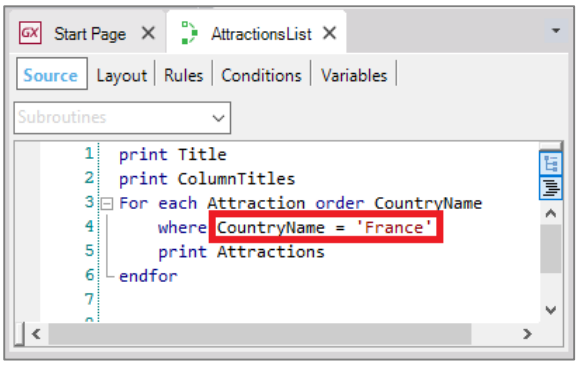
Comunicação entre objetos

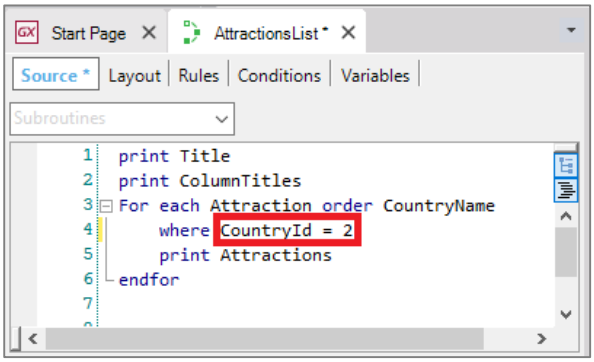
Em situações anteriores tivemos a necessidade de fazer chamada entre objetos, ou seja, um objeto que chama outro objeto GeneXus.

Por exemplo, quando criamos o objeto procedimento AttractionsList, tivemos a necessidade de filtrar as atrações que tivessem como nome de país “France”:



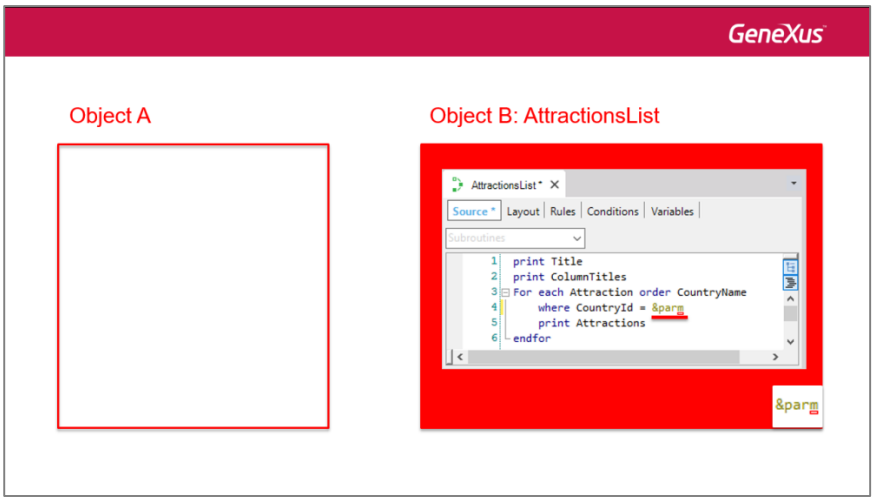
ou, uma outra alternativa de filtro, código de país igual a 2 (que corresponde a “France”):

…e para resolver, utilizamos valor fixo no código.

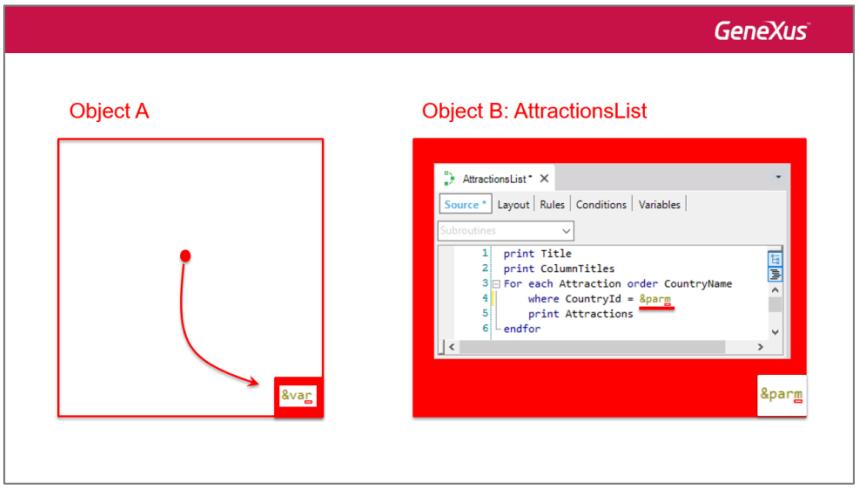


Mas, isso implica que se quiséssemos filtrar as atrações de um país diferente de France, teríamos que alterar o código do procedimento.

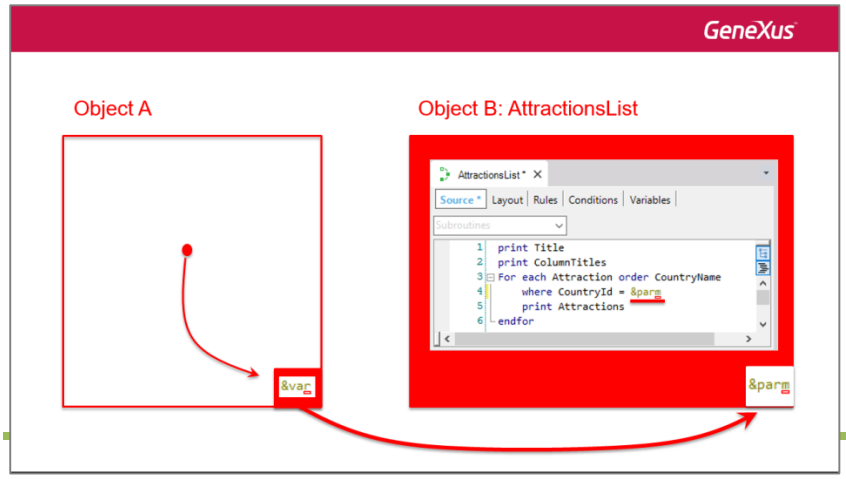
O ideal seria “receber” de alguma forma nesse objeto o valor que precisamos filtrar.



Em outras palavras, que outro objeto GeneXus permitisse ao usuario escolher esse valor:

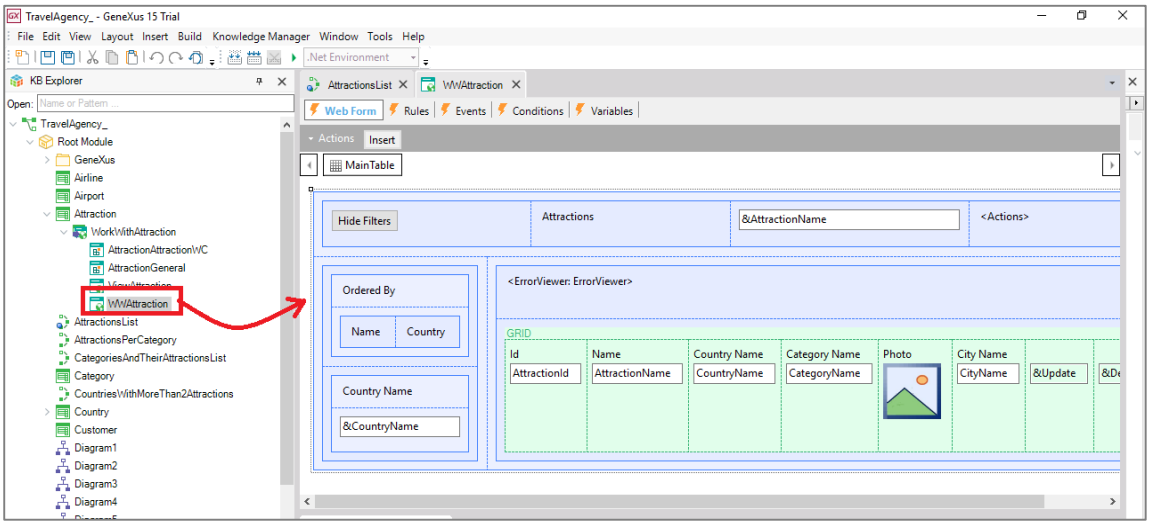


… e na sequência enviasse como parâmetro o valor para o objeto procedimento para que este exiba as atrações filtrando o país com o valor recebido.

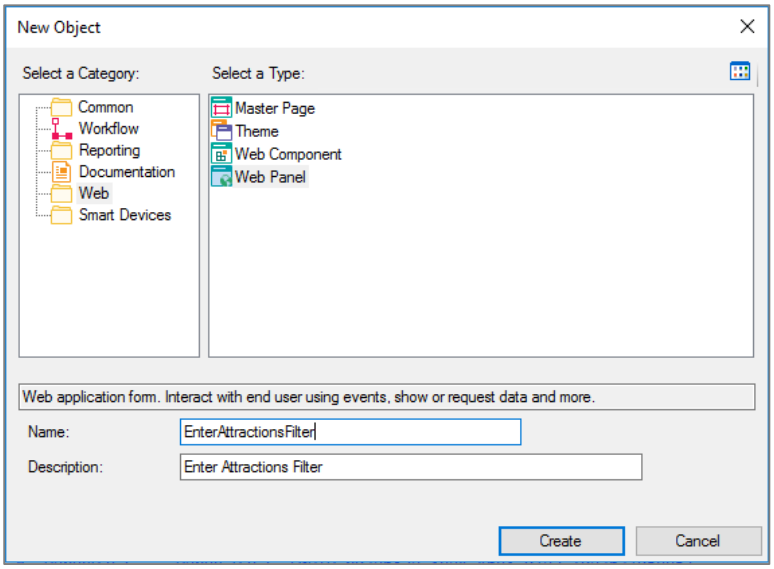


Veremos na continuação, mediante este exemplo, como implementar a comunicação entre objetos GeneXus.

