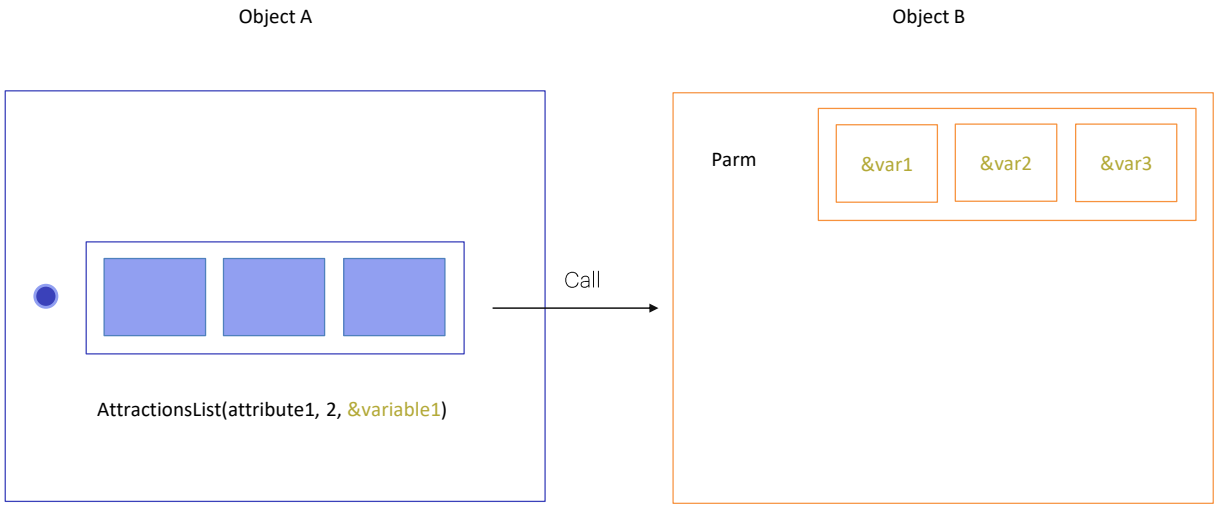
Invocações entre objetos (Cont.)

