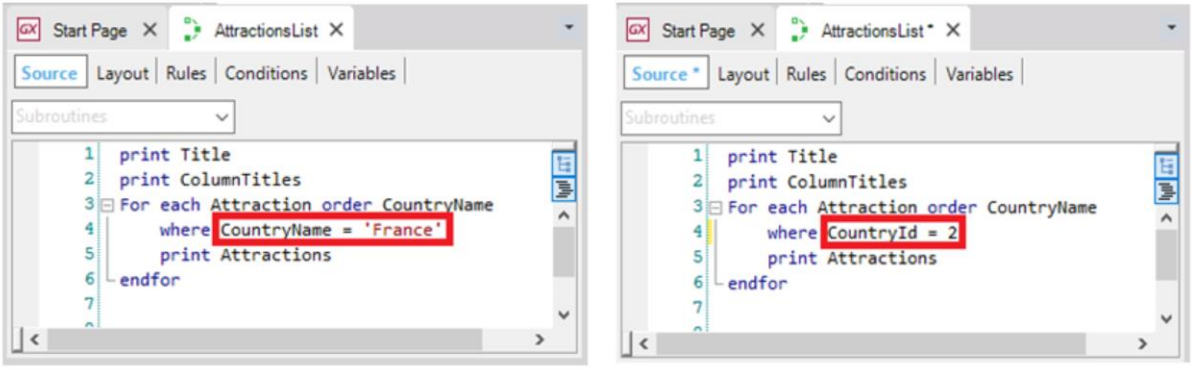
Communication between objects

Need for calling an object from another object

GeneXus 16

So far...

