Subroutines

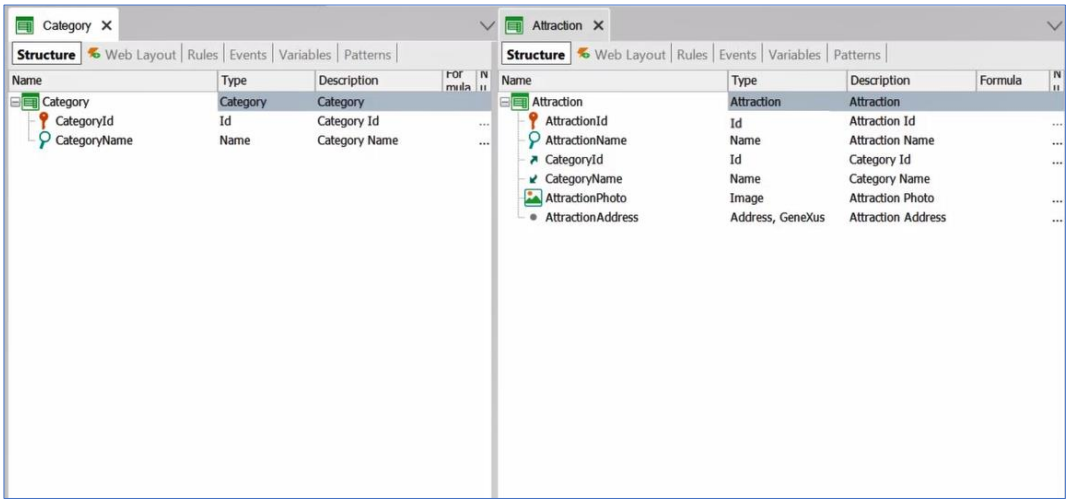
GeneXus™

Veremos neste vídeo o que são as sub-rotinas em GeneXus. Veremos seu funcionamento e implementação, e analisaremos para que nos podem ser úteis.

Uma sub-rotina é basicamente um bloco de código, o qual poderemos invocar quantas vezes quisermos, desde que esteja dentro do mesmo objeto.
Desta forma, poderemos executar esse mesmo bloco a partir de vários locais do objeto, ou a partir de um mesmo local, mas em várias ocasiões. A sub-rotina nos permite escrever esse código apenas uma vez, atribuir um nome a ele e, em seguida, simplesmente invocá-la pelo nome fornecido.

As sub-rotinas podem ser utilizadas em todos os objetos que aceitem programação, com exceção dos Data Providers. São eles: Web Panels, Procedimentos, Panels, Transações.

Vejamos seu funcionamento em um exemplo simples.



Faremos isso em uma aplicação que estamos fazendo para uma agência de viagens, entre suas transações temos uma chamada Category, para registrar as diferentes categorias, e outra chamada Attraction. Nesta última, além de seus próprios atributos, possui os atributos CategoryId e CategoryName da transação Category.

Vejamos um dos Web Panels da aplicação e analisemos sua implementação.

