Eventos em um objeto panel

Eventos do lado do cliente e do lado do servidor

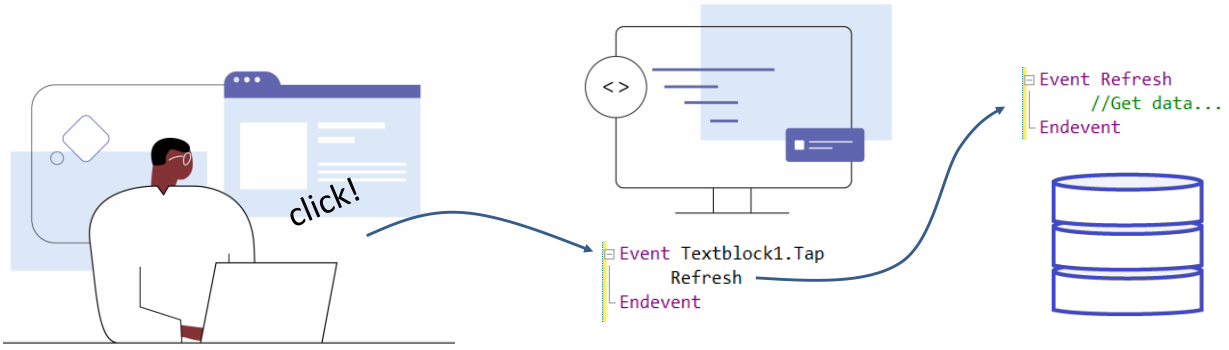
GeneXus™

Tanto em uma aplicação web gerada em Angular quanto em uma aplicação gerada para dispositivos móveis em Android ou iOS, utilizamos objetos panel para implementá-la.

Por esse motivo, os eventos disponíveis neste objeto são em sua maioria válidos para ambas as plataformas.

Neste vídeo, tudo o que falemos sobre eventos em um objeto panel será aplicável a ambos os tipos de plataformas e faremos uma exceção explícita quando for o caso.

Uso dos eventos

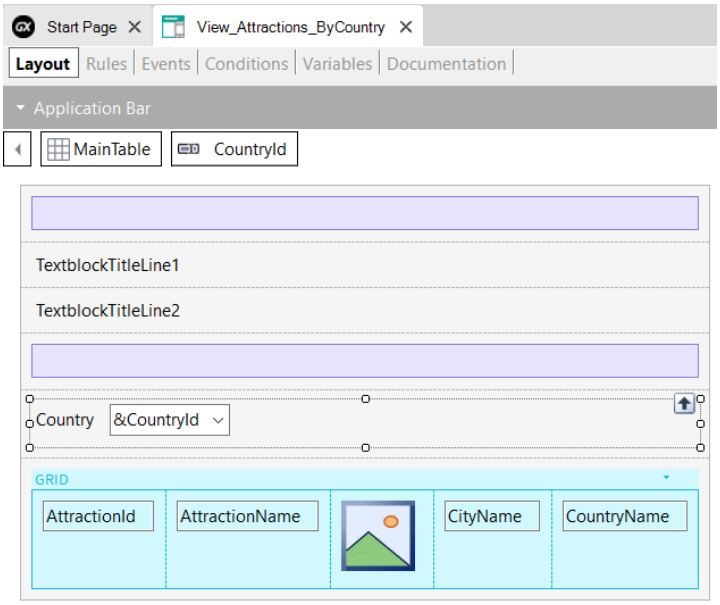


Como já sabemos, os eventos são mensagens que o software ou o sistema nos dão em determinados momentos da execução de nossa aplicação e isso nos permite programar ações a serem executadas nesses momentos.

Por exemplo, se o usuário clica em um botão, ou se digita algo em um controle de texto e sai dele, ou se é preciso que o servidor envie informações atualizadas, podemos programar uma resposta após que seja disparado o evento.

Suponhamos que quando vimos os dados das atrações, quiséssemos ter filtros para que os dados mostrados estivessem de acordo com a restrição escolhida.

Um exemplo simples

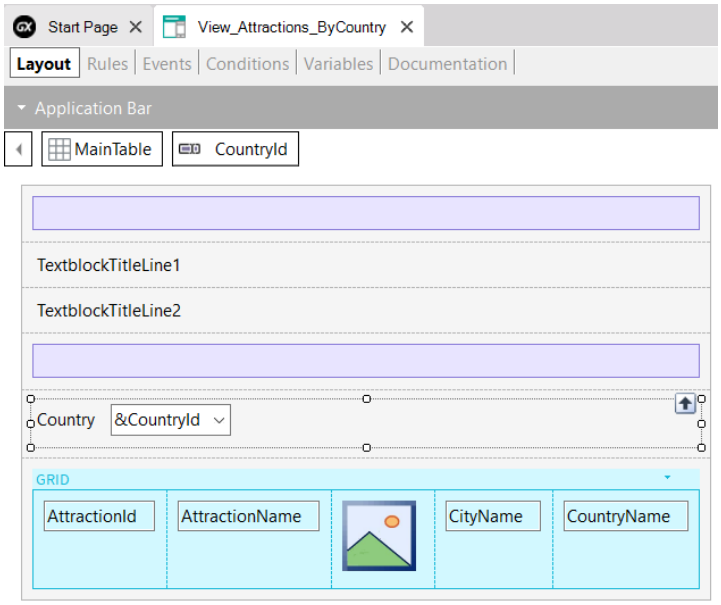


Para implementar isto, abrimos o panel View_Attractions e o salvamos com o nome View_Attractions_ByCountry.
Agora vamos para a seção de variáveis e adicionamos uma variável &CountryId que automaticamente fica do tipo do atributo CountryId.
A arrastamos para o form depois dos textblocks do título e em suas propriedades alteramos o rótulo CountryId para Country, alteramos a propriedade ControlType para Dynamic Combo e colocamos Item Descriptions em CountryName. Também atribuímos à propriedade EmptyItem o valor True, para que no início o combo apareça sem nenhum valor padrão.

Agora, para que o filtro tenha efeito nas atrações que vemos no grid, editamos as propriedades do grid e na propriedade Conditions escrevemos: CountryId = &CountryId when not &CountryId.IsEmpty() e fechamos com ponto e vírgula.
Aproveitamos que estamos nas propriedades do grid e configuramos a propriedade Auto Grow para o valor True para que o grid ajuste automaticamente seu tamanho e possa exibir todos os registros.

