

Telas interativas

Objeto Web panel (Continuação)

GeneXus® 16

Outra ação no nível das linhas: uma que atualiza a linha

The screenshot shows a GeneXus Web Panel interface. At the top, there are input fields for "Country Id" (dropdown) and "Attraction Name From" (&AttractionNameFrom). Below these are two more input fields: "Attraction Name To" (&AttractionNameTo) and "Total Trips" (&totalTrips). A grid section is labeled "GRID" and contains columns for "Id" (AttractionId), "Attraction Name" (AttractionName), "Country" (CountryName), "Photo" (image icon), "Trips" (&trips), and "New Trip" (&newTrip). A red arrow points from the text "Character(10)" to the "&newTrip" field in the grid. To the right of the grid, there is descriptive text: "Criar nova viagem no BD, com a atração da linha" followed by a database icon. Below the grid, there are two event blocks:

```

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

```

Event &newTrip.Click
 /* Chamar uma proc que insere viagem (trip) e devolve
 quantidade de trips de AttractionId */
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?

Outra ação no nível das linhas: uma que atualiza a linha

```

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

```

```

1 new
2   TripDate = Today()
3   TripDescription = "Created automatically from WnAttractionsFromScratch"
4 endnew
5 &tripid = TripId
6 new
7   TripId = &tripid
8   AttractionId = &AttractionId
9 endnew
10 &trips = Count( TripDate, AttractionId = &AttractionId )
11
12

```

Por exemplo, chamar um procedimento ao qual passemos o identificador da atração da linha e cria a trip com essa atração.

Observe que o implementamos com o comando **new**, para criar diretamente um registro na tabela Trip e um na tabela TripAttraction. Usamos esta solução para mostrar o uso destes comandos. No entanto, a solução mais aconselhável seria usar o business component de Trip para inserir. Neste curso, não vimos como carregar business component de dois níveis, mas é muito simples.

Vemos que depois de inserir a cabeça e linha de Trip, calculamos a quantidade de trips nas quais essa atração se encontra. Como a fórmula inline está disparando sem que esteja posicionada em qualquer tabela (está "solta" no código), temos que indicar a condição explícita de que queremos filtrar as trips da atração.

E esse valor é o que é retornado ao chamador desse procedimento. Então, a partir do evento click do controle (variável) `&newTrip`, invocamos este procedimento, passando-lhe o `AttractionId` da linha a partir da qual o clique é feito, e como o parâmetro que devolve é aquele que precisamos mostrar na coluna `&Trips`, diretamente o atribuímos à variável.

Logicamente, quando o usuário clicar sobre New Trip, deverá ver para essa linha, na coluna Trips, o valor que tinha antes do click, mais um. Ou seja, a linha deve ser atualizada.

DEMO

The screenshot shows two instances of a web panel titled "Application Name". The left instance is labeled "DEMO". Both instances have a "Country Id" dropdown set to "France". The main area contains a table with columns: Attraction Name, Country, Photo, and Trips.

Left Panel (DEMO):

Attraction Name	Country	Photo	Trips
Eiffel Tower	France		2 New Trip
Louvre Museum	France		0 New Trip
Matisse Museum	France		1 New Trip
Total Trips			3

Right Panel:

Attraction Name	Country	Photo	Trips
Eiffel Tower	France		2 New Trip
Louvre Museum	France		1 New Trip
Matisse Museum	France		1 New Trip
Total Trips			3

Annotations in red:

- A red arrow points to the "New Trip" link for the Louvre Museum entry in the left panel, indicating it was clicked.
- A red arrow points to the "New Trip" link for the Louvre Museum entry in the right panel, indicating it was clicked.
- The text "Atualizou somente a linha." (Updated only the line) is written next to the right panel's "New Trip" link.
- The text "Porém não atualizou o total!" (However, it did not update the total!) is written below the right panel's table.

Below the panels, the event code is shown:

```

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

```

Executemos para testar.

Por exemplo, vamos filtrar por França, e para a atração museu do Louvre, que por enquanto não está em nenhuma trip, cliquemos em New Trip. Vemos que a linha foi atualizada automaticamente.

Agora mostra a quantidade de trips do Louvre: 1.

No entanto, o que não foi atualizado foi a contagem total, que deveria dizer 4, e, no entanto, mantém o valor anterior. Por quê?

É que ao executar o evento de click associado à variável &NewTrip, somente seu código foi executado, e uma vez que dentro dele um valor foi atribuído a uma variável de grid se atualizou a linha, essa linha, somente. Sem executar nenhum outro evento. Nenhum, nem mesmo o Load.

