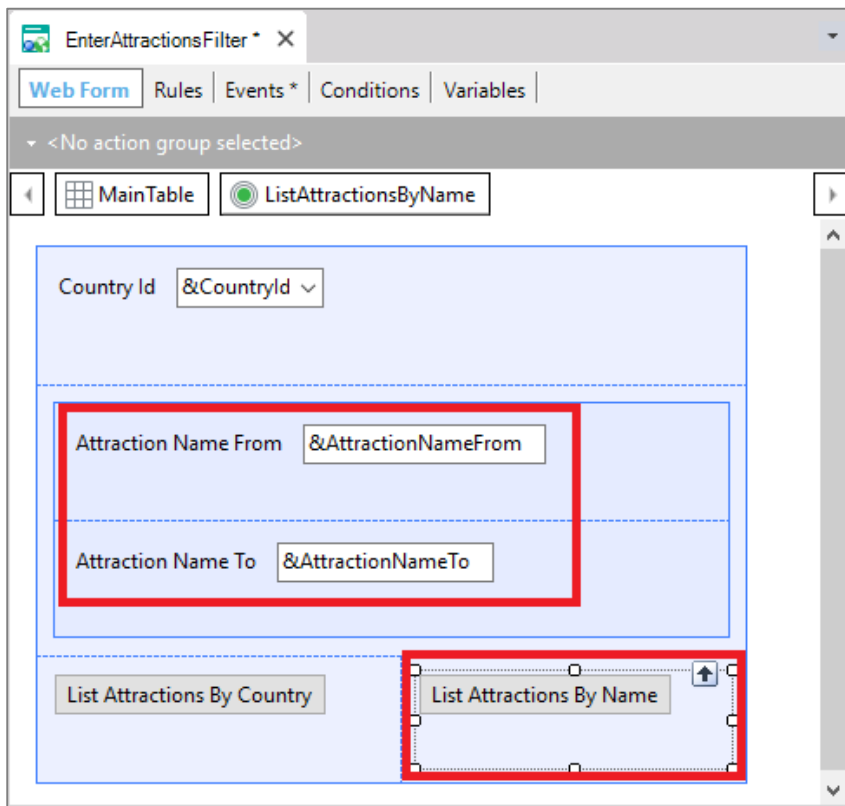


esperava que o usuário escolhesse um país dos carregados no combo e ao pressionar o botão “List Attractions By Country”, executava-se o evento associado...

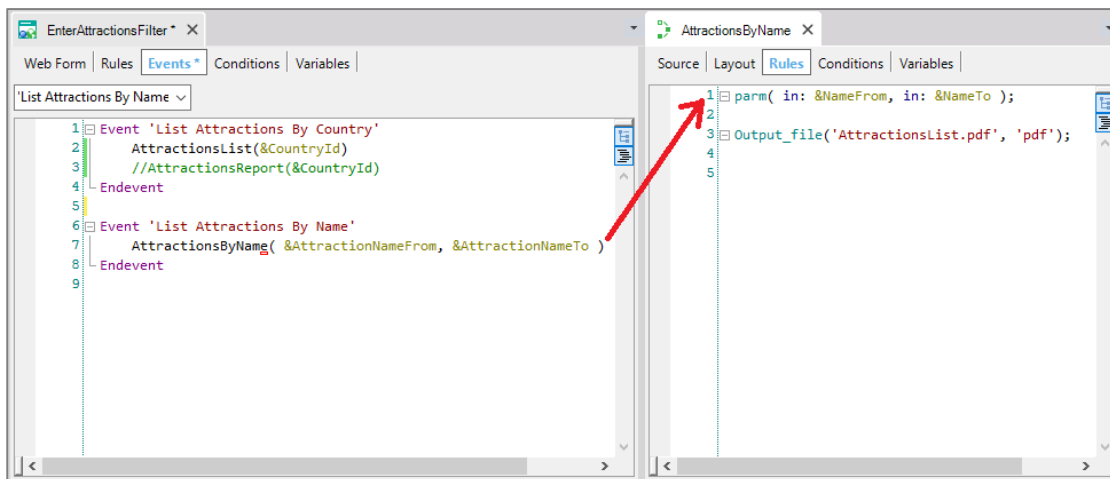


...chamando o pdf que listava as atrações desse país.

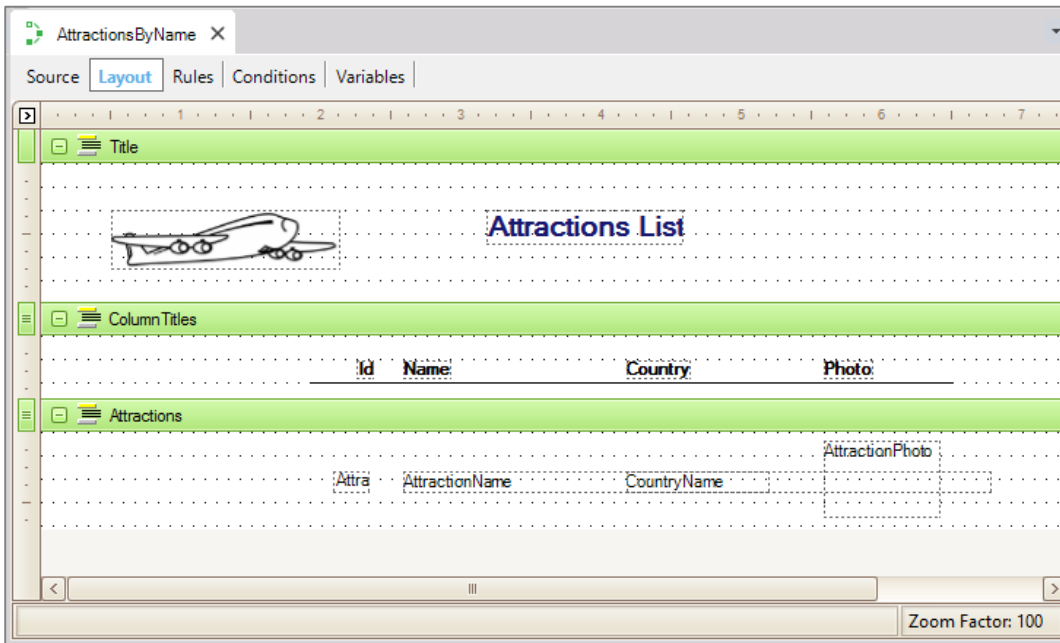
E nessas outras variáveis o usuário entrava com um range de nomes de atração para que, pressionando esse outro botão,



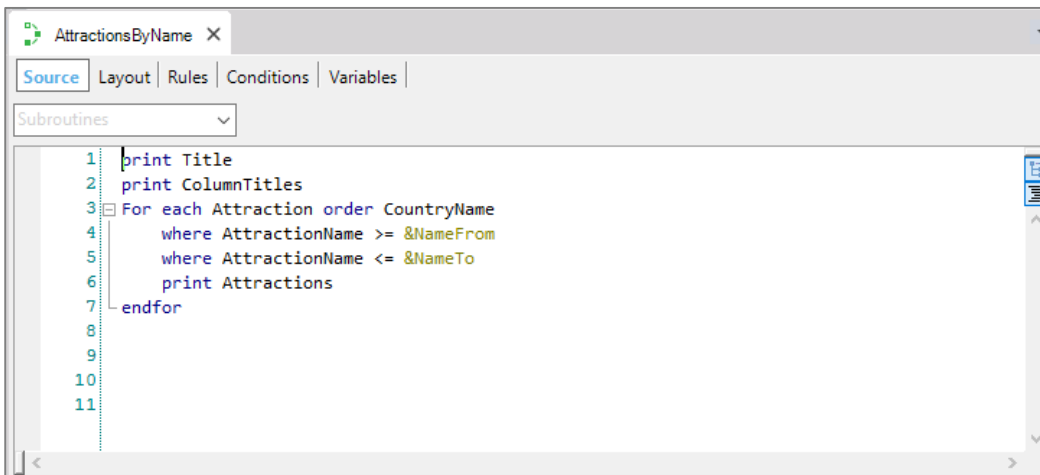
fosse chamado o relatório pdf que mostrava as atrações dentro desse range recebido via parâmetro.



Lembremos o layout:

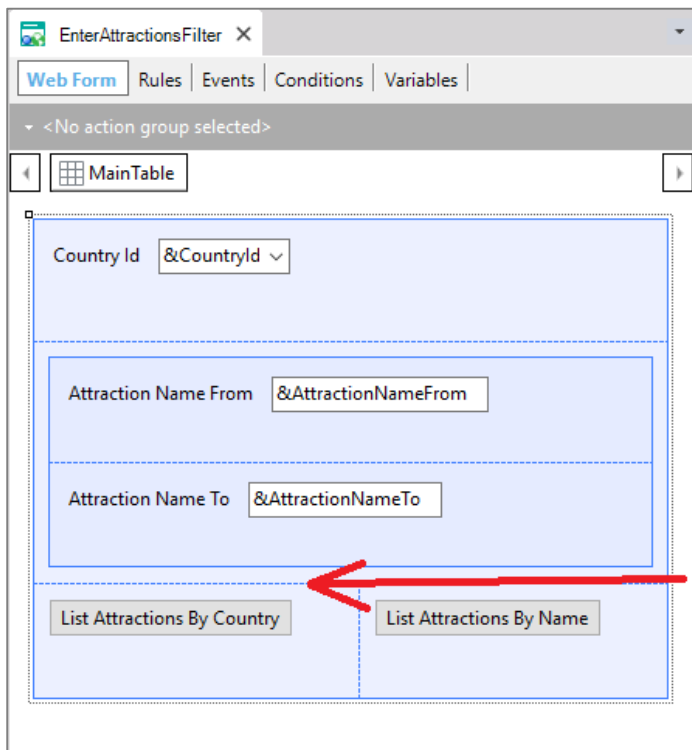


E no Source programávamos a consulta a base de dados com o for each, filtrando por nome:



Mas por que definir essas consultas através de relatórios pdf e não diretamente na própria tela na qual solicitamos ao usuário os dados para os filtros?

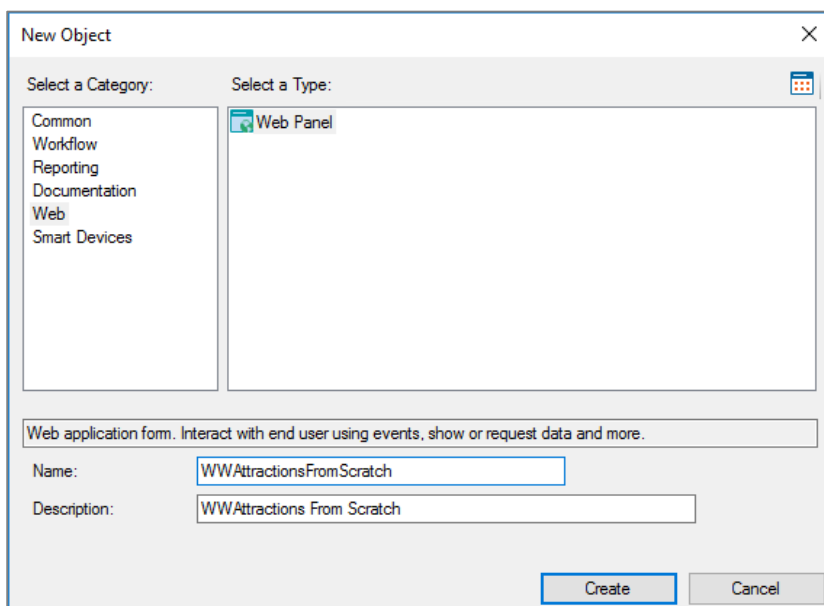
Por que não colocar aqui, no lugar dos botões, um grid que mostre as atrações desejadas?



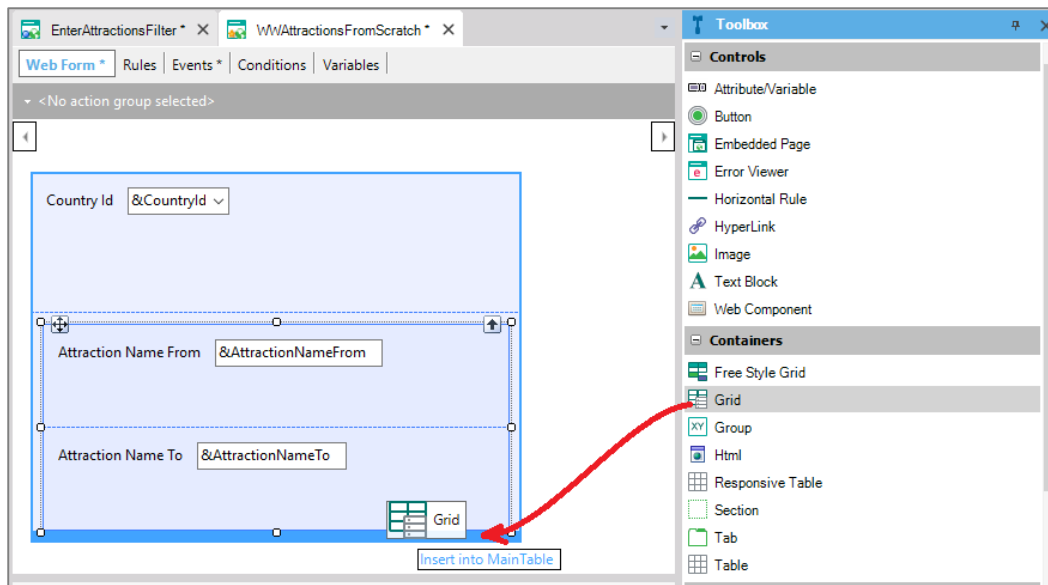
As web panels, além de permitir definir variáveis para utilizar nas ações programadas nos botões, permitem, -e é seu objetivo fundamental- implementar consultas **interativas** na base de dados.

A terminologia “**interativas**” refere-se a que o usuário pode informar na página da web panel uma e outra vez diferentes valores –**nas variáveis**- e consultar na sequência dados da base de dados que correspondam com esses valores informados, utilizando-os como filtros, como veremos agora.

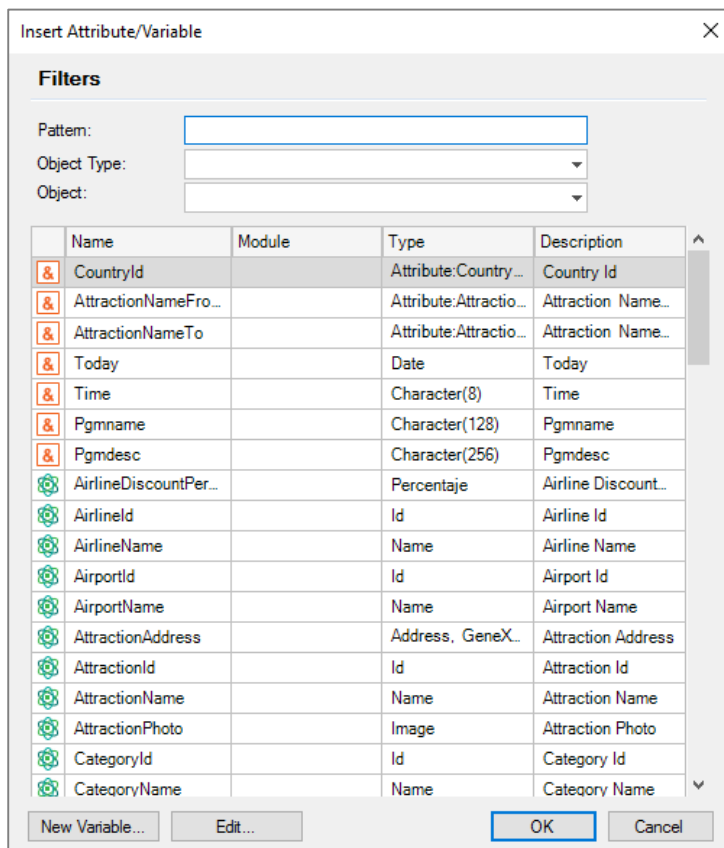
Salvemos essa web panel com outro nome. Como o que vamos implementar vai ser similar a um work with, chamaremos:



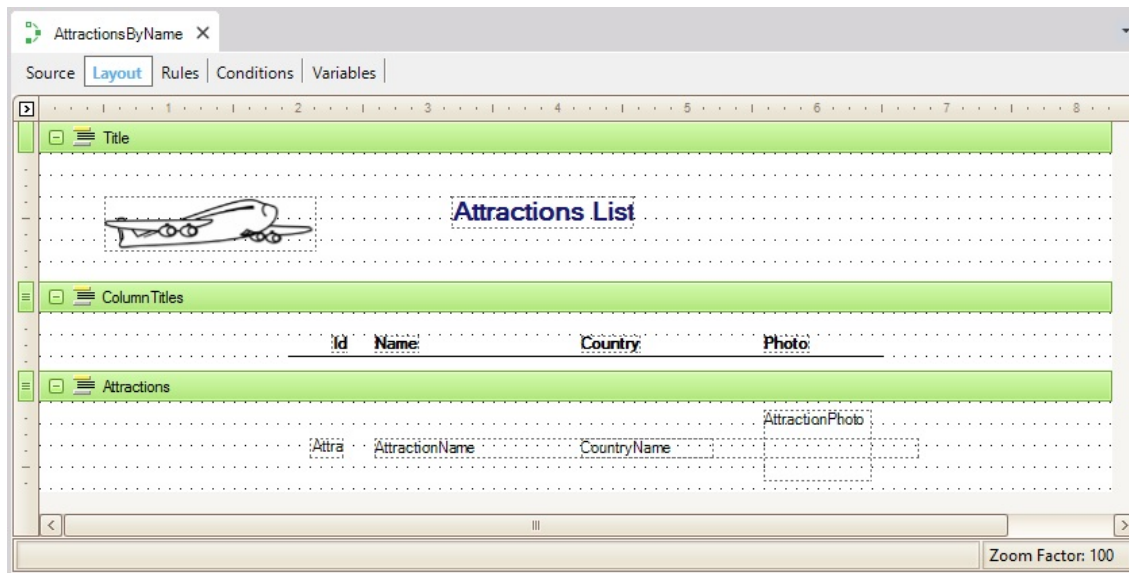
Eliminemos os botões, que já não serão necessários, assim como os eventos associados. Agora inserimos debaixo das variáveis um controle do tipo grid:



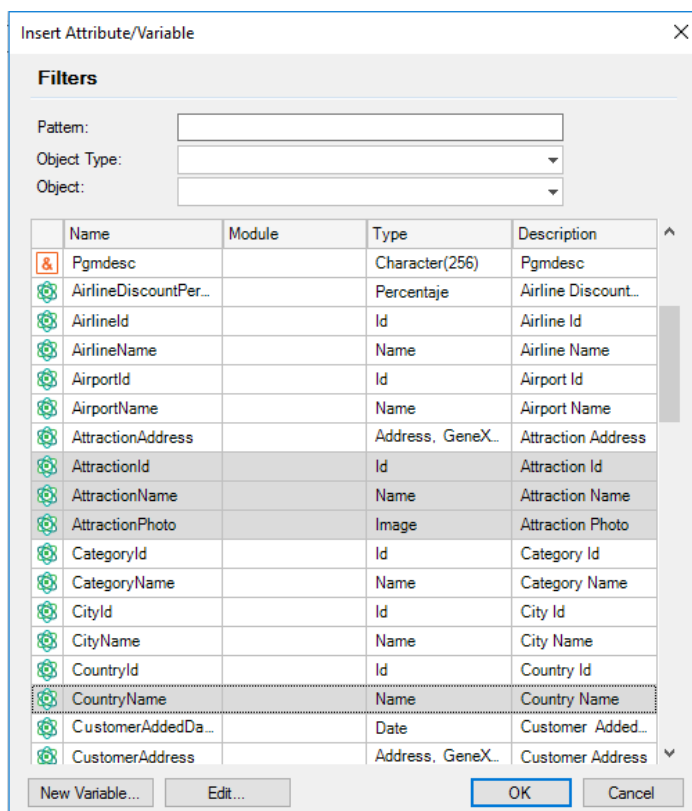
Abre-se essa tela para escolher os atributos e/ou variáveis que serão as colunas desse grid.



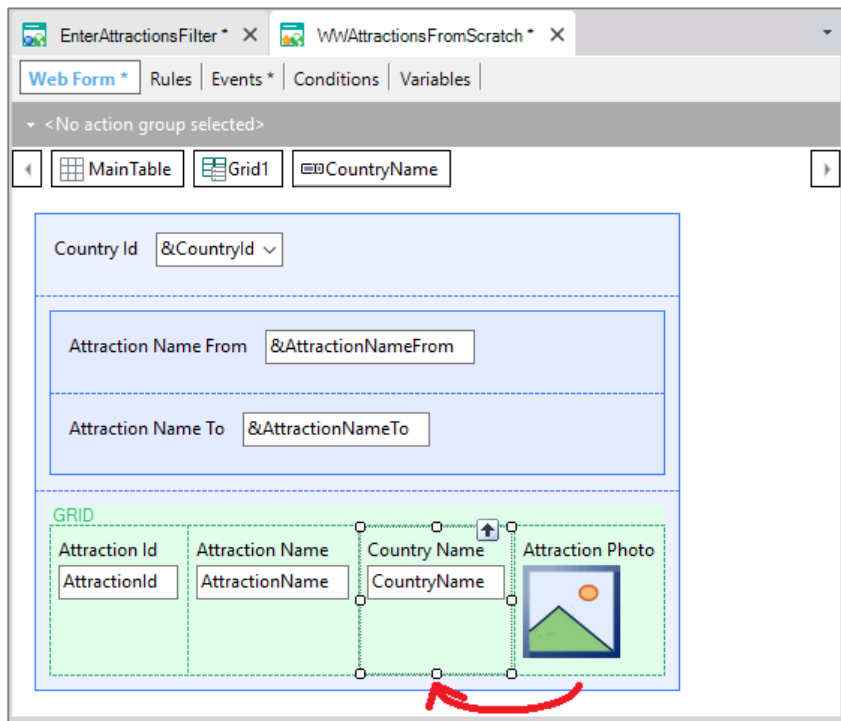
Como o que queremos é o mesmo que mostrávamos no relatório pdf...