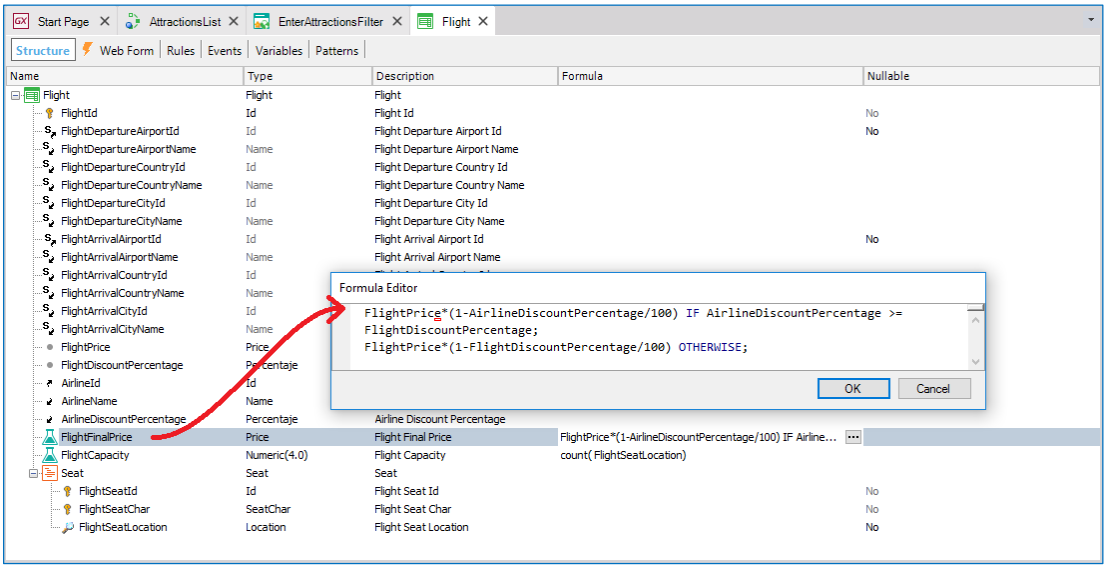
GeneXus™



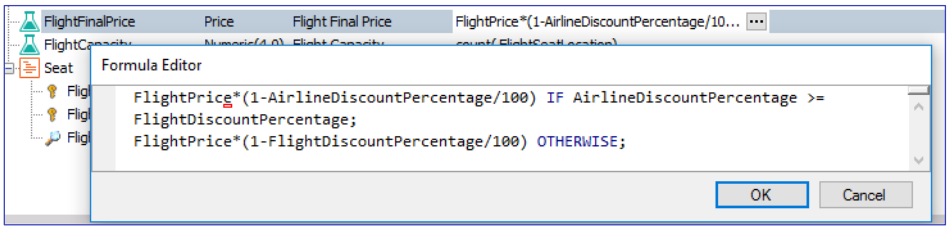
No vídeo anterior vimos como declarar num objeto parâmetros para permitir receber dados de outro objeto e realizar as ações pertinentes de acordo a esses dados. Para isso, fizemos o uso da regra Parm e de variáveis. Os exemplos que vimos eram com parâmetros de entrada, isto é, parâmetros que o objeto apenas recebe.

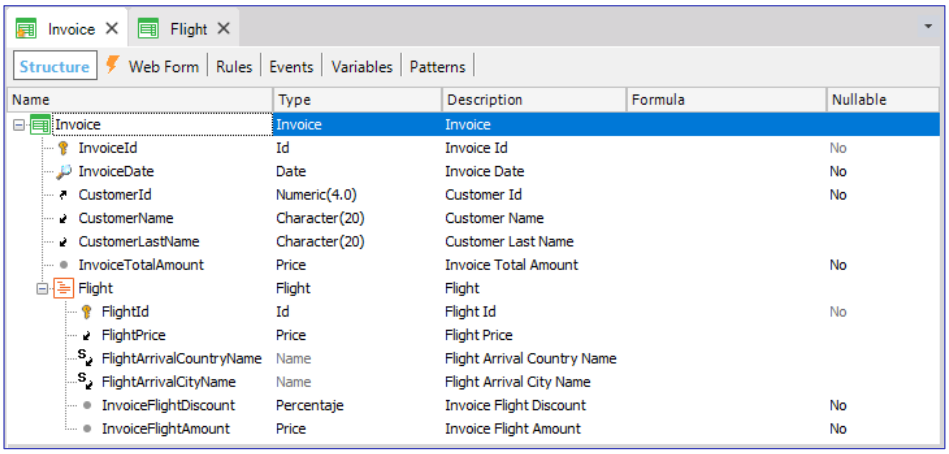
Se temos o objeto B com uma regra Parm declarada com três variáveis, todo objeto que queira chamar B deverá enviar os três valores, que como apredemos, poderão estar armazenados em atributos, ser uma expressão (como o caso de um valor fixo), ou estar armazenados em variáveis.

Agora veremos o que acontece quando o objeto B tem que devolver um valor ao chamador, ao finalizar a sua execução.



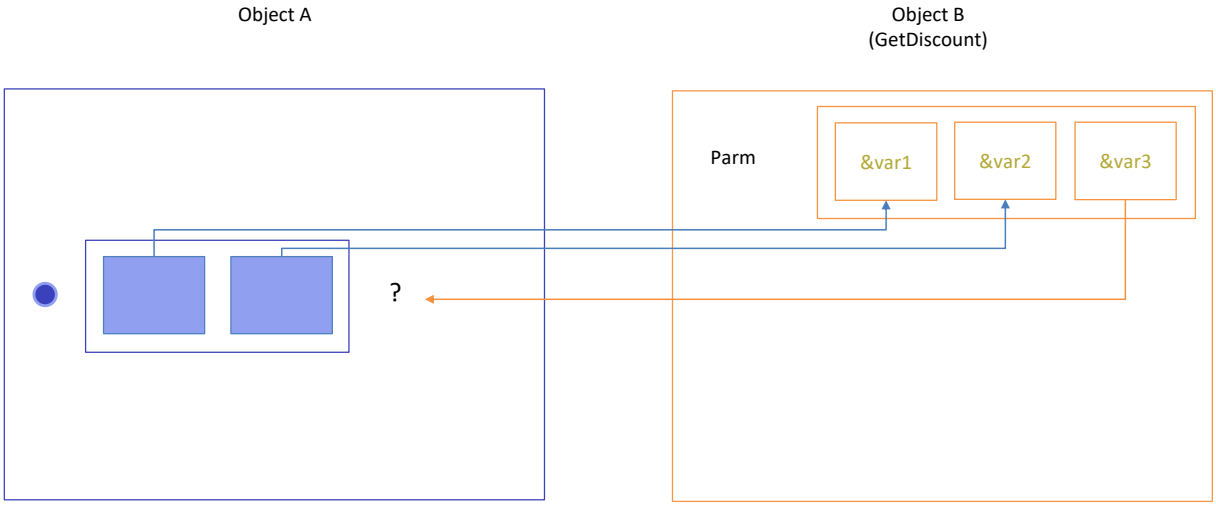
Tínhamos na transação Flight uma fórmula que calculava o preço de um voo de acordo a porcentagem de desconto que a companhia aérea oferecia e a porcentagem especificada para o próprio voo. O melhor desconto
era escolhido e esse era o desconto a ser aplicado.