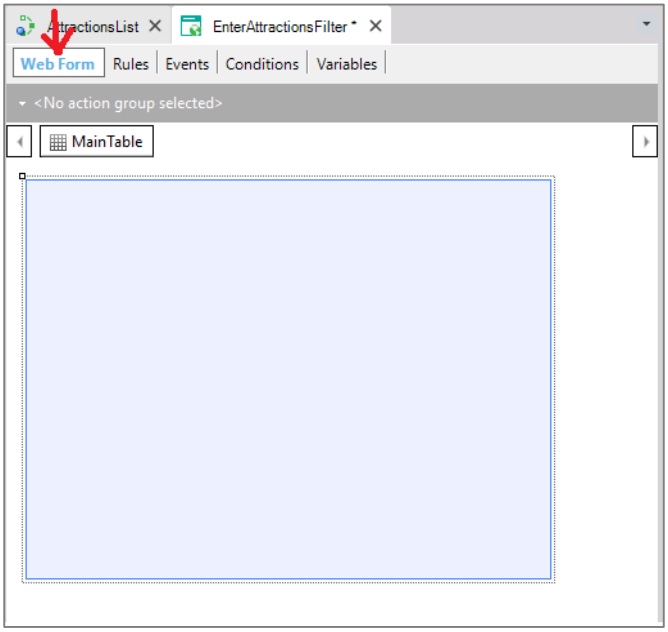
Para iniciar, devemos criar um objeto que seja capaz de oferecer uma tela para pedir valores ao usuário e fazer alguma coisa com esses valores. O objeto que permite isso é a **web panel**, que estudaremos em detalhe mais adiante. Neste momento digamos que se trata de um painel visual muito felxível que permite solicitar dados ao usuário, exibir informações da base de dados ou de outras fontes, entre outras coisas. Por exemplo, o Work With de atrações foi desenvolvido automaticamente pelo GeneXus como um objeto Web Panel.



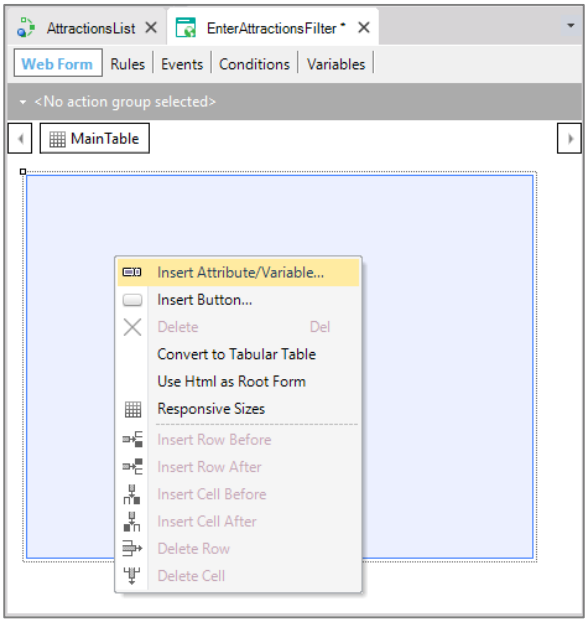
Então, vamos criar um objeto Web Panel com o nome de EnterAttractionsFilter:



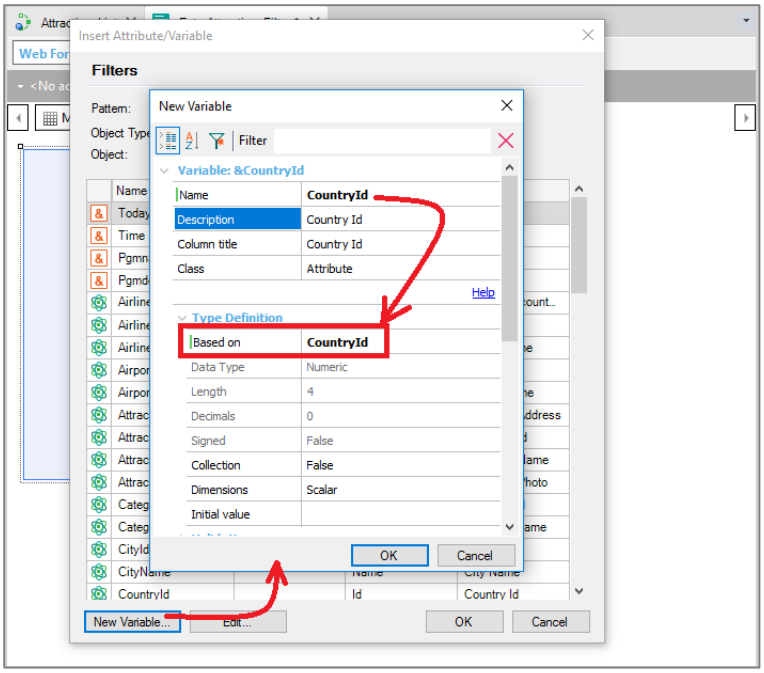
Veja que foi criado um Form Web, que será a tela do objeto. O Form Web contém unicamente uma tabela.



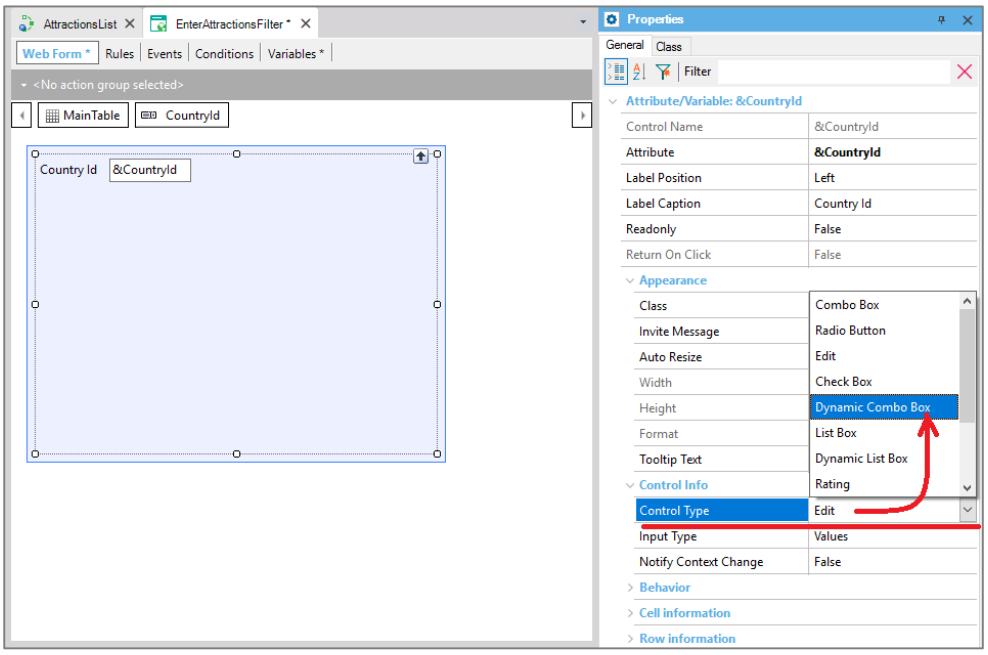
Adicionamos a variável CountryId



ao nomear uma variável igual ao nome do atributo ela ficará baseada no atributo, e portanto, assume o mesmo tipo de dados. Dessa forma, sempre que alterarmos o tipo de dados do atributo, por exemplo, para numérico 10 ao invés de numérico 4, a variável automaticamente será alterada refletindo esse novo valor.

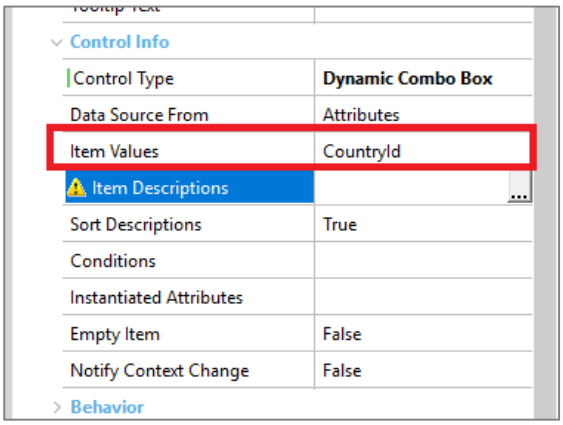


Agora alteraremos as propriedades da variável e veremos que sua propriedade Control Type está configurada com o valor Edit. Isso significa que quando a web panel for executado esse campo ficará esperando o usuário digitar um valor numérico, mas não fornecerá nenhuma ajuda para escolher quais valores existem na base de dados e nem a que país corresponde.
Vamos mudar o tipo de controle para Dynamic Combo Box:

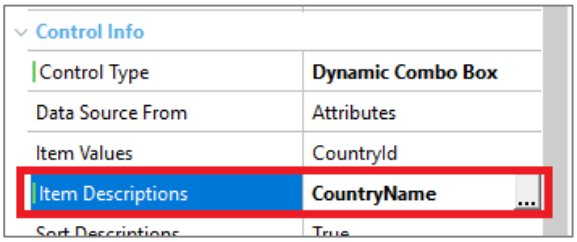


Dessa forma, será exibido para o usuário uma série de valores extraídos da base de dados, para que o usuário

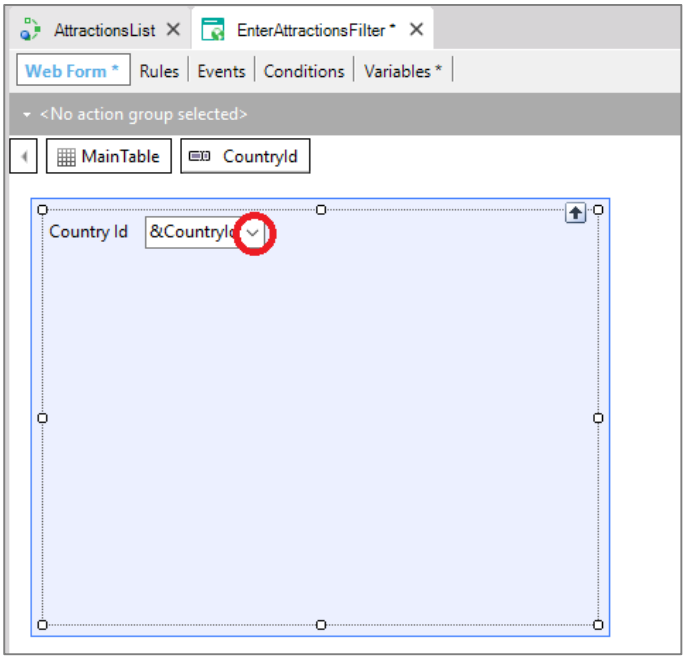
selecione o desejado. Quais valores? Os valores do atributo CountryId:



Significa que a tabela Country será lida e os valores de CountryIds serão carregados no combo. Mas, como os códigos não dizem muita coisa, se bem que a variável vai armazenar um código de país, o que será mostrado ao usuário na tela é o conteúdo do atributo que indicamos aqui… Escolhemos exibir o nome do país:

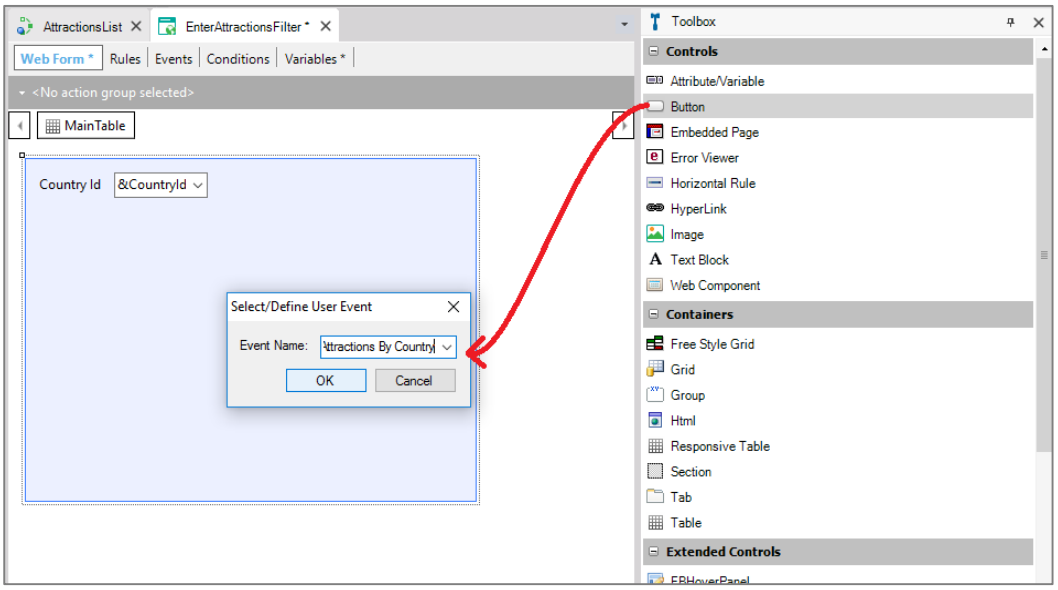


E podemos ver que aparece uma pequena flecha indicadora de combo:

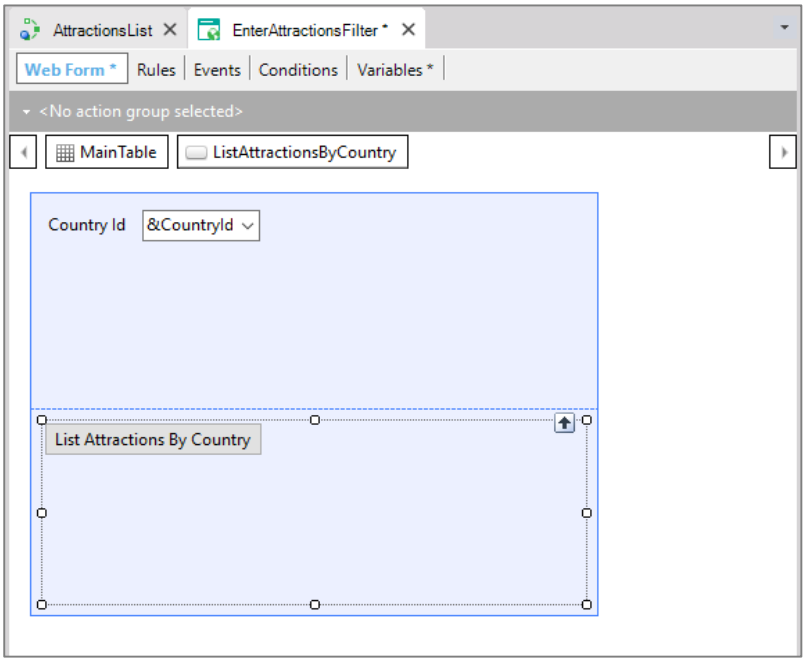


Resumindo, será exibido em execução um combo que apresentará a lista de países da base de dados para escolher o que nos interessa.

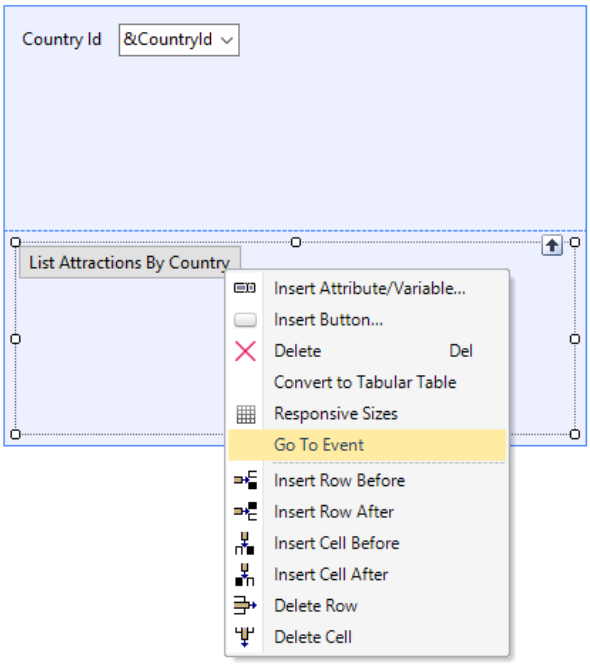
Adicionamos um botão na Web Form. É solicitado que indiquemos o nome do evento que esse botão terá associado. Colocamos: “List Attractions By Country”:



E podemos ver que o texto do botão assume o mesmo nome do evento por padrão:



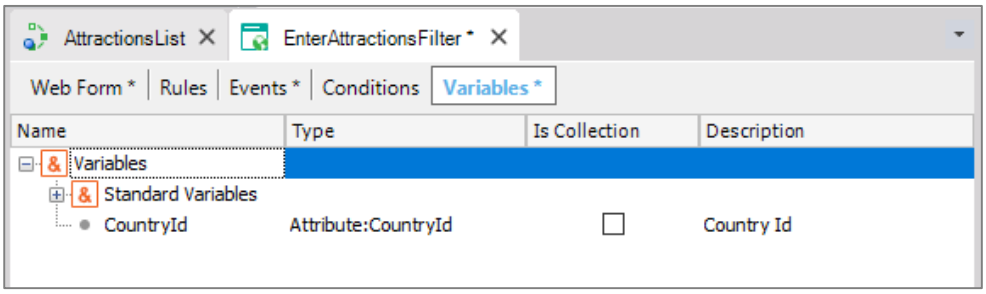
Se posicionamos sobre o botão, pressionamos o botão direito e escolhemos Go to Event…



…podemos ver que foi criado um evento com esse nome, e somos posicionados automaticamente na aba Events, e o cursor está a espera do código que será executado quando o evento for disparado. O disparo irá ocorrer quando o usuário pressionar o botão associado ao evento.

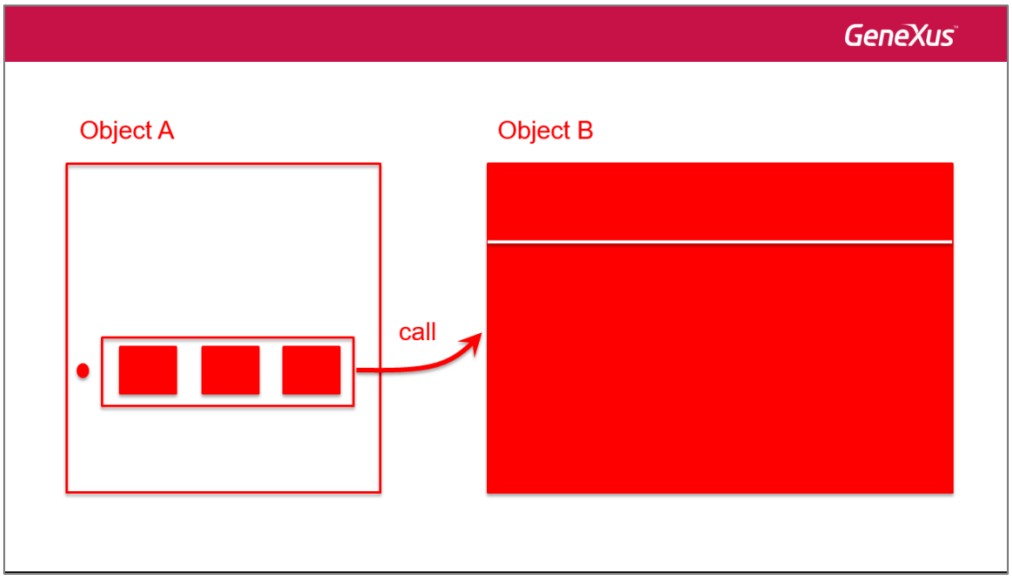
O que precisamos fazer nesse momento é chamar o objeto procedimento AttractionsList que exibe as atrações e enviar o país que queremos filtrar como parâmetro:

Observemos que no momento de pressionar o botão e executar esse código, a variável &CountryId conterá o código do país referente ao país escolhido pelo usuário no combo box na tela. Anteriormente aprendemos que uma variável é uma porção de memória na qual damos um nome e nos serve para guardar um dado de forma temporária, e que cada objeto tem sua seção de Variables,