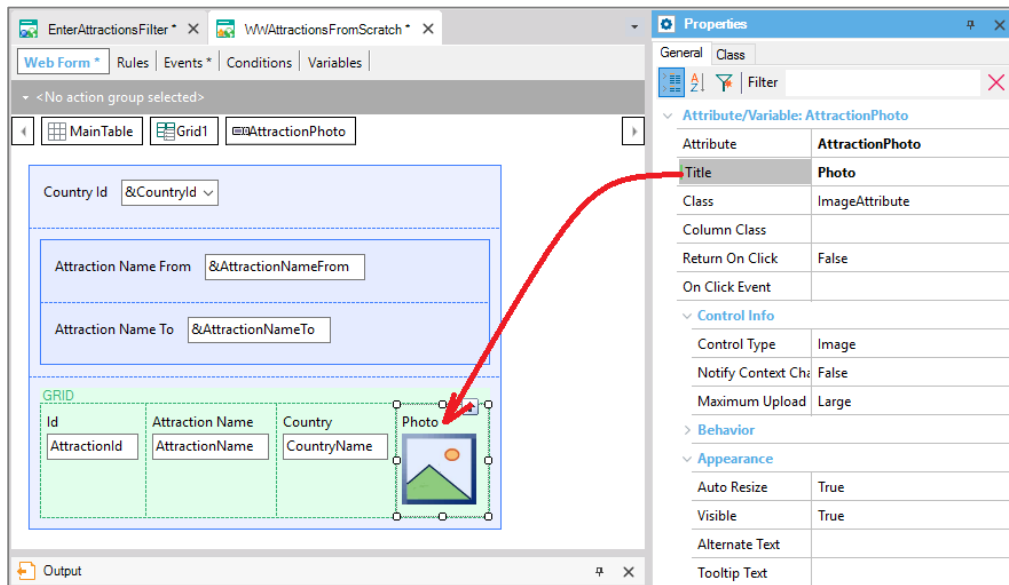
...escolhemos os atributos AttractionId, AttractionName, AttractionPhoto e CountryName...



e pressionamos OK. Visualizamos que foi criado um grid com essas colunas. Movamos CountryName para aqui:

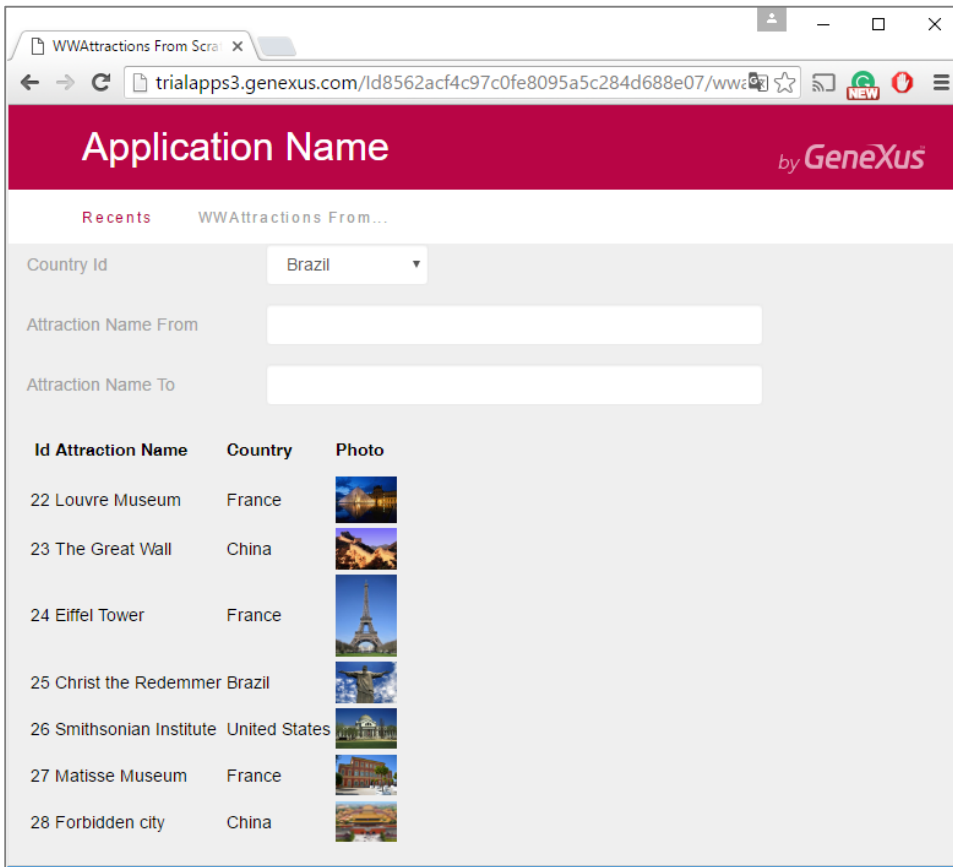


Podemos alterar os títulos de cada coluna, editando as propriedades de cada um dos atributos que pertencem as colunas do grid.



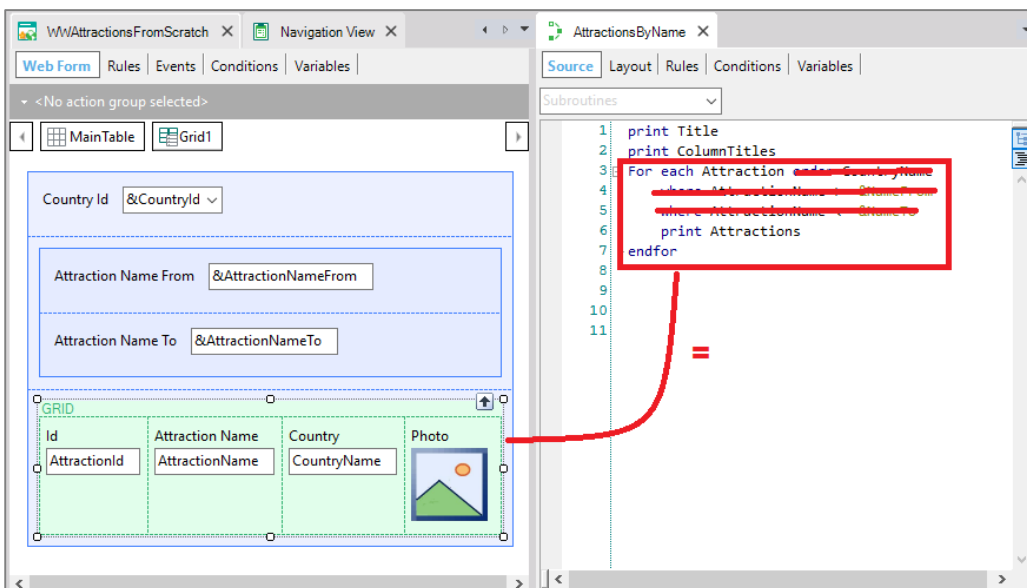
Pressionemos F5 para executar nossa nova web panel assim como temos feito até agora.

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

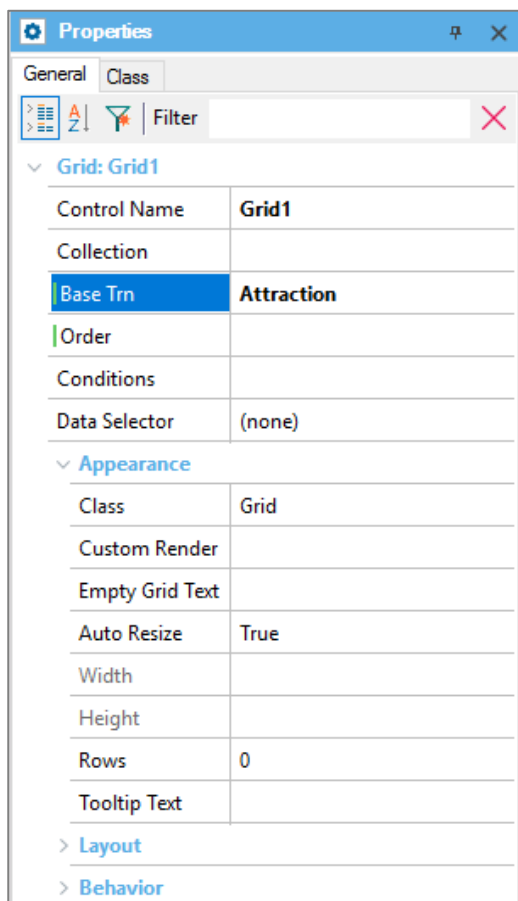


Visualizamos que saíram impressas todas as atrações, com os dados que indicamos (o id, nome, país e foto). Além do que, saíram ordenadas por AttractionId.

Tão somente em colocar um grid com esses atributos, GeneXus entendeu que deveria ir a base de dados, acessar a tabela Attraction, acessar também a tabela Country para trazer o país da atração, tal como fazíamos com o comando for each (sem essas cláusulas):

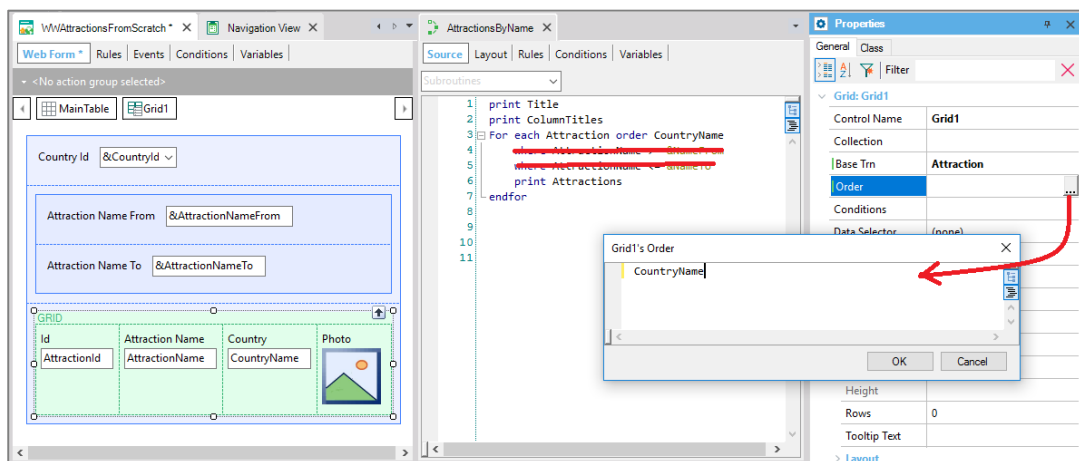


Se observamos as propriedades do grid, aparece de fato uma de nome **Base Trn**, que é análoga a transação base do for each. De fato, para assegurarmos de que para o grid seja eleita a tabela base Attraction, que é a que desejamos, é uma boa prática indicar a transação base, igual ao que é feito para o for each.

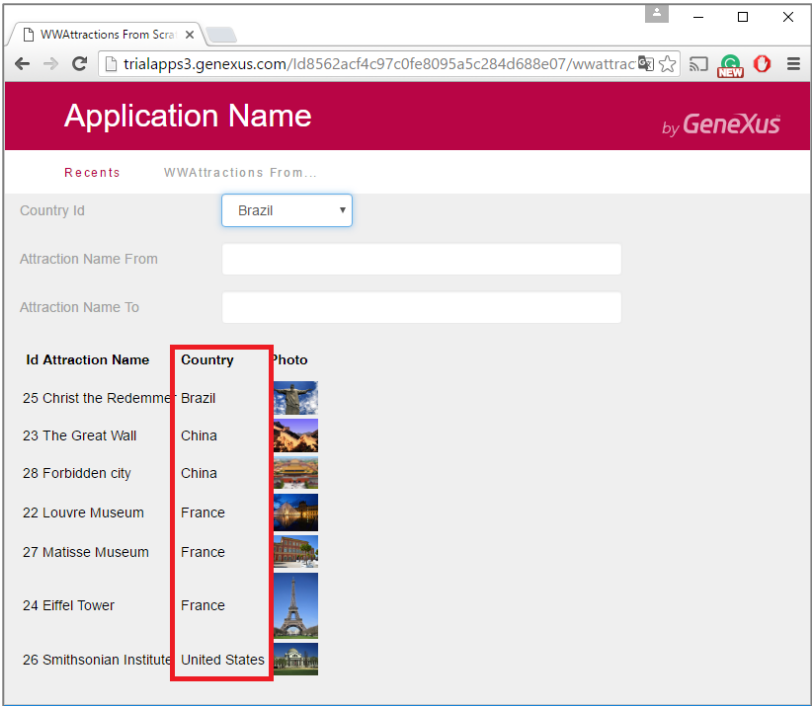


Por outro lado, observemos que temos uma propriedade **Order** para o grid. Essa propriedade corresponde-se com a cláusula Order do for each.

Assim, se quisermos ordenar por nome de país, como no relatório:

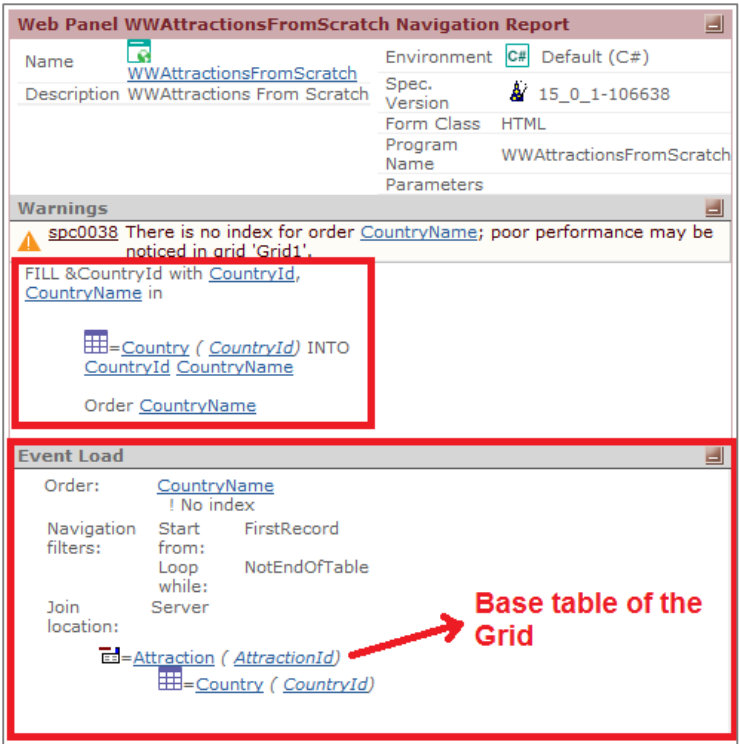


Pressionemos F5.



E visualizamos que agora o grid sai ordenado por nome de país.

Observemos o relatório de navegação da web panel:

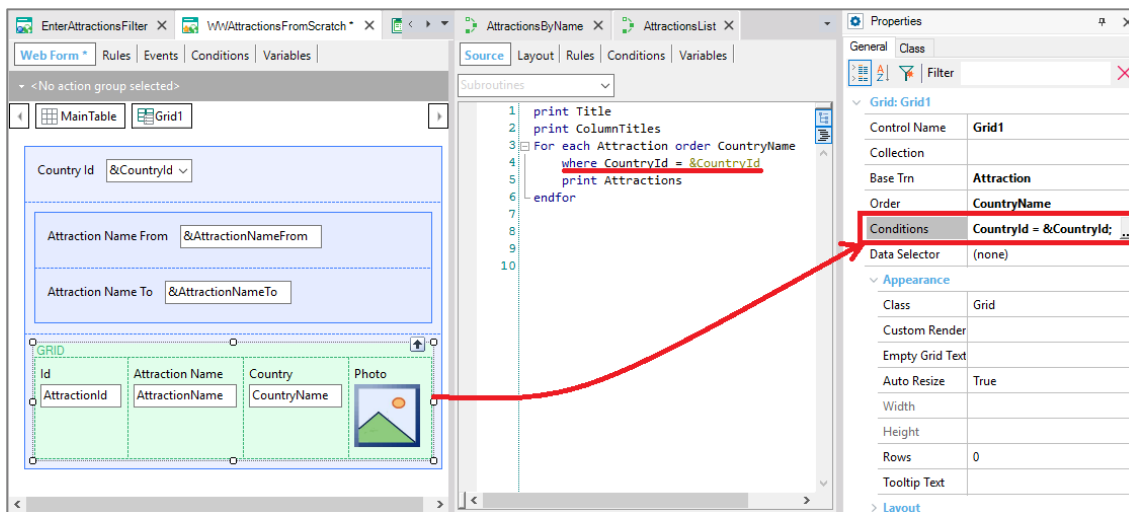


Aqui está indicando a navegação que tem que realizar para carregar o combo box da variável &CountryId, que por agora não estamos usando para nada.

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

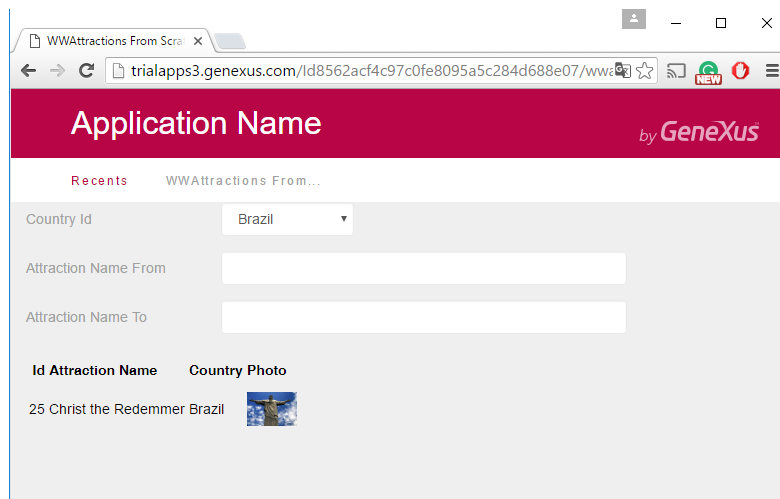
Até agora não fizemos nada com as variáveis. Mas queríamos utilizá-las para filtrar os dados que são mostrados no grid, tanto por país como por nome de atração.

Em AttractionList filtrávamos por país. Como indicamos esse filtro para o grid? Através da propriedade **Conditions**:



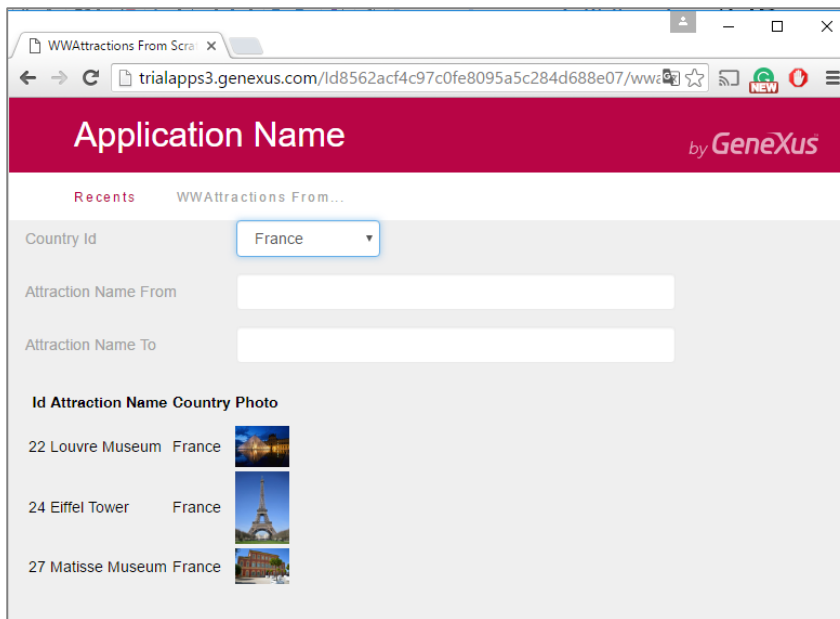
Aqui estamos dizendo ao GeneXus que queremos que quando leia a tabela base do grid, Attraction, filtre os registros para os quais o CountryId da atração é igual ao valor que o usuário tem escolhido no combo &CountryId.

Pressionemos F5.