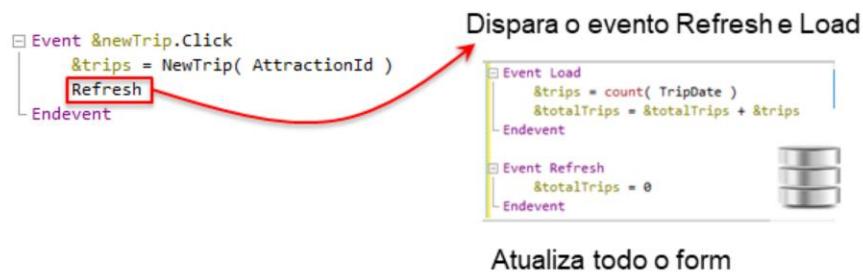
Comando Refresh

- Alternativa 1

```

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

- Alternativa 2



Portanto, temos duas alternativas para manter atualizado o Total de Trips carregado na grid quando esse evento ocorre.

Uma possibilidade é para &totalTrips adicionar um, uma vez que o procedimento só adicionou uma trip para essa atração.

Mas se, em seguida, o procedimento muda e se adicionam mais trips, teremos que lembrar de fazer a mudança de forma consistente neste evento.

Uma solução melhor parece ser poder pedir ao web panel que execute novamente os eventos Refresh e Load. Para isso, temos o comando Refresh. Ao fazer isto, ele tornará a ir à base de dados para carregar a grid.

Resumo

- Web Panel com um grid (com atributos)



No caso de um web panel com uma grid com atributos, GeneXus entende que essa grid tem uma tabela base associada, isto é, uma tabela que deverá navegar para carregar as linhas da grid. Carregará uma linha para cada registro dessa tabela base. Para filtrar, temos a propriedade Conditions da grid, para ordenar, a propriedade Order. Uma grid com tabela base é análoga a um for each.

Nos web panels, são produzidos eventos do sistema aos quais podemos programar o código a ser executado no momento em que são disparados. Vimos três desses eventos, que são disparados sempre que um web panel é aberto, ou seja, em sua primeira execução:

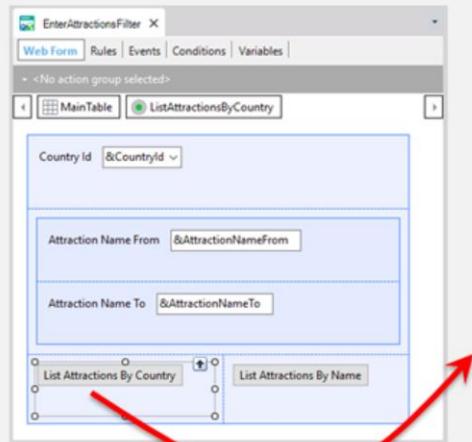
- O **Start** é ativado esta única vez. Ali podem ser inicializadas variáveis, por exemplo.
- O **Refresh** é disparado antes que sejam carregadas as informações da tela. Após este evento, ocorrerá o acesso à base de dados para trazer os dados a partir da tabela base e sua estendida.
- Para cada registro da tabela base que está prestes a ser carregado na grid, ocorre um evento **Load**. Portanto, aqui será o momento de programar todas as ações que queremos que se executem antes que a linha seja efetivamente carregada na grid. Os dados que são carregados na grid são exclusivamente aqueles das colunas visíveis ou invisíveis que foram incluídos nela.

Depois disso, o web panel estará carregado com as informações que extraiu da base de dados e se desconecta dela.

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

Por exemplo, o evento Click que programamos sobre esta imagem (&update) ou o Click sobre New Trip, ou ...

Resumo



1ª vez:

Start
Refresh
Load

N-ésima vez:

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

...mesmo no web panel que tínhamos definido muitas aulas atrás, o evento de usuário que associamos ao botão, ao qual damos este nome.

Resumo

- Web Panel com um grid (com atributos)



Esses eventos são conhecidos como **eventos do usuário** ou **eventos dos controles** (como o Click que vimos).

Quando o usuário provoca um evento destes, apenas seu código é executado, sem atualizar a tela. A única exceção é quando o evento ocorre no nível de uma linha da grid, como o caso que vimos de &newTrip, e dentro do seu código é atribuído valor a uma variável da mesma grid, que em nosso caso era &Trips. Nesse caso, o valor da variável é atualizado na tela, para essa linha, para que se exiba atualizada.