Suponha que estamos criando uma transação para registrar as faturas que são emitidas aos clientes quando compram passagens aéreas.

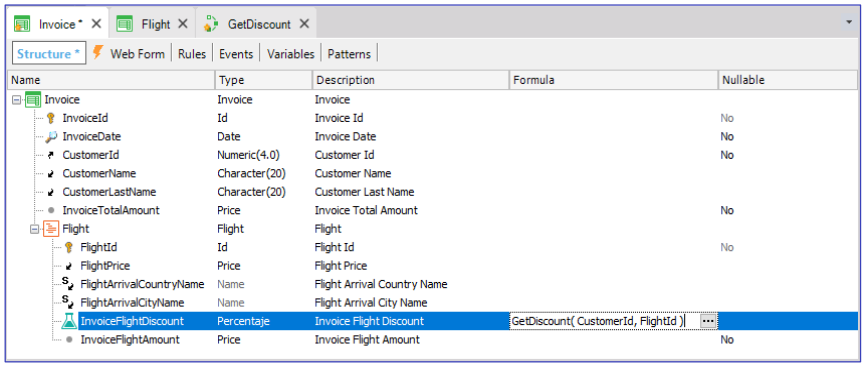
E suponhamos que o desconto é um cálculo mais complexo, que implica não só a passagem, senão, por exemplo, alguma condição relativa ao cliente que está comprando a passagem aérea. Por exemplo, a quantidade de passagens compradas antes, se o cliente é fiel, e, por exemplo, se um destino está em oferta. Dependendo de todas essas condições mais complexas, determina-se o percentual de desconto.

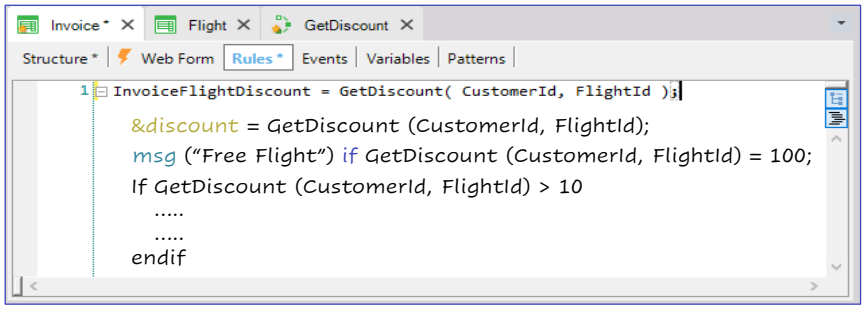


Para casos como esse, o interessante seria implementar um procedimento que realize os cálculos e devolva o valor resultante ao objeto chamador. Por exemplo, poderíamos criar um procedimento denominado GetDiscount.

Por ejemplo, podríamos llamarle a nuestro procedimiento GetDiscount.

O procedimento deverá receber como parâmetros de entrada um cliente específico e o voo desejado.
E devolverá o valor do desconto resultante.
A primeira pregunta é como receber o resultado do procedimento no objeto chamador? Temos que pensar como uma função, que chamamos e depois fazemos algo com o resultado devolvido.





Uma possibilidade é atribuir o resultado para um atributo. Por exemplo, poderíamos definir o atributo FlightDiscount na estrutura da transação Invoice como um fórmula que é calculada chamando o procedimento GetDiscount.

Dessa forma, em todo o objeto que o atributo InvoiceFlightDiscount for usado, a fórmula será calculada chamando o procedimento GetDiscount, que será executado, devolvendo ao finalizar o resultado que será exibido como valor do atributo fórmula.