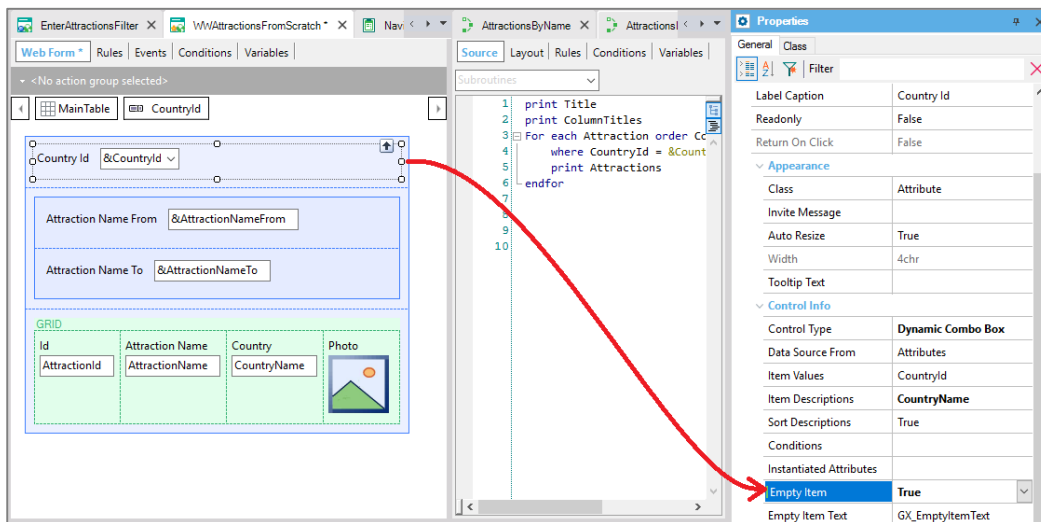
Observemos que por padrão, o combo tem o valor Brasil, e que no grid só visualizamos atrações do Brasil.

Se escolhemos França, vemos que a tela é atualizada, voltando-se a carregar o grid, agora com as atrações da França.

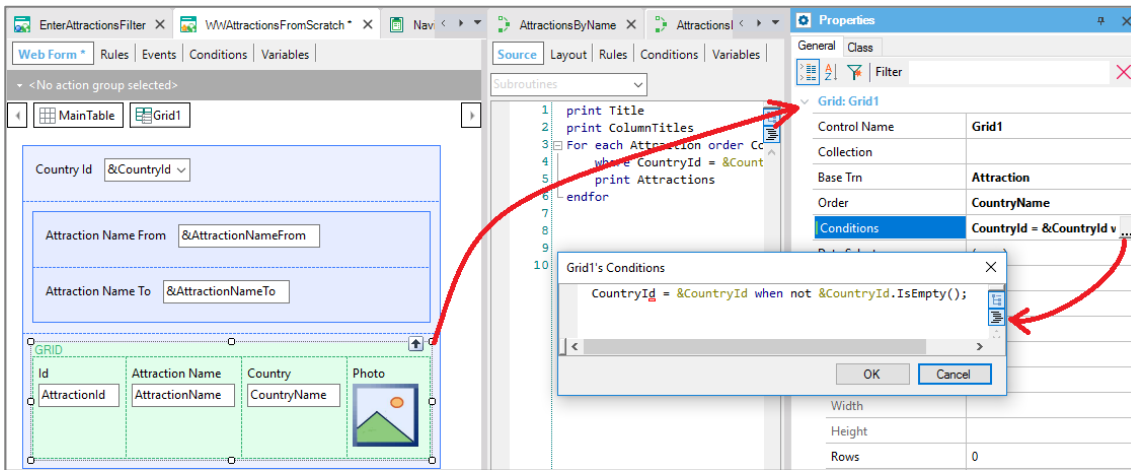


É bem provável que queremos que o combo apareça sem valor escolhido na primeira vez, e que nesse caso seja mostrado as atrações de todos os países.

Para isso, editemos as propriedades do combo box... e alteremos para True a propriedade de nome Empty Item. Isso fará que seja adicionado uma opção "(none)" ao combo. Corresponderá ao valor empty, ou seja, vazio, zero.

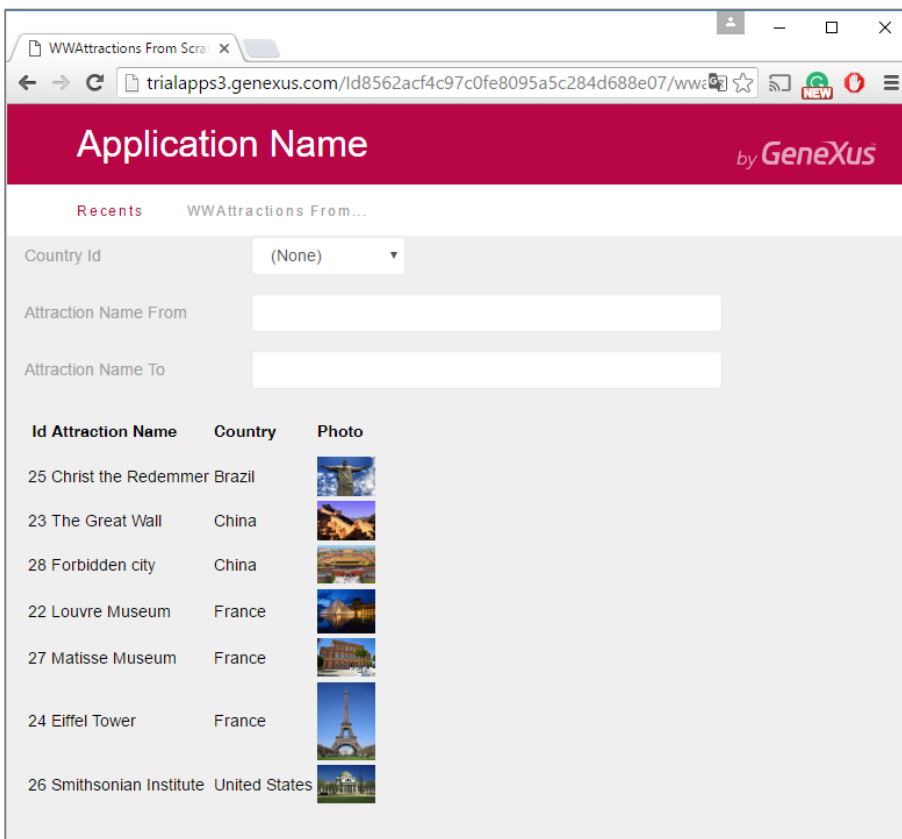


Então, vamos até a propriedade **Conditions** e especifiquemos que queremos que seja aplicada essa condição somente quando o combo não tem o valor vazio. Quando tenha valor vazio, que não aplique-a.

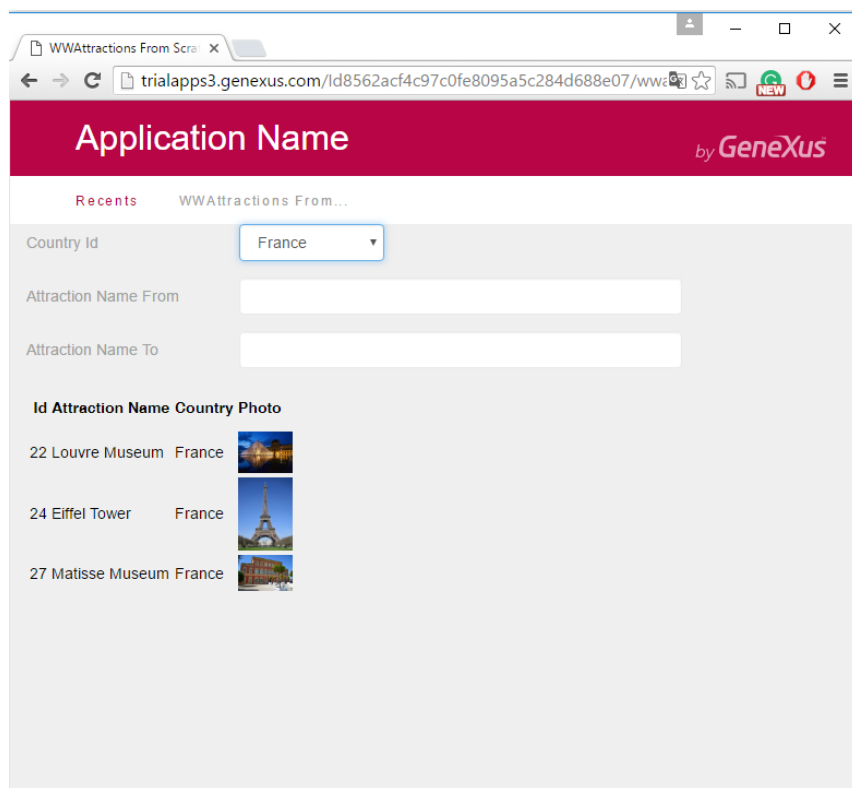


Executemos...

Aparece o valor (None) no combo, e além disso para esse caso, as atrações não estão sendo filtradas. Todas são mostradas:



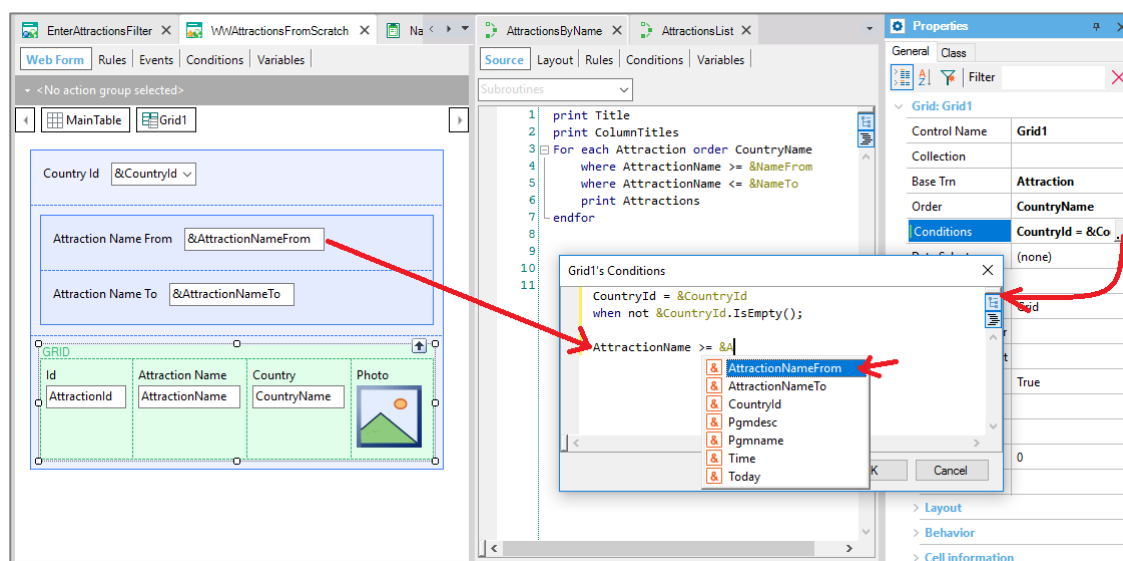
Se agora escolhemos, por exemplo, França:



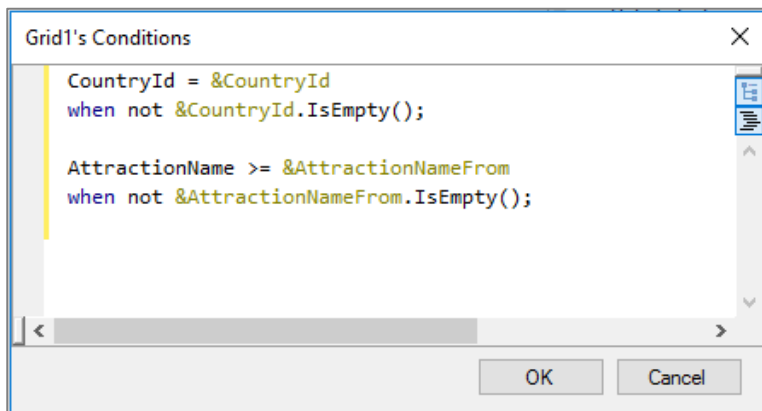
Como a variável não tem valor vazio, o filtro é aplicado e são mostradas as atrações da França.

Também teríamos que adicionar os filtros por nome da atração, que gostaríamos que fosse somado ao outro filtro.

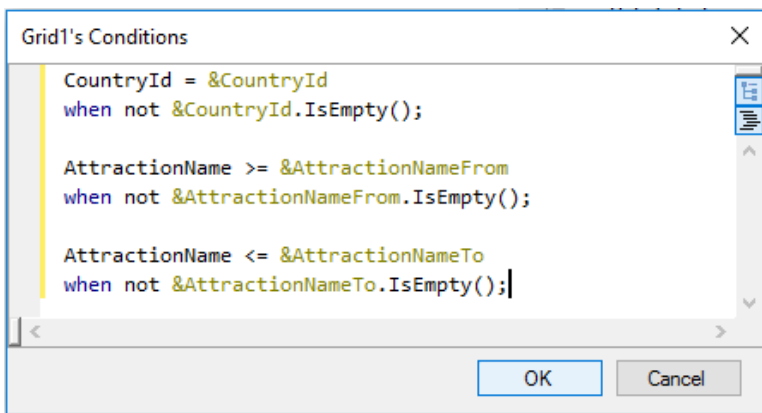
Então... se no relatório filtrávamos no for each com essas duas cláusulas where... no grid adicionaremos como condições:



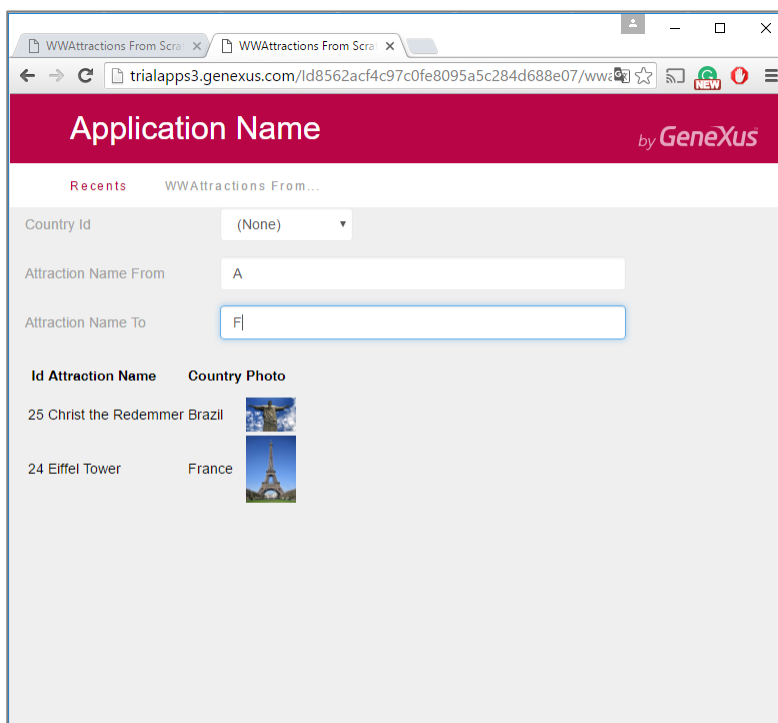
AttractionName maior ou igual ao valor da variável &AttractionNameFrom do form, que o usuário haverá digitado, se desejar. Outra vez, se o usuário não entra com valor na variável, não queremos que esse filtro seja aplicado. Então usamos a cláusula when. Essa cláusula também pode ser utilizada na cláusula where do for each, de uma forma completamente análoga.



E adicionamos o outro filtro:

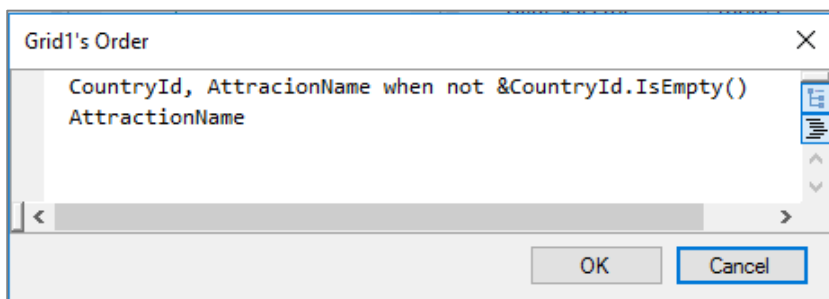


Executemos novamente. E escolhemos ver as atrações entre A e F:



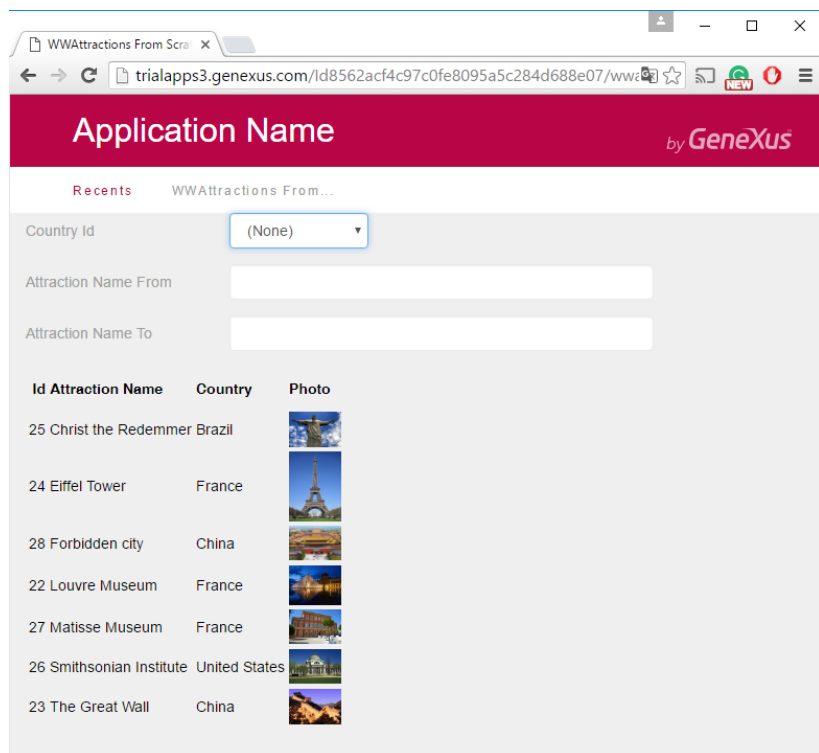
Podemos solicitar na web panel que se o usuário escolhe um país, então ordene a informação por CountryId e dentro de CountryId por AttractionName, caso contrário, ordene por AttractionName. Isso pensando em otimizar a busca dos registros da tabela.

Para isso, editamos no grid a propriedade Order e escrevemos primeiro a ordem, condicionada a que o usuário tenha escolhido um país no combo. Nesse caso será filtrado por esse país, mas além disso, as atrações sairão listadas alfabeticamente para esse país. E se o usuário deixar o combo com o valor “(none)”, isto é, vazio, então será escolhida a ordem seguinte, que é por AttractionName:

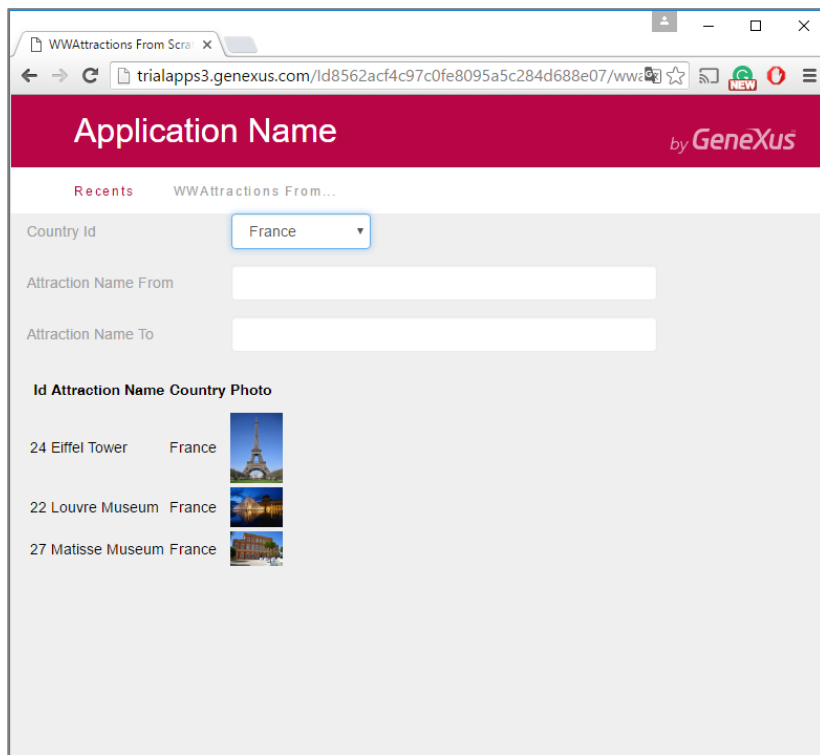


Não nos deteremos nisso aqui. Deixamos apontado unicamente para mostrar que também é possível condicionar a forma em que se quer ordenar a informação. Isso é idêntico a um for each.