Se precisarmos executar novamente o Refresh e voltem a se carregar as linhas na grid a partir da base de dados (por exemplo, para atualizar o total das linhas), então podemos escrever o comando **Refresh** no evento.

Resumo e além

- Web Panel sem grid porém com atributos diretamente no form

Parm(in: **AttractionId**);

Carrega somente **um** registro

Tabela Base

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

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 sem grid porém com atributos diretamente no form

The screenshot shows the GeneXus interface with two windows. On the left, a 'Web Form' window displays a form with fields for AttractionId, AttractionName, Country Name, Category Name, and City Name. On the right, a 'Conditions' tab is selected in another window, containing the code 'AttractionId = &AttractionId;'. Below these windows is a table titled 'Tabela Base' (Base Table) with three rows of data.

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

Tabela Base

Parm(in: &AttractionId);

Se precisarmos estabelecer outro tipo de condição, ou se, por exemplo, tivéssemos recebido em variável, em vez de em um atributo, temos a guia Conditions para definir o filtro.

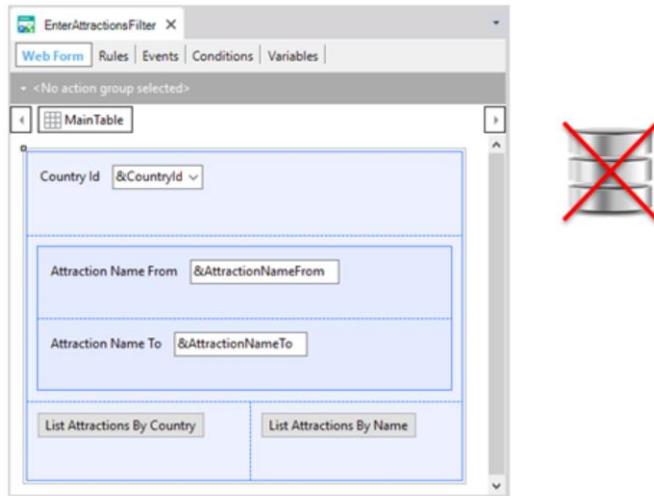
Web panels com tabela base



Vimos até aqui dois web panels que possuem tabela base:

- o primeiro com uma grid. Lá a tabela base da grid é a tabela base do web panel, ou seja, a tabela à qual o web panel decide automaticamente navegar para trazer as informações a carregar na tela.
- o segundo sem grid, mas com atributos. Lá a tabela base do web panel se encontra a partir desses atributos.

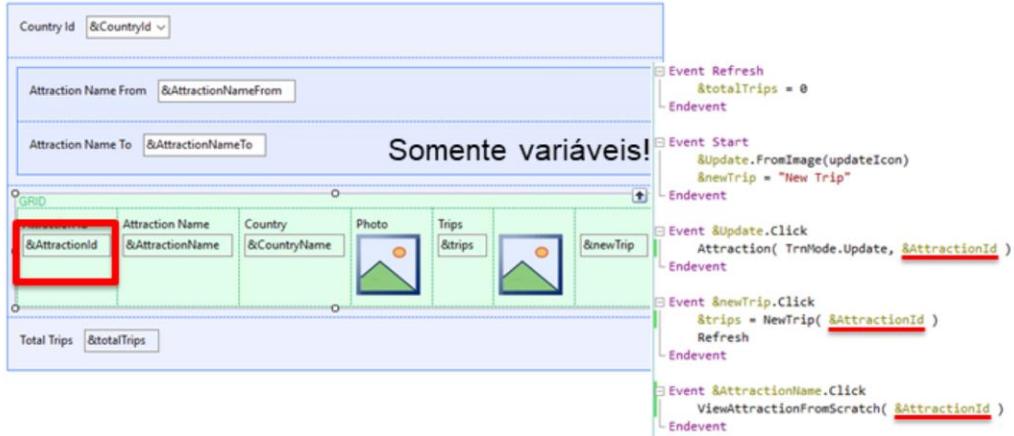
Web panels sem tabela base



Agora vamos ver o caso de web panels sem tabela base, ou seja, os web panels que não têm programada **automaticamente** uma consulta à base de dados.

O caso mais óbvio é quando não se consulta em absoluto a base de dados, como foi o caso do nosso web panel de partida, o que só pedia dados ao usuário e chamava a outros objetos.

Web panels sem tabela base



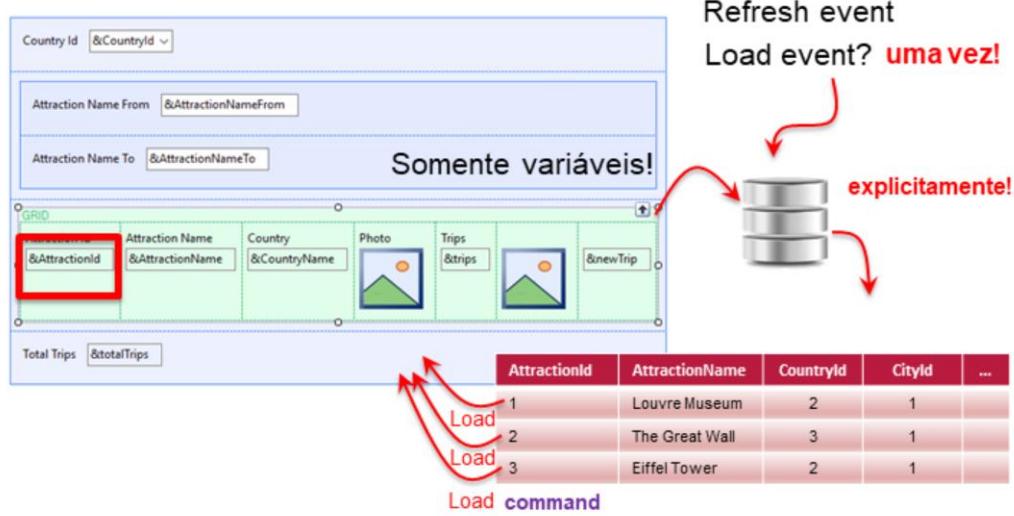
Mas também podemos ter um web panel que sim consulte a base de dados, só que essa consulta e carga na tela estão completamente nas mãos do desenvolvedor.

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 AttractionId, 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**, único local onde este comando é permitido, uma linha será inserida na grid.

Web panels sem tabela base

AttractionId **AttractionName** **CountryId** **CityId** ...

1	Louvre Museum	2	1	
2	The Great Wall	3	1	
3	Eiffel Tower	2	1	

Country Id &CountryId

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

Attraction Id	Attraction Name	Country Name	City Id
&AttractionId	&AttractionName	&CountryName	&CityId
Total Trips	&totalTrips		

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
  