We've used fixed values in filters... in this case by Country

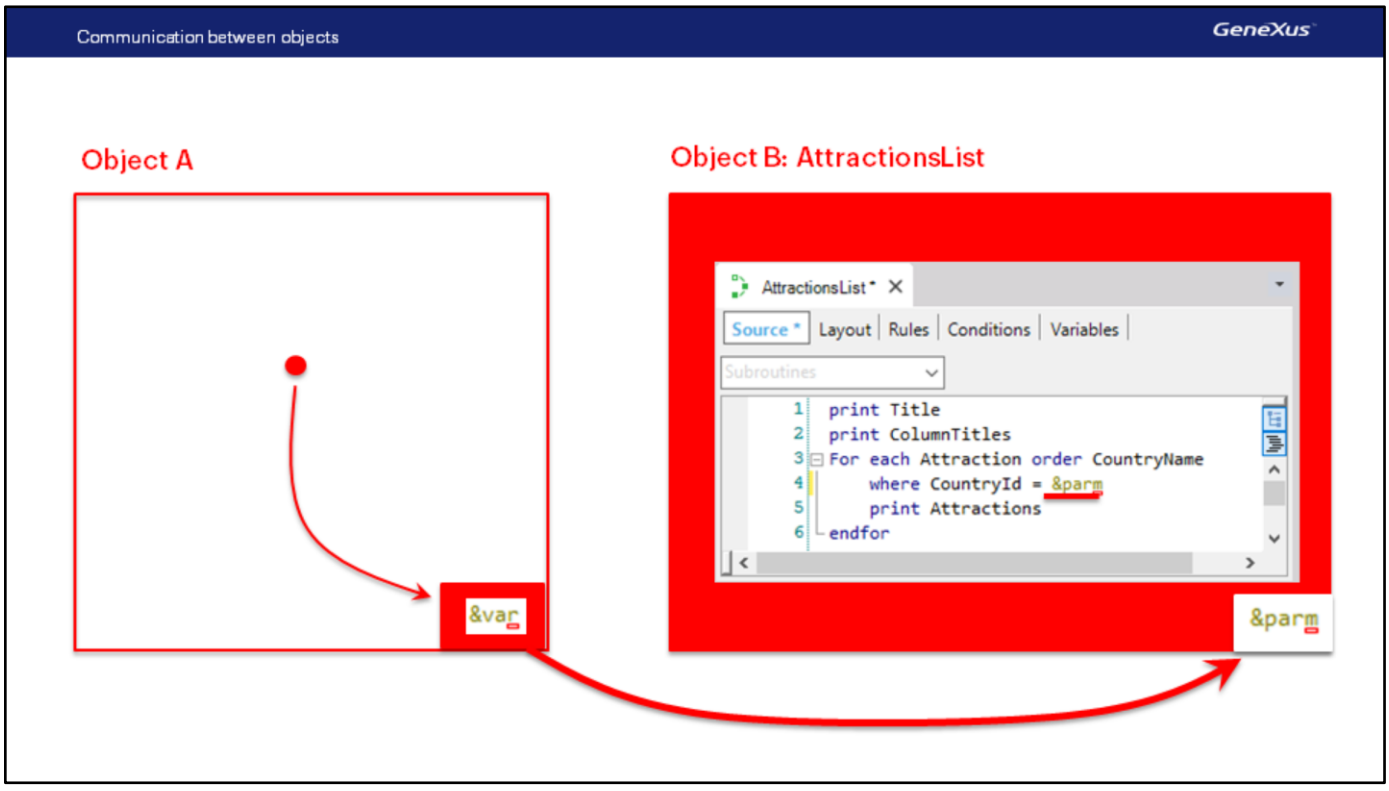


What if we want to make the list general and "receive" the country to filter by?

In previous situations we've had to call an object from another object.

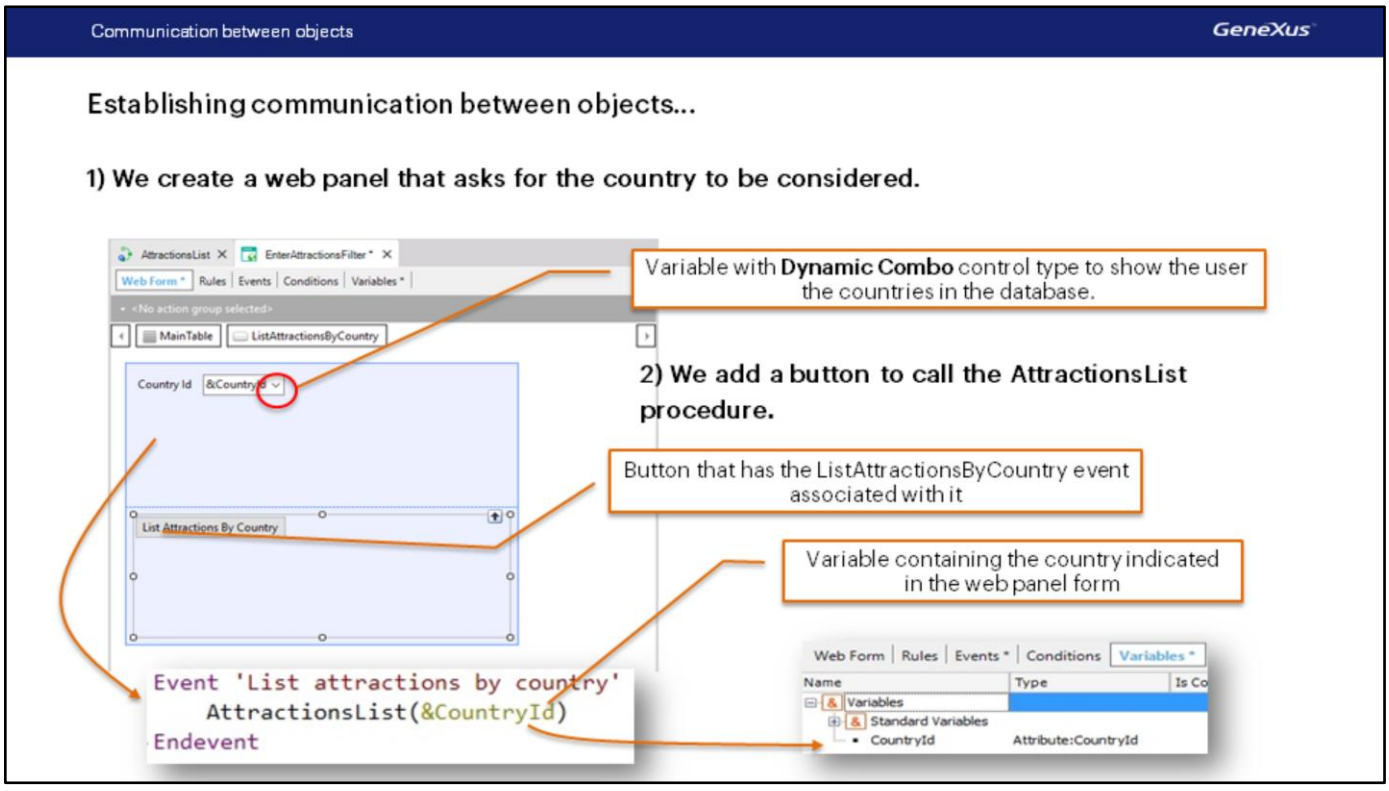
For example, when we implemented the AttractionsList procedure object, we needed to filter the attractions whose country name was "France", or, similarly, whose country identifier was 2 (which corresponded to "France")... To do so, we used fixed values in the code.