Executemos:

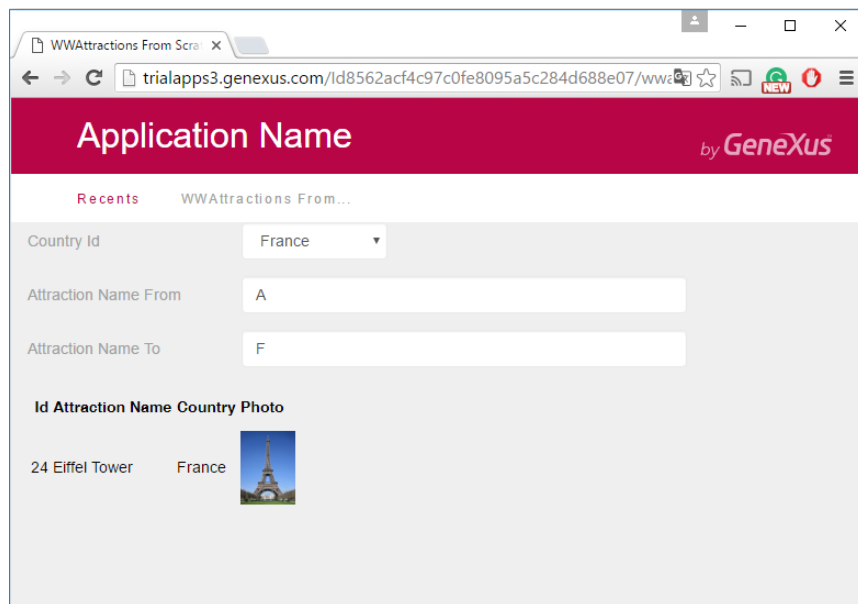


Aqui ordenou-se por AttractionName. E se escolhermos França, será ordenado pelo Id da França e dentro dele por AttractionName.

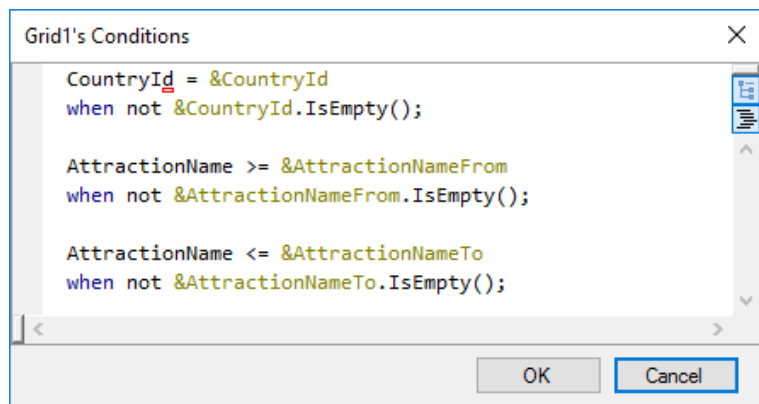


Em definitivo, sempre vemos as atrações ordenadas alfabeticamente.

Se dentro das atrações da França queremos as que se encontram entre A e F:



Vemos que o grid foi carregado filtrando pelas três condições que havíamos escrito:



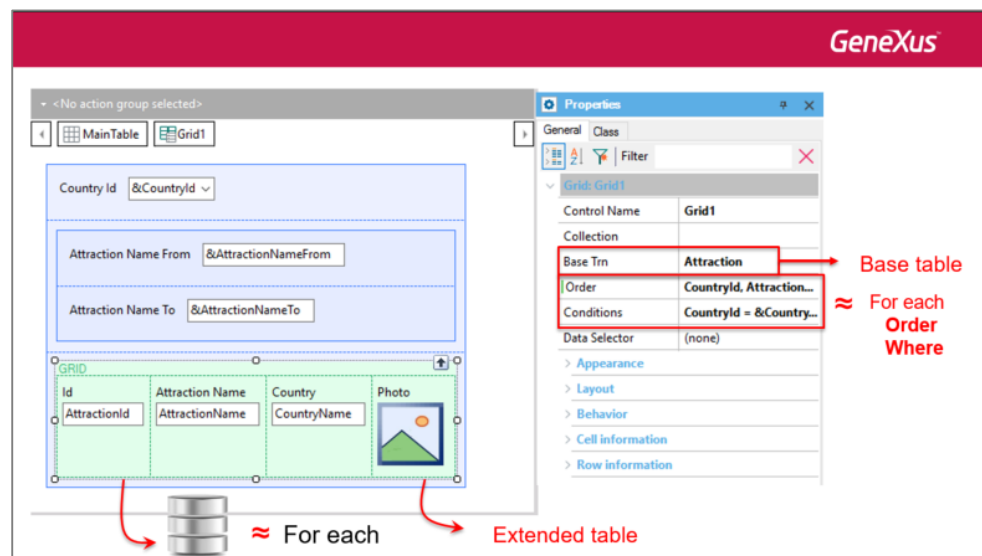
Resumindo o que foi feito até o momento:

Temos implementado uma web panel na qual incluímos algumas variáveis para que o usuário entre com valor, e inserimos um controle Grid com atributos.

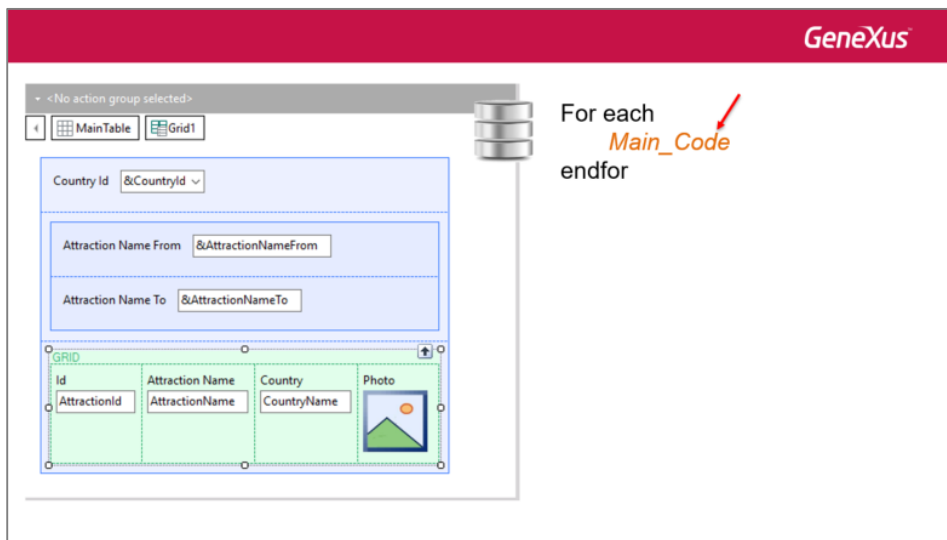
Os atributos correspondem a informação da base de dados, pelo que GeneXus entende que deve ir buscar essa informação. Um grid com atributos é como um for each. Por isso contamos com a propriedade **Base Trn**, como a do for each, para especificar o nível da transação cuja tabela associada queremos ler. Chamamos essa tabela, **tabela base do grid**. Caso não especifiquemos, como também ocorre com um for each, GeneXus infere na base os atributos que são utilizados. Esse caso não veremos aqui.

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

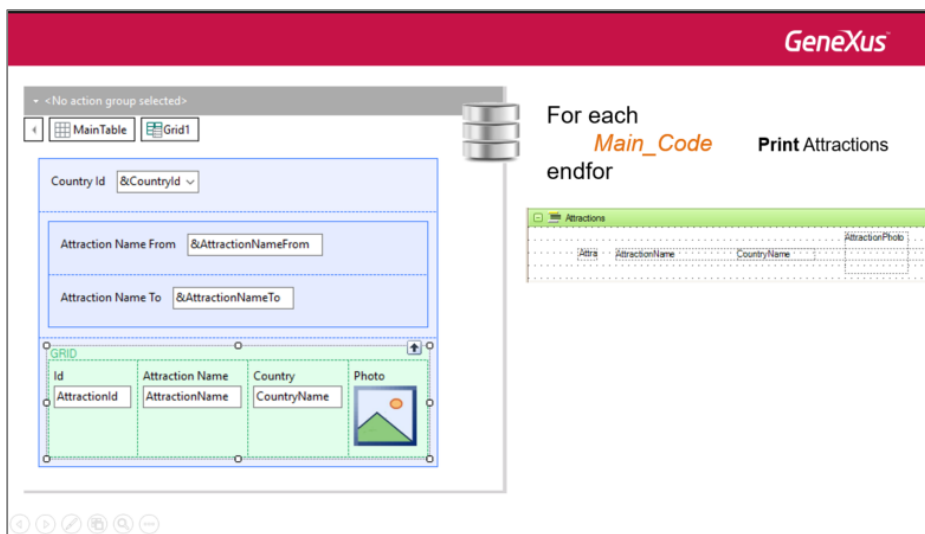
Assim como em um for each ordenamos a informação com a cláusula Order e filtramos os dados a serem devolvidos pela consulta com uma ou várias cláusulas where, para fazer o mesmo com o grid temos as propriedades Order e Conditions, respectivamente.



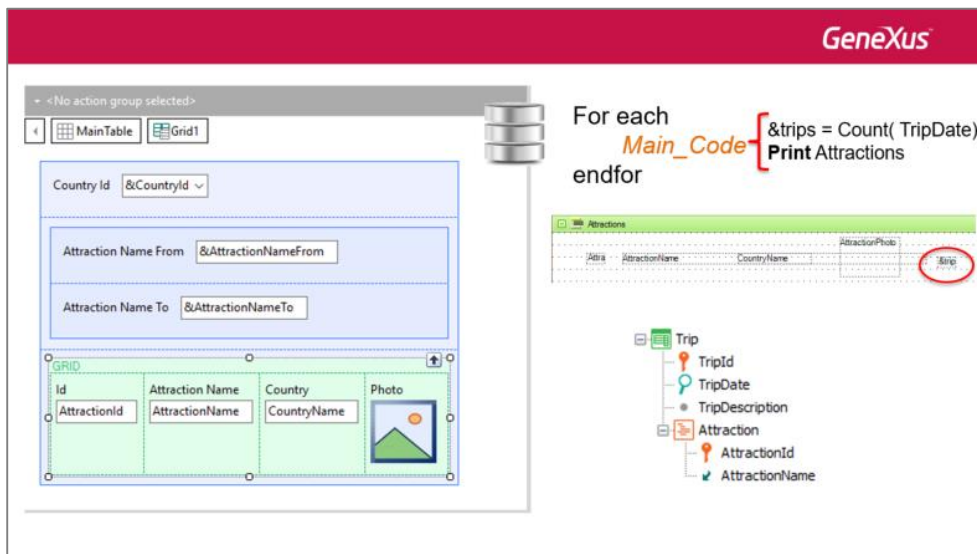
Agora bem, em um for each, programamos o que queremos que seja feito com cada registro que cumpra as condições, dentro do seu corpo.



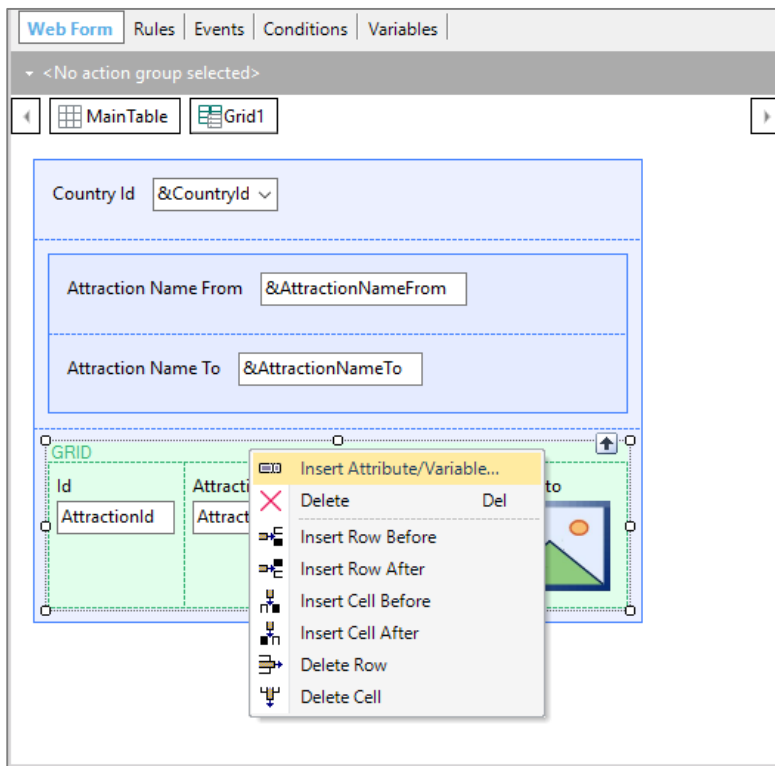
Por exemplo, no relatório de atrações turísticas, o comando print Attraction imprimia na saída o que em nosso caso seria a linha do grid. No caso do grid não é necessário especificar. Isso é feito automaticamente.



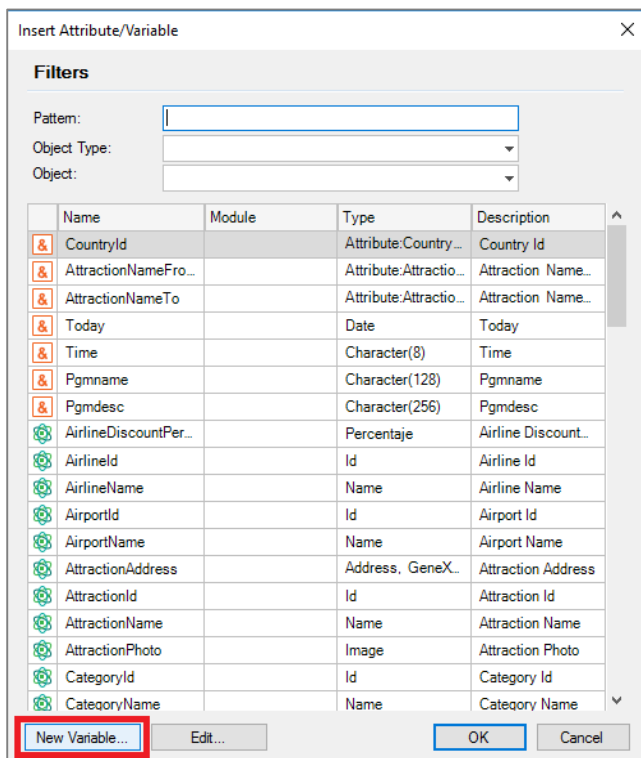
Mas, por exemplo, imaginemos que temos uma transação Trip que registra as excursões oferecida pela agência de viagens. De forma muito simplificada, imaginemos que cada excursão só registra a data em que ocorrerá e a descrição e na sequência são registradas as atrações turísticas que serão visitadas por essa excursão. Bem, então agora suponhamos que no relatório de atrações turísticas queremos ver também a quantidade de trips que tem associada cada atração. Para isso bastava definir uma variável &trips, numérica, e atribuir dentro do corpo do for each, (isso é, quando o for each está posicionado no registro de sua tabela base que está para processar) o resultado de contar os trips dessa atração. E colocar essa variável no print block.



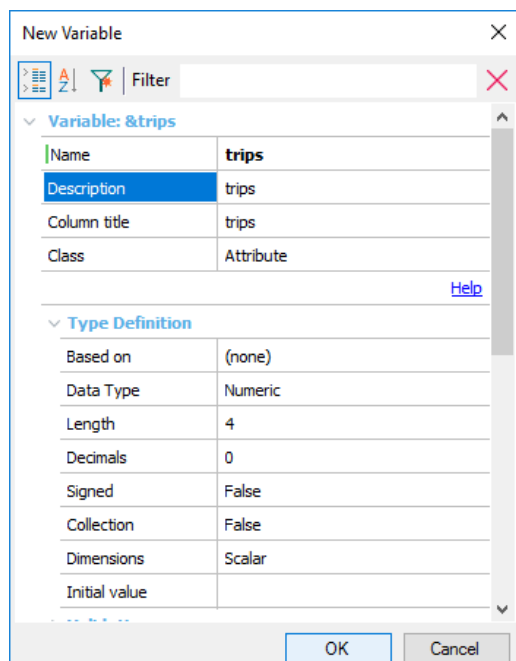
Para fazer o mesmo na web panel, clicamos com o botão direito sobre o grid e Insert Attribute/Variable



Depois New Variable:



E definimos a variável trips:




Ao pressionar OK... podemos ver que foi inserida como coluna do grid. Movemos ela para que ocupe a posição que nos interessa dentro do grid.

Country Id

Attraction Name From


Attraction Name To

GRID

trips	Id	Attraction Name	Country	Photo
&trips	AttractionId	AttractionName	CountryName	

E mudamos a propriedade **Title** para que o título da coluna saia em maiúscula.

GRID

Id	Attraction Name	Country	Photo	Trips
AttractionId	AttractionName	CountryName		&trips

Isso corresponde a ter inserido a variável no printblock. Mas onde informamos como é calculada?


GeneXus

Country Id

Attraction Name From

Attraction Name To

GRID

Id	Attraction Name	Country	Photo	Trips
AttractionId	AttractionName	CountryName		&trips

For each
Main_Code
 endfor

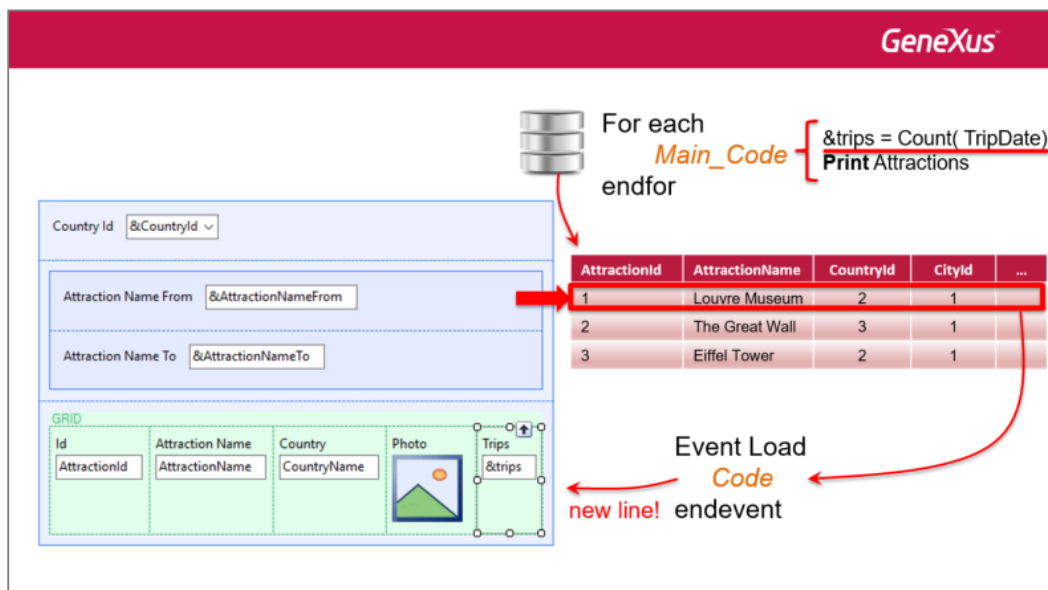
&trips = Count(TripDate)
Print Attractions

Attractions

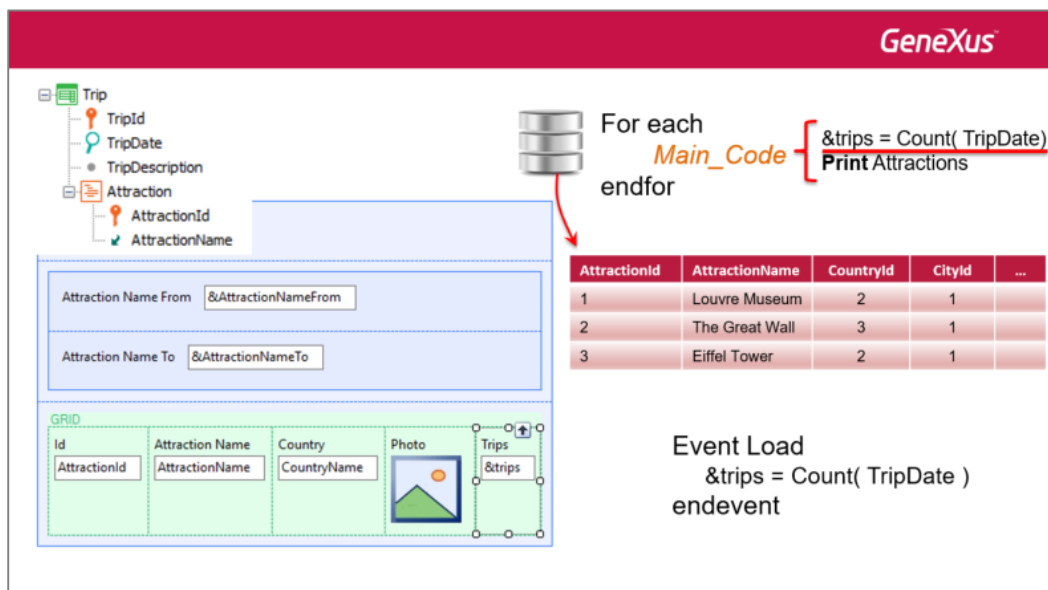
Attri	AttractionName	CountryName	AttractionPhoto	Trips
				&trips

No for each é dentro de seu corpo. E aqui?

