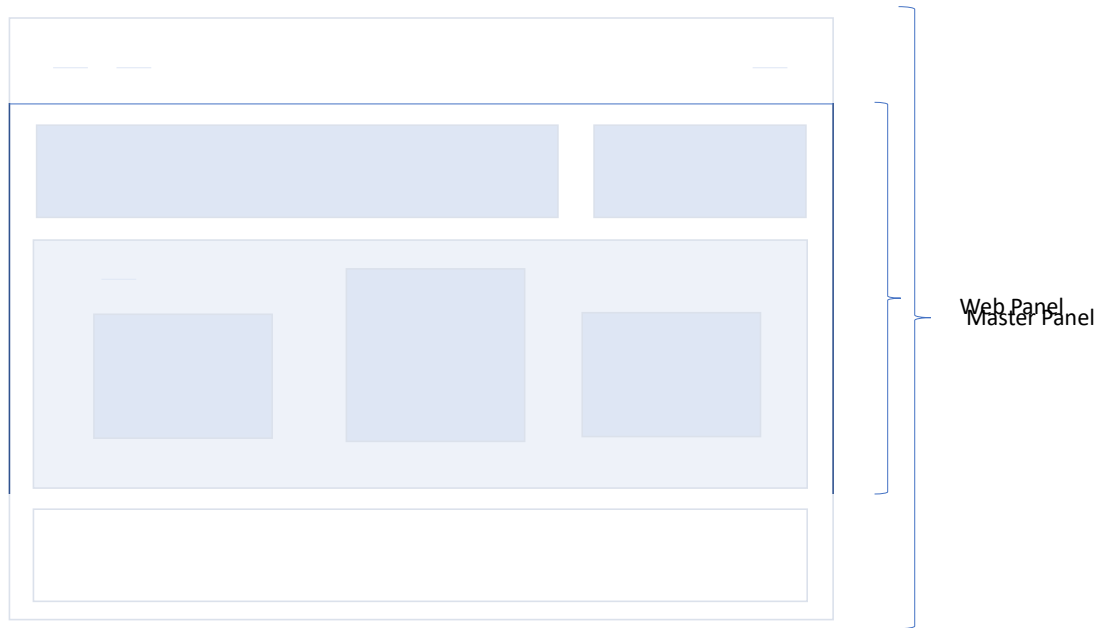


Eventos globais

Interação entre os componentes de uma mesma tela

GeneXus™

Screen composed by many screens



Como já sabemos, o uso de componentes permite reutilizar funcionalidades já desenvolvidas.

Por exemplo, é o caso de um web panel como o que já vimos, onde eram listadas as atrações turísticas que uma agência de viagens oferecia para visitar. Não trataremos aqui do desenho, que sem dúvida pode ser melhorado.

My favorite attractions

Travel Agency

Country Id

France


My favorite attractions: 0

Attraction Name From

Attraction Name To


Louvre Museum

Favorite



Eiffel Tower

Favorite



Estamos vendo uma tela não muito complexa, que apresenta uma parte fixa que vem da master page, e outra parte que é própria. Podemos filtrar por país e também por nome de atração.

Mas o que nos importa neste momento é observar que adicionamos a possibilidade de marcar atrações turísticas como favoritas. Por exemplo, marco esta e esta outra, ou seja, marcamos duas atrações como favoritas.

No entanto, se observarmos aqui acima, está indicando um zero, como se não tivéssemos marcado nenhuma como favorita.

Vamos atualizar a tela... e agora sim vemos o 2.

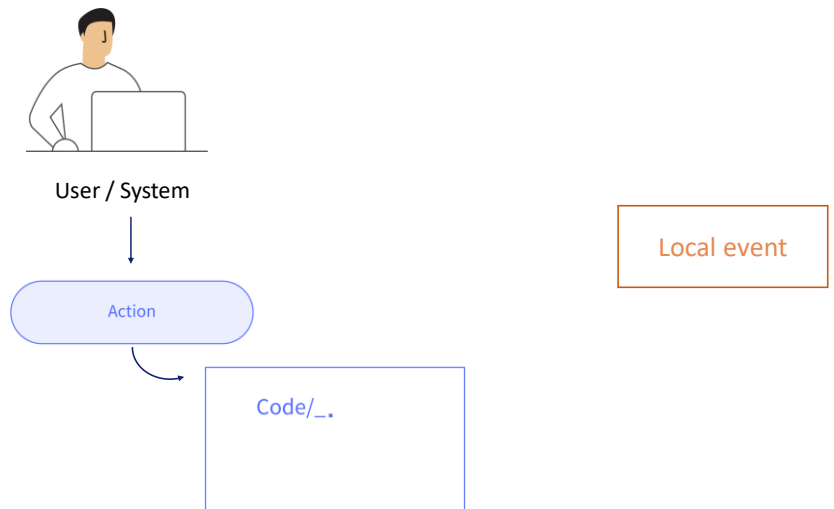
Pretendemos também que, se desmarcarmos uma atração favorita, este número seja atualizado indicando 1.