Uma vez que escolhemos um país, o panel deve ser atualizado para que o grid carregue apenas aquelas atrações que sejam desse país.
Para isso usamos o evento ControlValueChanged do combo dinâmico, então vamos para a guia Events e no menu escolhemos Insert event, clicamos sobre &CountryId e selecionamos o evento que dissemos.

Um exemplo simples (cont.)



Vemos que é aberto o código do evento para escrever o que queremos que seja executado quando seja disparado este evento. Escrevemos Refresh. Este comando fará com que o grid obtenha novamente os dados, desta vez filtrados pelo país escolhido.

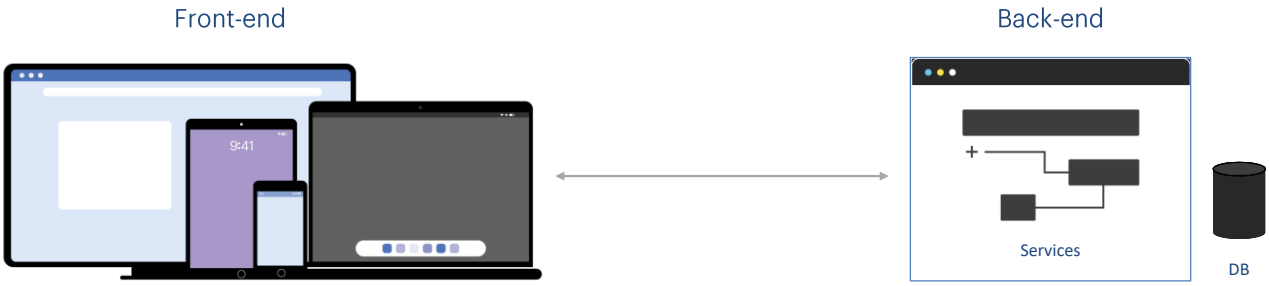
Para que o servidor entenda que queremos trazer novos dados, devemos fazer com que seja alterada a URL enviada ao servidor, para que ele interprete que é uma página nova e obtenha a informação correspondente para enviá-la ao cliente.
Para isso, adicionamos uma regra Parm com as variáveis que usamos nos filtros, para que ao alterar seu valor, seja atualizada a página. Neste caso, adicionamos apenas a variável &CountryId.

Clicamos com o botão direito do mouse no panel View_Attractions_ByCountry e escolhemos Run.

Vemos que, como havíamos colocado a propriedade EmptyItem como True, o combo aparece sem um país selecionado e são mostradas todas as atrações.
Se escolhemos China, vemos que o grid é recarregado e agora mostra apenas as atrações turísticas da China.

Aprofundemos um pouco mais no tema dos eventos.

Eventos do cliente e do servidor



Client

- ClientStart
- Back
- User Events
- Controls Events
- WorkWith Predefined Events
- Only for mobile devices:
  - Navigation.Start Events

Server

- Start
- Refresh
- Load

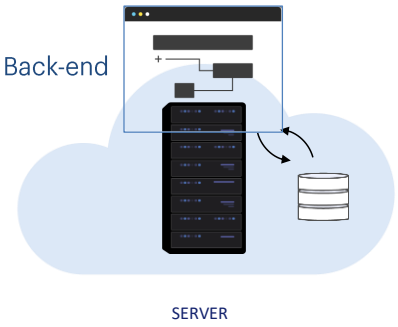
Em um objeto panel, temos dois tipos de eventos, os eventos que são disparados do lado do cliente e os eventos que são disparados no servidor.

Os eventos do servidor são os mesmos de quando trabalhamos com webpanels, os eventos Start, Refresh e Load. Estes eventos serão disparados sempre no servidor para o caso de aplicações Angular e aplicações móveis nativas on-line. No caso de estarmos implementando aplicações móveis nativas desconectadas, serão disparados internamente no dispositivo.

Os eventos do lado do cliente são o ClientStart, o Back, eventos definidos pelo usuário, eventos associados aos controles da tela e também os eventos predefinidos pelo padrão WorkWith, que é um padrão que quando aplicado permite gerar uma série de panels que cumprem a funcionalidade de trabalhar com uma entidade (para realizar inserções, modificações, exclusões, navegação de dados, filtros, etc.), de forma análoga ao padrão WorkWith que usamos quando trabalhamos com webpanels.

No caso em que usemos os objetos panel para gerar uma aplicação para dispositivos móveis, teremos os eventos associados aos estilos de navegação da informação, que dependem, por exemplo, do tipo de dispositivo e sua orientação ao iniciar a aplicação.

Eventos do lado do servidor



- Start
- Refresh
- Load

- Start Event execute only once.
- Refresh Event is executed after Start Event
- Refresh Event trigger Load Event
- Load Event is the last of system Events executed (only if a grid exist)
  - Grid with Base Table: Load is executed as many times as records in base Table.
  - Grids without Base Table: Load is executed only once.
  - SDT based Grids: Load is not triggered.

Vejamos agora os eventos do lado do servidor.

- O primeiro que se executa é o evento Start. É executado apenas uma vez, quando é aberto o panel, e não será executado novamente a menos que saiamos do objeto e entremos nele novamente.

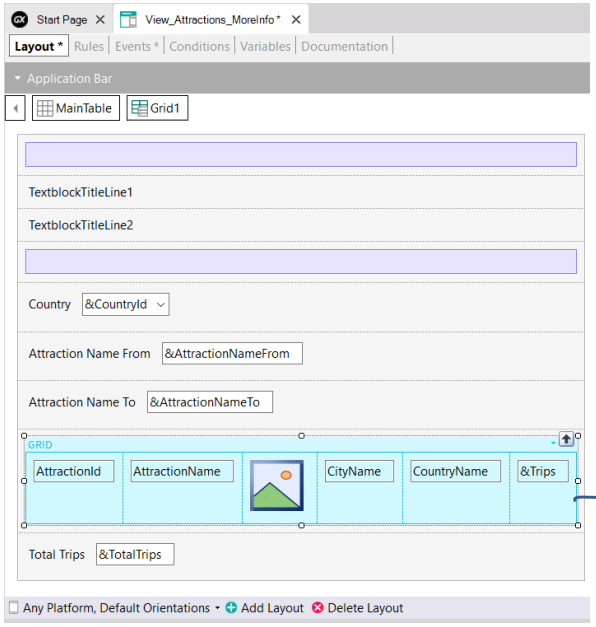
- O evento Refresh é executado após o Evento Start, normalmente apenas uma vez, mas pode ser invocado novamente por meio do comando Refresh, neste caso será executado mais de uma vez e será o primeiro evento, pois o Start já não voltará a ser executado.