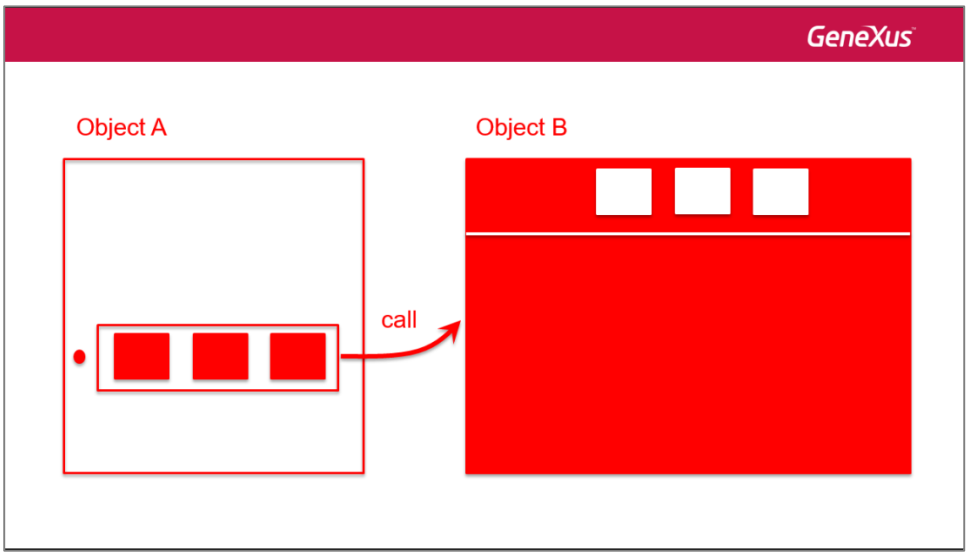


ou seja, as variáveis que são definidas num objeto são conhecidas somente dentro do escopo desse objeto. Sendo assim, se dois objetos tem a variável CountryId definida, mesmo possuindo o mesmo nome, se tratará de duas variáveis distintas.

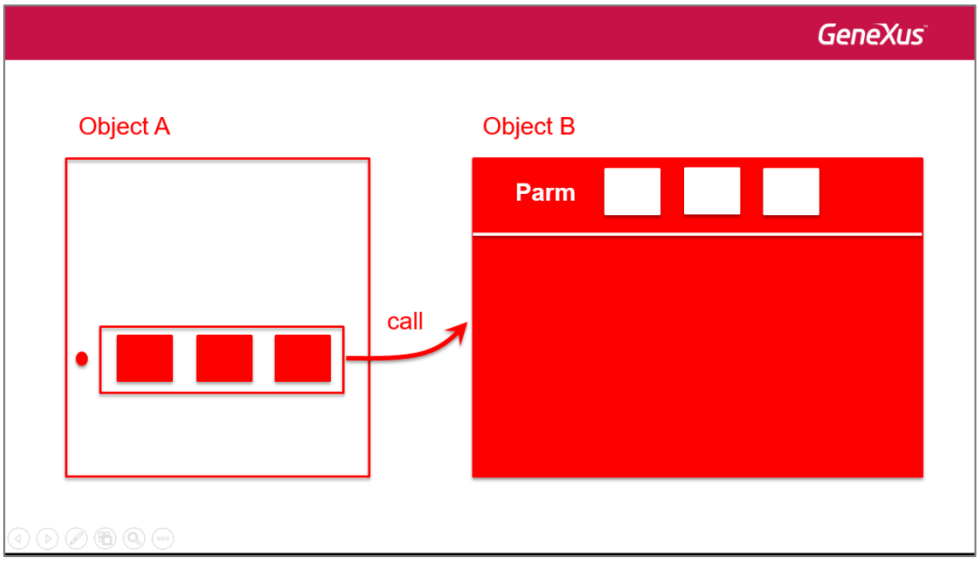
Então, como fazer para que um objeto A possa chamar um objeto B em um dado momento passando valores?:



e que o objeto B possa carregar em suas variáveis internas os valores que foram enviados, para poder fazer alguma coisa com essa informação:

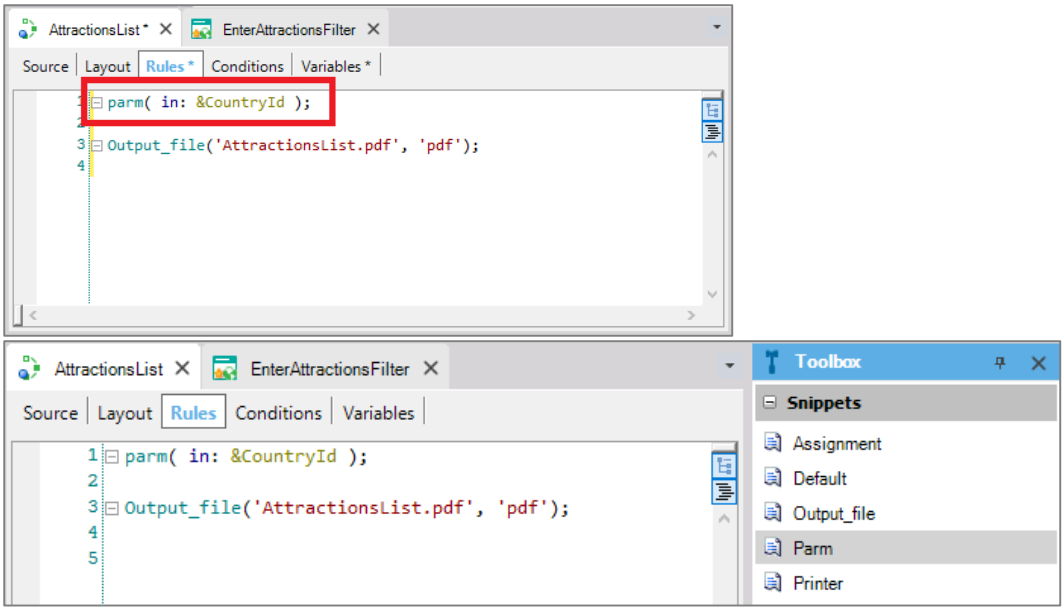


Para que um objeto possa receber valores (comumente chamados de **parâmetros**), devemos ir a sua seção Rules e escrever a regra **Parm**. Essa regra **Parm** declara os parâmteros que o objeto possa receber e/ou devolver ao objeto chamador.

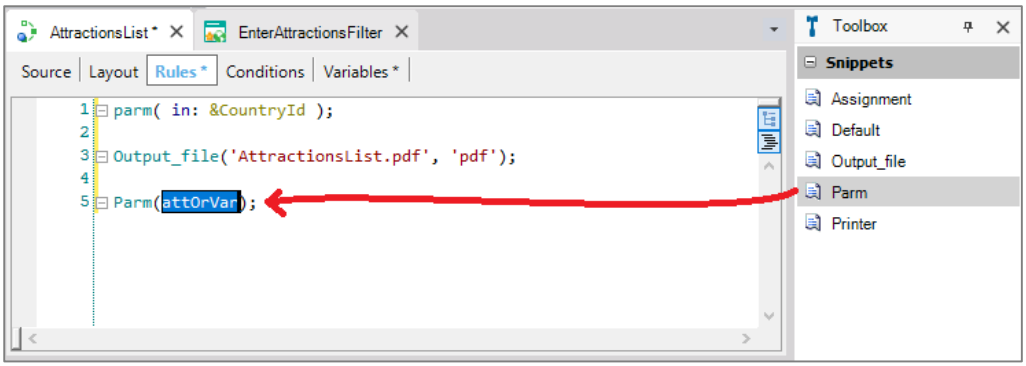


Como em nosso exemplo quem vai receber os valores é o objeto procedimento AttractionsList, abrimos o objeto e vamos a sua seção de regras.

Escrevemos:



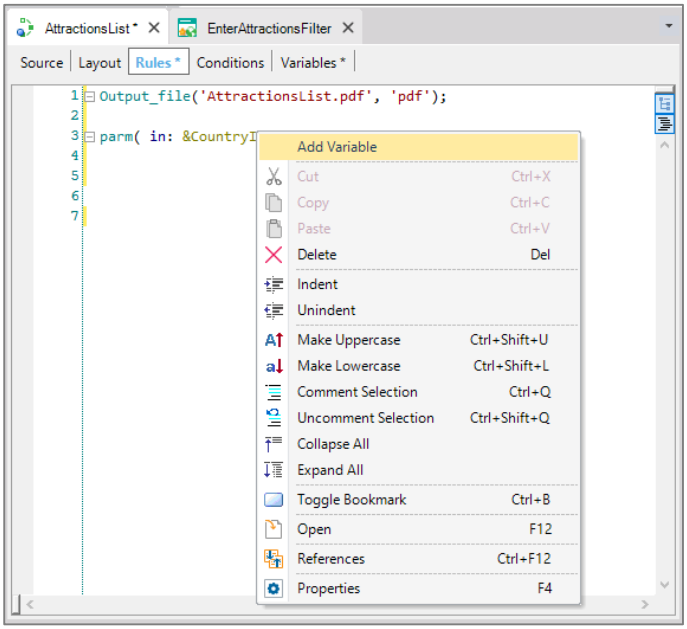
Vejamos que na Toolbox é disponibilizada todas as regras que esse objeto suporta. Entre elas, a Parm. Poderíamos ter arrastada a regra da Toolbox no lugar de digitá-la.



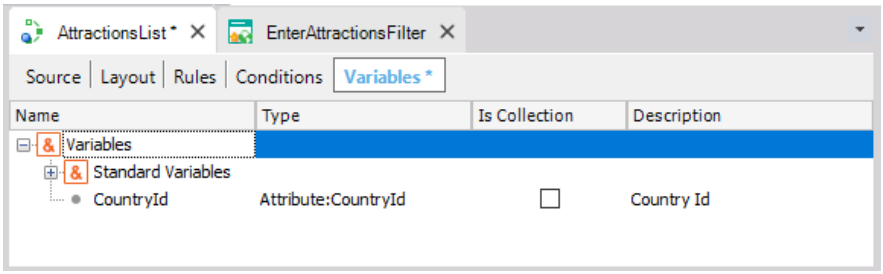
Na sintaxe da regra Parm é suportado trabalhar com atributo ou variável. Depois veremos o caso dos atributos. Nesse momento vamos trabalhar apenas com o caso das variáveis.