Então, como conseguimos que este valor seja atualizado automaticamente conforme seja marcada ou desmarcada uma atração turística como favorita?

Global Events

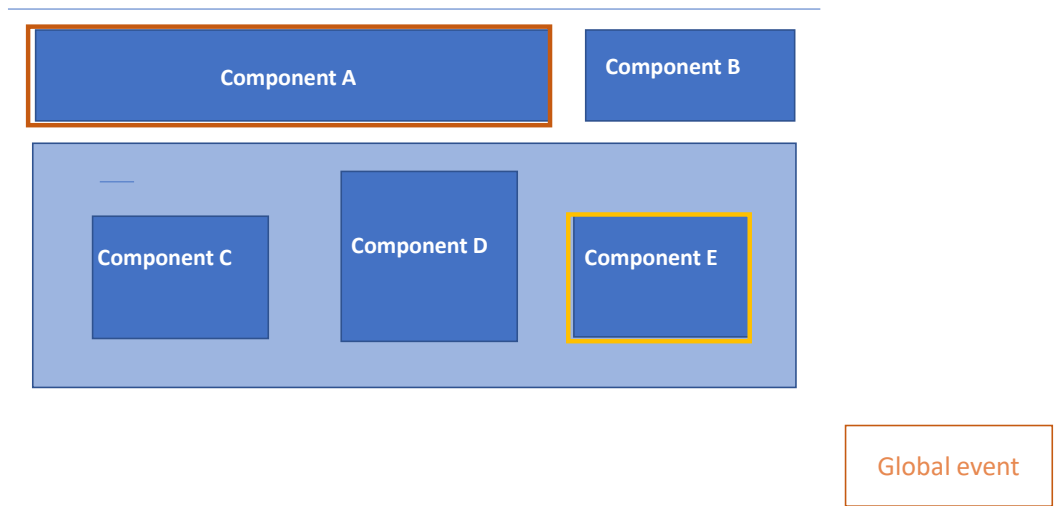
Bem. Vamos apresentar então o conceito de Evento global.

Events



Geralmente, em objetos interativos, como é o caso dos objetos web panels, os eventos são ações operadas pelo programa e acionadas pelo usuário. Os eventos costumam ser locais para o programa no qual são definidos.

Events



Em vez disso, os eventos globais permitem definir eventos que se aplicam a todos os componentes de uma aplicação.

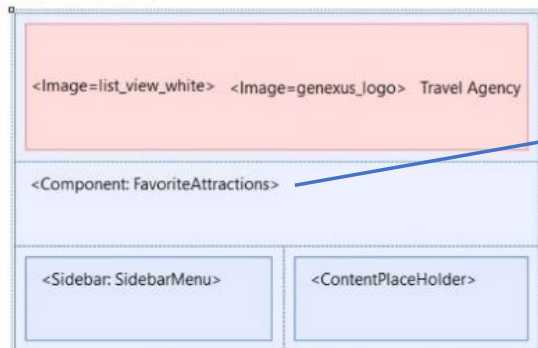
Enquanto os eventos locais são acionados em resposta a uma ação do usuário, os eventos globais são códigos que permanecem inativos até serem invocados a partir de outro evento, a partir de qualquer outro componente.

Na imagem que vemos, um evento global definido por exemplo no "Componente E" poderia ser invocado a partir do "Componente A". Qualquer combinação é possível, independentemente do nível de aninhamento dos componentes que compõem a tela.