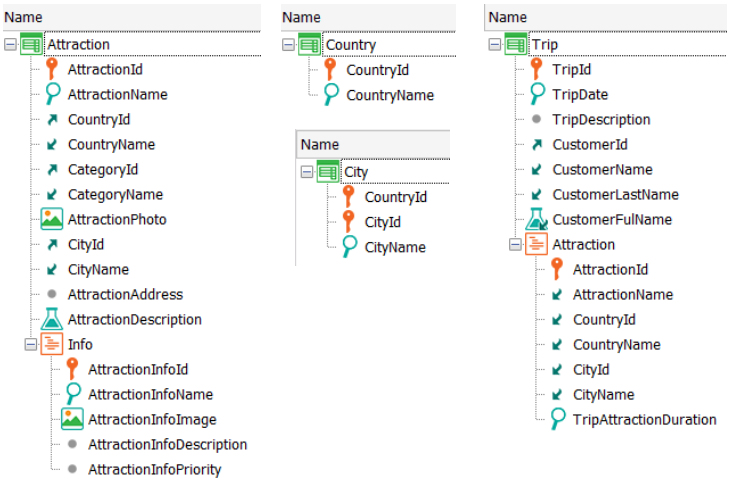
- Quando o evento Refresh é chamado, ao finalizar será disparado o evento Load. Isto apenas se houver pelo menos um grid no panel e o grid não for uma variável SDT coleção.

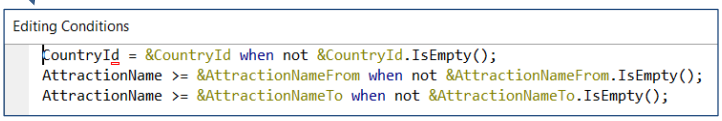
- O evento load é o último dos eventos do sistema a ser executado e
  - Se tiver Tabela Base, será executado tantas vezes quanto registros existam na tabela base.
  - Se o grid não tem tabela base será executado apenas uma vez
  - E se o grid está baseado em um SDT, não será executado o evento load.

No caso de estarmos em uma aplicação para dispositivos móveis, no código dos eventos Start, Refresh e Load, não há acesso aos recursos do dispositivo, por exemplo a câmera, o GPS, etc.

Exemplo de eventos do lado do servidor







Vamos estudar o caso do seguinte panel, que nos permite ver um grid de atrações e filtrar o grid por país e nome de atração.

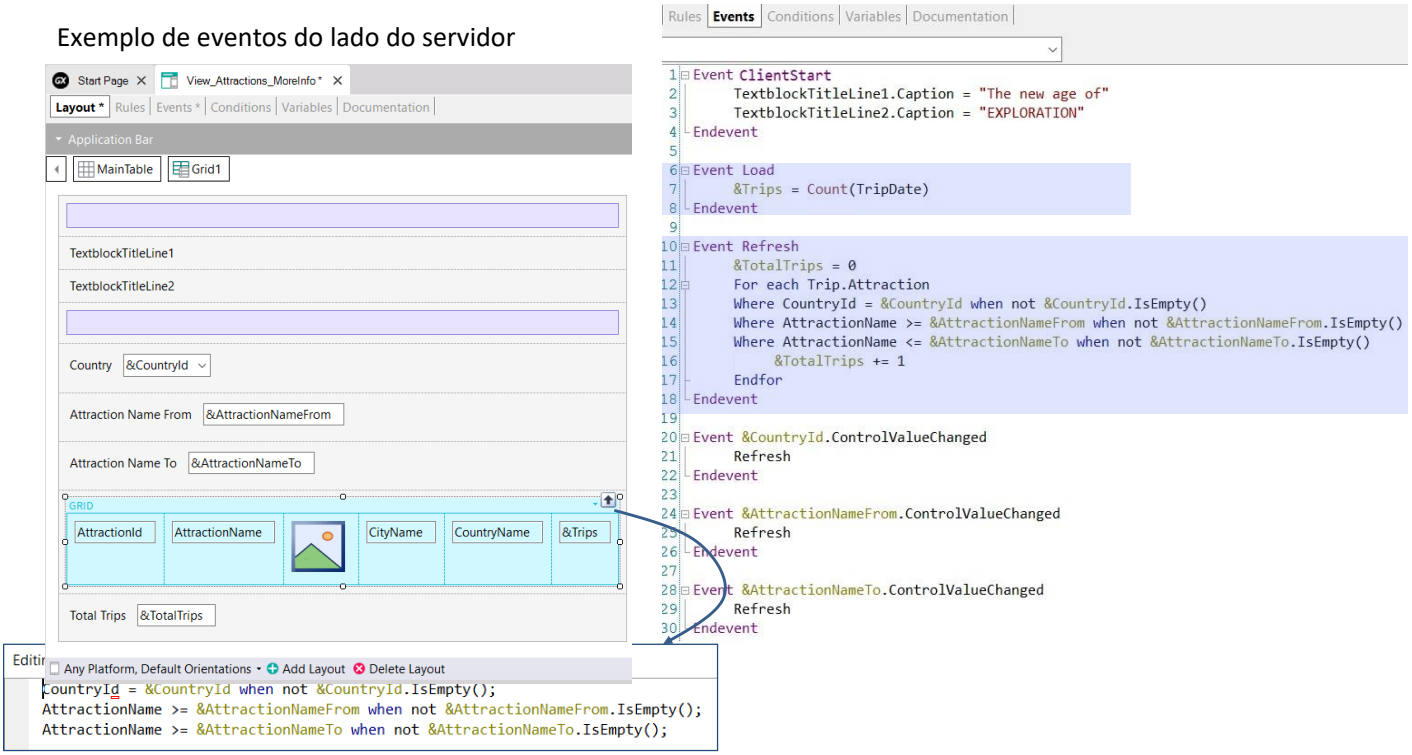
O panel View_Attractions_ByCountry que acabamos de ver, salvamos com o nome View_Attractions_MoreInfo, adicionamos variáveis para filtrar por nome de atração e também algumas variáveis para totalizar as viagens. Aqui vemos o panel com todas as alterações.