Se não desejarmos definir esse atributo como fórmula, deixá-lo apenas como um atributo armazenado na tabela correspondente, e que somente ao executar a transação se amarzene o resultado do procedimento, escreveríamos nas regras assim.

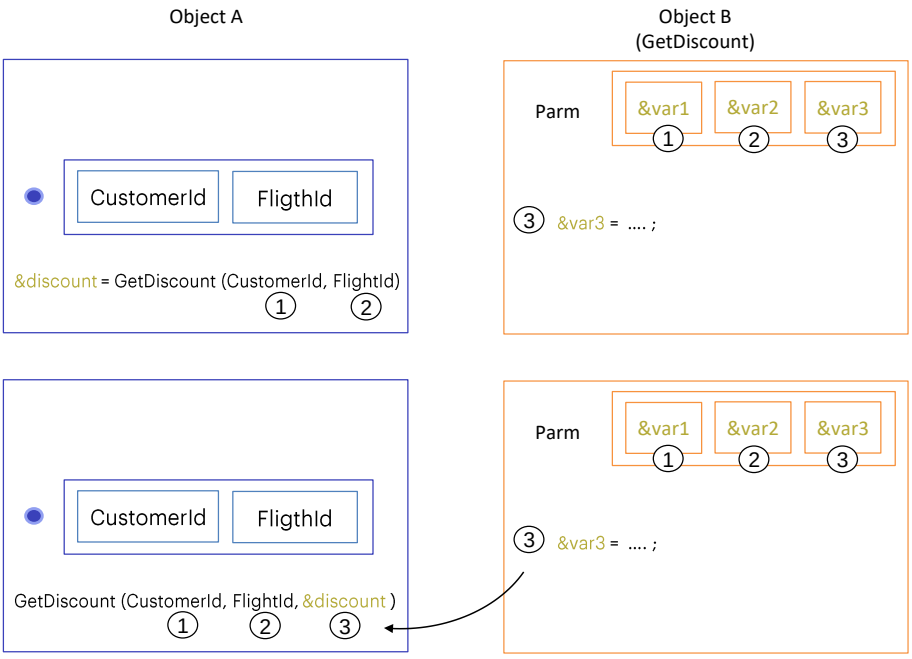
Também poderíamos atribuir o resultado da execução do procedimento em uma variável.

Ou não atribuir o resultado a nada, usá-lo apenas com uma expressão. Por exemplo, para condicionar o disparo de uma regra.

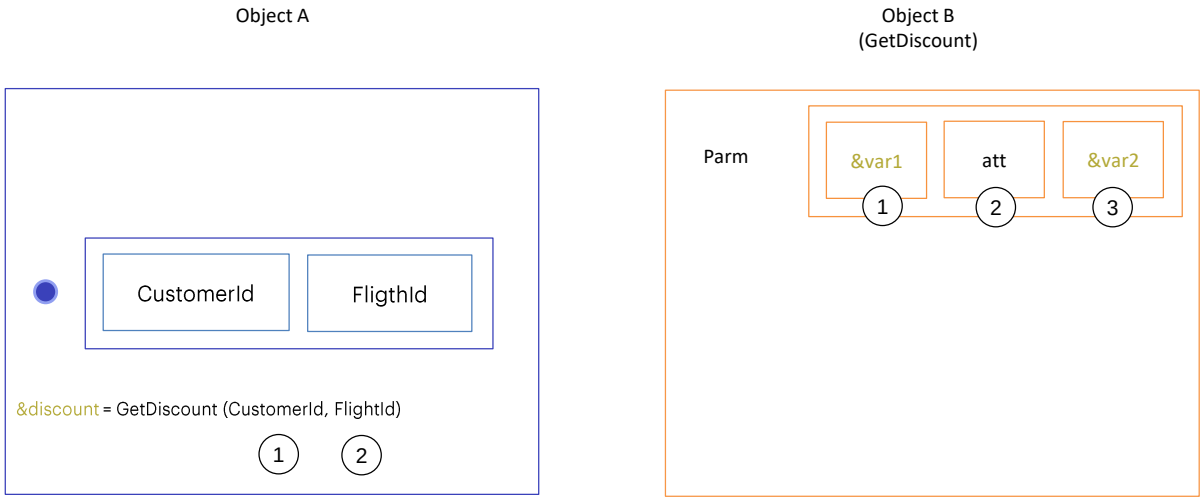
Na codificação de um procedimento ou na programação de um evento.

```
If GetDiscount( CustomerId, FlightId) > 10
    ...
Endif
```

Não veremos como implementar o procedimento GetDiscount, pois não tem relevância no tema estudado. Mas é importante que vejamos como é declarado no objeto chamado a regra Parm quando na sintaxe da chamada
assume-se que o objeto devolve um valor, como é o caso dos exemplos que acabamos de explorar.