```

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.

Web panels sem tabela base

Web Panel WWAttractionsFromScratch2 Navigation Report

Name	WWAttractionsFromScratch2	Environment	Default (C#)
Description	WWAttractions From Scratch2	Spec. Version	15_0_1-106638
Form Class	HTML	Program Name	WWAttractionsFromScratch2
Parameters			

FILL &CountryId with [CountryId](#), [CountryName](#) in

```

    [grid]-Country (CountryId) INTO CountryId CountryName
    Order CountryName

```

Event Load

For Each Attraction (Line: 2)

Order:

```

        CountryId , AttractionName WHEN not &CountryId. isempty()
        ! No index
        AttractionName OTHERWISE
        ! No index

```

Navigation filters:

```

        Start from: FirstRecord
        Loop while: NotEndOfTable

```

Constraints:

```

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

```

Join location:

```

        Server

```

```

    [grid]-Attraction (AttractionId) INTO AttractionName CountryId AttractionPhoto.Url AttractionId AttractionPhoto
    [grid]-Country (CountryId) INTO CountryName
    [grid]-count(TripDate) navigation (AttractionId)

```

Formulas

Navigation to evaluate: count([TripDate](#))

Given: AttractionId
Index: IATTRACTION
Group by: AttractionId

```

    [grid]-TripAttraction
    [grid]-Trip (TripId)

```

Agora, se olharmos para a lista de navegação do web panel sem tabela base... vemos que aparece o for each no Load, indicando a navegação.

Resumindo

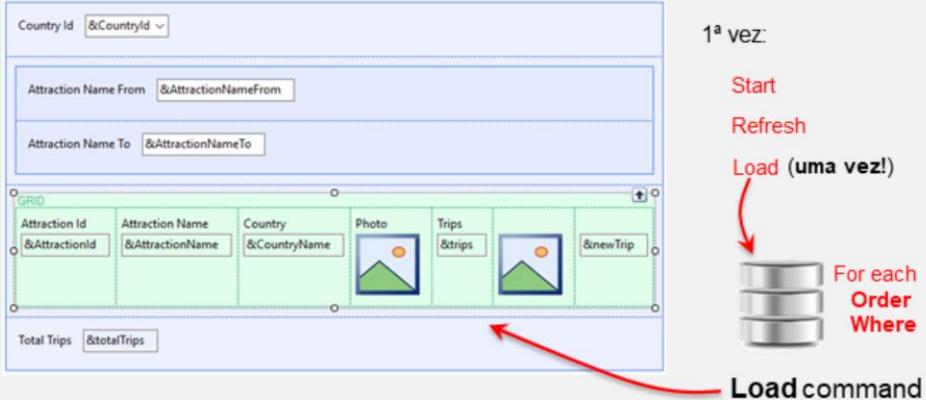
- Web panel com um grid (com tabela base)