Agora, no grid estão presentes os atributos AttractionId, AttractionName, AttractionPhoto, CityName, CountryName e uma variável &Trips que nos mostrará as viagens que foram realizadas para cada atração.

Abaixo do grid, vemos o total global de viagens realizadas para todas as atrações.

Também podemos ver o modelo das transações utilizadas neste panel e as conditions do grid que nos permitem realizar os filtros.

Exemplo de eventos do lado do servidor



Na guia de eventos, vemos que estão programados os eventos Refresh e Load que são disparados do lado do servidor.

No evento Load, calculamos o total de viagens de uma atração mediante uma fórmula que acessa a tabela TripAttraction, correspondente ao segundo nível da transação Trip. Observemos que na fórmula Count estamos fazendo referência ao atributo TripDate e, devido aos atributos presentes no grid, a tabela base do grid (ou seja, da parte variável do panel) será TripAttraction, portanto, serão contadas as viagens correspondentes a cada atração.

No evento Refresh, programamos um For Each para acessar a tabela TripAttraction para contar o total global de viagens das atrações, de forma que quando for desenhada a parte fixa, já estejam disponíveis os dados correspondentes ao total de viagens.

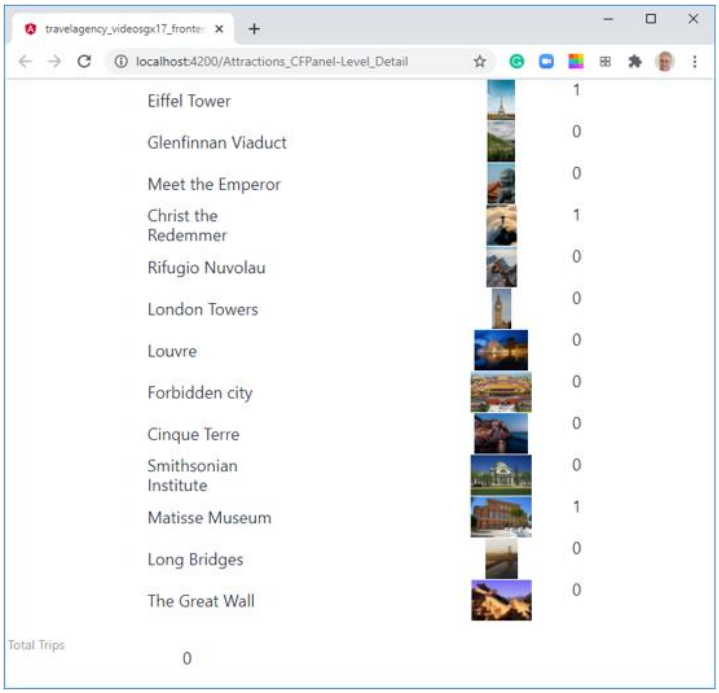
[Se fosse um Web Panel seria feito no evento Load, mas como é um Painel seria feito no evento Refresh]

Total global de viagens mal calculado!

Layout * | Rules | **Events *** | Conditions | Variables

Events

```
Event Load
      &Trips = Count(TripDate)
      &TotalTrips = &TotalTrips + &Trips
Endevent

Event Refresh
      &TotalTrips = 0
Endevent
```



[*Vamos ver um trecho do vídeo Panel Object. Primeiros passos*]

Se vamos para a aplicação executando no navegador, vemos que aparece o total de viagens de cada atração corretamente, mas que o total acumulado de viagens sai em 0.
O que fizemos de errado? A variável &TotalTrips está sendo incrementada no evento Load como fizemos no webpanel e temos a segurança de que este evento está sendo disparado porque vemos que algumas atrações têm viagens... Então?

A razão é que os objetos panels não funcionam como os web panels. Nos objetos panel, a parte fixa é carregada de forma independente do grid.
Neste caso, a variável &TotalTrips está na parte fixa do painel, que é o que se carrega primeiro e então ocorre a carga do grid, portanto quando se mostra a variável ainda não pôde carregar o grid, ainda não foi disparado o evento Load e não foi possível acumular o valor de &TotalTrips.

Recordemos que em um objeto panel usado para desenvolver aplicações com foco em customer-facing, para carregar a tela no dispositivo cliente, invoca-se a serviços localizados no servidor que são os que acessam a base de dados. Estes serviços são data providers, que são independentes para a parte fixa e para o grid (ou cada grid) da tela.

Quando a execução do panel é iniciada, é disparado um evento local no cliente que invoca o data provider para carregar a parte fixa e faz com que sejam disparados os eventos Start e Refresh no servidor. Em seguida, é executado um segundo data provider que dispara o evento Load no servidor N vezes e carrega o

grid.

Em nosso exemplo, ao ser disparado o Refresh primeiro, inicializa-se a variável &TotalTrips e carrega-se a parte fixa, logo depois é disparado o evento Load que é onde &TotalTrips é carregado com o valor correto e atualiza-se o grid, mas a parte fixa já foi carregada antes e não volta a ser desenhada.

Devido a esta característica não podemos programar o objeto panel como se fosse um webpanel.

Em outro vídeo entraremos em detalhes no disparo dos eventos e na determinação das tabelas base, mas por enquanto, para que o exemplo funcione, vamos mudar a programação.

Solução correta

```
Event Load
      &Trips = Count(TripDate)
Endevent

Event Refresh
      &TotalTrips = 0
      For each Trip.Attraction
            &TotalTrips += 1
      Endfor
Endevent
```

A solução é incluir no evento Refresh um For Each que acesse a tabela TripAttraction e conte o total global de viagens. Não nos esqueçamos de eliminar o cálculo que tínhamos no evento Load.

A razão pela qual incluímos a atualização no evento Refresh, utilizando um For Each, é porque o evento Refresh será disparado quando for carregada a parte fixa, que será antes de ser carregado o grid.
Portanto, ao carregar a parte fixa o valor de &TotalTrips terá o valor correto e logo depois se atualizará a parte do grid.