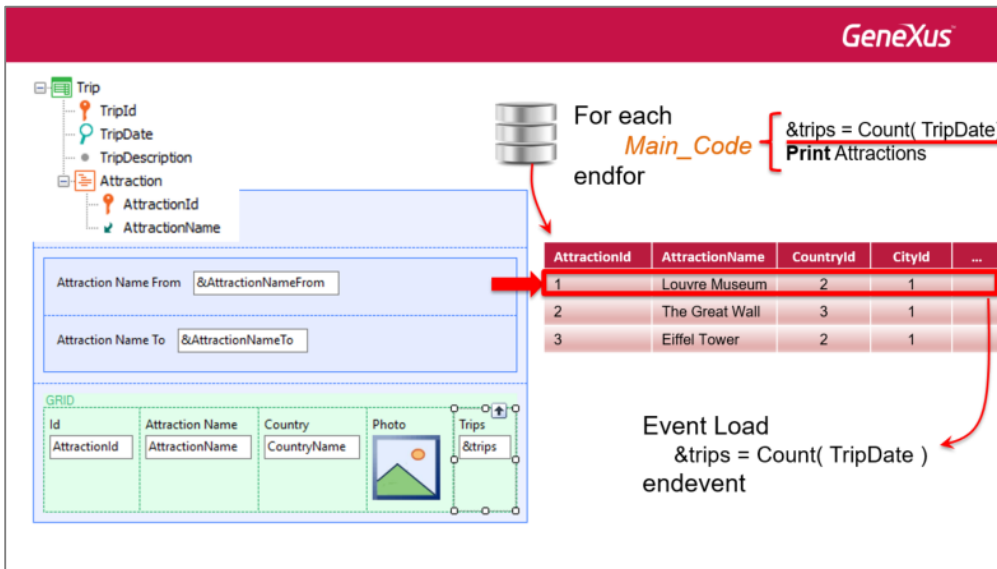
Contamos com o evento **Load** do sistema. Ali dentro programaremos o que queremos que seja executado quando se está posicionado em um registro da tabela base do grid, imediatamente antes que a linha correspondente seja carregada no grid.



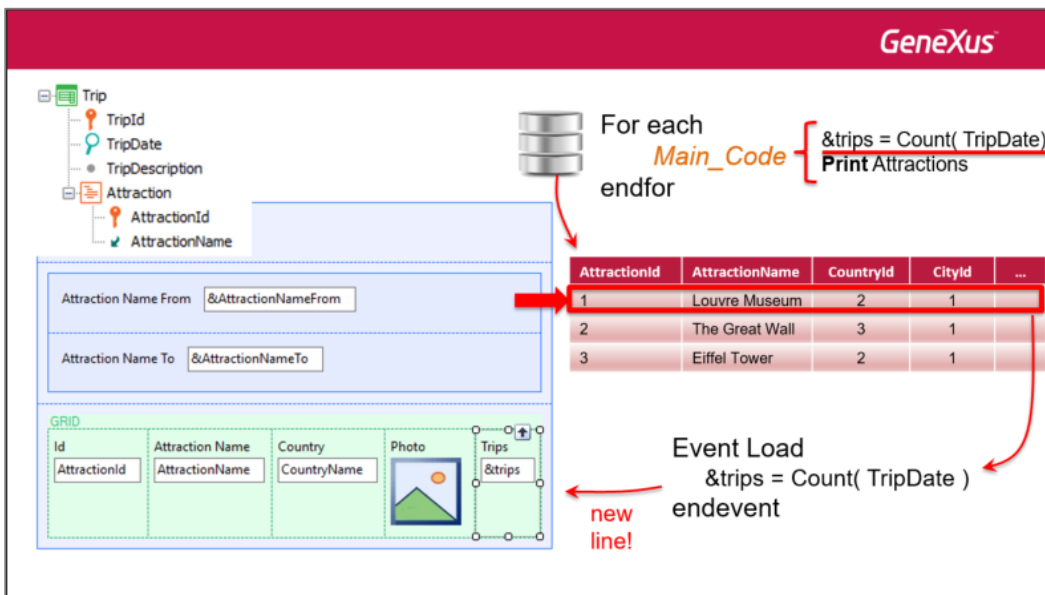
Em nosso caso, ali é onde atribuiríamos valor na variável &Trips:



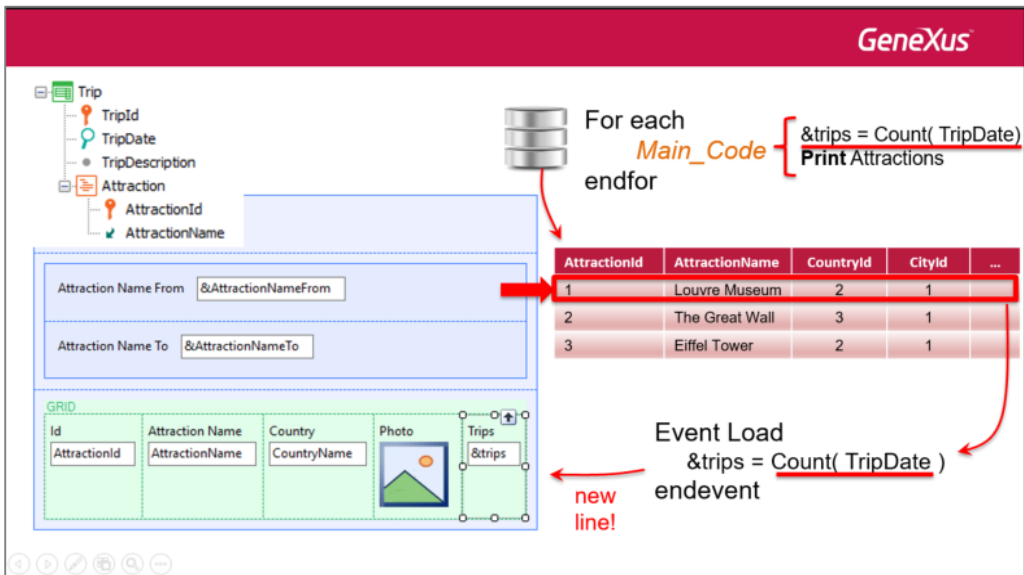
O evento Load vai ser executado automaticamente para cada registro



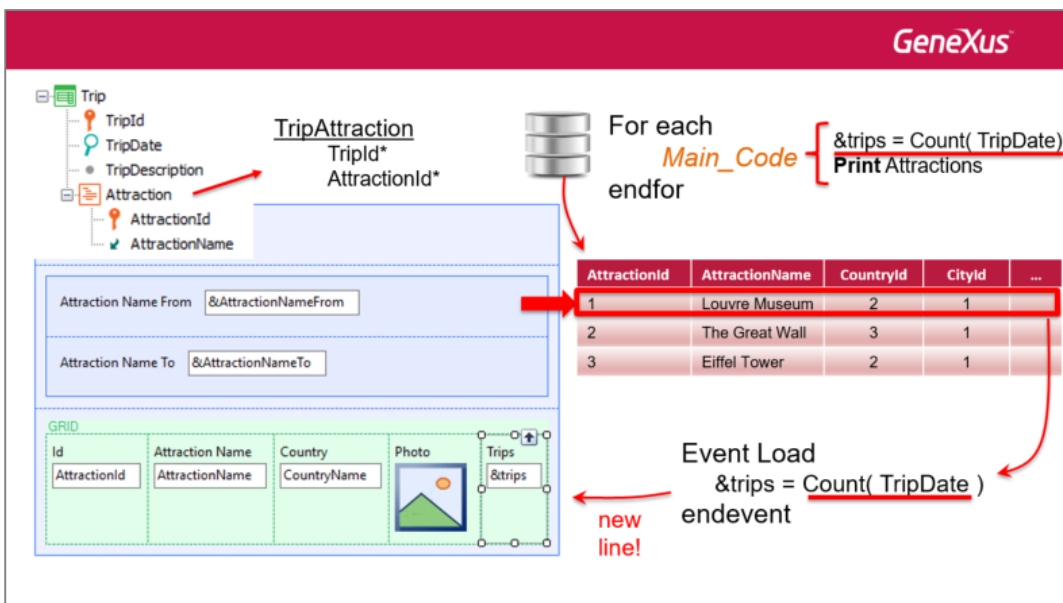
da tabela base do grid que cumpra com as condições de filtro, imediatamente antes que seja inserida a linha no grid.



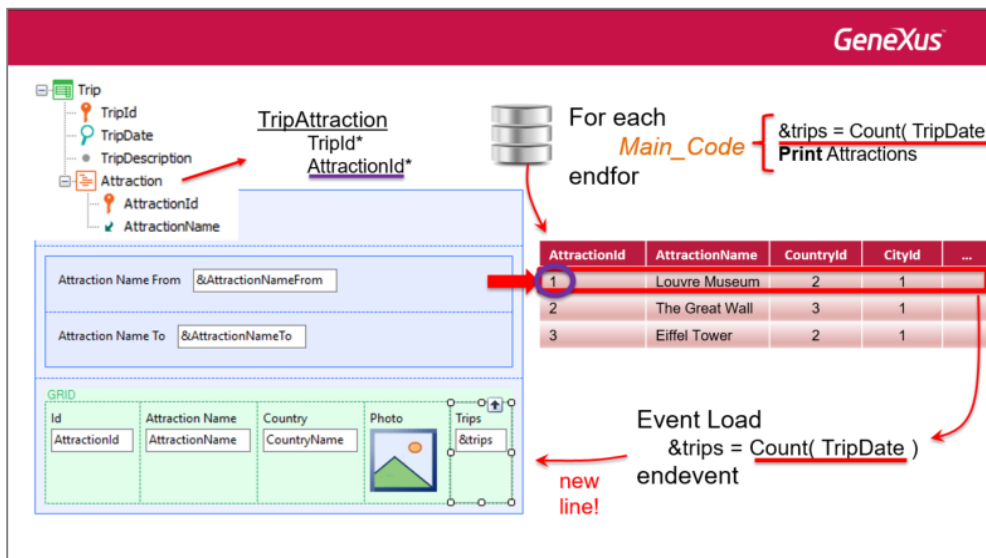
É por essa razão que ao ser executado o seu código sabe-se que estamos trabalhando com um registro da tabela base e sua estendida, e essa fórmula inline:



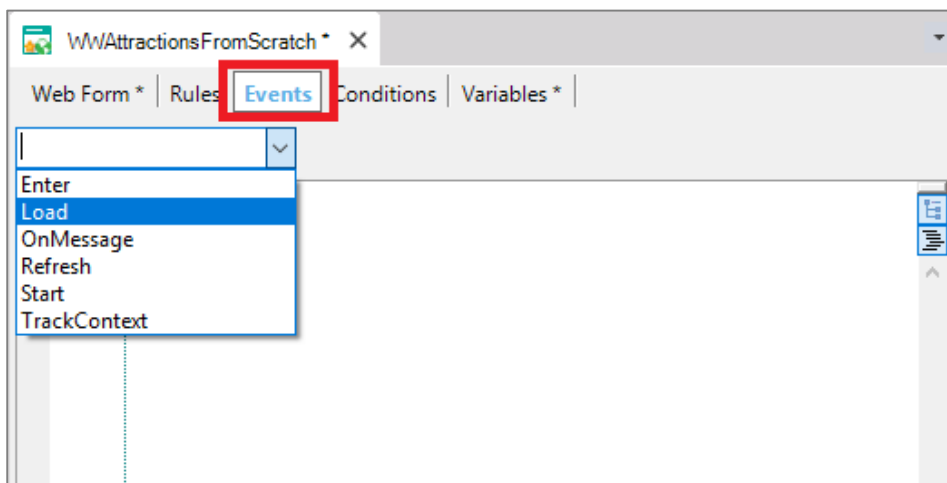
não vai contar todos os trips



só aqueles da tabela TripAttraction que correspondam a esse AttractionId, o do registro de Attraction que estamos a ponto de carregar no grid.



Implementemos em GeneXus. Já temos a transação Trip criada. Vamos até a seção de eventos da web panel. Nesse combo são disponibilizados os eventos predefinidos, ou seja, os eventos do sistema que são produzidos em momentos específicos, nos quais poderemos programar código.



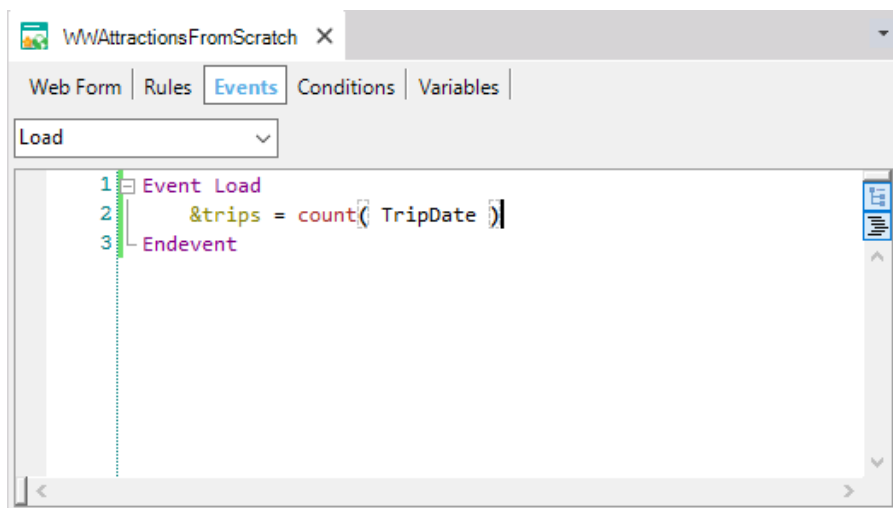
Entre eles, temos o evento Load. Ao escolhê-lo, aparece esse código:

```

Event Load
|
Endevent

```

Aqui dentro programaremos o que queremos que seja executado a cada vez que se está posicionado em um registro da tabela Attraction, antes de carregar a linha no grid. Em nosso caso...



Executemos... Terá que reorganizar a base de dados para adicionar as tabelas correspondentes a nova transação Trip.

Aqui mostramos a execução com um par de trips já cadastrados.

Trip

« < > » SELECT

Id: 1

Date: 09/12/16

Description: A

Attraction

Attraction Id	Attraction Name
24	Eiffel Tower
25	Christ the Redemmer
27	Matisse Museum

0

Trip

« < > » SELECT

Id: 2

Date: 09/13/16

Description: B

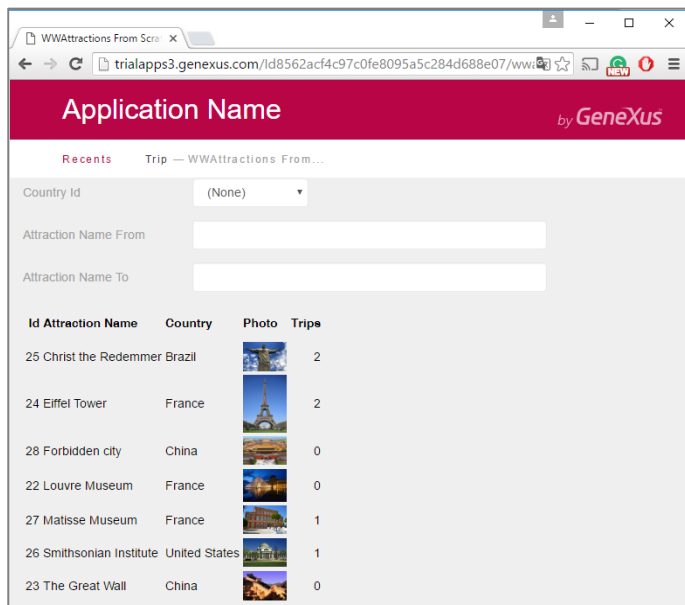
Attraction

Attraction Id	Attraction Name
24	Eiffel Tower
25	Christ the Redemmer
26	Smithsonian Institute

0

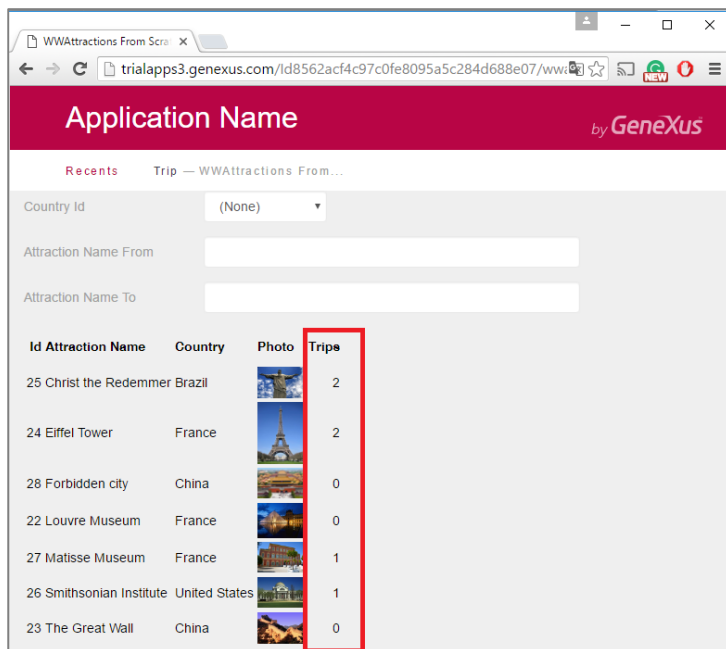
Vemos que a torre Eiffel está em um par de Trips, o Cristo Redentor também, Smithsonian Institute em um, e o museu Matisse também em um.

Agora executemos nossa web panel.

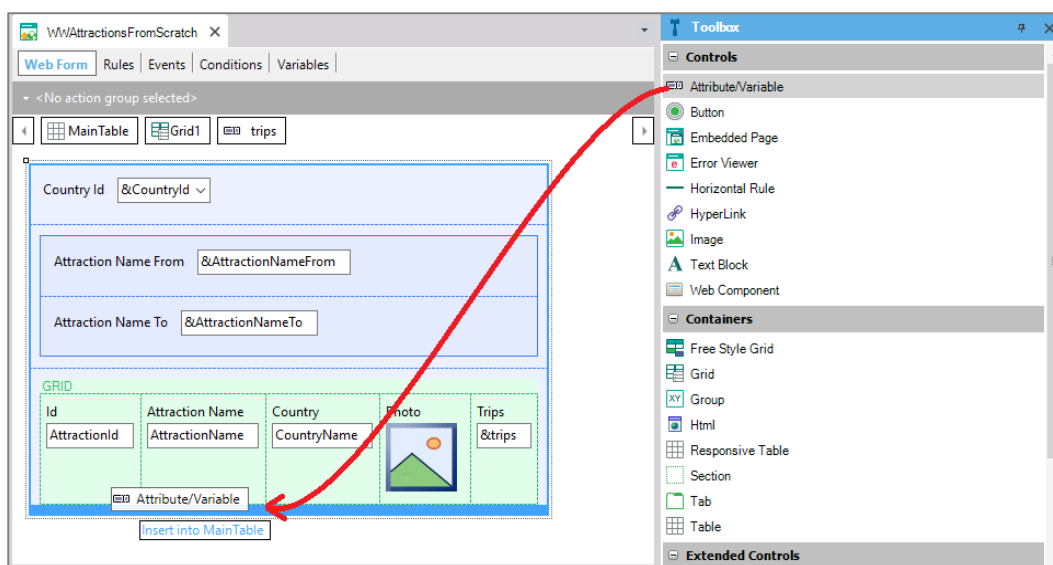


E vemos que está mostrando corretamente o que queríamos.

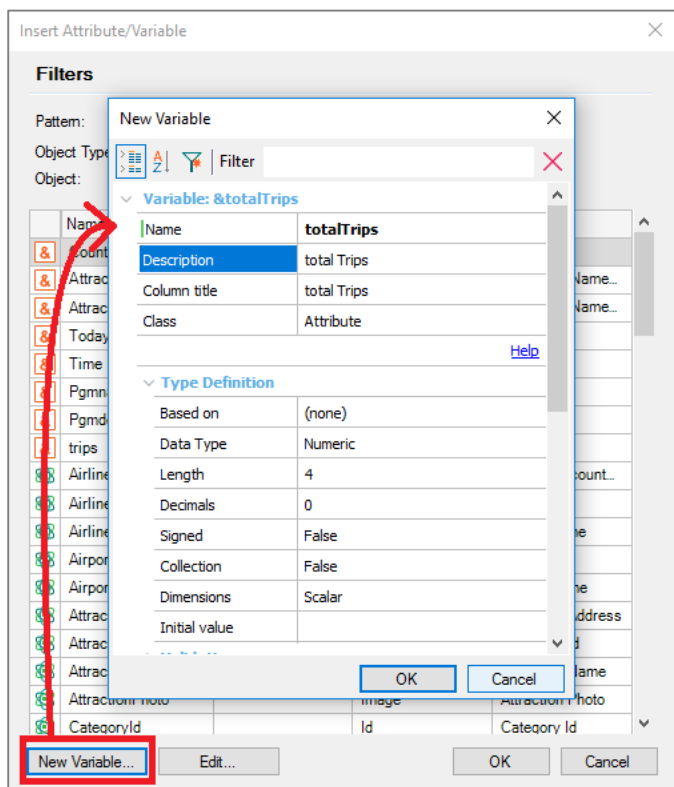
Agora queremos adicionar a soma de excursões nas quais as atrações que são mostradas no grid em cada oportunidade estão incluídas. Isso é, a soma dos valores dessa coluna:



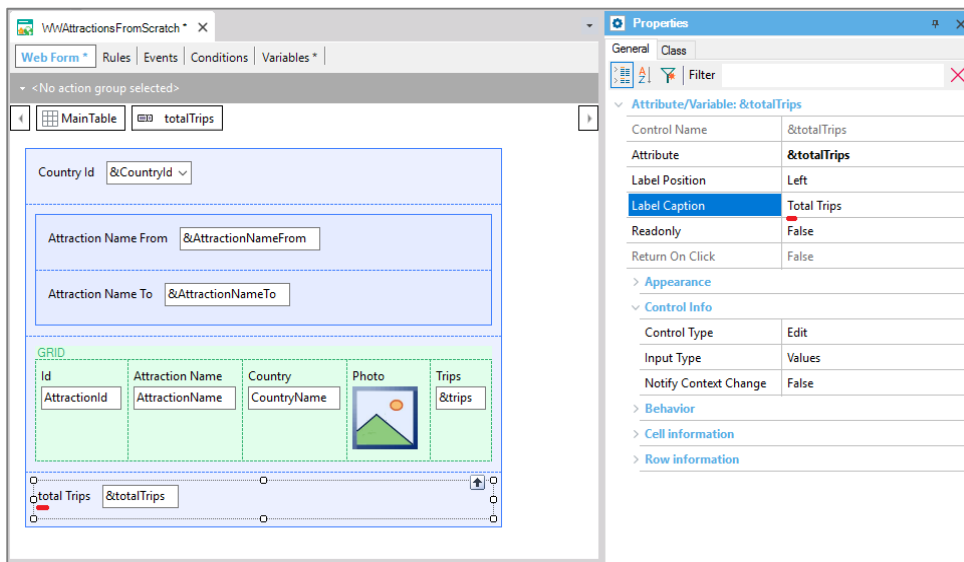
Para isso, adicionamos uma variável fora do grid:



Colocamos o nome de totalTrips:



Alteramos a propriedade **Label Caption** para que sua etiqueta apareça em maiúsculas, já que o nome possuímos em minúsculas:



Um forma eficiente de calcular o valor exibido pelas variáveis é... a cada vez que uma linha for carregada no grid, somar o valor da variável &Trips dessa linha ao valor calculado até o momento em &totalTrips.