Quando tínhamos o web panel com uma grid com tabela base, ao abrir o web panel, se executava o evento Start, imediatamente o Refresh, após o qual a tabela base era acessada automaticamente, filtrando de acordo com as condições da grid e ordenando de acordo com os atributos indicados na propriedade Order da grid. Dizemos que há um tipo de for each implícito, que GeneXus coloca internamente, sem que o desenvolvedor tenha que fazer nada. E para cada registro a ser carregado como linha do grid, antes de fazê-lo, se executa o evento Load. É por isso que o evento Load será disparado, neste caso, **tantas vezes quanto as linhas serão carregadas na grid**.

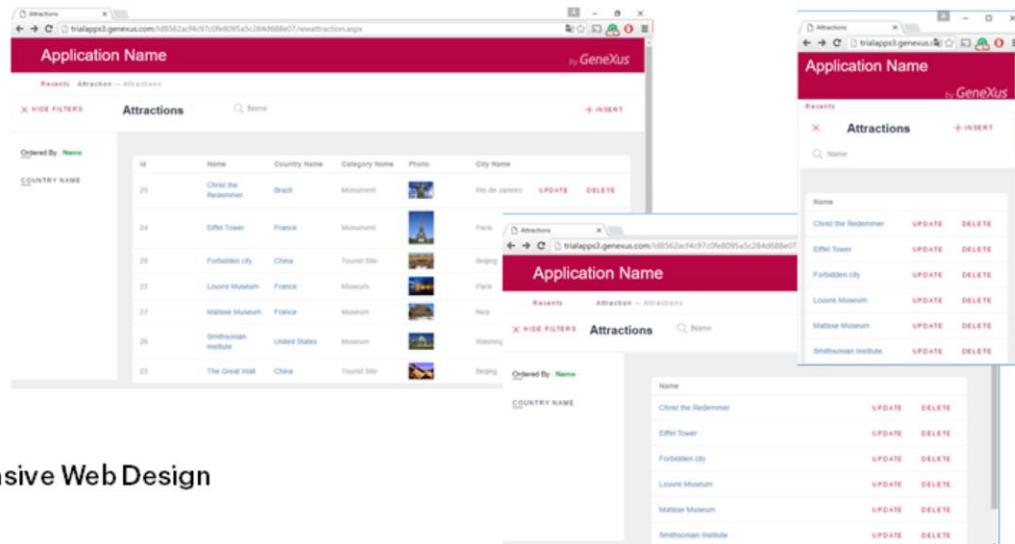
Resumindo

- Web panel com um grid (sem tabela base)



Ao contrário, quando o web panel não possui tabela base, na grid existem apenas variáveis e não há nenhum for each implícito.

Ao abrir o web panel, o evento Start é executado como antes, o Refresh como antes e o Load apenas uma vez. Se precisa ir à base de dados para recuperar informações, deve fazê-lo explicitamente dentro deste evento. Em outras palavras: devemos escrever o for each e lá utilizar as cláusulas Order e Where. Para carregar cada linha, isso deve ser feito explicitamente com o comando Load, para cada uma.

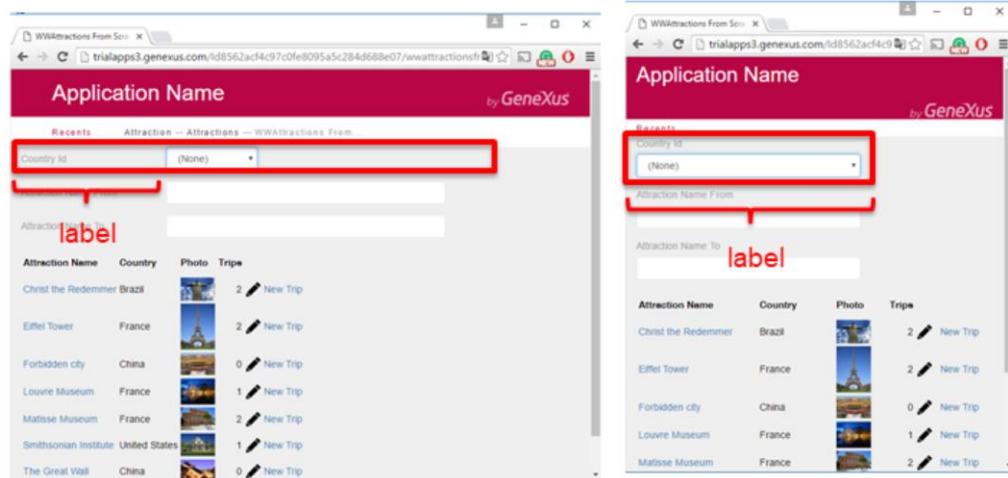


Responsive Web Design

Para concluir, vejamos que as aplicações projetadas por GeneXus autoajustam as informações que se apresentam na tela de acordo com o tamanho dessa tela. Por exemplo, vamos executar o Work With Attractions que criou o pattern. Estamos executando em um navegador que está ocupando toda a tela de um notebook.