[Fim da revisão]

Eventos do lado do cliente

- Types:
  - System Events: ClientStart, Back
  - User Events
  - Screen Controls Events
  - Others: WW events & Navigation.Start
- Call to services to access server resources
- Do not trigger Server Side Events (unless Refresh command is used)
- In mobile apps, Access to devices resources
- Use different Grammar & use of Composite

Vejamos agora os eventos do lado do Cliente. Estes eventos são a resposta da aplicação à interação do usuário.

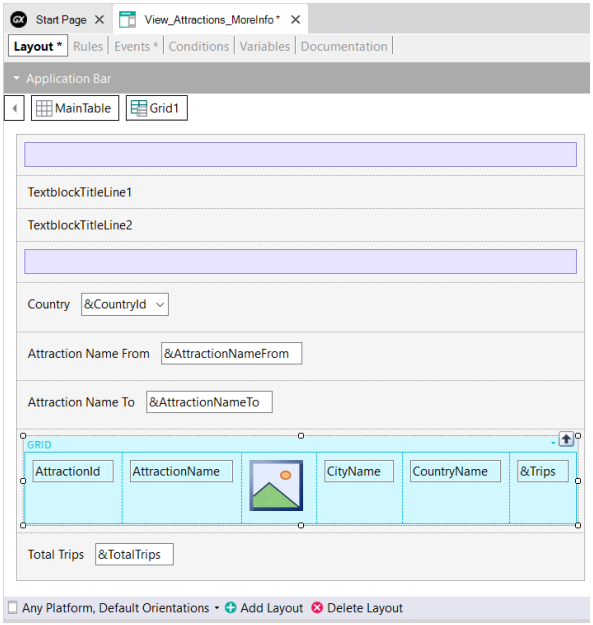
- Existem vários tipos de eventos no cliente: os eventos de sistema como o ClientStart e o Back, os de usuário, eventos de controles em tela e outros que veremos a seguir.
- O código associado a estes eventos é executado no cliente, a menos que seja necessário acessar um recurso do servidor, por exemplo, se é desejado acessar a base de dados. Neste caso, o cliente deverá chamar um serviço do servidor.
- Durante a execução de um evento do lado do cliente, os eventos do lado do servidor não são executados, a menos que sejam requeridos explicitamente através do comando Refresh, que faz com que seja disparado o evento Refresh do servidor, seguido do evento Load (se houver pelo menos um grid, como vimos antes).
- Se estamos executando uma aplicação nativa para dispositivos móveis, os eventos do lado do cliente têm acesso a todos os recursos de hardware e software do dispositivo, como a câmera, GPS, microfone, calendário, contatos, etc.
- Estes eventos do lado do cliente terão uma gramática particular diferente dos eventos do lado do servidor.

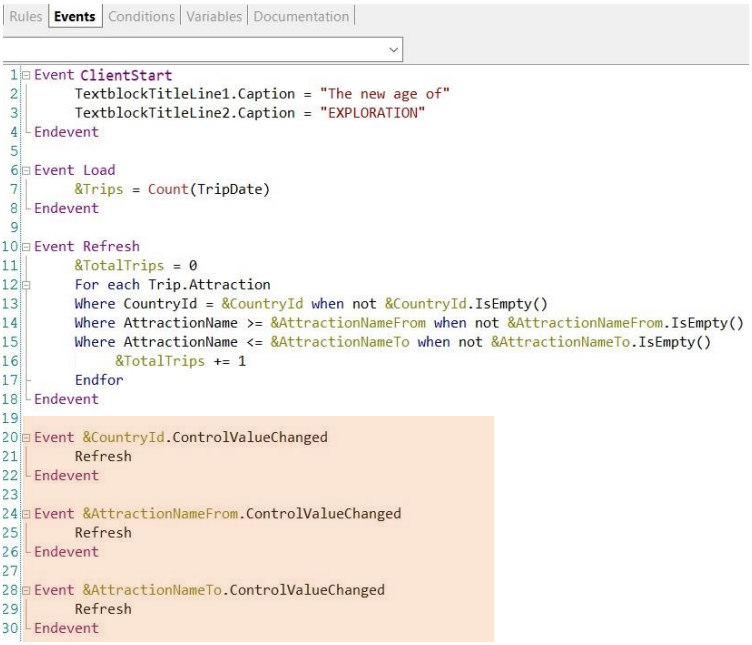
Eventos do lado do cliente

- ClientStart
- Back
- User Events
- UI Controls Events
- WorkWith Predefined Events
- Only for mobile devices:
  - Navigation.Start Events

Vejamos brevemente cada evento do lado do cliente.
O evento **ClientStart** é o primeiro evento que é disparado, mesmo antes do evento Start que é disparado no servidor e sem a necessidade de qualquer interação do usuário com a aplicação. É utilizado para inicializar a tela inicial e aspectos relacionados com a UI.
O evento **Back** é utilizado em plataformas móveis e permite capturar que o usuário pressionou o botão de retorno em dispositivos Android ou que foi realizado o gesto de retorno em dispositivos iOS e programar alguma ação apropriada.
Os eventos de **Usuário** têm um nome dado pelo usuário e permitem que o desenvolvedor associe um determinado código que seja executado quando é ativado um determinado controle em tela, ao clicar (ou tap) sobre ele.
Os **controles em tela** têm eventos próprios, dependendo do controle, como: click (ou tap), double-click (ou duplo tap), drag, swipe, ControlValueChanged, etc. Estes eventos são disparados quando ocorrem algumas das ações mencionadas e o desenvolvedor pode programar uma resposta da aplicação à interação do usuário.
O padrão **WorkWith** possui eventos predefinidos que são disparados dependendo da ação que realizamos sobre os dados da entidade à qual aplicamos o padrão. Entre eles estão os eventos Insert, Update, Delete, Save, Cancel, entre outros. Conforme a parte do objeto WorkWith que estamos executando (lista, detalhe, etc.) serão os eventos que estarão disponíveis
No caso de dispositivos móveis, conforme o tipo de dispositivo e a orientação, quando iniciamos a aplicação, será disparado um evento **Start** que depende da navegação. Por exemplo, se estamos usando um Tablet, por padrão a aplicação inicia em modo Split, o que significa que a tela é mostrada dividida em duas seções, com uma lista à esquerda e o detalhe do elemento selecionado à direita. Neste caso, quando a aplicação é iniciada, é disparado o evento Split.Start.
No caso dos telefones, a tela mostrará, por exemplo, apenas a lista e para ver o detalhe, será aberta outra tela independente. Este modo é denominado Flip e é o comportamento padrão nestes dispositivos, portanto, ao iniciar a aplicação, será disparado o evento Flip.Start.
Embora conforme o dispositivo exista uma navegação padrão, os objetos main de aplicações móveis possuem uma propriedade que nos permite alterar a forma em que se inicia a aplicação.
Estes eventos Start associados ao tipo de navegação são disparados no início da aplicação móvel e imediatamente após o evento ClientStart.