No evento Load, após o cálculo do valor da variável &trips, atribuir a variável &totalTrips o valor que contém até o momento mais o valor da variável &trips:

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

Para a primeira linha a ser carregada no grid, será executado pela primeira vez o evento Load, calcula-se &Trips e quando for calculada &totalTrips, o valor atual estará em zero,

GeneXus

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

0

assim que nesse caso &totalTrips assumirá o valor de &Trips.

GeneXus

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

Country Id &CountryId v

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

Id	Attraction Name	Country	Photo	Trips
AttractionId	AttractionName	CountryName		&trips

Total Trips &totalTrips

Para a segunda linha a ser carregada, calcula-se o valor de &trips e &totalTrips conterá o valor anterior

GeneXus

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 1

&totalTrips 2

Country Id &CountryId v

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

Id	Attraction Name	Country	Photo	Trips
AttractionId	AttractionName	CountryName		&trips

Total Trips &totalTrips

ao que será somado ao valor da variável &trips para essa segunda linha, e assim sucessivamente.

GeneXus

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 1

&totalTrips 3

Country Id &CountryId v

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

Id	Attraction Name	Country	Photo	Trips
AttractionId	AttractionName	CountryName		&trips

Total Trips &totalTrips

Logo que seja carregada a última linha, a variável &totalTrips terá o valor desejado.

GeneXus

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 1

&totalTrips 3

Country Id &CountryId

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

Id	Attraction Name	Country	Photo	Trips
AttractionId	AttractionName	CountryName		&trips

Total Trips &totalTrips

Executemos para testar.

Application Name by GeneXus

Country Id (None)

Attraction Name From

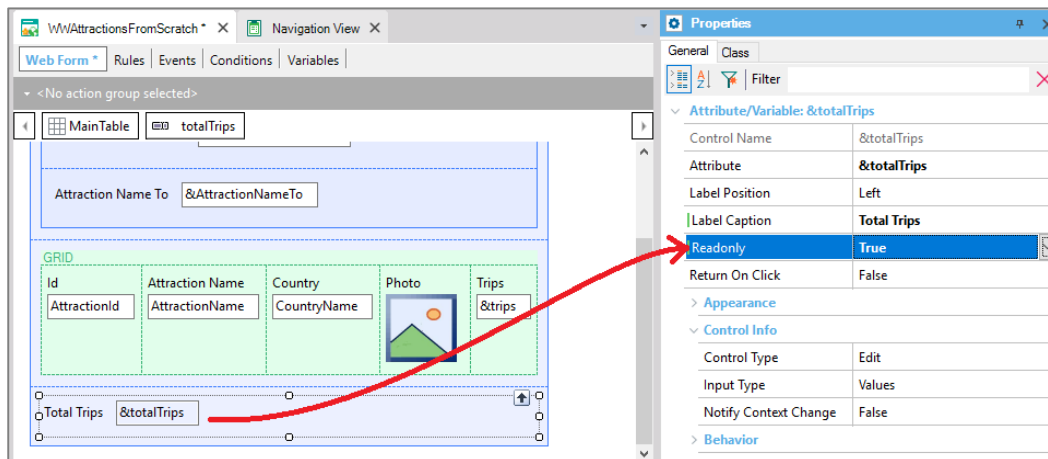
Attraction Name To

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

Total Trips
6

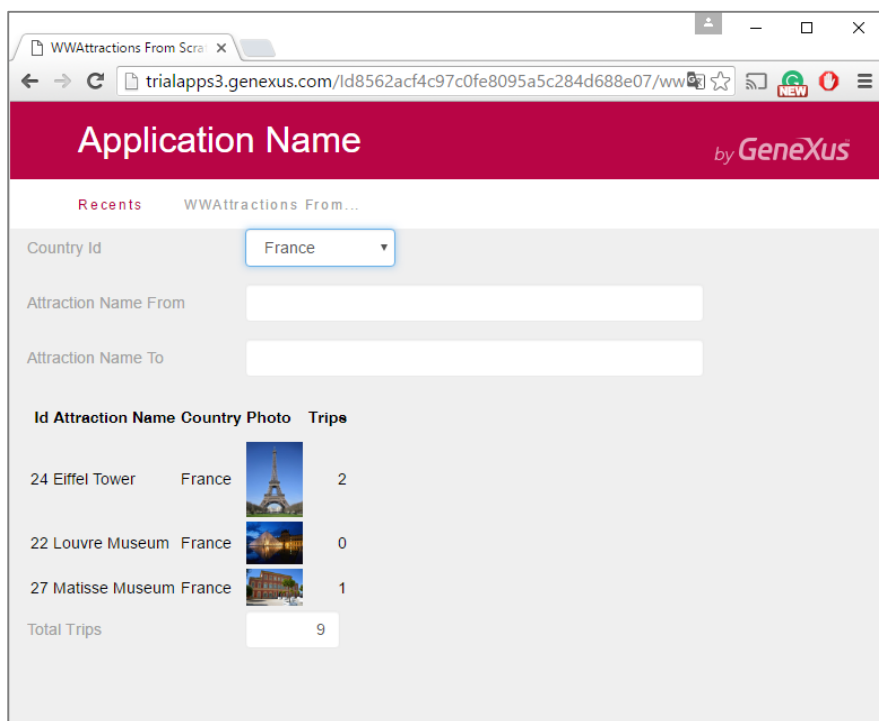
Podemos ver duas coisas: primeiro, que está somando corretamente; segundo: por tratar-se de uma variável, é de entrada, pelo que o usuário pode alterar o valor, o que não tem sentido. Assim, em primeiro lugar, configuraremos a propriedade Readonly.

Para isso nos posicionamos sobre a variável no form e entre suas propriedades, alteramos a ReadOnly para True:



Pode chamar a nossa atenção, também, que &trips também é uma variável, está sendo exibida como readonly, e porém, não fizemos nada para isso. As variáveis nos grids, quando não se tem programado eventos no nível das linhas, e nem são percorridas com for each line via código, serão readonly. Voltaremos a esse tema logo.

Mas também, vejamos o que passa se filtrarmos, por exemplo, por França.



No lugar de mostrarmos um total de 3, está sendo mostrado 9, que é a soma do valor que era mostrado antes, 6, mais os 3 que agora deveria estar sendo mostrado. Por que aconteceu isso?

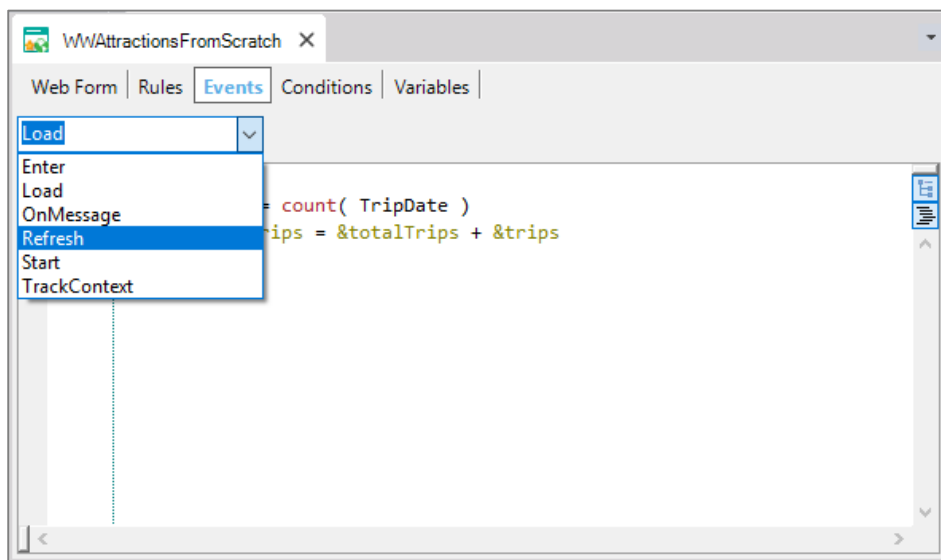
Ao alterar uma das variáveis de filtro, a web panel voltou a carregar o grid. Isso é, voltou a consultar a tabela

Attraction da base de dados, e voltou a executar o evento **Load** para cada registro que cumpra os filtros. O problema é que a variável `&totalTrips` teria que ser zerada, antes de lançar a carga do grid.

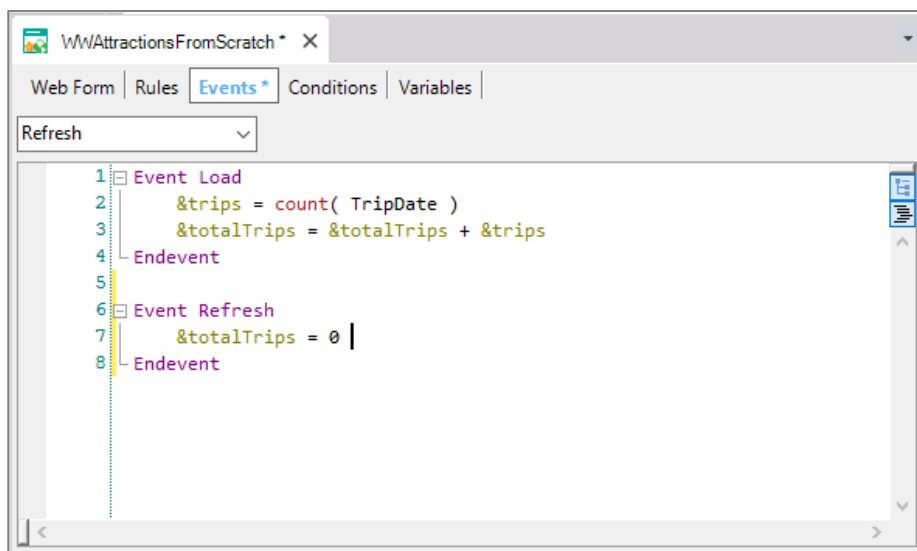
Onde fazemos isso? No **evento Refresh**.

Esse evento do sistema é produzido sempre que a web panel é carregada, imediatamente antes de ir a base de dados buscar a informação para carregar no grid. Isso é, imediatamente antes de executar os eventos Load para cada linha a ser carregada.

Assim, vamos até a aba de eventos, e escolhemos no combo o Refresh...



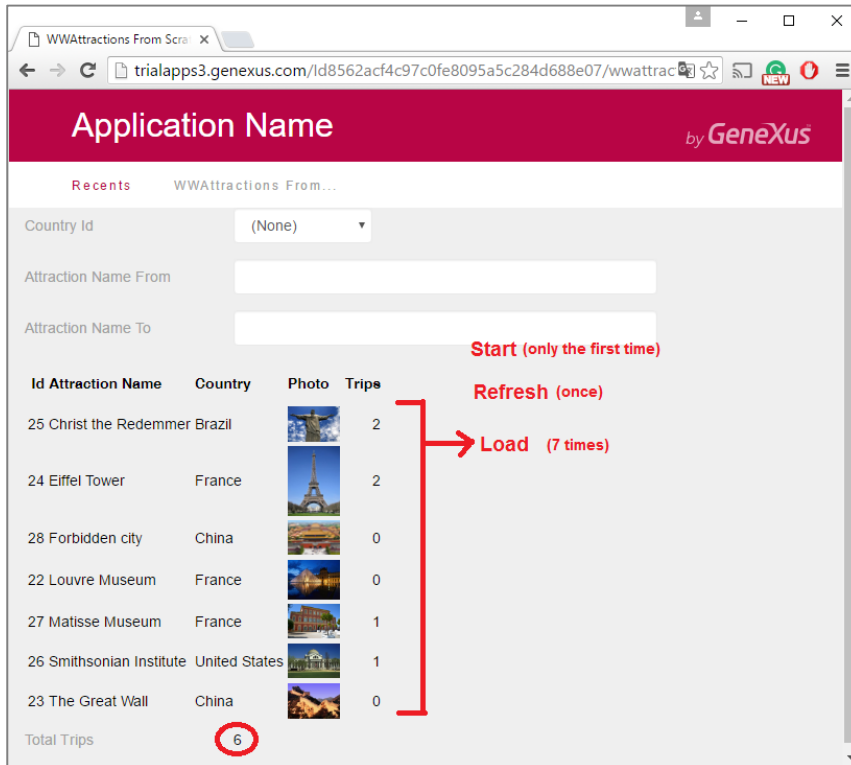
Esse é o momento para colocar zero na variável.



Observemos que a ordem em que os eventos ficam escritos não tem a menor importância. Aqui somente indica-se o código que será executado ao produzir cada um deles.

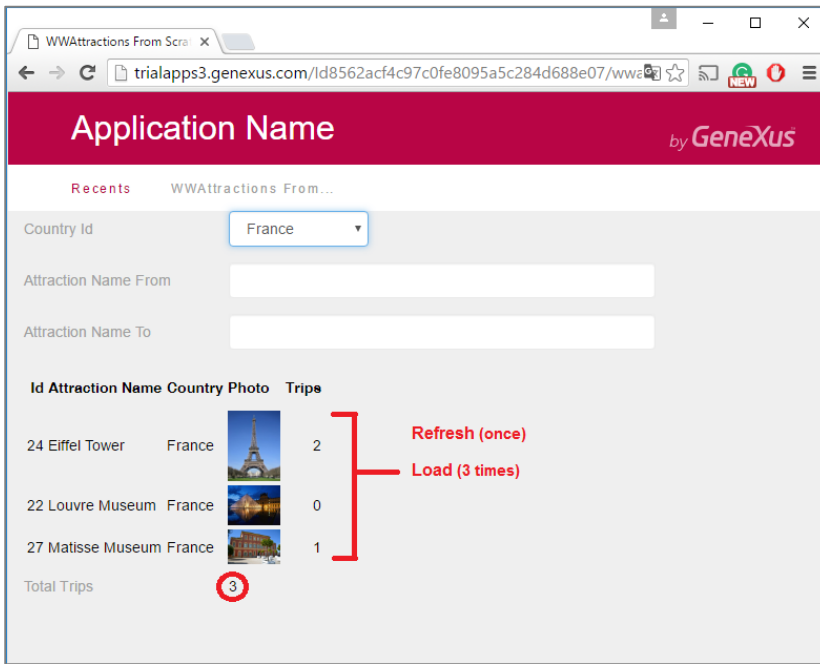
Pressionemos F5.

Podemos ver que agora o total de trips aparece Readonly. Ao executar esta web panel pela primeira vez, são disparados três eventos de forma consecutiva: o **evento Start**, que executará somente ao ser aberta a web panel, na primeira vez, o **evento Refresh**, que põe a variável em zero, e o **evento Load**, tantas vezes forem as linhas carregadas no grid. Nesse caso foram 7.

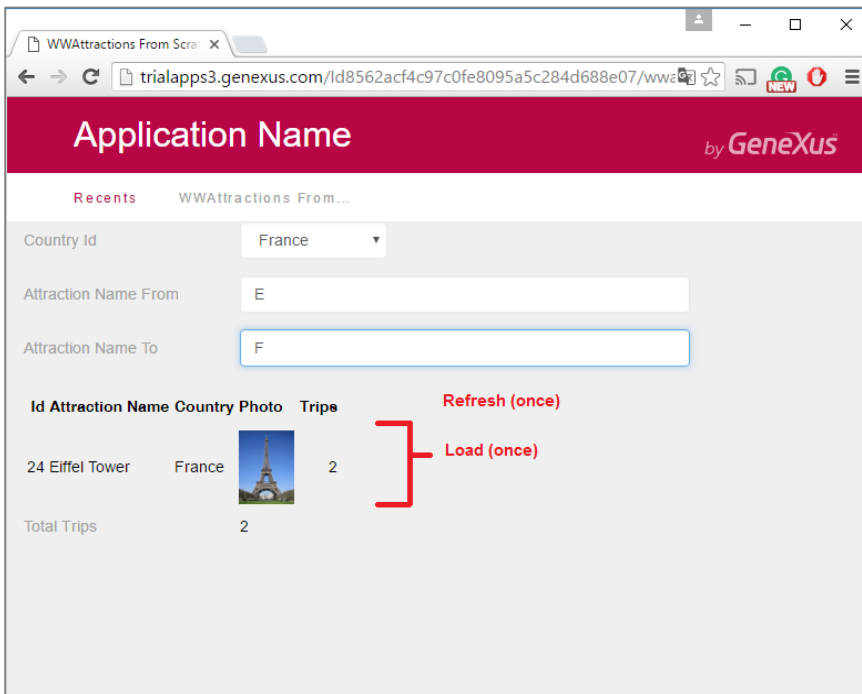


Agora, se escolhemos filtrar por um país, por exemplo, França, visualizamos que está sendo calculado bem o número de trips.

Ao mudar o valor de uma variável que tem efeitos sobre as condições que devem cumprir os registros para carregar-se no grid, volta-se a disparar o evento **Refresh** (e portanto a variável &totalTrips volta a ficar com zero) e a ida a base de dados para voltar a filtrar e carregar os registro no grid. Portanto o **Load** volta a ser disparado para cada atração da França a ser carregada.

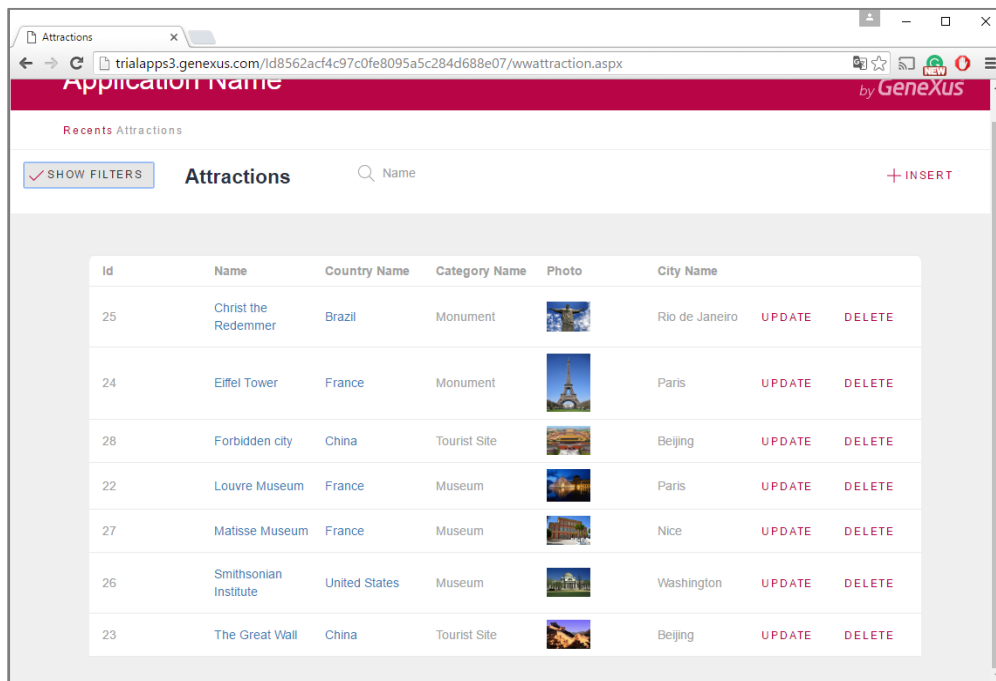


Se agora escolhemos ver as atrações entre E e F, também volta-se a atualizar a tela. Volta-se a colocar em zero a variável &totalTrips no Refresh, e volta-se a ler a tabela base do grid e sua estendida, desta vez recuperando apenas um só registro, pelo que o Load será executado uma única vez.