Com "in" estamos indicando que a variável &CountryId será um parâmetro de entrada. Isso significa que somente será utilizada para receber um valor do objeto chamador. Não para devolver. Podemos omitir essa informação e deixar que GeneXus descubra automaticamente.

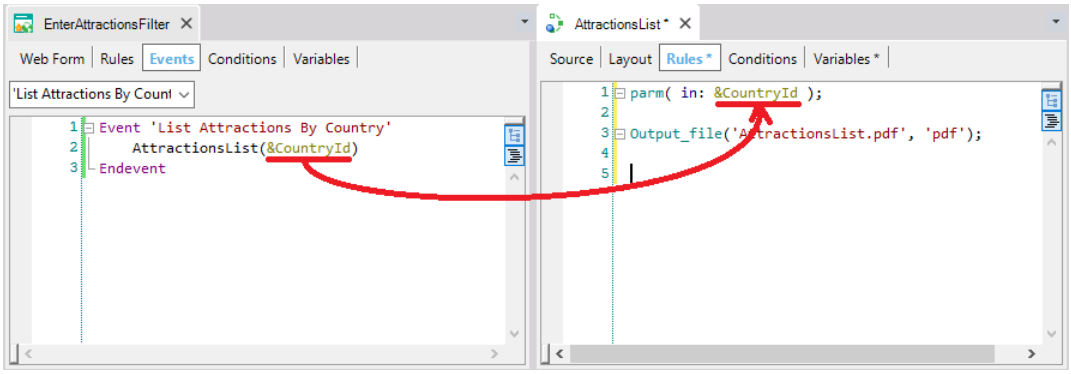
Digitamos o **nome** da variável, mas não criamos ela como variável no objeto. Para fazer isso, uma das formas é posicionarmos o cursor sobre o nome, acionar o botão direito do mouse e escolher "Add variable":



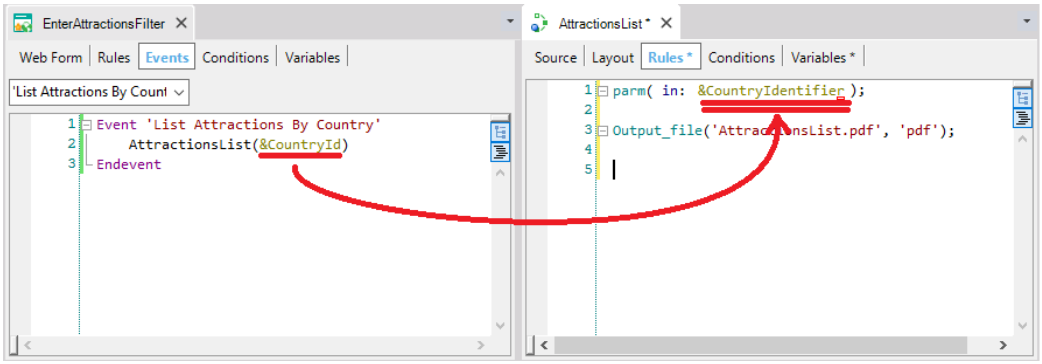
Se passamos para a aba Variables podemos ver que a variável foi definida baseada no tipo do atributo CountryId. Isso se deve ao fato de nomearmos a variável igual ao nome de um atributo.



No procedimento definimos a variável com o mesmo nome e tipo de dados que usamos na web panel onde o usuário informa o país:



No entanto, como foi comentado, são duas variáveis diferentes. Uma válida somente na web panel e a outra no procedimento. Poderíamos ter colocado nomes diferentes nas variáveis em ambos os objetos. Mas, para que a comunicação e a passagem de informação seja correta, o tipo de dados deve coincidir entre objeto chamador e chamado.

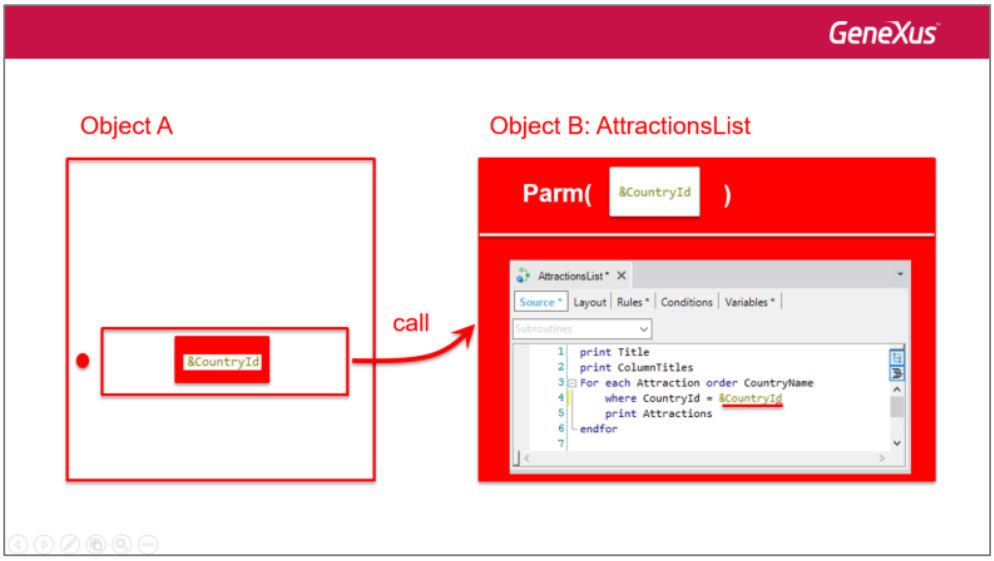


Agora, nosso procedimento está preparado para receber um código de país enviado pela web panel EnterAttractionsFilter.

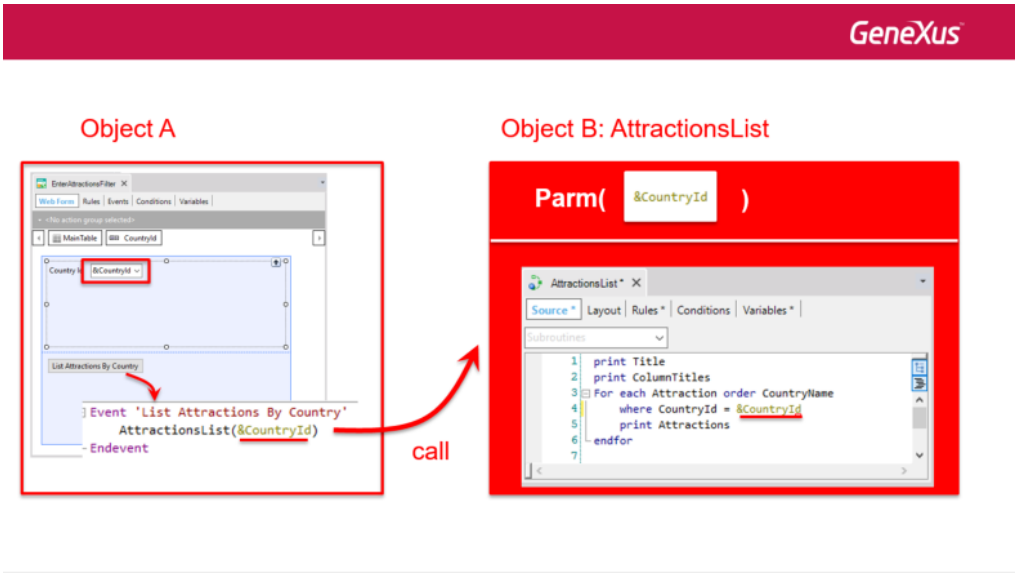
Agora resta eliminarmos o filtro fixo que tínhamos (o valor 2 de país) no For each, e trocar pela variável cujo valor é recebido como parâmetro.

Observemos que ao ter definido a regra Parm dessa forma, de agora em diante qualquer objeto que chame o procedimento poderá (e deverá) passar o valor do código do país.

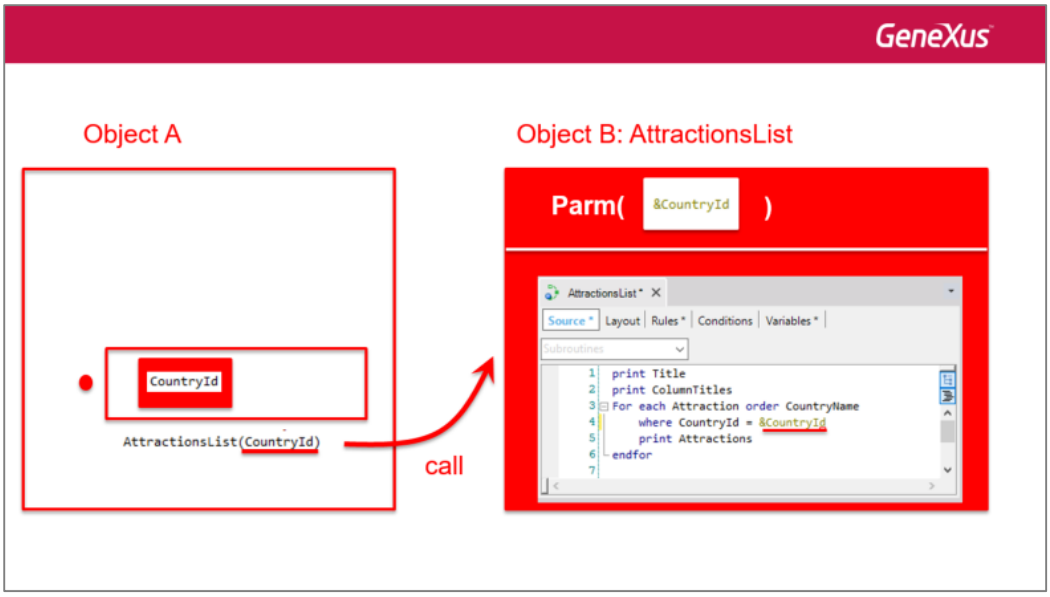
Esse procedimento já não poderá ser chamado sem enviar um valor desse tipo. É por esta razão que o procedimento AttractionsList não aparece mais no Developer Menu.