Exemplos de eventos do lado do cliente



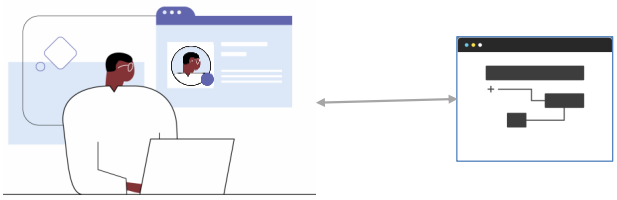



```
Event ClientStart
     TextblockTitleLine1.Caption = "The new age of"
     TextblockTitleLine2.Caption = "EXPLORATION"
Endevent

Event Load
     &Trips = Count(TripDate)
Endevent

Event Refresh
     &TotalTrips = 0
     For each Trip.Attraction
     Where CountryId = &CountryId when not &CountryId.IsEmpty()
     Where AttractionName >= &AttractionNameFrom when not &AttractionNameFrom.IsEmpty()
     Where AttractionName <= &AttractionNameTo when not &AttractionNameTo.IsEmpty()
          &TotalTrips += 1
     Endfor
Endevent

Event &CountryId.ControlValueChanged
     Refresh
Endevent

Event &AttractionNameFrom.ControlValueChanged
     Refresh
Endevent

Event &AttractionNameTo.ControlValueChanged
     Refresh
Endevent
```

No objeto que vimos antes, programamos apenas eventos associados a controles, não programamos nenhum evento de usuário.

Para cada variável dos filtros em tela, programamos seu evento ControlValueChanged, para disparar o método Refresh do Grid, o que fará com que sejam disparados os eventos Refresh e Load do servidor para atualizar o conteúdo do grid com os filtros inseridos.

O que podemos fazer em eventos no cliente



CLIENT

- Call to other panels
- Call to Rest Services
- Use of Business Component
- Call to WorkWith objects. In mobile call to Menu
- Call Externals Objects of GeneXus module For Angular apps see: https://wiki.genexus.com/commwiki/servlet/wiki?46456
- Call Subroutines
- We can do:
  - Control properties assignments
  - Simple or SDT variable assignment
  - For each Line and Selected Line in grids
  - Use If-Else, Do-Case and Do-While code blocks.

Vejamos em resumo o que podemos fazer em um evento do lado do cliente

- Podemos invocar outro objeto panel
- Também podemos invocar Serviços Rest e, quando invocamos Procedimentos ou Data Providers, automaticamente estes serão expostos como serviços Rest no servidor. Se esses procedimentos ou DPs tivessem sido chamados apenas a partir de um evento do lado do servidor (dentro de eventos Start, Refresh ou Load), não seriam expostos como serviços REST porque não seria necessário.
- Podemos usar Business Components para recuperar ou atualizar informações, neste caso também esses Business Components serão expostos como serviços Rest automaticamente.
- É possível invocar diretamente qualquer nó do padrão Work With. No caso de aplicações nativas, podemos invocar um objeto Menu.
- Podemos invocar os objetos externos definidos no módulo GeneXus. Algumas destas API's fazem sentido para uma plataforma específica, por exemplo, apenas para dispositivos móveis, outras apenas em aplicações web e outras podem ser utilizadas em ambas as plataformas. Para conhecer quais não podem ser acessadas por aplicações Angular, visite a página da wiki mostrada na tela (https://wiki.genexus.com/commwiki/servlet/wiki?46456)
- A partir de um evento no cliente, podemos chamar sub-rotinas
- E também podemos fazer:
  - Atribuição de propriedades aos controles
  - Atribuição de variáveis simples ou de variáveis SDT
  - Execução de For Each Line e For each Selected Line em grids
  - Uso dos blocos IF-Else, Do-Case e Do-While

Caso tentemos utilizar comandos não permitidos em um evento do lado do cliente, veremos um erro na tela de saída, para aquelas linhas que não obedecem às restrições da gramática, por exemplo, se tentarmos usar um comando For Each, um New ou qualquer tentativa de acessar ou modificar informações que são acessadas apenas a partir do servidor.

Gramática de eventos do lado do cliente

COMMANDS

Composite

<Control>.<Property> = <value>

If **<Bool_expr>**
Do case... endcase
Do while <Bool_expr>
Do-sub (except Menu for Smart Devices)

For each selected line

Simple variable assignation: &var = ***<expr>***

SDT or BC elements assignation:
&SDT.A = <value>
&BC.A = <value>

Return
Refresh

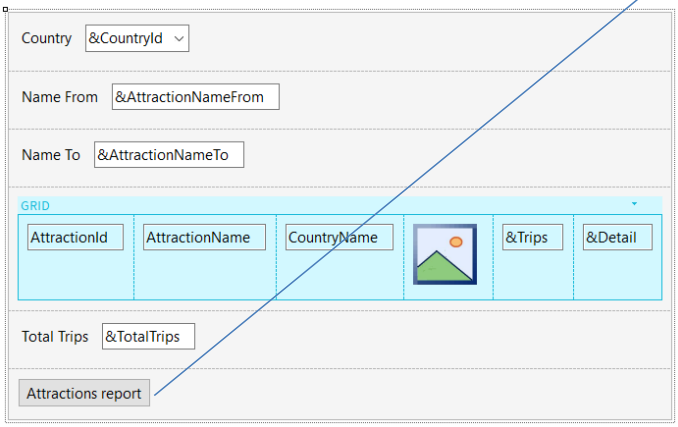
Inside an **expression**:
- Variables
- Attributes
- Constants
- Methods
- Functions
- Control properties
- Operators (+, -,, /, ^)

Aqui, vemos um resumo dos comandos que podemos usar no código dos eventos do lado do cliente.