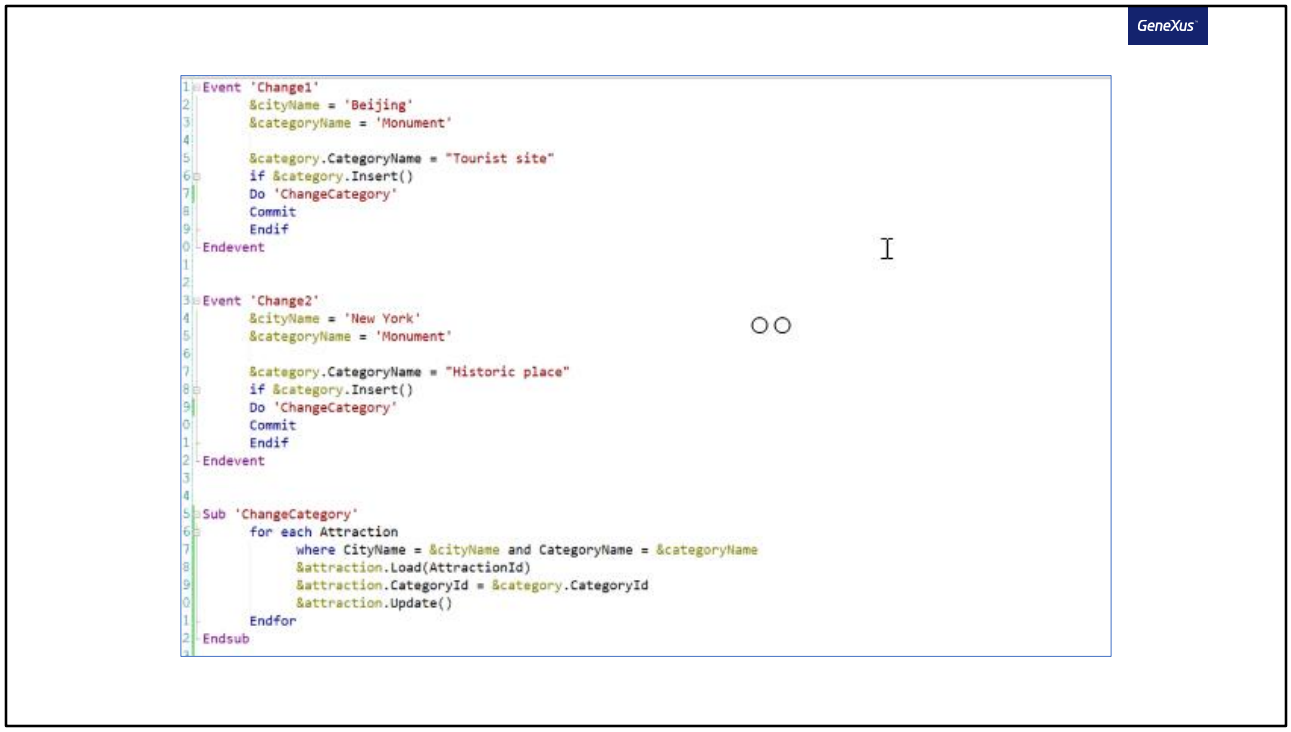

```
Event 'Change1'
      &cityName = 'Beijing'
      &categoryName = 'Monument'

      &category.CategoryName = "Tourist site"
      if &category.Insert()
            for each Attraction
                  where CityName = &cityName and CategoryName = &categoryName
                  &attraction.Load(AttractionId)
                  &attraction.CategoryId = &category.CategoryId
                  &attraction.Update()
            Endfor
      Commit
      Endif
Endevent


Event 'Change2'
      &cityName = 'New York'
      &categoryName = 'Monument'

      &category.CategoryName = "Historic place"
      if &category.Insert()
            for each Attraction
                  where CityName = &cityName and CategoryName = &categoryName
                  &attraction.Load(AttractionId)
                  &attraction.CategoryId = &category.CategoryId
                  &attraction.Update()
            Endfor
      Commit
      Endif
Endevent
```

No Layout, vemos dois botões, um denominado "**Change category Monument** in Beijing" e o outro "**Change category Monument** in New York". O primeiro terá como nome de evento Change1 e o segundo Change2.

Na seção eventos, vemos o código atribuído precisamente a estes dois eventos. Vejamos o primeiro. São declaradas duas variáveis, cityname e categoryname, e atribuído o texto Beijing e monument respectivamente.
Então temos uma variável category, que é do tipo Business Component Category, à qual é informado que o nome da categoria será Tourist site. E mais tarde tentará inserir esta nova categoria na tabela Category.
Se foi inserido o registro corretamente, então será percorrida a tabela attraction, filtrando apenas os registros onde CityName tenha o mesmo valor da variável CityName e CategoryName tenha o mesmo valor da variável categoryName.
Para esses registros, será atualizada a categoria da atração, mudando para a que acabamos de inserir, neste caso Tourist Site.

O segundo botão fará o mesmo que o primeiro, mas com dados diferentes. O nome da categoria a ser inserida será Historic Place, e serão filtradas as atrações de New York que tenham como categoria Monument.

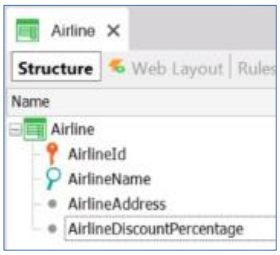
Se observarmos o primeiro e o segundo eventos, têm bloco de código exatamente igual.
Neste caso, poderíamos declarar esse código em um único lugar e simplesmente chamá-lo a partir do evento que quisermos. Vamos fazer.



O comando Sub nos permitirá definir uma sub-rotina, e então devemos atribuir um nome a ela. Com o endsub marcamos a sua finalização.
Dentro, devemos inserir o código que invocaremos mais tarde. O copiamos e o colamos aqui.
Removemos esse bloco dos eventos e, devemos em seu lugar chamar a sub-rotina. Fazemos isto por meio do comando Do, seguido do nome.
Com esta implementação, o funcionamento será exatamente o mesmo que o de antes de declarar a sub-rotina.