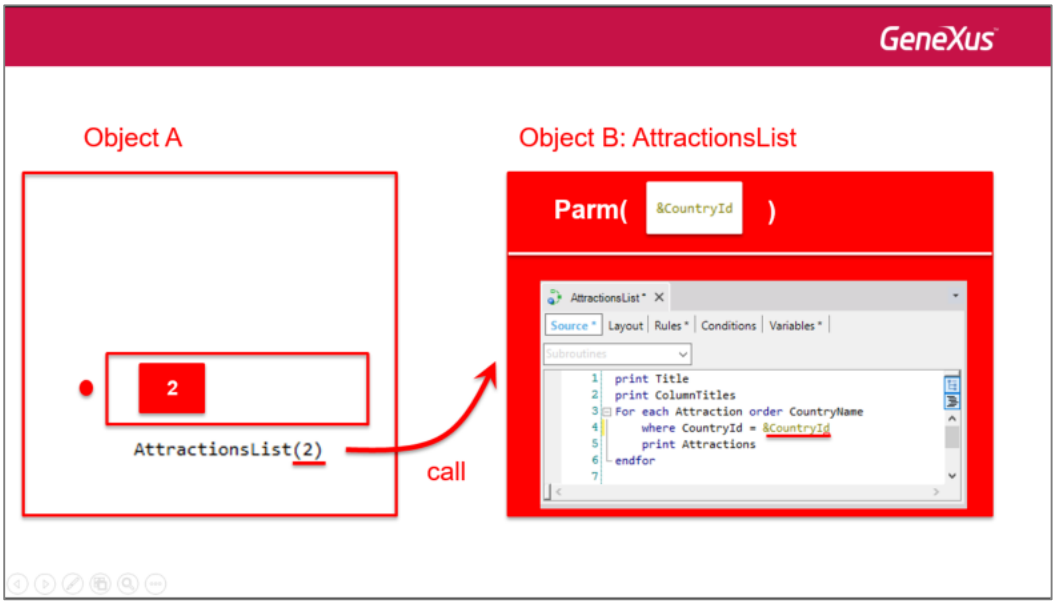
No caso da web panel tínhamos esse valor numa variável:



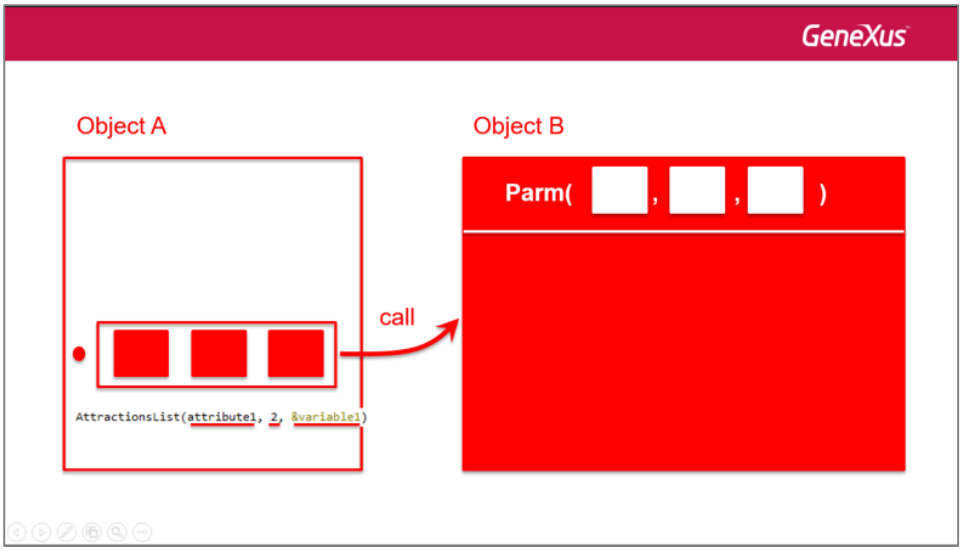
Mas se tivéssemos o dado num atributo, incluiríamos dentro do parêntese o atributo correspondente,



Também poderíamos passar diretamente um determinado valor:



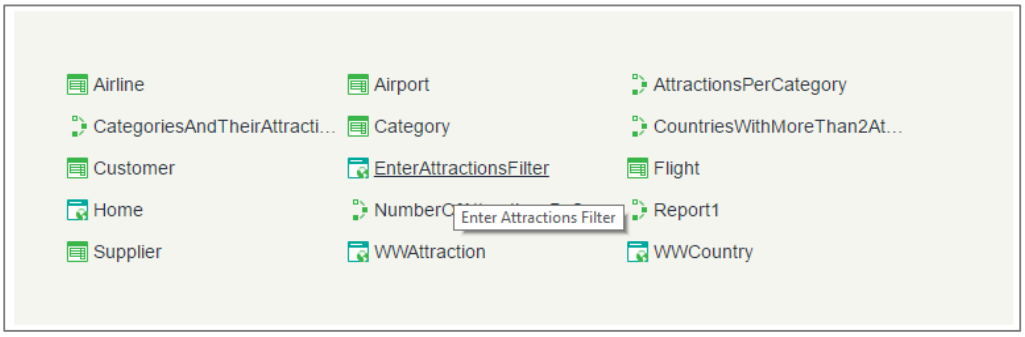
e no caso de ter que passar dois ou mais valores, enviaríamos vários atributos, e/ou valores explícitos, e/ou variáveis, separadas por vírgula:



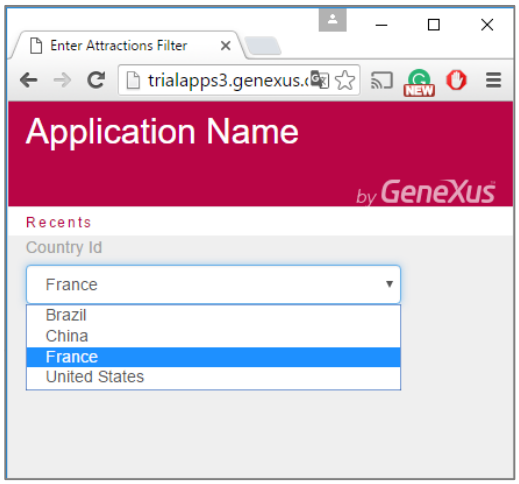
Esses parâmetros também são declarados na rega Parm de forma ordenada e separados por vírgulas.

Evidentemente que um objeto que não recebe parâmetros não deve declarar a regra Parm.

Vamos testar o que foi feito: aperte F5. O procedimento AttractionsList já não aparece no Developer Menu. Agora somente podemos chamá-lo através da web panel…

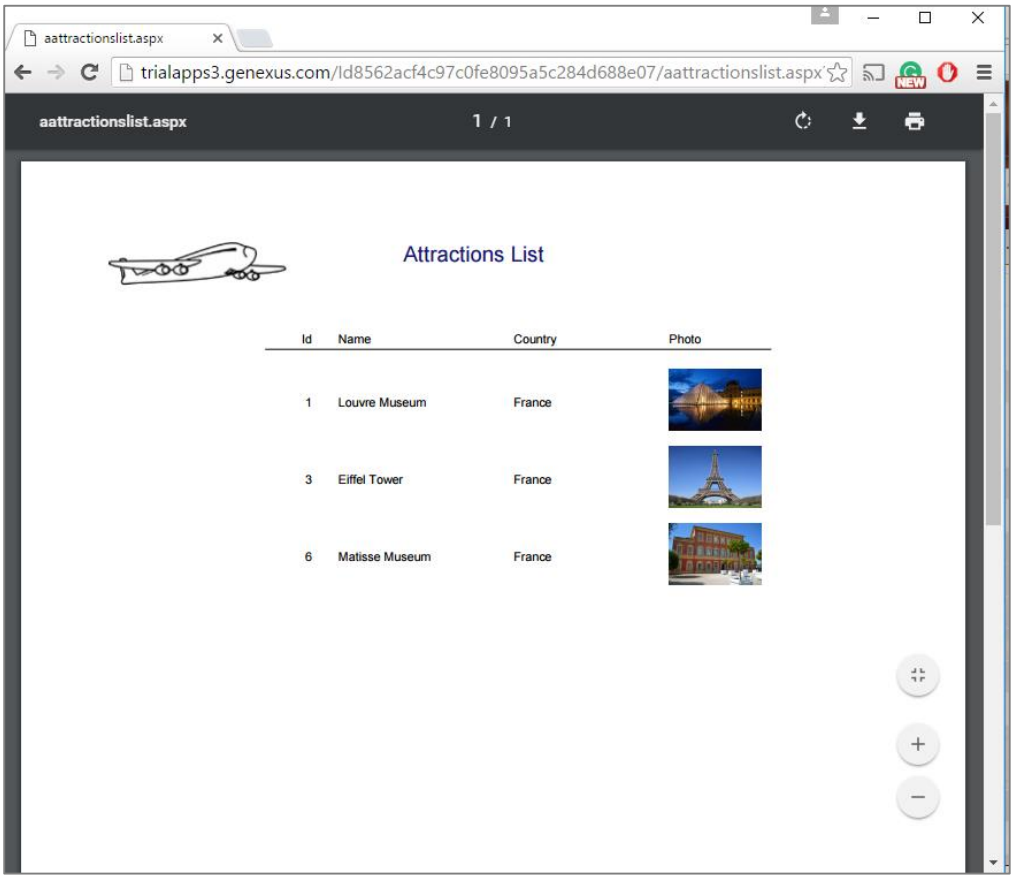


No combo do país, escolhemos France…



Ao escolher o valor France no dynamic combo, internamente foi selecionado o valor do código France (neste caso o valor 2) e esse valor é o que se envia no procedimento AttractionslList.