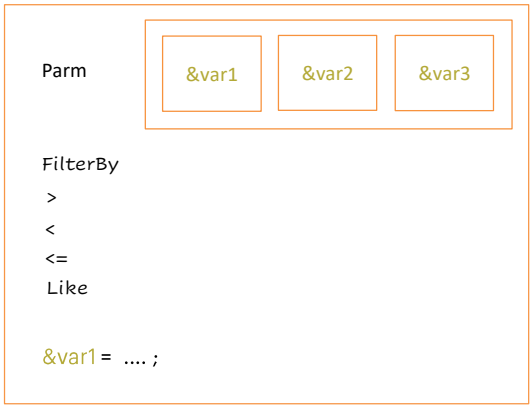
Na seção de regras do procedimento GetDiscount deveremos declarar a regra Parm com a quantidade de parâmetros escritos na chamada. Com mais um parâmetro no final que deverá ser uma variável cujo valor seja carregado no código do objeto (em nosso caso, no Source do procedimento). O valor com o qual ficará a variável no final da execução do código será o valor devolvido para o objeto que o chamou.



Por último, abordaremos o caso em que um parâmetro da regra Parm será usado como atributo no lugar de ser uma variável.

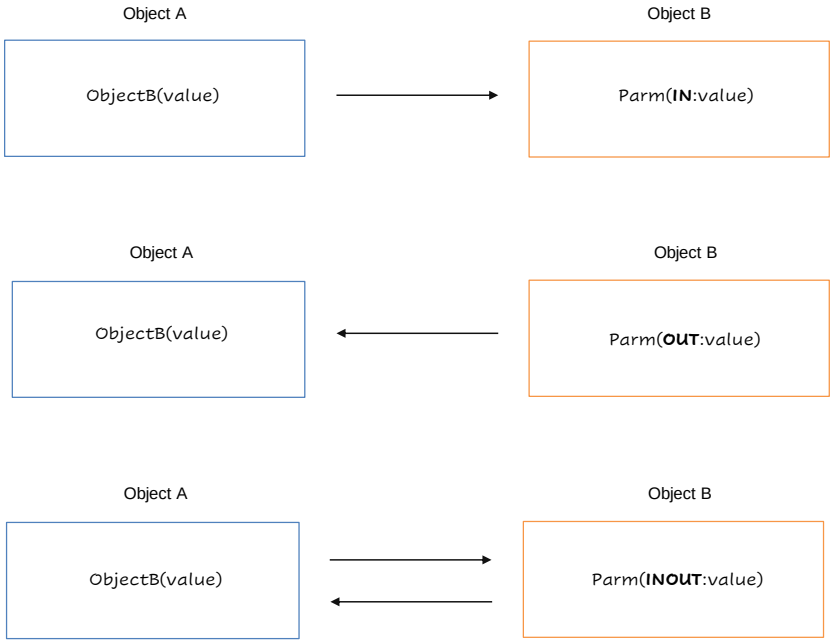
Qual é a diferença entre usar uma variável ou um atributo na regra Parm do objeto chamado?

Object B




```
parm (inout: &var1, in: &var2, out: &var3);
```

Se o valor é recebido numa variável, a mesma poderá ser utilizada livremente na programação: poderá ser utilizada como condição de filtro por igualdade, filtro por maior, maior ou igual, etc... poderá ser usada para alguma operação aritmética ou no que for necessário. É um espaço de memória com nome que utilizamos dentro do objeto através de instruções explicítas para fazer o que desejarmos.