Desta maneira, conseguimos modularizar nosso código, tornando-o mais claro e facilitando sua leitura. Outra vantagem é que, se devemos alterar algo deste bloco, fazemos apenas uma vez e é aplicado em todos os locais onde é chamado. E desta forma, temos a possibilidade de reutilizar este código através da sub-rotina quantas vezes quisermos, desde que esteja dentro deste mesmo objeto, neste caso o Web Panel InsertCategoriesAndAttractions.

Vejamos agora este outro exemplo, para entender um pouco mais seu funcionamento.

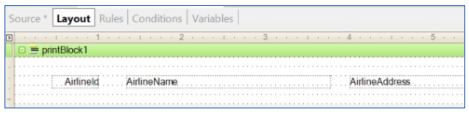



```
&Id = 1

For each Airline
      Where AirlineId = &Id
      AirlineAddress = '77 West Wacker Drive, Chicago'
      &NextId = AirlineId + 1
      do 'ChangeName'
Endfor

sub 'ChangeName'
      for each Airline
            where AirlineId = &NextId
            AirlineName = 'American Airlines, Inc'
      endfor
endsub
```

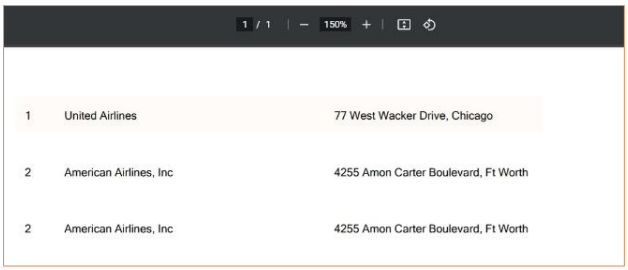
Em nossa aplicação, temos a transação Airline, onde registramos as diferentes companhias aéreas, com os seguintes atributos.

Neste objeto procedimento, temos o seguinte código no source.
Através deste for each, será percorrida a tabela Airline, filtrando pela companhia aérea com id 1, pois é o valor que atribuímos a esta variável.
Para esse registro, queremos atualizar seu endereço, através do atributo AirlineAddress.
Então temos uma variável chamada nextId, à qual atribuiremos o valor de AirlineId mais um, neste caso será dois. E depois chamamos a sub-rotina ChangeName.
Esta sub-rotina realiza um for each, também na tabela Airline, filtrando pelo registro que tenha AirlineId igual a dois.
Se esse registro for encontrado, seu nome será atualizado.

Agora, quando termina de ser executada a sub-rotina, e retorna ao for each principal, em qual registro estaremos parados? Se aqui tivéssemos um atributo da transação Airline, por exemplo este (AirlineDiscountPercentage), e atribuíssemos um valor a ele, a qual registro aplicará? Ao registro com id um, que é onde estávamos parados antes de chamar a sub-rotina? Ou ao registro com id dois?




```
&Id = 1

For each Airline
      Where AirlineId = &Id
      AirlineAddress = '77 West Wacker Drive, Chicago'
      &NextId = AirlineId + 1
      Print PrintBlock1
      do 'ChangeName'
      Print PrintBlock1
      AirlineDiscountPercentage = 25
Endfor

sub 'ChangeName'
      for each Airline
            where AirlineId = &NextId
            AirlineName = 'American Airlines, Inc'
            Print PrintBlock1
      endfor
endsub
```



Vejamos isto declarando três atributos no layout, para mostrar o Id, o nome e o endereço da companhia aérea.