O relatório é executado mostrando somente as atrações do país France:

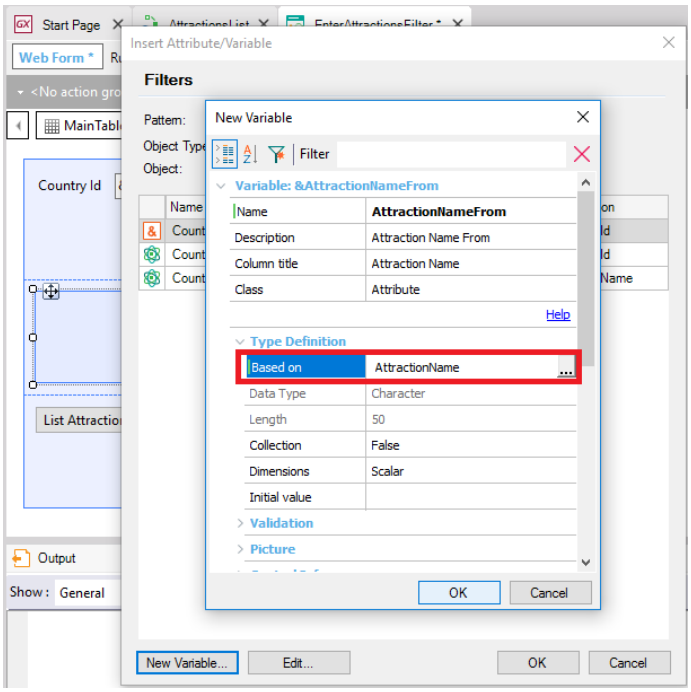


Vamos imaginar que agora queremos listar todas as atrações cujos nomes se encontram entre dois valores escolhidos pelo usuário. Por exemplo, entre as letras “A” e “D”.

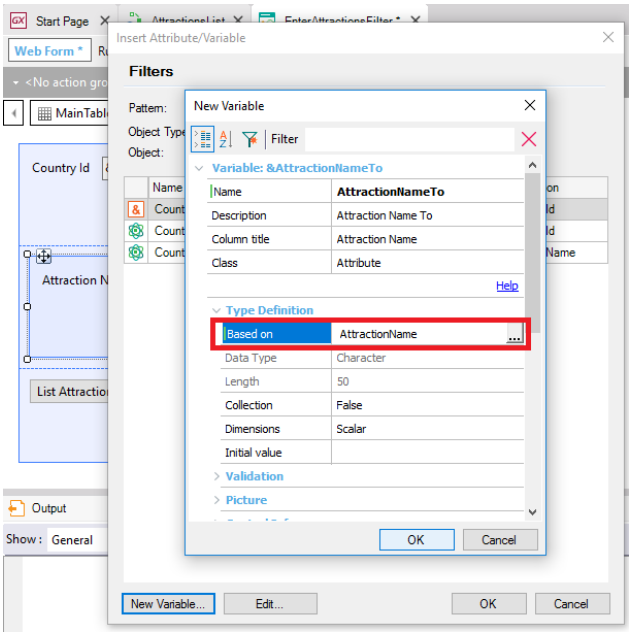
Para isso, vamos adicionar na web panel que criamos previamente a possibilidade do usuário informar um nome inicial e um nome final para que, pressionando um botão, chame um relatório que exiba todas as atrações turísticas cujos nomes se encontram dentro deste range.

Vamos até a web panel EnterAttractionsFilter e adicionamos uma tabela com duas variáveis.

- &AttractionNameFrom… baseada na definição do atributo AttractionName:

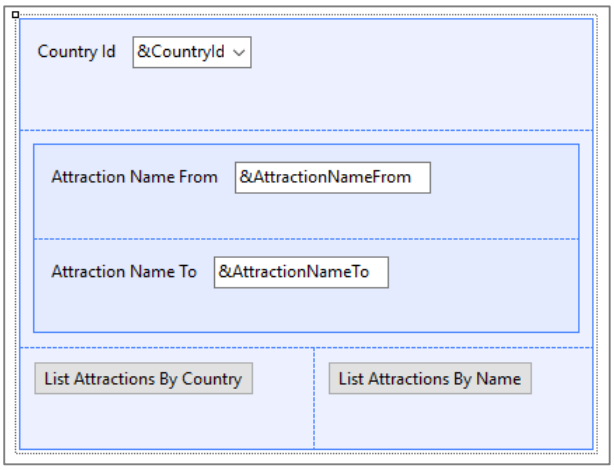


- e &AttractionNameTo, também baseada na definição de AttractionName:



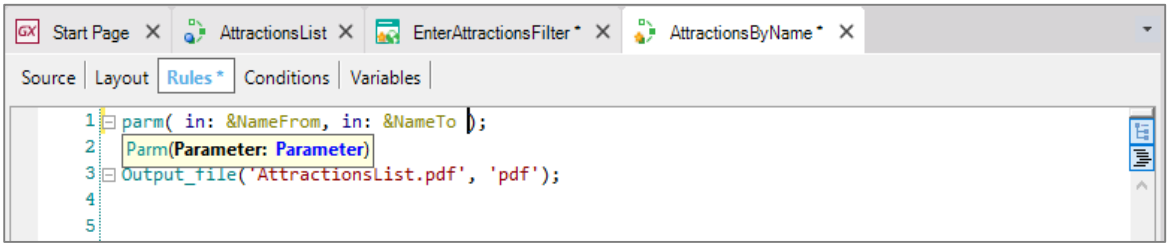
Como já comentamos, isso siginifica que a definição da variável está amarrada com a definição do atributo e quando alteramos o tipo de dados do atributo, automaticamente é modificada a definição da variável.

Depois adicionamos um botão com o evento “List Attractions By Name”:

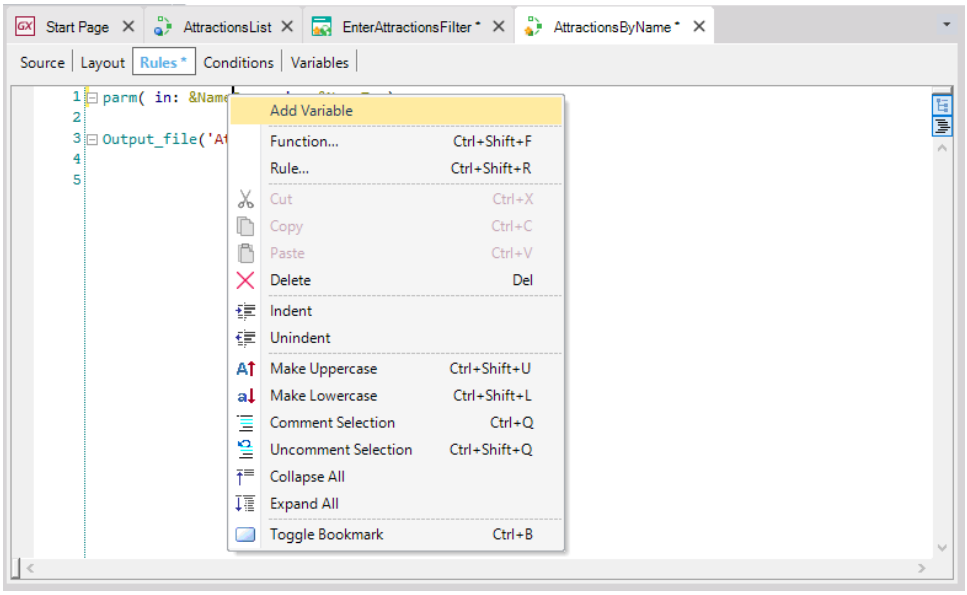


Posicionamos no botão que acabamos de adicionar, pressionamos o botão direito e escolhemos Go to event. Temo que chamar dentro do evento o procedimento que imprimirá as atrações turísticas do range.

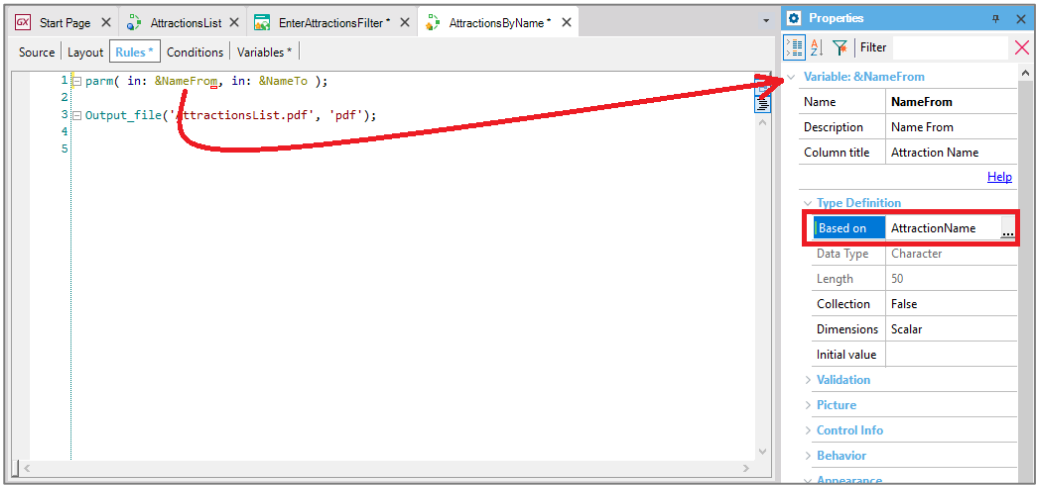
Já tínhamos o relatório AttractionsList… mas que recebia por parâmetro o código do país, não o range de nomes. Vamos salvá-lo com outro nome, AttractionsByName, e alterar sua regra Parm, para que agora receba dois parâmteros de entrada: a variável &NameFrom, e a variável &NameTo:



Temos que defini-lás como variáveis e especificar o tipo de dados. Clicamos com o botão direito sobre a primeira variável e pressionamos “Add Variable”



e editando suas propriedades, baseamos sua definição no atributo AttractionName:



Fazemos o mesmo com &NameTo.