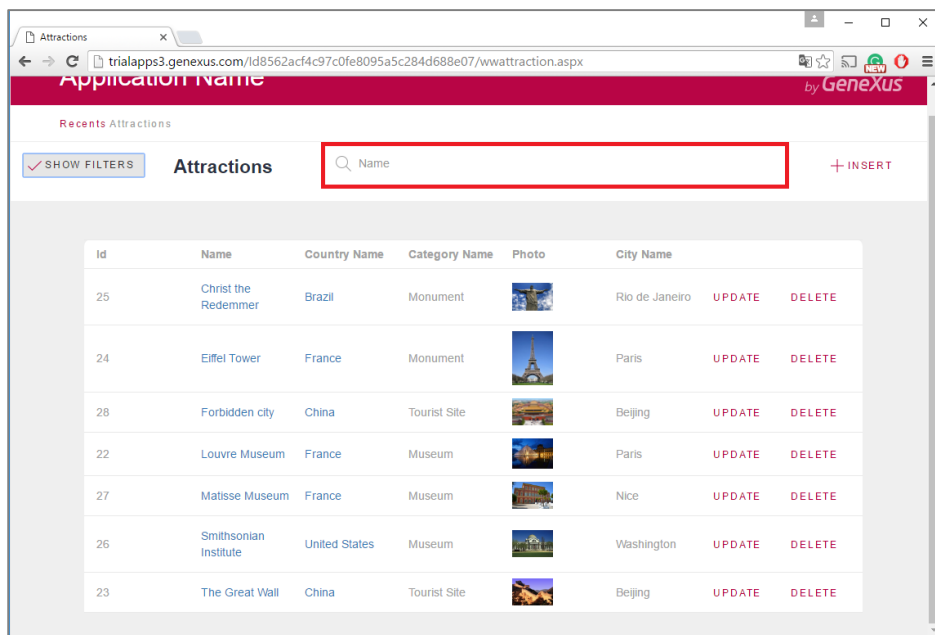


Agora bem, se observarmos a web panel que temos implementada até o momento, poderemos apreciar que vai aparecendo o objeto que havia criado o Pattern WorkWith aplicado na transação Attraction.

Esse objeto era, evidentemente, uma web panel.



Aqui filtrava-se por uma única variável:

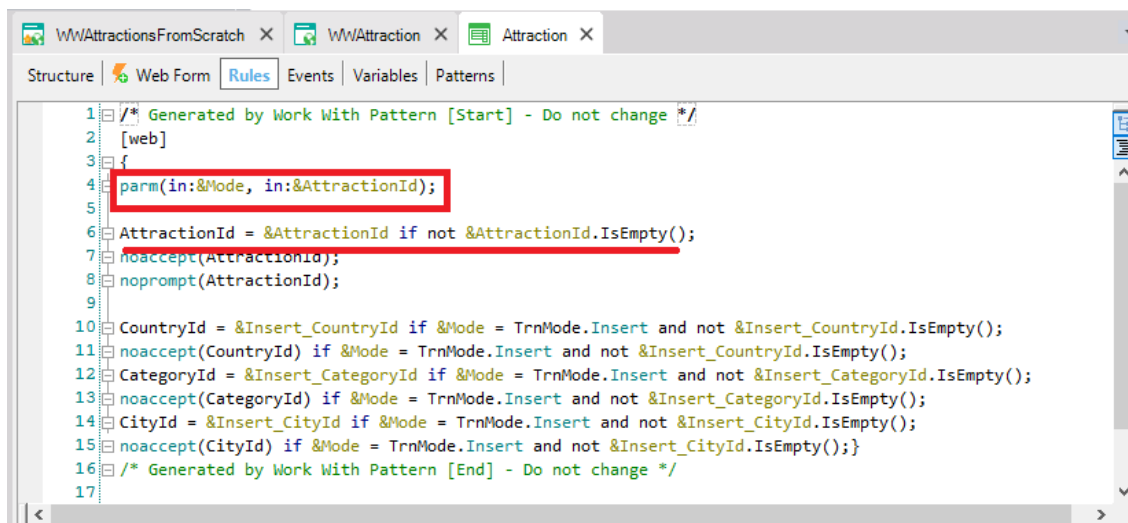


Mas, o mais interessante é que além de permitir filtrar os dados do grid, o Work With oferece executar ações

sobre os dados. Por exemplo, poder atualizar os dados de uma atração ou eliminar a atração assim como inserir uma nova.

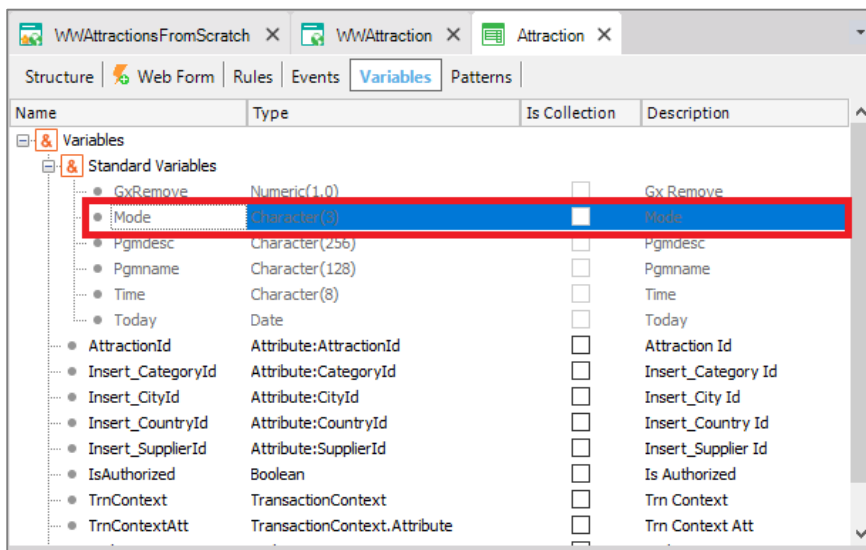
Para isso o pattern inseriu dois controles no nível das linhas do grid, e um fora. Em qualquer dos três casos, a ação associada a cada controle consiste em chamar a transação Attraction, enviando como parâmetro o **modo** em que deve ser aberta a transação, isto é, se é chamada para atualizar, eliminar ou inserir. Nos dois primeiros, Update ou Delete como corresponderão a eventos de uma linha contaremos com o id da atração da linha que será enviado como segundo parâmetro na transação, de modo a atualizar os dados **dessa** atração, ou eliminar **essa** atração. No caso do controle Insert fora do grid, será enviado **0** como segundo parâmetro, já que as atrações no Insert são autonumeradas.

É por isso que o pattern alterou a transação Attraction, adicionando, entre outras coisas, a regra Parm:



```
1 /* Generated by Work With Pattern [Start] - Do not change */
2 [web]
3 {
4   parm(in:&Mode, in:&AttractionId);
5 }
6 AttractionId = &AttractionId if not &AttractionId.IsEmpty();
7 noaccept(AttractionId);
8 noprompt(AttractionId);
9
10 CountryId = &Insert_CountryId if &Mode = TrnMode.Insert and not &Insert_CountryId.IsEmpty();
11 noaccept(CountryId) if &Mode = TrnMode.Insert and not &Insert_CountryId.IsEmpty();
12 CategoryId = &Insert_CategoryId if &Mode = TrnMode.Insert and not &Insert_CategoryId.IsEmpty();
13 noaccept(CategoryId) if &Mode = TrnMode.Insert and not &Insert_CategoryId.IsEmpty();
14 CityId = &Insert_CityId if &Mode = TrnMode.Insert and not &Insert_CityId.IsEmpty();
15 noaccept(CityId) if &Mode = TrnMode.Insert and not &Insert_CityId.IsEmpty();
16 /* Generated by Work With Pattern [End] - Do not change */
17
```

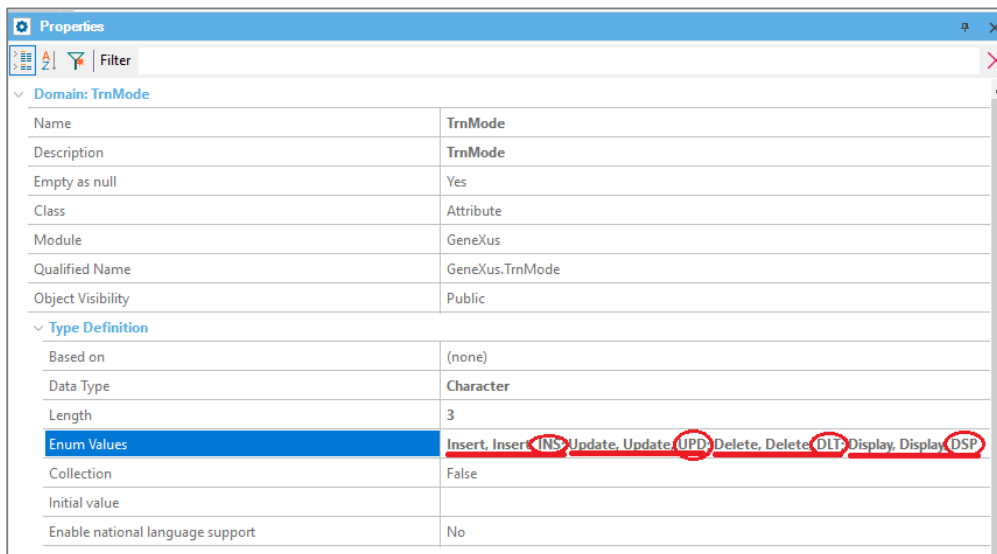
que como vemos, recebe duas variáveis: a variável &Mode é uma variável padrão nas transações, Character de 3:



Name	Type	Is Collection	Description
Standard Variables			
GxRemove	Numeric(1,0)		Gx Remove
Mode	Character(3)		Mode
Pgmdesc	Character(256)		Pgmdesc
Pgmname	Character(128)		Pgmname
Time	Character(8)		Time
Today	Date		Today
AttractionId	Attribute:AttractionId		Attraction Id
Insert_CategoryId	Attribute:CategoryId		Insert_Category Id
Insert_CityId	Attribute:CityId		Insert_City Id
Insert_CountryId	Attribute:CountryId		Insert_Country Id
Insert_SupplierId	Attribute:SupplierId		Insert_Supplier Id
IsAuthorized	Boolean		Is Authorized
TrnContext	TransactionContext		Trn Context
TrnContextAtt	TransactionContext.Attribute		Trn Context Att

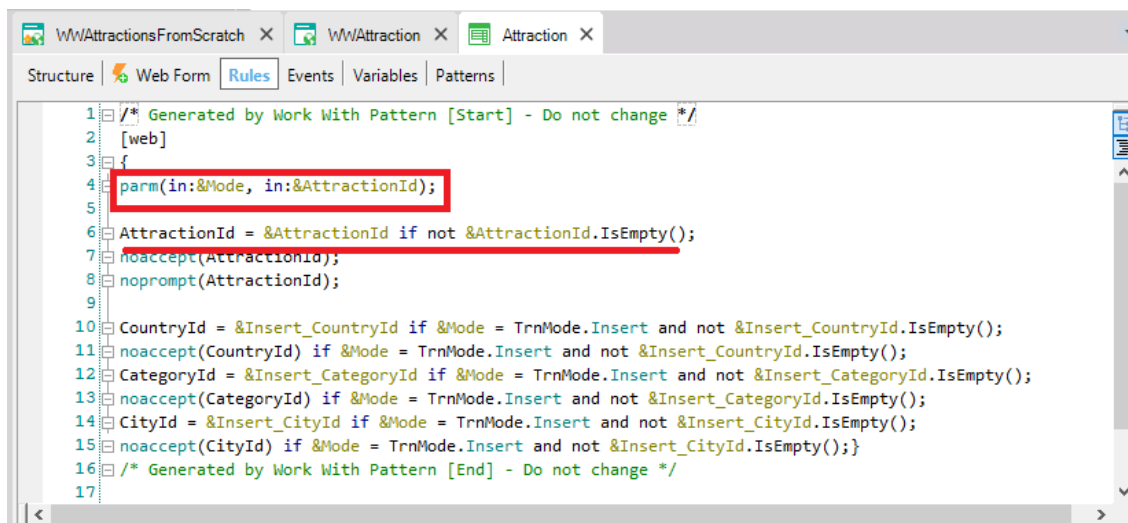
que aceita um de quatro valores especificados no domínio enumerado TrnMode:

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



Recebendo um desses quatro valores, a transação saberá em que modo deve ser aberta.

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

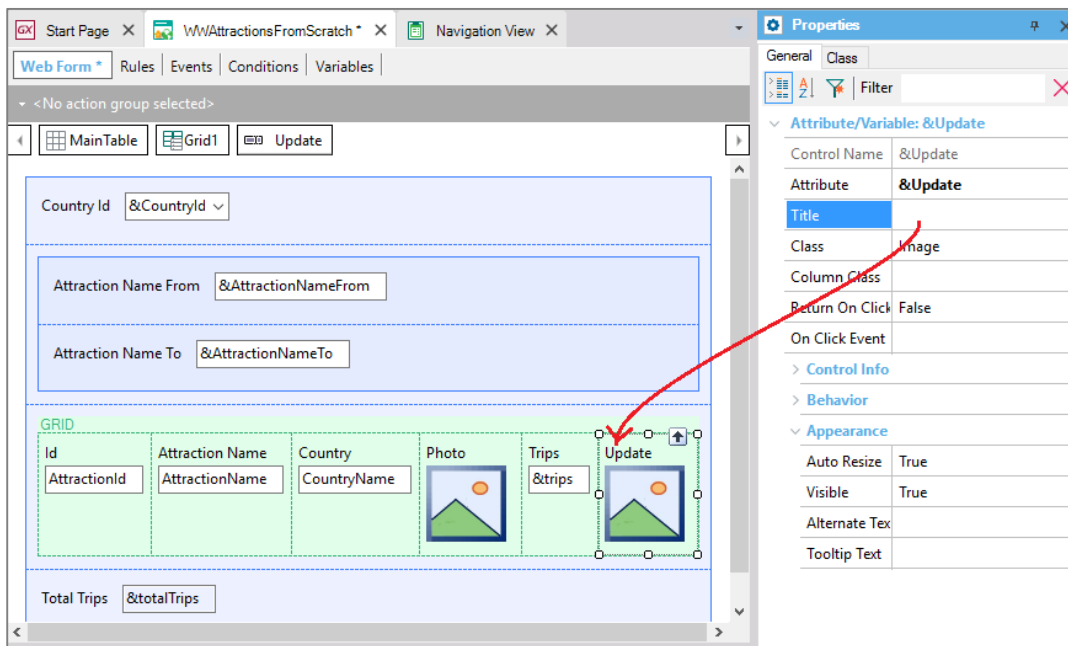


Em nossa web panel implementaremos uma dessas ações sobre as atrações. Por exemplo, a de Update. A intenção é mostrar um exemplo de ações sobre os dados.

Teremos que inserir um controle no grid. No caso do pattern é inserida uma variável character chamada update,

Chamamos a imagem de updateIcon.

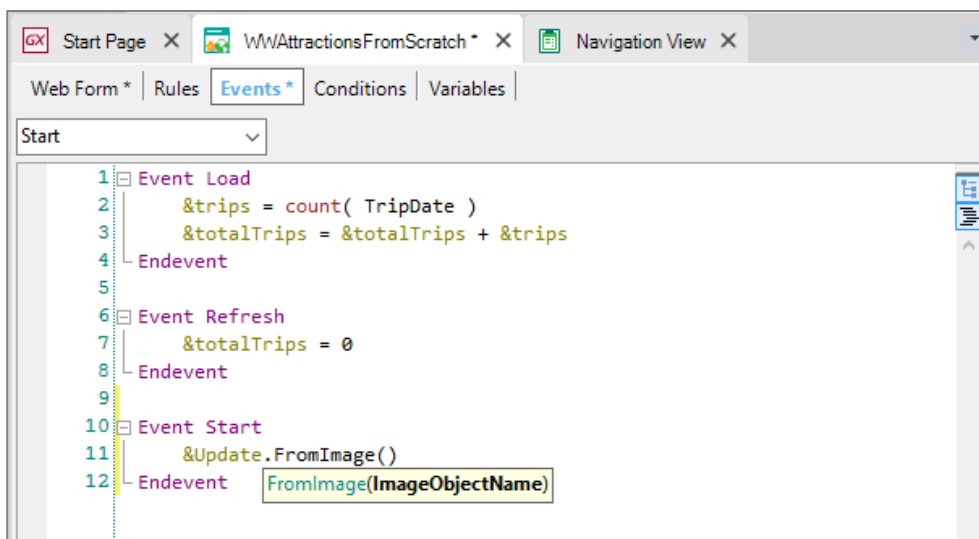
Agora vamos na web panel e arrastamos da toolbox o controle Attribute/Variable até a última coluna do grid, definimos a nova variável como &Update, mas não do tipo character, e sim do tipo Image:



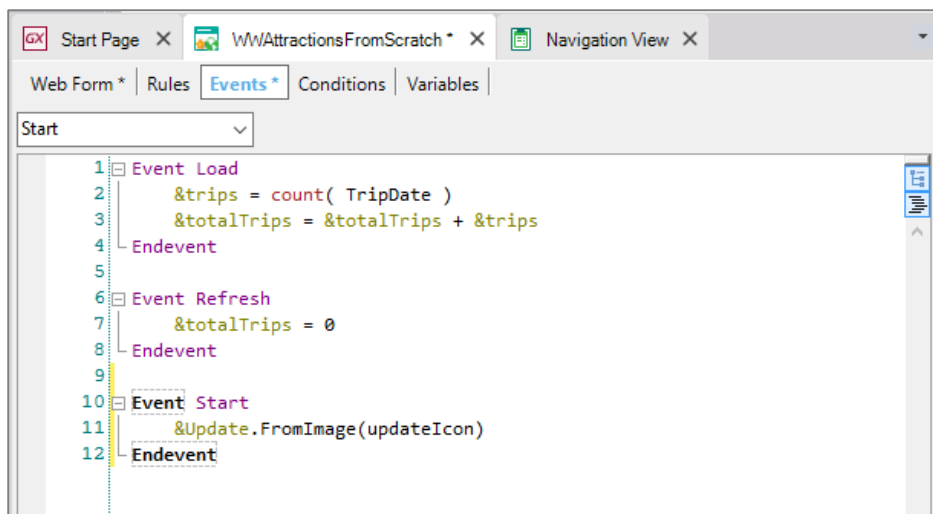
Retiramos o título para que não figure como título da coluna, e nos resta carregar essa variável com a imagem que acabamos de inserir na KB. Onde fazemos?

Se a imagem fosse variar por linha do grid, faríamos no evento Load, mas a imagem será a mesma para cada linha, e não irá variar nunca, assim que uma boa opção é fazer no **evento Start**, que será executado uma única vez, quando a web panel é aberta, e não mais.

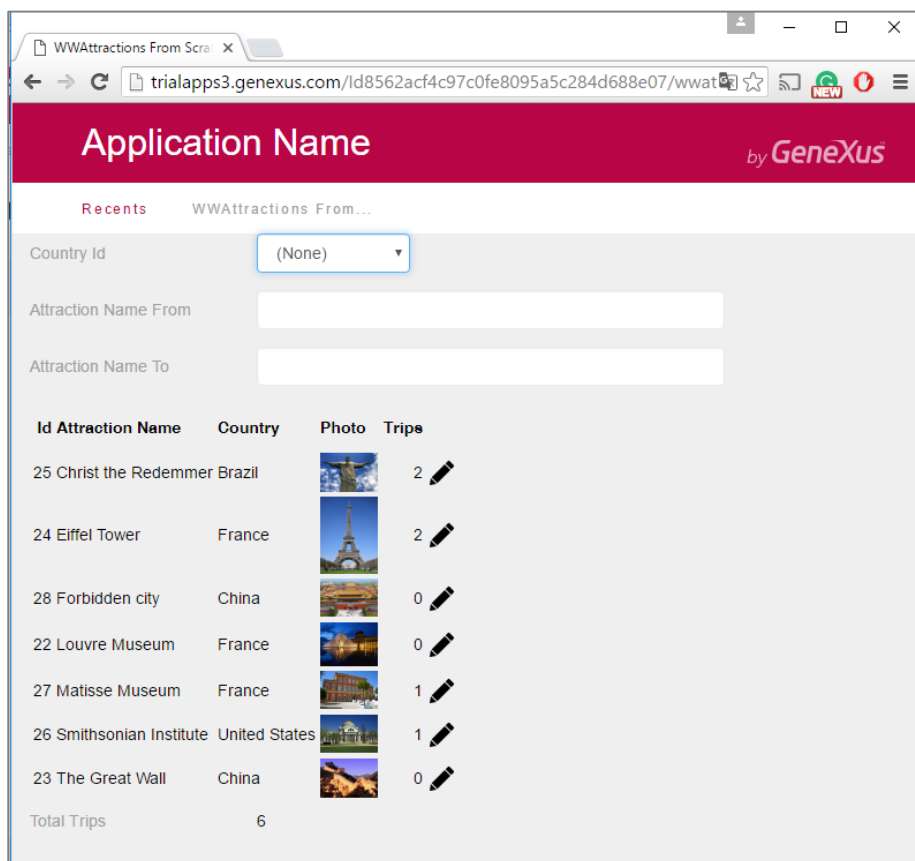
Assim que inserirmo no evento Start, e ali escrevemos:



E como parâmetro do método, escrevemos o nome da imagem da KB, isto é, updateIcon.