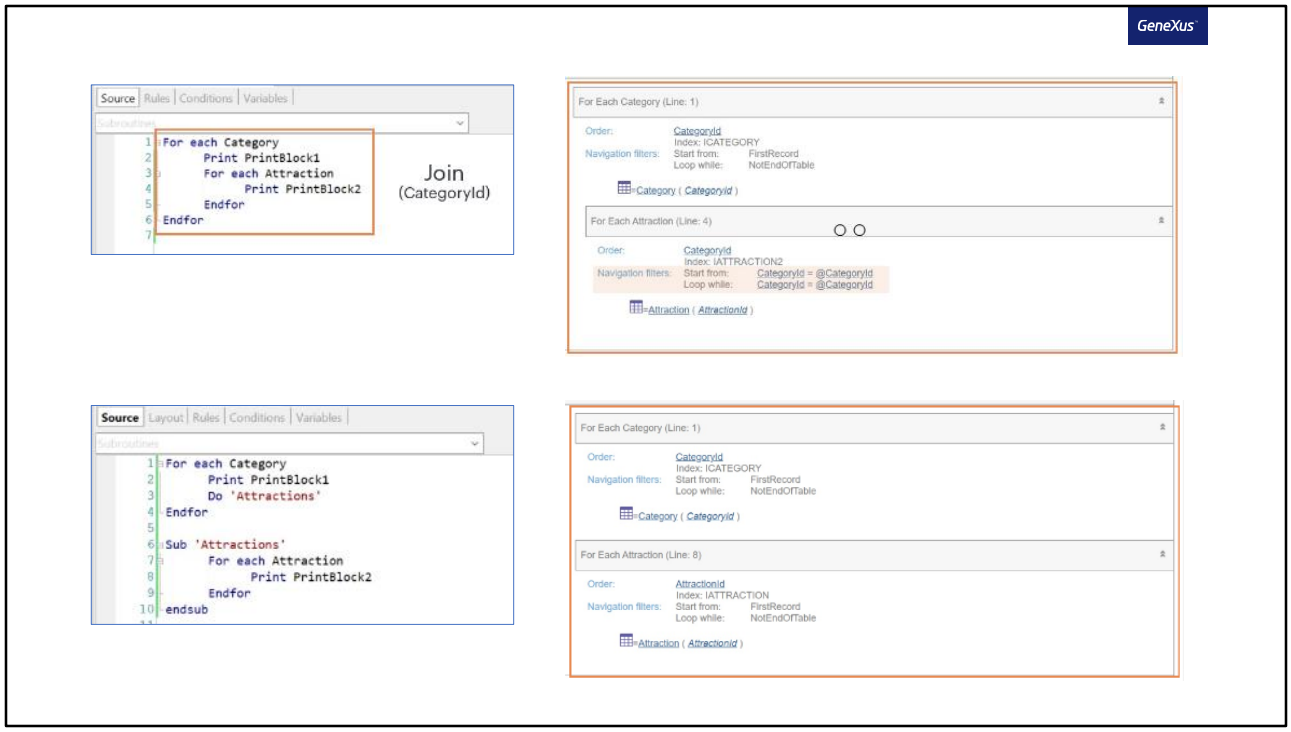
E no source, adicionamos três Print Printblock, para ver em qual companhia aérea estamos posicionados em cada momento. Colocamos um antes de chamar a sub-rotina, outro durante a execução da sub-rotina e outro imediatamente após sairmos dela.

Executemos para testar.
Vemos que no primeiro print, antes de chamar a sub-rotina, estamos posicionados na companhia aérea com Id 1, já com o endereço atualizado.
O segundo print executado corresponde ao que está dentro da sub-rotina. Vemos que é impresso na tela o registro com o Id dois, e já com o novo nome atualizado.
O terceiro print é executado assim que saímos da sub-rotina. E vemos que, nesse momento, seguimos posicionados no registro com Id dois, que é onde estávamos dentro da sub-rotina. E não no registro com Id um, que é onde estávamos posicionados antes de chamar a sub-rotina.
Portanto, se neste momento atualizamos o atributo AirlineDiscountPercentage com um valor, isso será feito no registro com Id dois.
Este funcionamento ocorre porque os atributos declarados serão globais do objeto. Portanto, se um atributo assume um valor em uma determinada seção de um objeto, e posteriormente é chamada uma sub-rotina que também atribui um valor ao mesmo atributo, ao retornar da sub-rotina invocada e consultar o valor do atributo, ele terá o valor atribuído na sub-rotina. Acabamos de ver isto com o atributo AirlineId.

As sub-rotinas não suportam a passagem de parâmetros, portanto, para troca de dados utilizamos variáveis, as quais são globais para os objetos.

Se não quiséssemos este comportamento que acabamos de ver, em vez de utilizar uma sub-rotina, poderíamos chamar um procedimento, por exemplo. Passando, nesse caso sim por parâmetro, a variável pela qual queremos então utilizar para filtrar.

Vejamos agora um terceiro exemplo.