However, this implies that if we wanted to filter the attractions of a country other than France, we would have to change the procedure code every time!



Ideally, we should be able to "receive" in this object the value that we want to filter by. In other words, that another GeneXus object allows the user to choose that value... and then sends it to this procedure object to have the attractions listed according to the country received.

Next, we will use this example to see how to implement this communication between GeneXus objects.



First, we need to create an object that provides a screen to prompt the user for values and do something with those values. The object that enables this is the **web panel**, which will be studied in detail later. For now, let's say it is a visual, flexible panel that can prompt the user for data, as well as show information from the database or other sources, among other things. For example, the attractions Work With element was automatically implemented by GeneXus as a Web Panel.

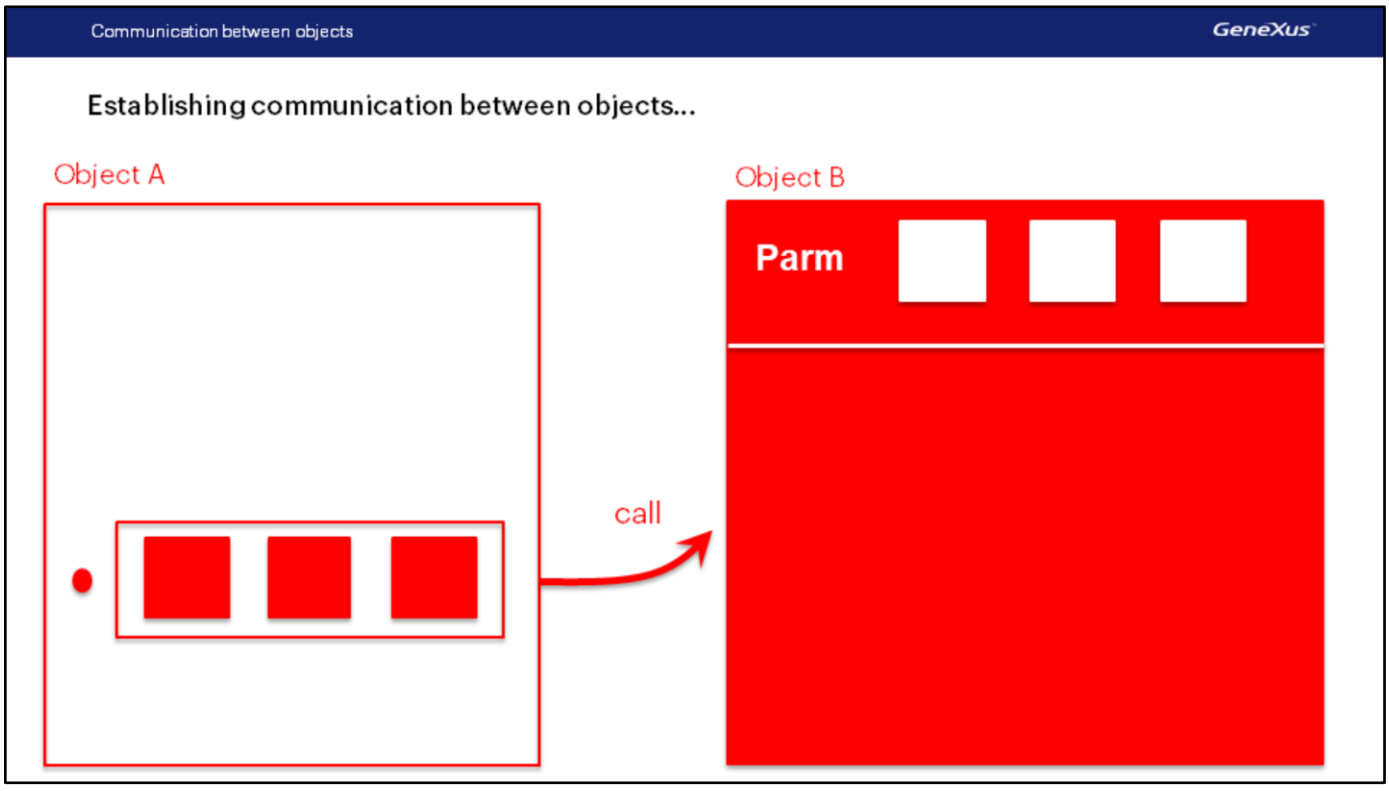
So, we will create an object of this type, and call it EnterAttractionsFilter. Note that a Web Form will be created to be the object screen. It contains a single table