Estas restrições são somente para os eventos do lado cliente, não para os eventos do lado do servidor (Start, Refresh e Load) onde podemos usar todos os comandos e funções disponíveis no GeneXus.

Os comandos aceitos atualmente são os mostrados.

Comando Composite




```
Event "Attractions report"
    Composite
        AttractionsByName("C", "M")
        Return
    EndComposite
Endevent
```

- Only for Client Side Events with more than 1 line of Code.
- Stop execution on Error.
- Automatic Error Handling.

Vejamos agora o comando composite que mencionamos anteriormente.

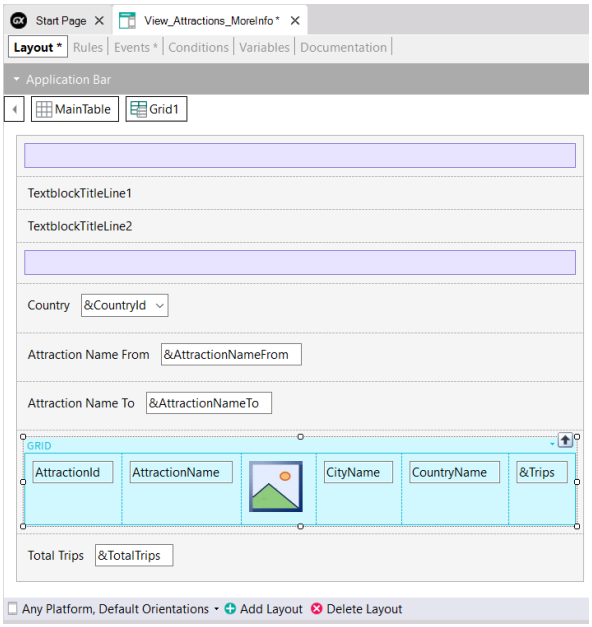
Este comando é utilizado em eventos do lado do cliente em objetos panel

A importância deste comando é que, quando ocorre um erro na sequência de chamadas, a execução é interrompida e são tratados os erros automaticamente, exibindo-os na tela sem a necessidade de implementação de nenhuma programação.

Esta é uma grande diferença com os webpanels, pois neles, quando dentro de um evento um objeto chamado produz um erro, não é interrompida a execução, continua na sentença seguinte e é o desenvolvedor quem deve se encarregar de tratar os erros e programar as ações a serem tomadas.

Este comando Composite é opcional, se não o utilizamos o funcionamento será idêntico ao dos webpanels.

Ordem de execução dos eventos



Fixed Part

Grid Part

Fixed Part

- ClientStart
- Only for mobile: Navigation.Start
- *Call to Fixed part Data Provider*
  - Start event
  - Refresh event
- ***Fixed Part is Drawn***
- *Call to Grid Data Provider*
  - Load event
- ***Grid Part is Drawn***

Agora vejamos o que ocorre com os eventos quando executamos um objeto panel.

Primeiro é executado o evento ClientStart, é executado apenas uma vez no cliente. No caso de aplicações móveis, em seguida, é disparado o evento Start correspondente ao tipo de navegação estabelecida no objeto main, mediante a propriedade Navigation Style.

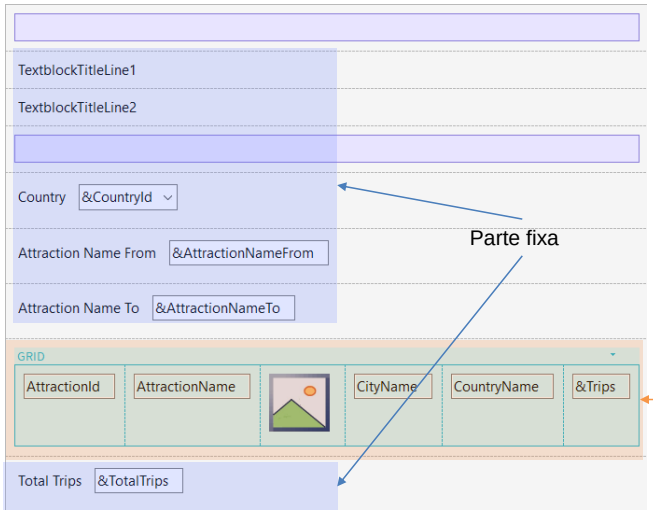
Em seguida, é executado um data provider que retornará os dados necessários para carregar a parte fixa do panel. Este data provider integra a execução do código dos eventos Start e Refresh que serão executados no servidor e retorna em um único resultado, a informação para carregar a parte fixa.

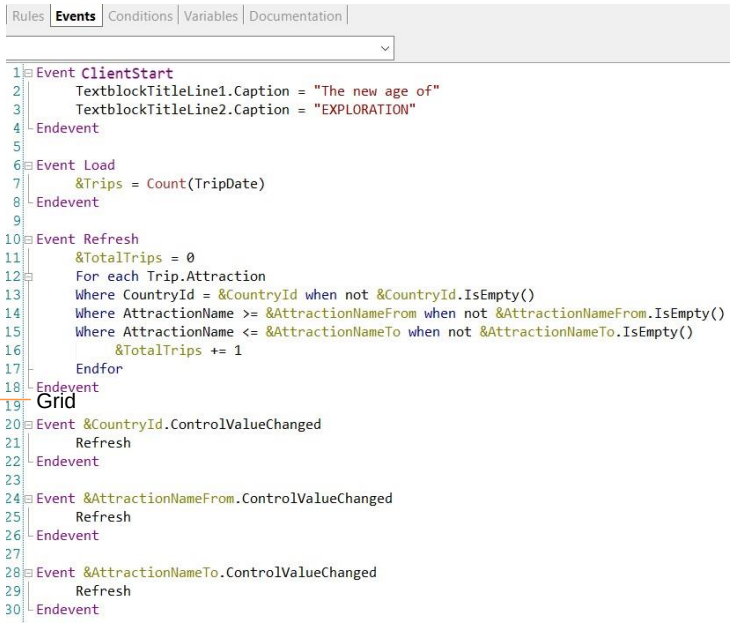
Em seguida, é desenhada a parte fixa do panel.