Mas vejamos o que acontece se tornarmos a tela do navegador menor. Se a reduzirmos a partir da direita ... Vemos que em um momento a grid começa a mostrar apenas o nome da atração e as ações. E se continuarmos a encolher... Já nem sequer vemos a coluna esquerda. Assim sairá em um telefone celular que execute a aplicação a partir do seu navegador.

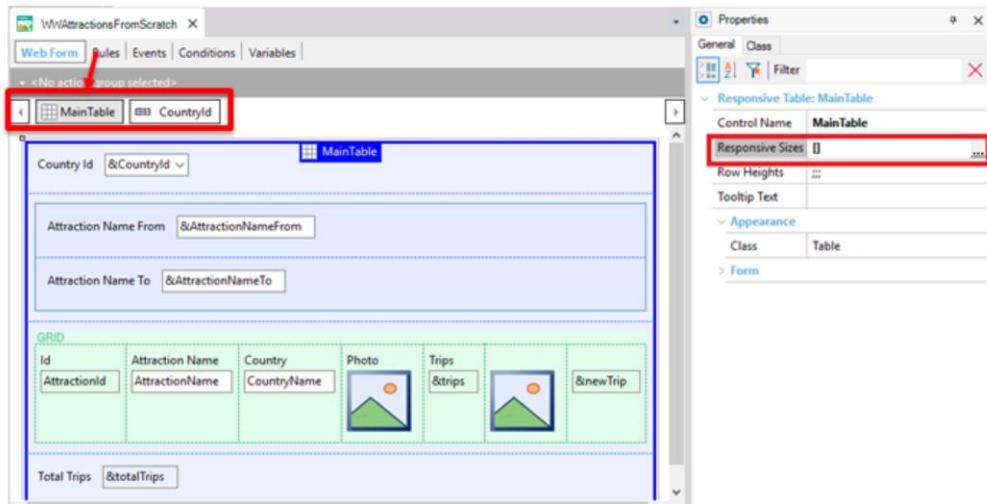
Responsive Web Design



Se vamos agora ao nosso Work With,o que desenvolvemos a partir do zero, vemos que, embora não fizemos nada, ele possui algum autoajuste mínimo.

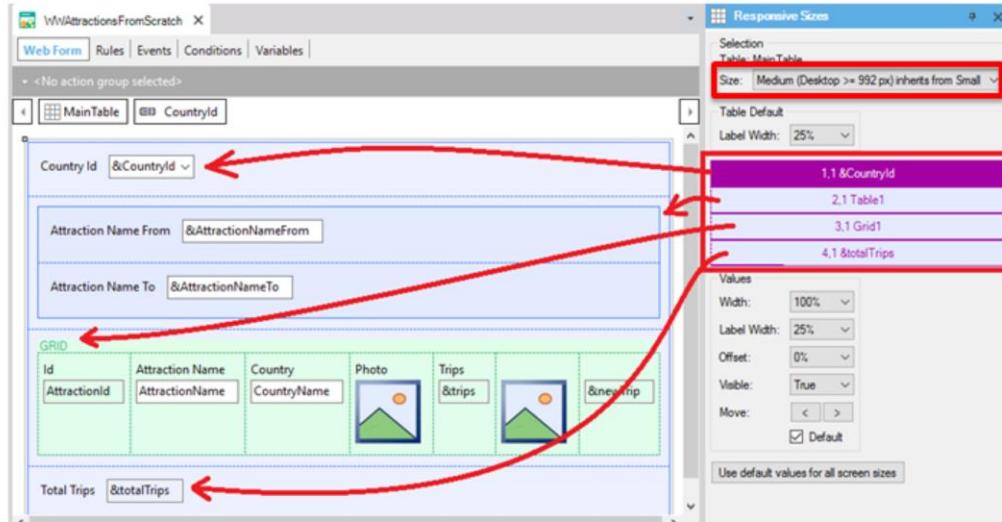
Por exemplo, prestemos atenção às variáveis de cima, que passarão de ter o rótulo à esquerda e ocupando uma certa largura antes do início do controle variável propriamente dito, para ter o rótulo acima, ocupando 100% da largura.