We add a CountryId variable... Since its name is the same as that of the attribute, it is based on it and, therefore, takes its same data type. In this way, if we change the attribute's data type, for example, from numeric (10) to numeric (4), the variable will automatically take this new value.
Now we edit the variable properties and see that its **Control Type** property takes the **Edit** value. This means that when the web panel is executed, this field will expect the user to enter a numeric value, but it will not provide any help to choose from the values existing in the database or indicate the corresponding country. We will change the control type to **Dynamic Combo Box**. In this way, the user will be offered a series of values extracted from the database for him to choose one. What values? Those of the CountryId attribute. That is to say, the Country table will be run through and the existing CountryIds will be loaded in the combo box. But, since identifiers don't usually provide any details, even though the variable will store a country identifier, the user will be shown the content of the attribute indicated in the **Item Descriptions** property of the variable... We choose to show the country name. Note that the arrow that indicates the combo is displayed. In sum, at runtime it will offer a combo box with a list of the countries stored in the database to choose the one we're interested in.

Also, we add a button. We're asked to enter the name of the event that will be associated with that button. We call it: “List Attractions By Country”. We see that the button's text takes the same default name. If we click on it, right-click, and select Go to Event... we see that an event with that name has been created, and automatically changed from the Web Form tab to the Events tab. Also, the cursor is waiting for us to enter the code that will be run when this event is triggered. That is to say, when the user presses the associated button.

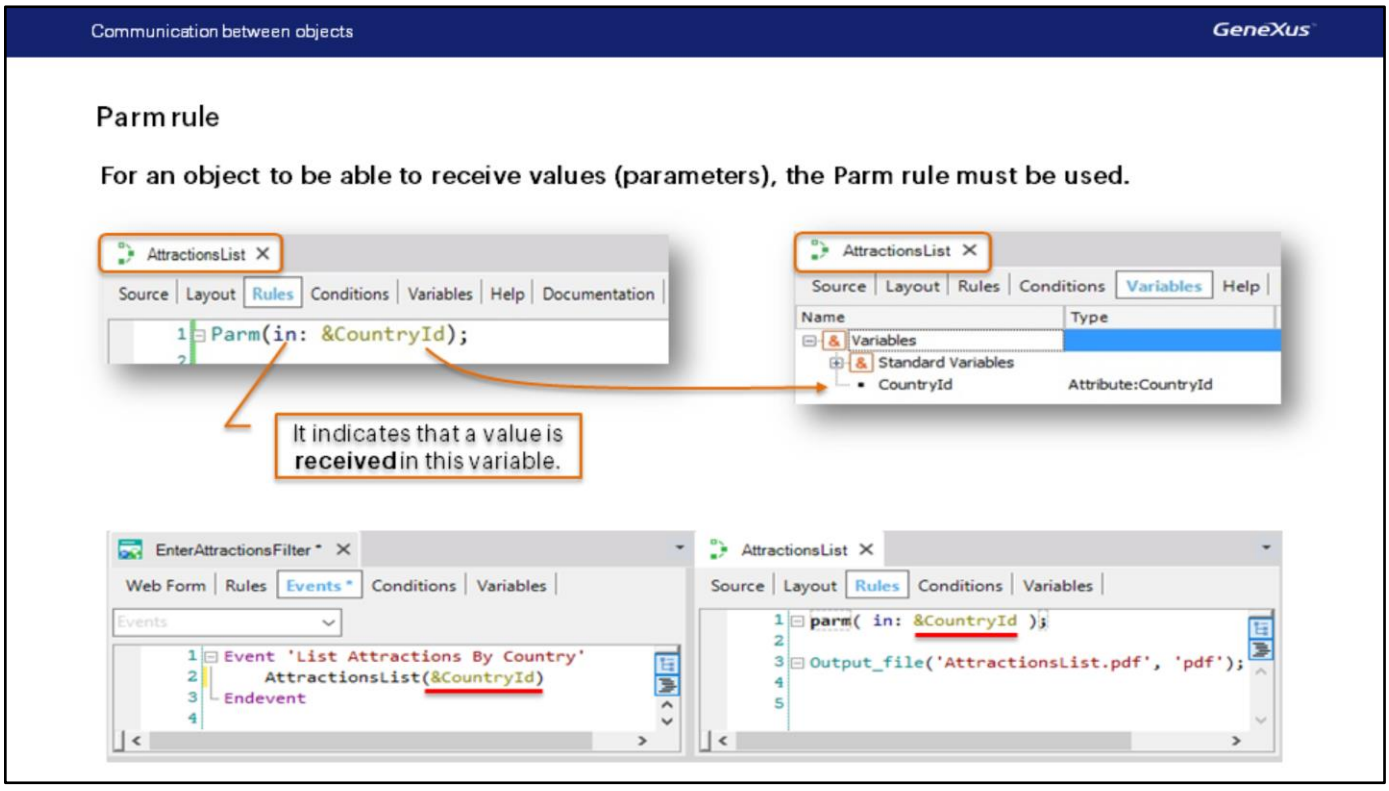
What we need to do now is call the AttractionsList procedure object that lists the attractions and send the country that we want to use to filter them.