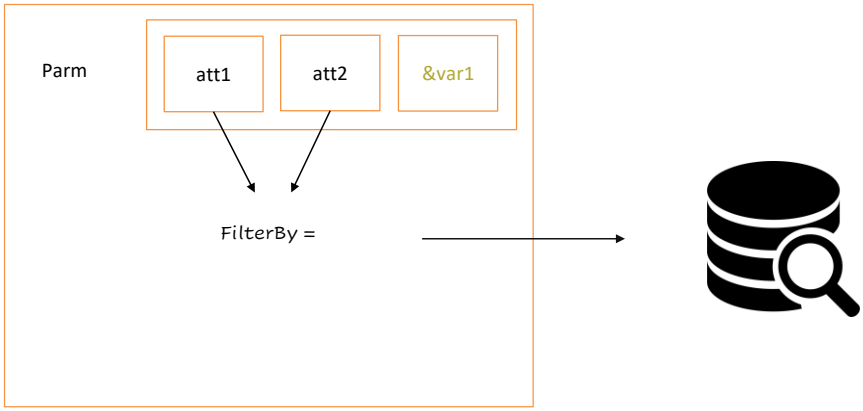
Para cada parâmetro declarado na regra Parm, podemos definir se o parâmetro é usado para receber um valor, para devolver um valor ou para ambas as coisas. Fazemos isto usando os operadores in, out e inout respectivamente.

O operador IN indica que o parâmetro é de entrada, ou seja, o parâmetro chega com um valor e esse valor não pode ser alterado.

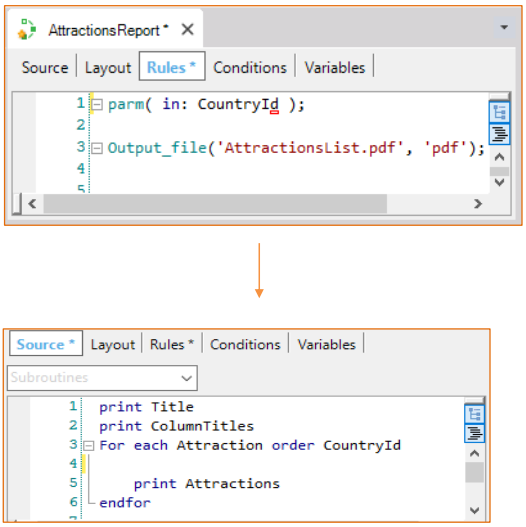
Os parâmetros com o operador OUT são os parâmetros de saída. Estes não chegam com nenhum valor e, após a execução do objeto chamado, o parâmetro de saída conterá o valor resultante que será devolvido ao objeto chamador.

Por último, temos o operador INOUT, que faz com que o parâmetro seja de entrada e saída ao mesmo tempo.
Com este operador, o parâmetro chega com um valor e pode ser alterado durante a execução do objeto.
Quando finaliza, o parâmetro conterá o valor que é devolvido ao objeto que o chamou.

Se não declaramos nenhum destes operadores na regra parm, GeneXus atribuirá o operador INOUT a todos os parâmetros, mesmo que isso não apareça na tela.



Se o valor é recebido num atributo, isto significa ter um sentido fixo, determinado, implícito. Recebemos o parâmetro em atributo quando dentro do objeto vamos acessar a base de dados. Em particular uma tabela estendida que se encontra o atributo. Sendo assim, ao receber por parâmetro um valor nesse atributo, será aplicado um filtro por igualdade. Serão considerados os registros que tenham esse valor para o atributo. Veremos isso com um exemplo.