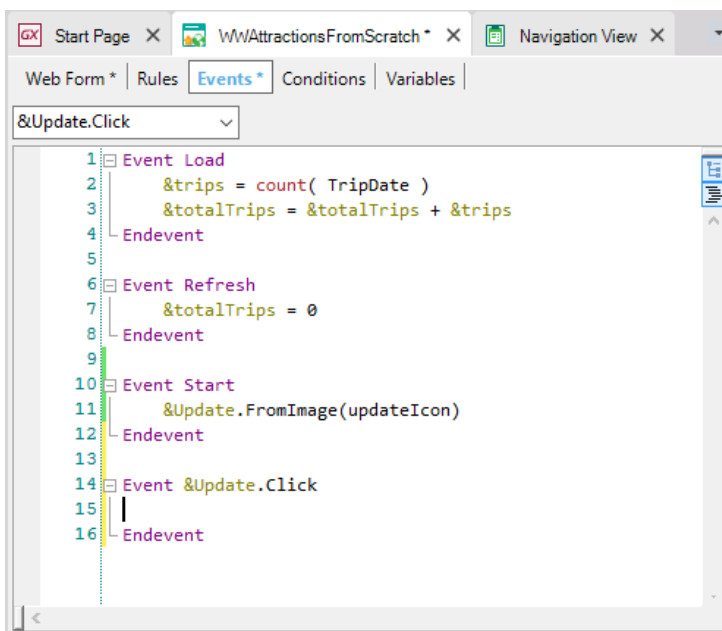
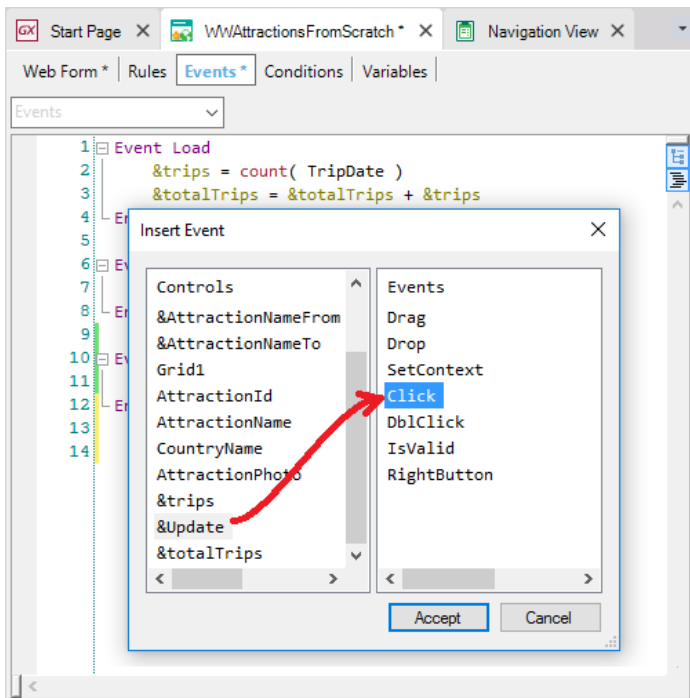
Vejamos em execução.



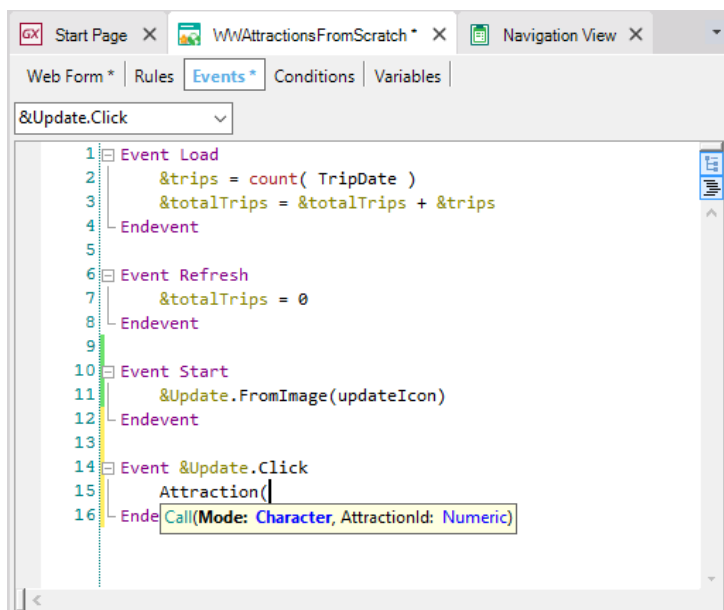
Agora nos resta associar um evento a essa imagem, de tal maneira que quando o usuário clique sobre ela, produza esse evento e execute o seu código, no qual chamaremos a transação Attraction.

Há várias alternativas para realizar isso. Uma delas é ir até a aba de eventos e com Insert/Event vemos à

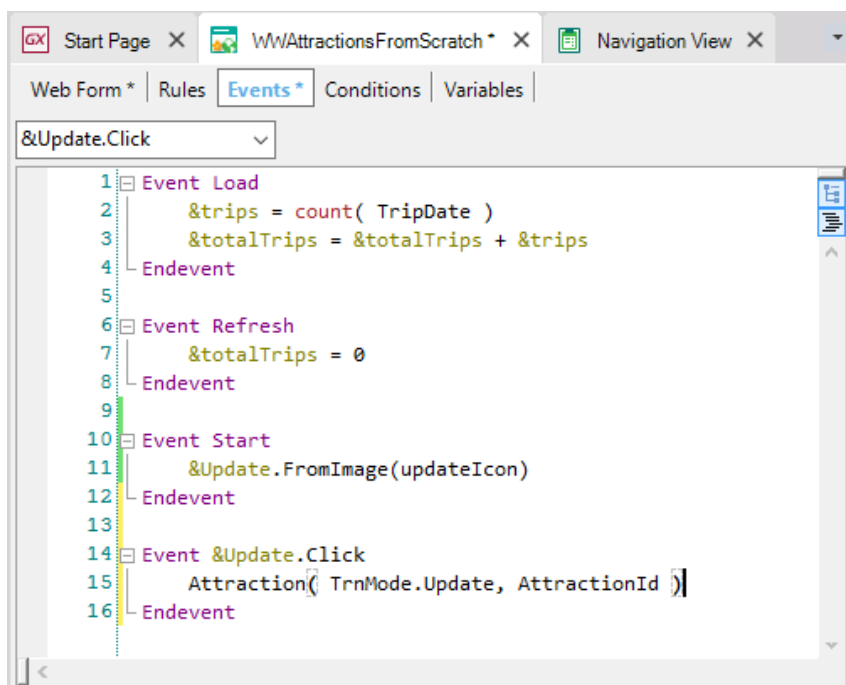
esquerda todos os nomes dos controles que temos inseridos no form. Escolhemos o que desejamos, a variável &Update, e vemos que à direita é mostrado os eventos que podem ser associados, por exemplo, o evento click:



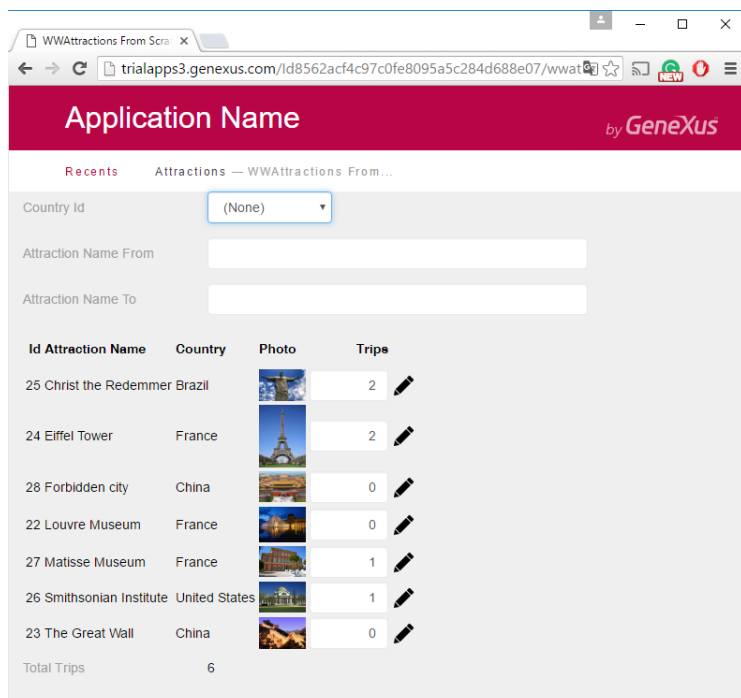
Desse modo, quando o usuário clique sobre a imagem para uma linha, executará o código que escrevemos dentro desse evento. O que queremos fazer nesse caso é chamar a transação Attraction:



Passando o modo Update, isto é, o valor Update do domínio enumerado TrnMode que vimos antes, e o valor de AttractionId correspondente a linha do grid onde foi clicada:

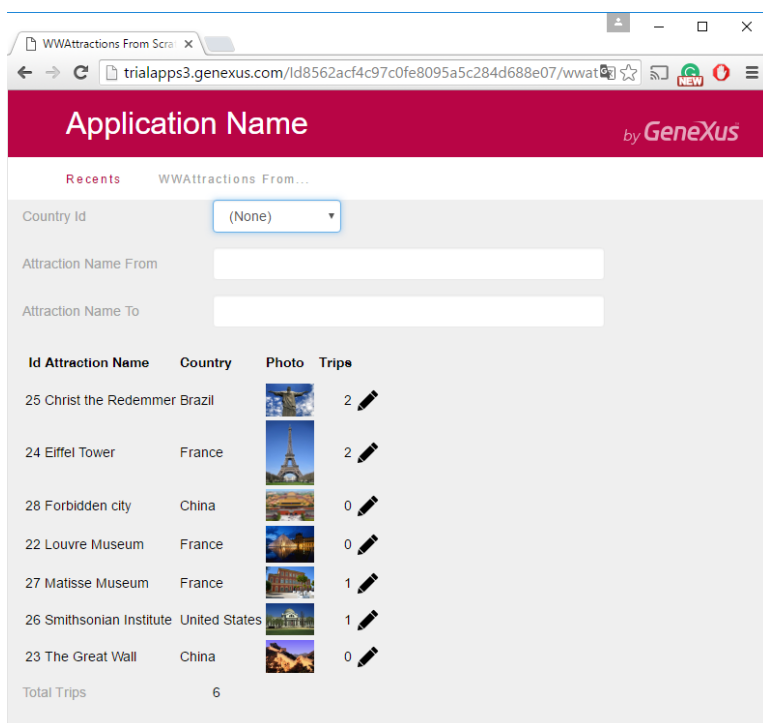


Executemos para testar.

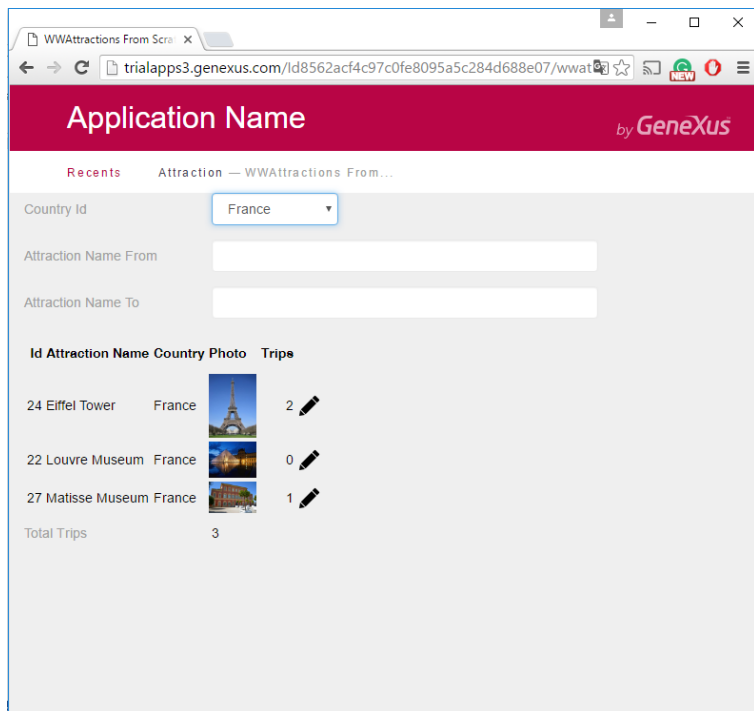


Primeiro observamos que a coluna da variável &Trips aparece agora como editável. Não havíamos definido como Readonly explicitamente porque ao executar vimos que já estava. Como falamos antes, as variáveis do grid em princípio são colocadas como readonly, exceção quando é definido algum evento no nível das linhas, como é o nosso caso, ou outras exceções que agora não veremos.

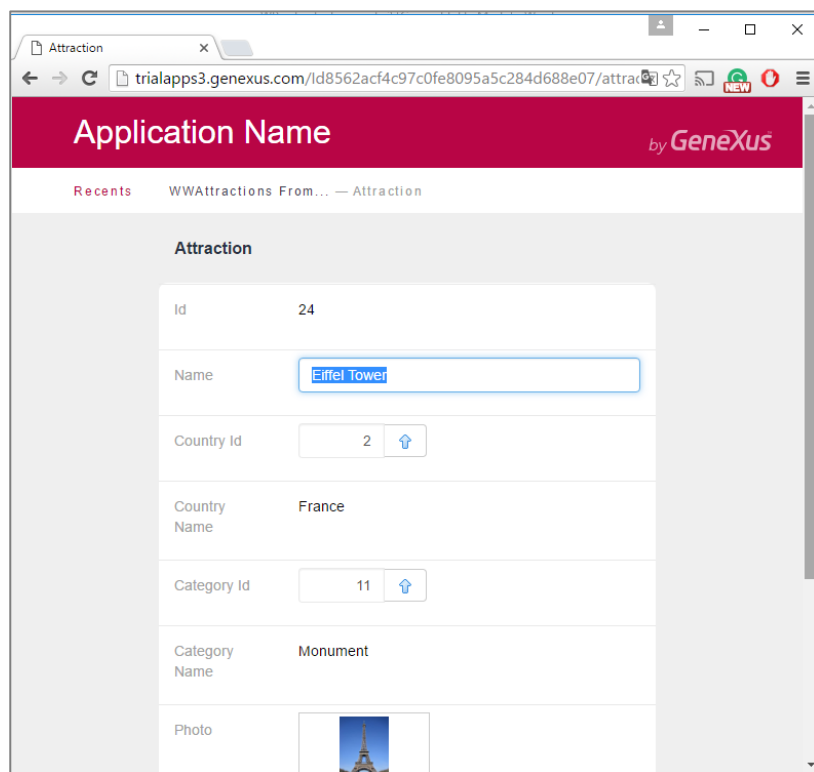
Configuremos como Readonly. Voltamos a executar.



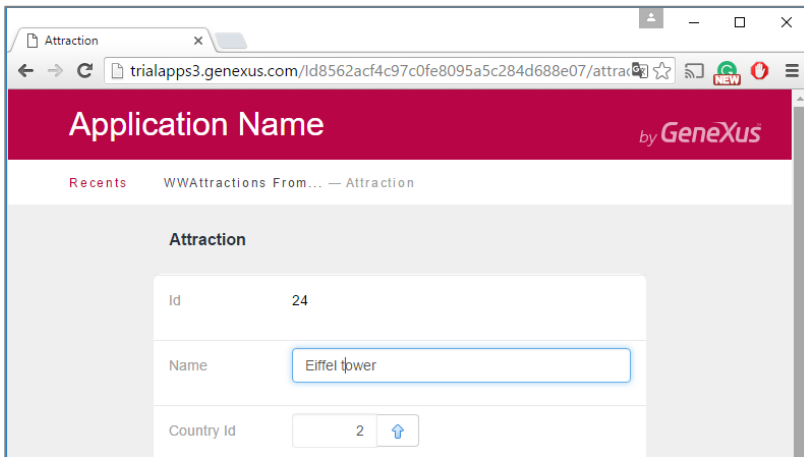
Selecionamos, por exemplo, o país França:



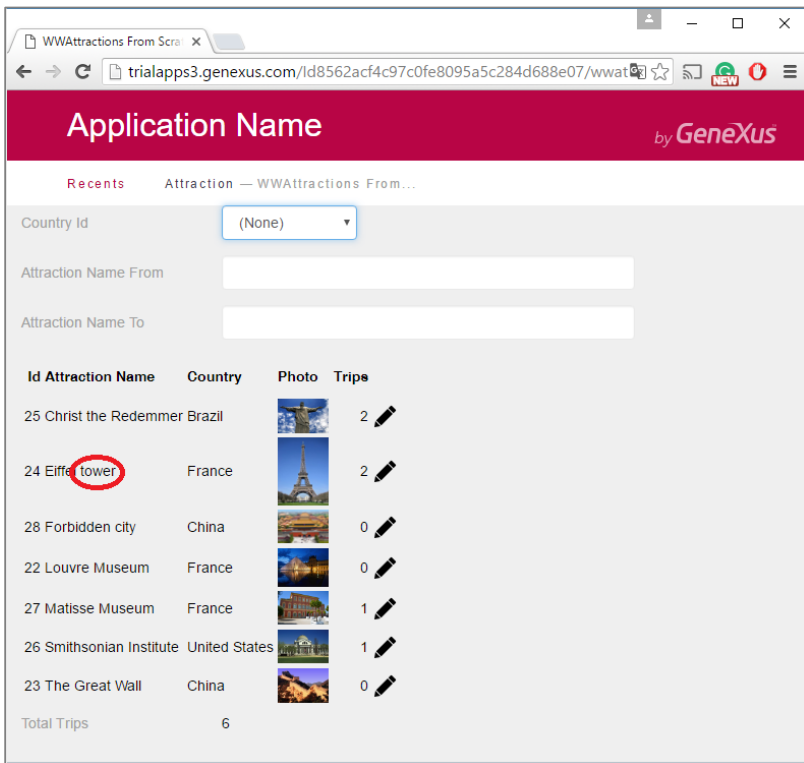
E agora clicamos sobre a imagem de update para a Torre Eiffel:



Vemos como é chamada a transação em update. Alteremos algo... por exemplo, passamos de maiúscula a minúscula a letra T de Tower:



Confirmamos... e como o pattern Work With, embora não o vimos, adicionou na transação o comando Return, retorno, para voltar ao objeto que a chamou, neste caso a web panel. Este comando Return é como chamar a web panel pela primeira vez, pelo que será executado o evento Start, seguido do Refresh e do Load tantas vezes quantos registros forem carregados:



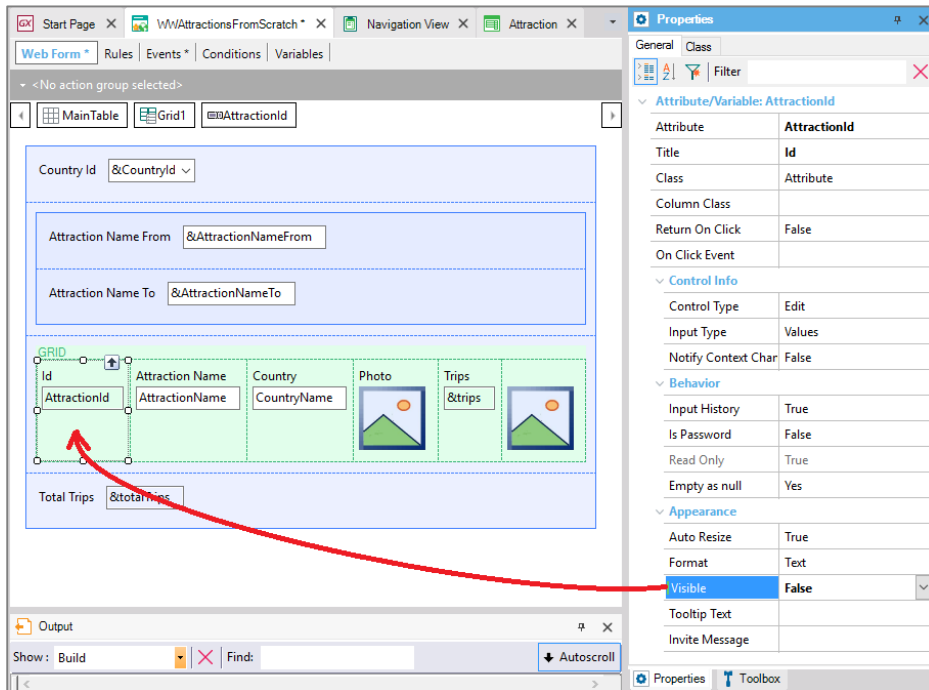
É por isso que vemos que ao voltar são carregadas todas as atrações, sem filtros.

Que passaria se não tivéssemos colocado o atributo AttractionId no grid? Ao clicar sobre a imagem para atualizar, que AttractionId seria enviado como parâmetro a transação? Não teria esse valor para enviar.

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


Como vamos usar um evento no nível das linhas, que será disparado depois que as linhas tenham sido carregadas, não podemos remover o `AttractionId` do grid. É que aqui já não se está mais na base de dados. O grid armazenou ao carregar-se com o Load todos os valores de suas colunas e nada mais. Um evento posterior trabalhará unicamente sobre os dados carregados no grid. Então, o que podemos fazer se não queremos ver essa coluna no grid é ocultá-la. Continuará estando presente, mas invisível.

Para isso utilizamos a **propriedade Visible com o valor False**:



Executamos.

No próximo vídeo veremos o que acontece com um evento no nível das linhas que altera o valor de uma variável do grid. Veremos uma web panel sem grid mas com atributos no form. Faremos um resumo conceitual de tudo o que foi visto, veremos web panels sem tabela base, e mencionaremos mais casos avançados de uso de web panels e suas características.