My favorite attractions

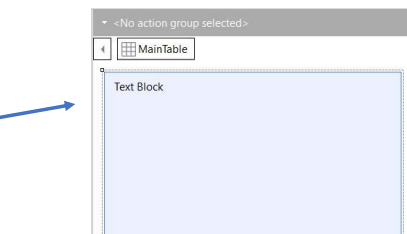
Master page: MasterUnanimosidebar

MainTable



Transaction: FavoriteAttraction

Name	Type	Description	Formula	Nullable
FavoriteAttraction	FavoriteAttraction	Favorite Attraction		
DeviceId	DeviceId	Device Id		No
AttractionId	Id	Attraction Id		No



```

1 Event Start
2   Do 'GetCount'
3 Endevent
4
5 Sub 'GetCount'
6   &FavoriteAttractionss = Count(AttractionId, DeviceId = ClientInformation.id, 0)
7   Textblock1.Caption = "My favorite attractions: " + &FavoriteAttractions.ToString
8 EndSub

```

Vamos ao GeneXus para ver este Web panel.

Vemos, antes de mais nada, que possui uma master page associada, que é a default. Portanto, o conteúdo de sua tela será carregado dentro do content place holder da referida master page.

Mas também adicionamos outro componente para mostrar a quantidade total de atrações turísticas que o usuário marcou como favoritas.

Para guardar as atrações que um usuário marca como favorita, deveríamos ter uma tabela de usuários, para poder indicar as atrações favoritas por usuário. Se ainda não quisermos trabalhar com usuários (por exemplo, porque ainda não aplicamos o GAM e ainda estamos resolvendo se teremos uma tabela própria de usuários ou não), então podemos provisoriamente manter favoritos por instância de browser.

Para fazer isso, criamos esta transação com este atributo, para o qual especificamos este novo domínio que nós também criamos, com o mesmo tipo de dados desta propriedade...

ClientInfomation é um external object do módulo GeneXus, cuja propriedade Id permite identificar o dispositivo cliente que está executando, tanto de forma nativa quanto web.

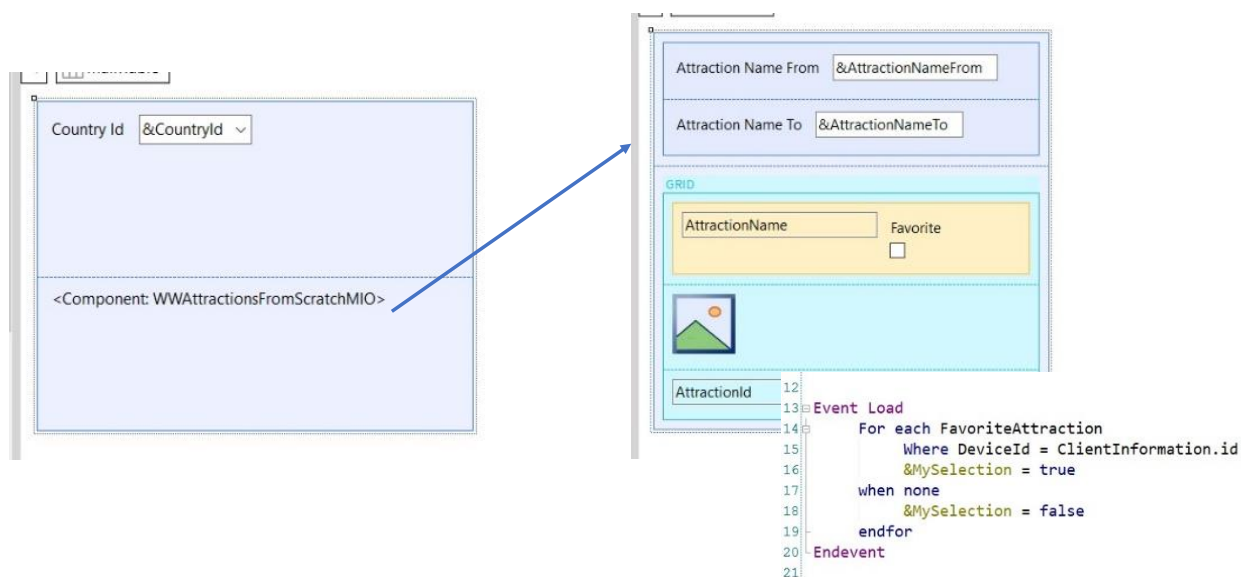
No caso de aplicações Web, a propriedade retorna um identificador do usuário, que persiste entre todas as sessões com o mesmo navegador e para a mesma aplicação. Então, o que fizemos foi adicionar uma transação com chave composta pelo

dispositivo e a atração turística. Na tabela associada a esta transação é que guardaremos os favoritos.