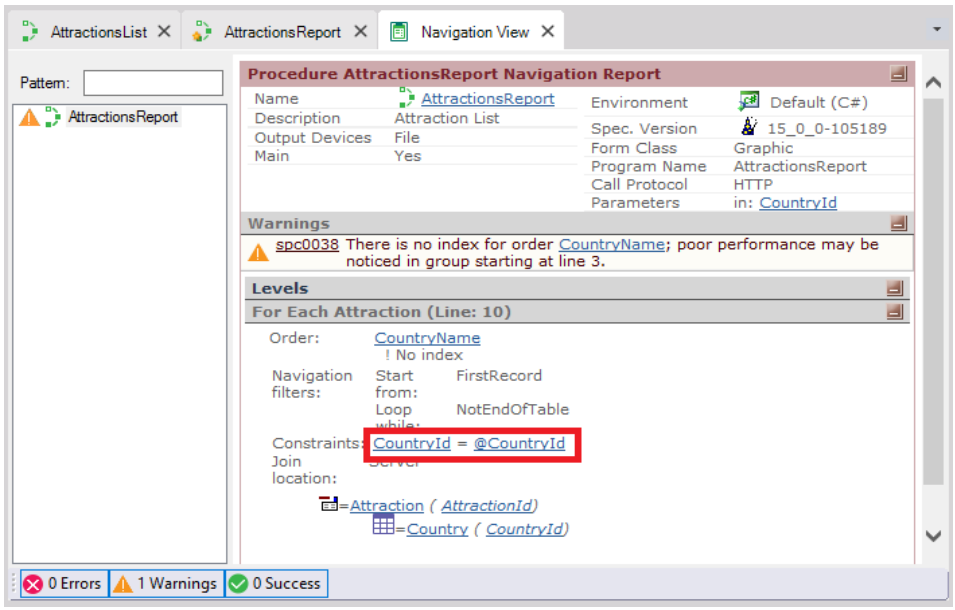


Façamos uma cópia do procedimento AttractionsList com o nome AttractionsReport.
Vale lembrar que o procedimento do qual fizemos a cópia utilizava uma variável como parâmetro: &CountryId.
E utilizava-a para filtrar as atrações da tabela Attraction por país.

Visualizamos que está sendo feito um filtro por igualdade: exibirá unicamente as atrações cujo CountryId corresponda ao valor da vaiável &CountryId recebida por parâmetro.
Poderíamos ter implementado extatamente o mesmo sem necessidade de estabelecer esse filtro de forma explicíta.

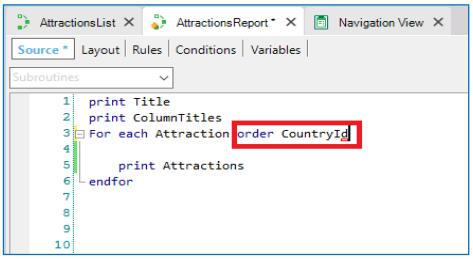
Como? Recebendo diretamente o atributo CountryId.
Quando recebemos o valor em um atributo na regra Parm, GeneXus filtra por igualdade, ou seja, somente será acessado os registros que tenham esse valor de código de país.

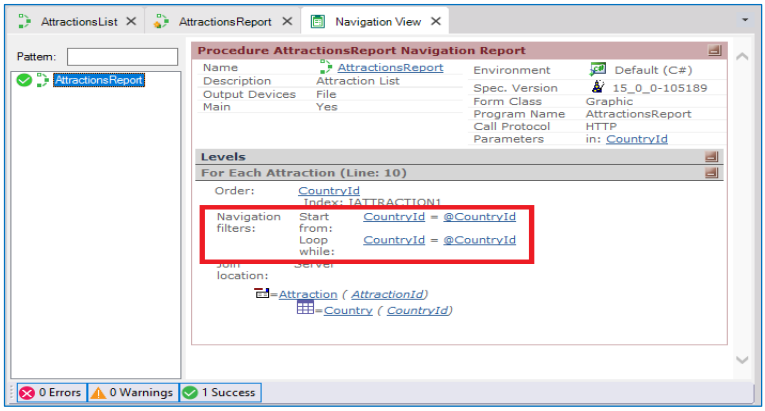
Analisando o mapa de navegação do objeto.



visualizamos que está sendo realizado o filtro mesmo sem nós termos escrito o Where.

O interessante é observar como o For Each está lendo a tabela ordenado por CountryName, e veja que é necessário ler toda a tabela para filtrar pelos valores de CountryId que correspondam ao parâmetro.





Em contrapartida, se ordenamos pelo atributo que filtramos.
Podemos ver que a navegação já não lê toda a tabela.
Executemos.
Enviamos tudo o que foi desenvolvido ao GeneXus Server.

Se tivéssemos recebido mais de um valor utilizando atributos, seria acessado unicamente os registro que tivessem o mesmo valor para cada atributo.
E para esses atributos não poderemos mudar o valor.

Se nosso objetivo não é receber valores para filtrar por igualdade, então no lugar de utilizar atributos a solução é receber os valores em variáveis.
Além do que, poderemos utilizar os valores livremente na programação, por
exemplo, para atribuir outros valores se for necessário.
A comunicação entre objetos é vital para qualquer aplicação GeneXus, já que é o mecanismo principal para que um objeto inicie a execução de outro e possa enviar ou receber informação do mesmo.

Existem outras formas de comunicação entre objetos em que não há passagem de dados por parâmetros. Um exemplo disto é quando persistimos informações em memória usando variáveis do tipo WebSession ou quando fazemos uso dos eventos globais. Não veremos estes casos neste vídeo, mas é importante saber que existem
várias e diversas formas de comunicação entre objetos.