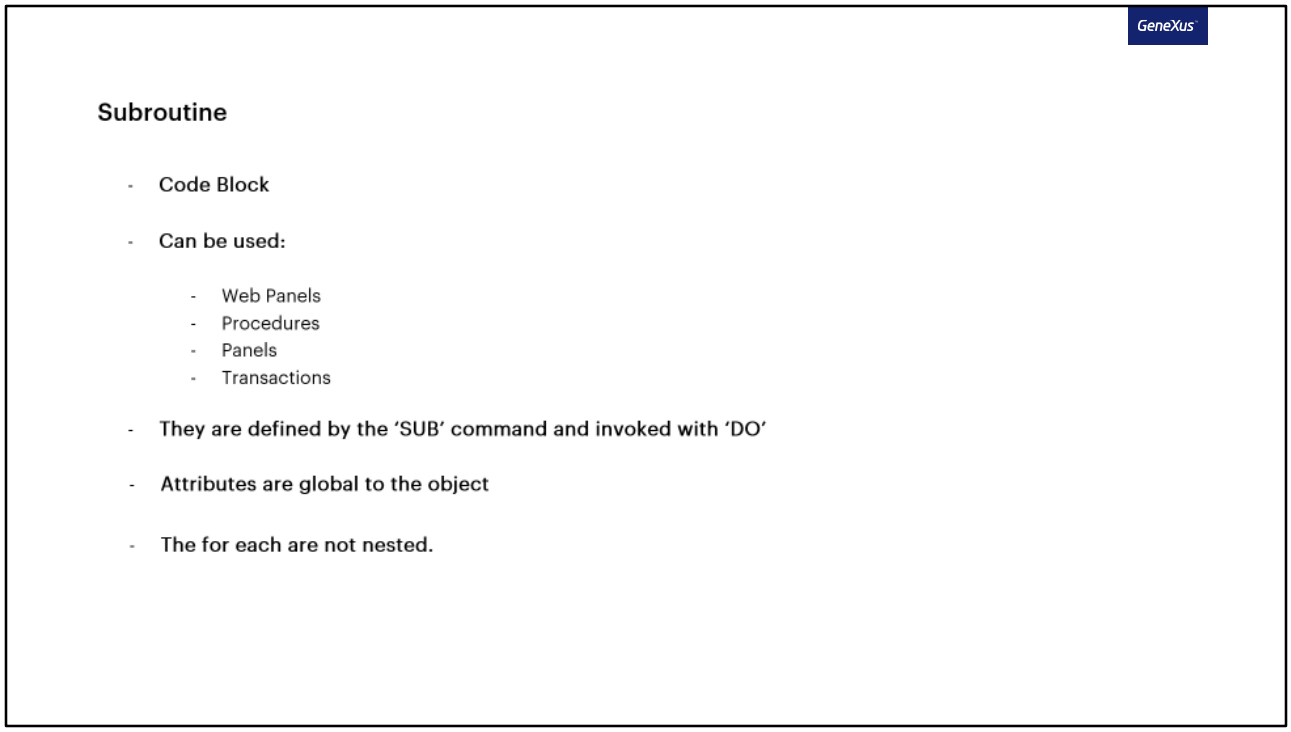
Neste caso temos o seguinte procedimento, que realiza um for each que navega na tabela Category, e um for each aninhado que navegará na tabela Attraction. Como sabemos, neste caso GeneXus realizará um Join por CategoryId. Uma vez que toda atração terá uma categoria atribuída. E CategoryId é o atributo comum que permitirá unir ambas as tabelas. Vejamos isto na lista de navegação.

Voltando ao source do procedimento vemos que implementamos dois prints. Um imprimirá o nome da categoria e o outro o nome da atração.

Se executamos, vemos que realmente é impresso o nome da categoria, e dentro as atrações que tenham essa categoria associada.
Agora, no caso em que não quiséssemos este comportamento, ou seja, que não se realize o Join, mas que se imprima a categoria e depois se imprimam todas as atrações, independentemente de qual categoria pertençam. Como poderíamos implementar?
Uma opção é colocar este código dentro de uma sub-rotina.
Vejamos agora a lista de navegação.
Vemos que efetivamente é percorrida toda a tabela Category e, em seguida, é percorrida toda a tabela Attraction, do primeiro ao último registro, sem aplicar qualquer tipo de filtro.

Desta forma, GeneXus já não realiza a inferência automática e não faz o filtro por CategoryId. Serão duas navegações independentes.

Até aqui vimos em diferentes exemplos o uso e funcionamento das sub-rotinas.



Façamos um pequeno resumo.
As sub-rotinas:

- São blocos de código que nos permitem modularizar o mesmo. Podendo invocá-las quantas vezes quisermos dentro do mesmo objeto.
- Podem ser utilizadas em Web Panels, Procedimentos, Panels, Transações
- São definidas mediante o comando SUB e então invocadas mediante o comando DO.
- Não suportam a passagem de parâmetros, para trocar dados são utilizadas variáveis.
- Se um atributo tem um valor e ao chamar a sub-rotina ele muda. Ao retornar da sub-rotina e consultar o valor, terá o que foi atribuído nela, já que os atributos são globais para o objeto.
- Se for chamada de dentro de um for each e a sub-rotina também tem um comando for each, eles não serão aninhados, ou seja, não serão feitas inferências nem filtros.

Para mais informações sobre este tema, convidamos a visitar nossa Wiki.


GeneXus™