Se observamos o Web component que estamos carregando aqui, vemos que contém apenas um text block, com este nome, cuja caption carregamos em execução, apenas no evento Start, invocando esta sub-rotina que a primeira coisa que faz é calcular a quantidade de atrações turísticas para este cliente, que se encontram na tabela que contém estes dois atributos (que é FavoriteAttraction).

Aqui temos o valor que estamos buscando. Em seguida, transformamos esse numérico em string, para atribuí-lo ao textblock que será exibido.

My favorite attractions



Por outro lado, se observamos o que será carregado no ContentPlaceholder para nosso Web panel... vemos que é uma tela com este combo box para escolher um país, e abaixo temos outro componente, ao qual é enviado o valor desse país, sempre que o alterarmos.

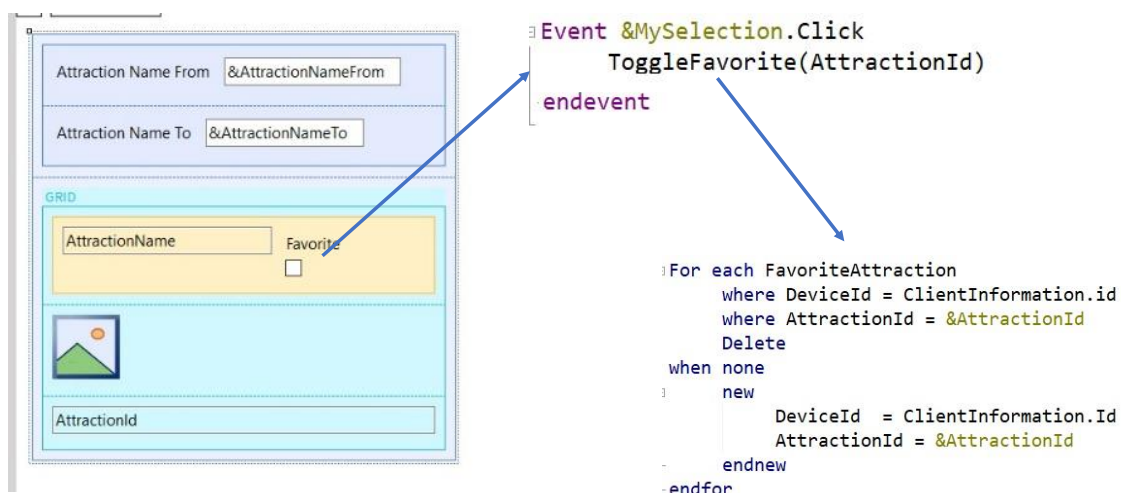
E se observarmos quem está sendo carregado nesse web component, é outro objeto web, de tipo componente, que tem um grid que filtra as atrações turísticas que vai mostrar, de acordo com o país recebido por parâmetro e os próprios filtros da tela.

Escolhemos um grid Flex, que contém uma tabela Canvas, para poder sobrepor a foto da atração turística juntamente com seu nome e uma variável booleana para permitir ao usuário marcar ou desmarcar a atração turística como favorita.

Sobre o desenho de tudo isto não vamos parar por aqui.

Apenas observemos que no evento Load do grid, que tem como tabela base Attraction, para cada atração turística que vai ser carregada, é executado este for each, que busca na tabela subordinada, FavoriteAttraction, se existe um registro para este dispositivo, o que está executando.

My favorite attractions



Mas disso tudo, o que verdadeiramente interessa é a programação do evento click nesta variável.

Uma vez carregado o grid de atrações, quando o usuário clica sobre o favorito de uma atração, chamamos um procedimento que verifica se a atração já estava marcada como favorita, caso em que a remove da tabela. E se for ao contrário, faz o oposto, ou seja, a insere.

Agora, ao fazer isto, o total de atrações favoritas deveria ser atualizado.

Ou seja: queremos que um evento de usuário de um componente carregado neste panel permita realizar uma ação sobre um componente com o qual não tem outra relação a não ser encontrar-se indiretamente reunidos na mesma tela.