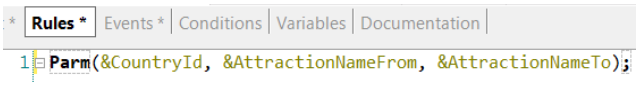
A seguir, é executado um segundo Data Provider que resolverá a recuperação dos dados requeridos pelo grid. Dentro da execução deste data provider é executado o código do evento Load no servidor. Este evento Load será executado N vezes quando o grid tiver tabela base, uma vez para cada registro, apenas uma vez se o grid não tiver tabela base e nenhuma vez se o grid for de uma variável SDT coleção.

Ao finalizar a execução do data provider, ele retornará as informações geradas por todas estas execuções do evento Load em um único resultado, com o que será carregado o grid, e então terminará de desenhar o grid.

Uso do comando Refresh em eventos do cliente







Uma das coisas que mencionamos foi que a parte fixa do panel é carregada de forma independente da carga do grid, invocando data providers diferentes, que são publicados no servidor como serviços e acessam a base de dados para recuperar as informações de cada parte.

Quando alteramos o valor de um filtro, é necessário atualizar as informações do grid. Para isto, é necessário que adicionemos o método Refresh, para que dispare os eventos Refresh e Load do servidor, que assumimos que fará com que seja carregado novamente o grid, aplicando as conditions programadas e mostre os resultados filtrados como esperamos.

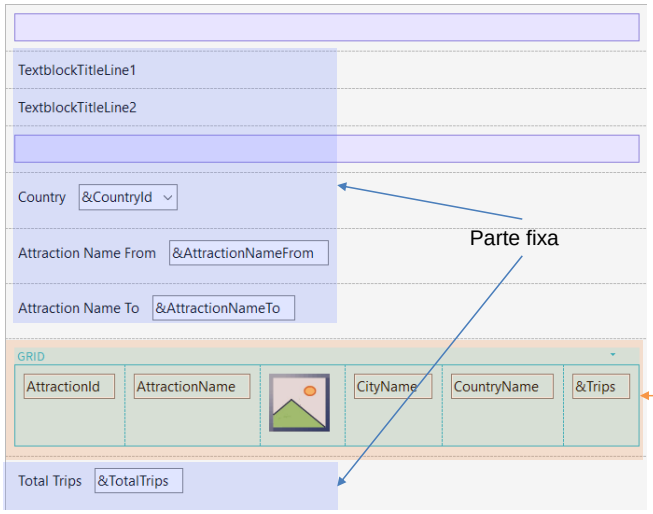
Como o comando Refresh devemos invocar após alterarmos o valor da variável do filtro, usamos o evento ControlValueChanged de cada variável para invocar o método. Desta forma, após alterar um valor do filtro, ao sair do campo será disparado o evento correspondente, que esperamos acabe atualizando o conteúdo do grid.

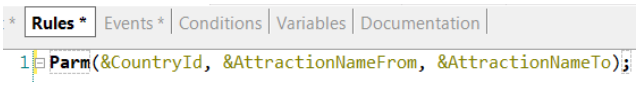
Mas o que acontece quando é invocado o comando Refresh dentro de um evento do lado do cliente?

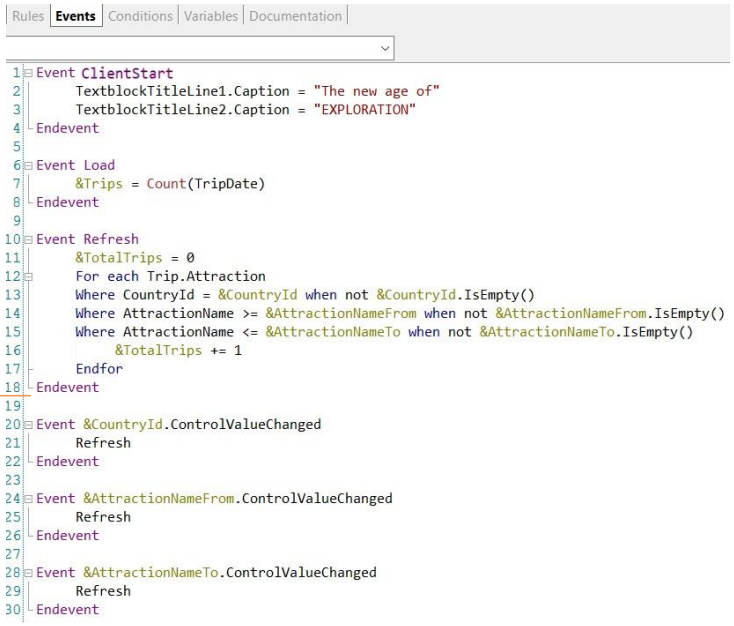
Nesta arquitetura, como o objetivo é que a página seja carregada a menor quantidade de vezes possível, prioriza-se o cache de dados, ou seja, que se trata sempre de recuperar as informações previamente armazenadas. Quer dizer que dependendo da configuração de armazenamento em cache, é decidido se deve ir ou não recuperar dados do servidor.

Caso seja decidido ir ao servidor, se não houver alterações nos dados do servidor, nada é devolvido ao cliente.
Caso contrário, são executados os eventos Refresh e Load (se houver um grid que não seja baseado em um SDT, como é nosso caso) e são atualizadas a parte fixa e a parte variável do panel, como esperávamos

Uso do comando Refresh em eventos do cliente (cont.)







Para que o servidor entenda que queremos trazer novos dados, adicionamos uma regra Parm, que contenha os valores das variáveis que usamos nos filtros, de forma que se altera o valor de uma variável, seja atualizada a página com os novos dados.
Executemos o panel para testar tudo isto que vimos.

Vemos que com os filtros vazios são mostradas todas as atrações e para cada atração é exibido corretamente o total de viagens, calculado no evento Load, que atualizou as informações do grid.

Vemos também que o total global de viagens que está na parte fixa é mostrado corretamente, pois foi calculado no evento Refresh e este foi executado antes de desenhar a parte fixa.

Se agora escolhemos as atrações do país França e optamos por ver as atrações que começam com a letra A até a letra N, vemos que o valor dos filtros presentes nas conditions do grid funciona corretamente e que os comandos Refresh invocados causaram o disparo dos eventos Refresh e Load, portanto as informações foram recuperadas do servidor corretamente atualizadas e somente vemos as atrações da França com o nome no intervalo escolhido.

Nos vídeos a seguir, veremos outros aspectos do design e implementação de objetos panel.

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