Responsive Web Design



Para alcançar esse comportamento, os objetos web usam as tabelas responsivas: responsive tables. Não vamos estudá-las neste curso, mas se vamos ao nosso web panel e nos posicionamos dentro do form no controle Country Id, vemos que aqui em cima nos mostra o controle tabela dentro do qual está inserido. É a tabela principal. Se clicarmos lá, vemos que entre as propriedades desta tabela está **Responsive Sizes**, ou seja, o tamanho de resposta ou “responsivos”. E se clicarmos lá...

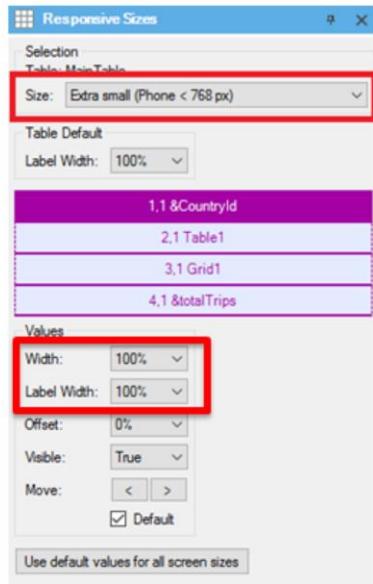
Responsive Web Design



...nos edita esta tela. Que se vemos tem os controles que são inseridos dentro dessa tabela principal. Alguns, como esta tabela (a que tem &AttractionNameFrom e & AttractionNameTo) ou como a grid, tem por sua vez controles dentro.

Mas também temos um combo que nos permite selecionar os tamanhos de tela. Estamos em Medium. Mas se escolhermos agora Small ou o Extra Small...

Responsive Web Design



The screenshot shows a web browser window titled 'WWAttractions From Scr' with the URL 'trialapps3.genexus.com/ld8562acf4c9'. The page has a red header 'Application Name by GeneXus'. It features a 'Recent' section with a dropdown menu set to '(None)'. Below it are two input fields for 'Attraction Name From' and 'Attraction Name To'. A table lists attractions with columns for 'Attraction Name', 'Country', 'Photo', and 'Trips'. The data is as follows:

Attraction Name	Country	Photo	Trips
Christ the Redemmer	Brazil		2 New Trip
Eiffel Tower	France		2 New Trip
Forbidden city	China		0 New Trip
Louvre Museum	France		1 New Trip
Matisse Museum	France		2 New Trip

Vemos que os valores mudam.

Em particular, note que para &CountryId na tela Extra small diz que o rótulo vai ocupar 100% da largura, e o controle variável em si ocupará o 100% restante. E é por isso que em execução vimos o rótulo sobre a variável.

Responsive Web Design

The screenshot shows the GeneXus development environment. On the left, the 'Responsive Sizes' dialog is open for a 'Table: Main.Table'. A red box highlights the 'Size' dropdown set to 'Medium (Desktop >= 992 px) inherits from Small'. Another red box highlights the 'Label Width' setting at 25% for a specific column. A red arrow points from this setting to a corresponding 'Label Width' input field in a table row within the browser preview on the right. The browser preview shows a travel application with a table of attractions like Christ the Redeemer and the Eiffel Tower.

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

Se agora escolhermos o tamanho Medium, vemos que a etiqueta ocupa 25% da largura, e o controle variável propriamente dito, o tamanho restante até completar 100%.

Isto coincide com o que vimos na execução. Também vejamos que temos a propriedade Visible para ser capaz de ocultar algum controle para um tamanho de tela específico.

Então, jogaríamos com os controles para que se possa ajustar o que é exibido em execução de acordo com o tamanho da tela na qual o form é mostrado em cada oportunidade.

Isto é conhecido como Responsive Web Design ou Desenho Web Responsivo. Aqui apenas o introduzimos.

Mais

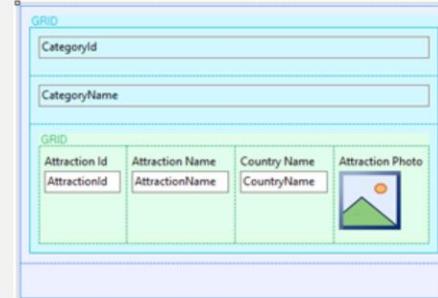
- Web panel sem grid ou com um grid
Tabela base automática?
- Web panels com múltiplos grids