External object: GlobalEvents

Structure	Type	Is Collection	Description
GlobalEvents			External Object for defin...
Properties			
Methods			
Events			
UpdateAttractionFavorites	None	<input type="checkbox"/>	

Event trigger

```

Event &MySelection.Click
    ToggleFavorite(AttractionId)
    GlobalEvents.UpdateAttractionFavorites()
endevent

```

É então que para permitir a interação, toda kb conta com este objeto predefinido, Global Events, para que possamos modificá-lo.

Poderíamos salvá-lo com outro nome e criar quantos objetos particulares quisermos.

No nosso caso, vamos modificar o predefinido, que vem vazio, adicionando um evento que chamaremos assim.

Os eventos que vão sendo adicionados poderão manipular parâmetros, caso sejam necessários para a comunicação entre componentes.

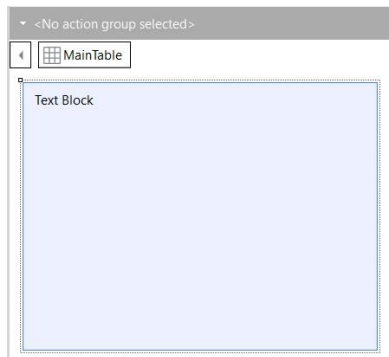
No nosso caso, precisamos apenas que um fique sabendo que outro o disparou. Não precisamos de parâmetros.

Então, temos esta forma global de comunicação, então vamos utilizá-la.

Vamos ao panel que deverá disparar o evento. O disparo será cada vez que for clicado sobre a variável. Neste momento invocamos o objeto externo GlobalEvents, evento: o único que existe no momento. Com isso, está sendo reportado o disparo do evento.

External object: GlobalEvents

Structure	Type	Is Collection	Description
GlobalEvents			External Object for definin...
Properties			
Methods			
Events			
UpdateAttractionFavorites	None	<input type="checkbox"/>	



Listen event

```

|Event Start
  Do 'GetCount'
-Endevent

|Sub 'GetCount'
  &FavoriteAttractionss = Count(AttractionId, DeviceId = ClientInformation.id, 0)
  Textblock1.Caption = "My favorite attractions: " + &FavoriteAttractions.ToString()
-endSub

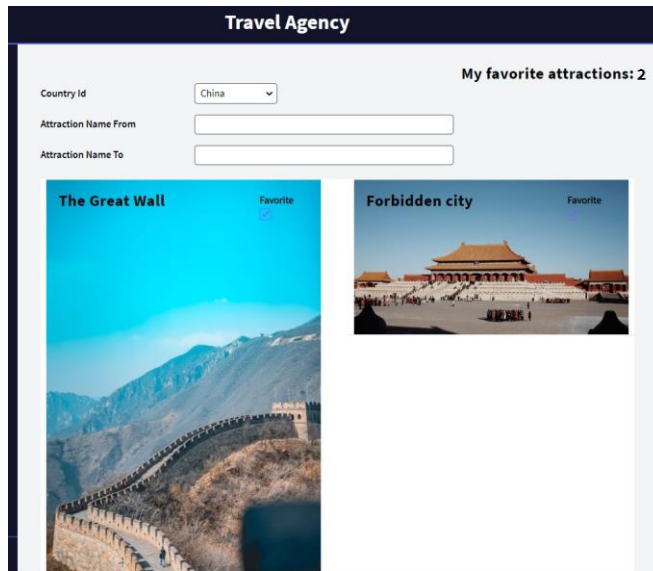
|Event GlobalEvents.UpdateAttractionFavorites()
  Do 'GetCount'
-endevent

```

Então, o que temos que fazer é, que o componente que conta as atrações para mostrá-las se inscreva a este evento, para que possa escutá-lo sempre que se produza.

Para fazer isso, simplesmente declaramos escutar o evento. E ali programamos o que queremos que seja executado quando o evento ocorre, que neste caso é calcular novamente a quantidade de atrações.

Run...



Vamos executar para testar.

Desmarcamos e vemos que está sendo atualizado. Agora marcamos, e esta outra... A comunicação está sendo efetuada com sucesso.

Embora aqui tenhamos visto um exemplo de comunicação entre painéis web com componentes, o mesmo vale para comunicar painéis multi-experience com componentes, por exemplo, para aplicações nativas.

Para aprender mais sobre este tema, você pode ir à nossa wiki.

*GeneXus*TM

training.genexus.com
wiki.genexus.com