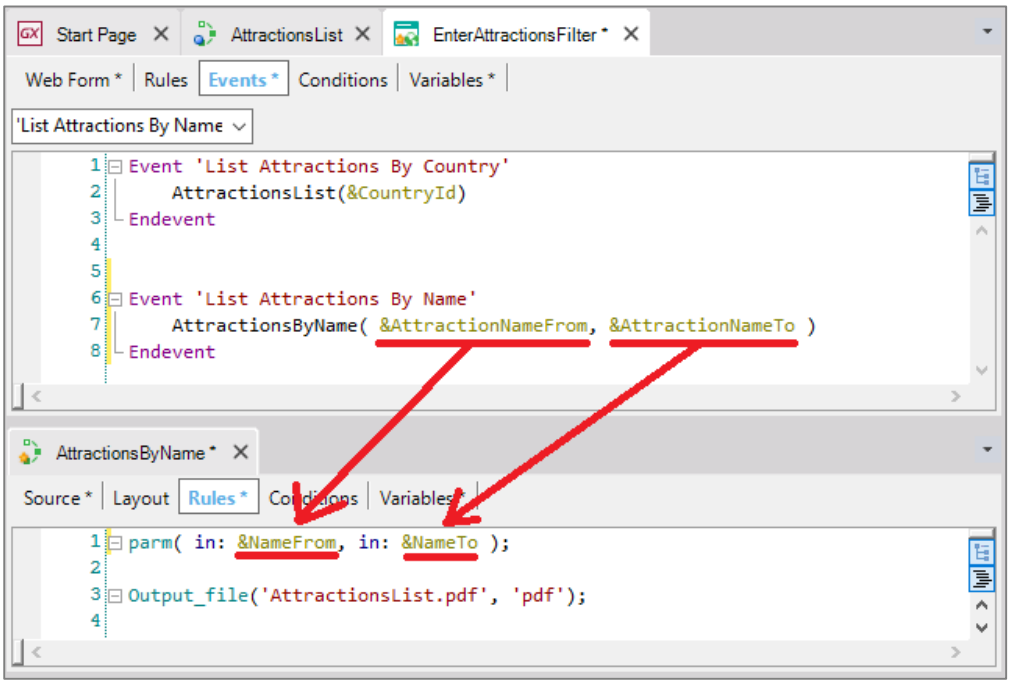
Agora vamos utilizar essas variáveis/parâmetros no For each…, para ficarmos com as atrações que cumpram que seu atributo AttractionName seja maior ou igual ao valor da variável &NameFrom, assim como menor ou igual ao valor da variável &NameTo:

```
Print Title
Print columnTitles
For each Attraction order CountryName
    Where AttractionName  >= &NameFrom
    where AttractionName <= &NameTo
    Print Attractions
Endfor
```

Com isso o procedimento está pronto e somente nos resta a chamá-lo por meio da web panel.

Vamos na web panel e adicionamos a chamada. Arrastamos o relatório AttractionsByName da janela para não ter que digitar e entre parênteses colocamos os parâmetros separados por vírgula, neste caso as variáveis &AttractionNameFrom e &AttractionNameTo que são oferecidas ao usuário na tela:

Observemos que o primeiro parâmetro que escevemos na chamada será carregado no primeiro parâmetro definido na regra Parm do objeto chamado e o segundo parâmetro da chamada será carregado no segundo parâmetros do objeto chamado.



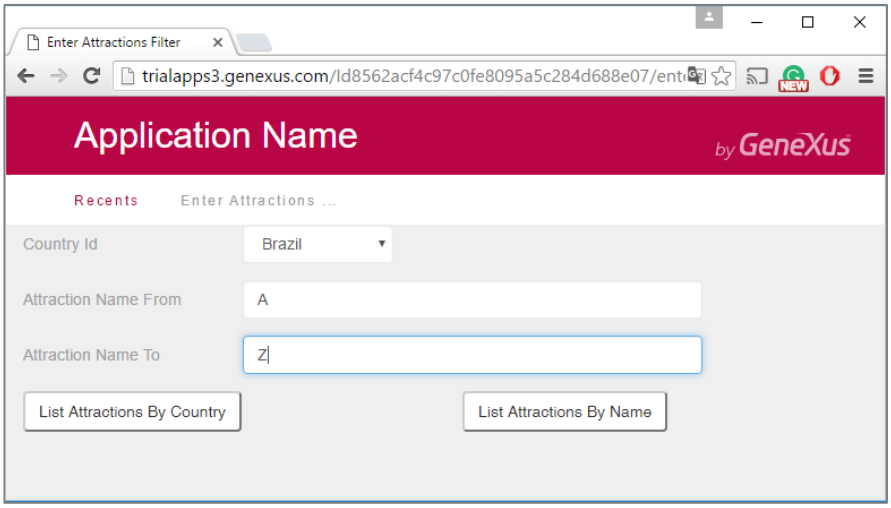
Devemos ser cuidadosos de respeitar a ordem na chamada e a definição na regra Parm. É uma boa prática usar

nomes relacionados como comentamos aqui, com o objetivo de entender melhor o código.

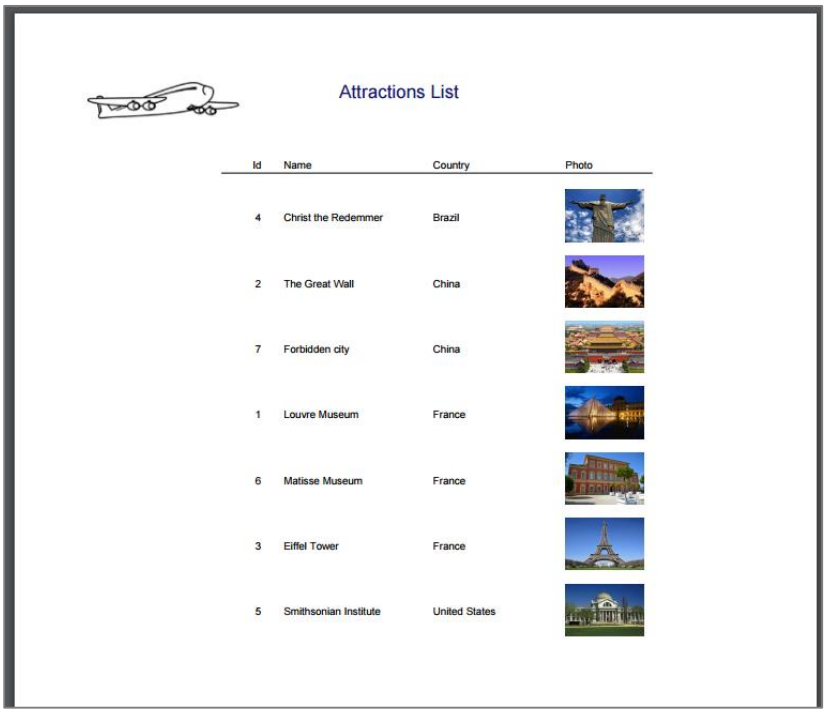
Notemos que os nomes escolhidos para as variáveis da web panel e do procedimento são diferentes. Como já informamos, o importante é que os tipos de dados enviados e recebidos sejam os mesmos.

Pressionemos F5 para executar…

Abrimos a web panel e para começar queremos ver todas as atrações entre as letras “A” e “Z”:



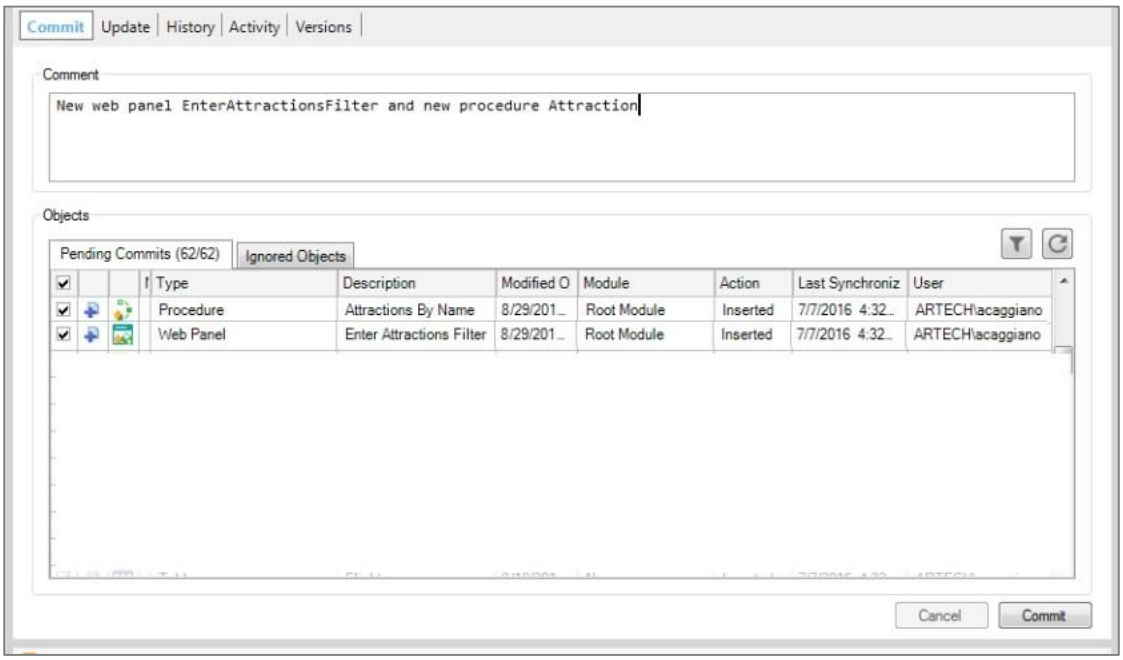
Pressionamos o botão “List Attractions By Name”… e visualizamos que são exibidas todas as atrações.



Agora vamos diminuir um pouco mais o range, colocamos entre as letras “A” e “F”:

E visualizamos o funcionamento do filtro:

Para finalizar enviaremos tudo o que desenvolvemos para o GeneXus Server…



No próximo vídeo veremos outras formas de enviar e receber parâmetros, incluindo os efeitos de colocar um atributos na regra Parm, no lugar de uma variável.