Paralelo



Aninhado



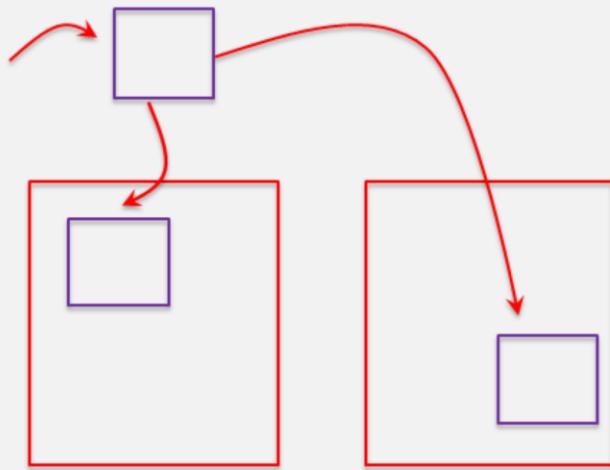
Há muito mais a saber sobre web panels.

Por exemplo, quando o web panel não tem nenhuma grid ou tem uma, onde exatamente busca GeneXus a aparição de atributos para determinar se associa uma tabela de base implícita ou não? E, se atributos aparecem nesses lugares, quais critérios devem atender? Embora possamos imaginar que serão os mesmos que para um for each: pertencem à tabela estendida.

É possível implementar web panels com várias grids: sejam elas paralelas ou aninhadas. É análogo a ter for eachs paralelos ou aninhados.

Mais

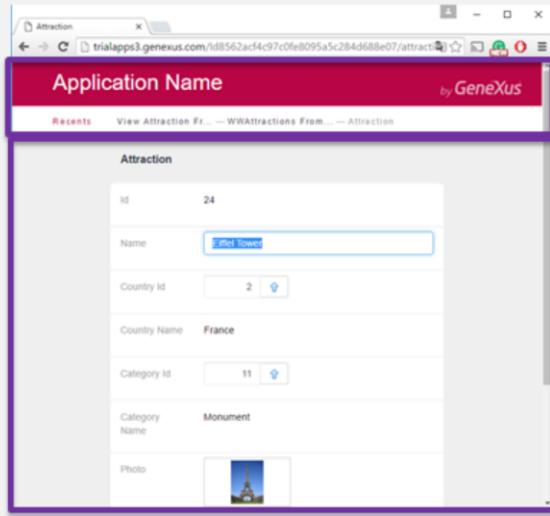
- Tipos de Web Panels:
 - Web page
 - Component
 - Master Page



Existem três tipos de web panels. Nestes vídeos, só vimos o tipo Web Page, mas há Web panels que podem ser definidos como componentes, para o caso em que uma parte de uma página web precise ser repetida em vários web panels. Ao defini-lo como um componente, pode ser inserido em outros objetos com tela web.

Mais

- Tipos de Web Panels:
Web page
Component
Master Page



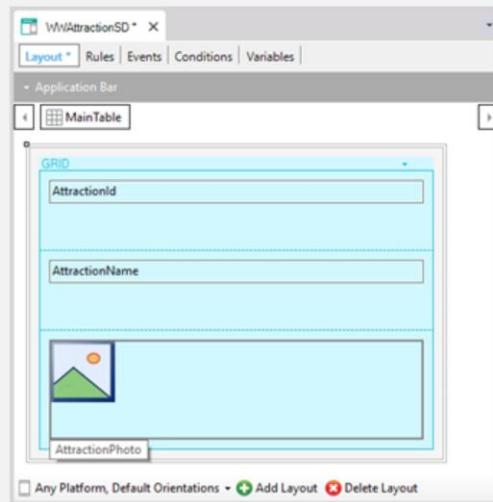
The screenshot shows a web browser window with the title "Attraction". The address bar displays a URL starting with "trialapps3.genexus.com". The main content area has a header "Application Name by GeneXus" and a sub-header "Attraction". Below this, there is a table with the following data:

Attraction	
ID	24
Name	<input type="text" value="Eiffel Tower"/>
Country ID	2
Country Name	France
Category ID	11
Category Name	Monument
Photo	

Por outro lado, temos web panels que são definidos como Master Pages, ou seja, páginas mestre. Toda KB é criada com páginas mestre, que são a estrutura dentro da qual são carregadas todas as páginas que se executam. Aqui podemos ver esta parte superior da transação, que é parte da Master Page da aplicação. A tela de transação é aberta na área designada para ela na Master Page.

Mais

- Panels para Smart Devices



Por último, mencionamos que, para o desenvolvimento de aplicações móveis para dispositivos inteligentes (Smart Devices), também temos painéis. Eles são chamados de Panels for Smart Devices. Em outro vídeo, veremos uma introdução a este tipo de aplicações.



Vídeos

training.genexus.com

Documentação

wiki.genexus.com

Certificações

training.genexus.com/certifications