Note that when we press the button and run this code, the &CountryId variable will contain the country identifier of the country selected by the user in the combo box displayed on the screen. Previously, we saw that a variable is a portion of memory that is given a name and used to temporarily save a data item. Also, that each object has its variables section. That is to say, variables defined in an object are only known in this object. So, if two objects have a CountryId variable, even if they have the same name, they will be two different variables.



So, how do we make an object A call another object B at a given moment, sending it values? Also, this object B should be able to load in its internal variables the values sent to it, in order to do something with that information.

For an object to be able to receive values (which we call **parameters**), we must open its Rules section and write a **Parm** rule. This **Parm** rule declares the parameters that the object can receive and/or return to the caller.



Since in our example the values will be received by the AttractionsList procedure object, we open the object and go to its rules section. We type the above parm rule.

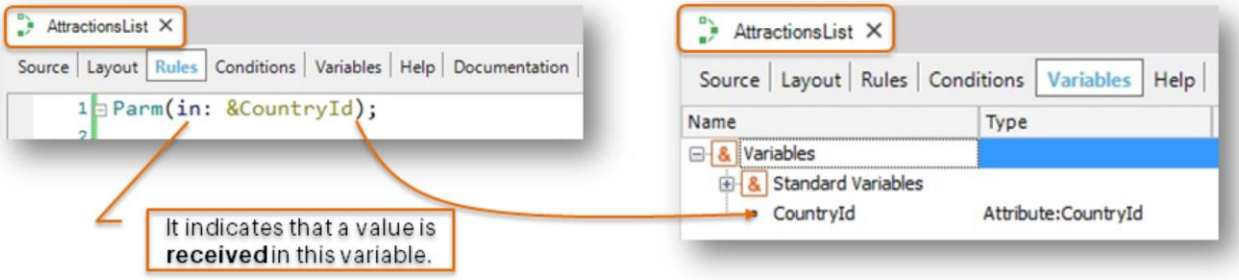
With "in" we indicate that the &CountryId variable will be an **input** parameter. This means that it will only be used to receive a value from the caller. It will not return any values. We can skip this information and have GeneXus infer it.

In this object (AttractionsList) we have created the variable with the same name and data type as the one we used in the web panel for the user to enter the country.
However, as we've said, they are two different variables. One is only valid in the web panel and the other in the procedure. We could have used different names in both objects, but for the communication and sharing of data to be correct, the **data type** of the caller and called objects must be the same.

Now, our procedure object is ready to receive a country identifier, in this case from the EnterAttractionsFilter web panel.

Parm rule

For an object to be able to receive values (parameters), the Parm rule must be used.



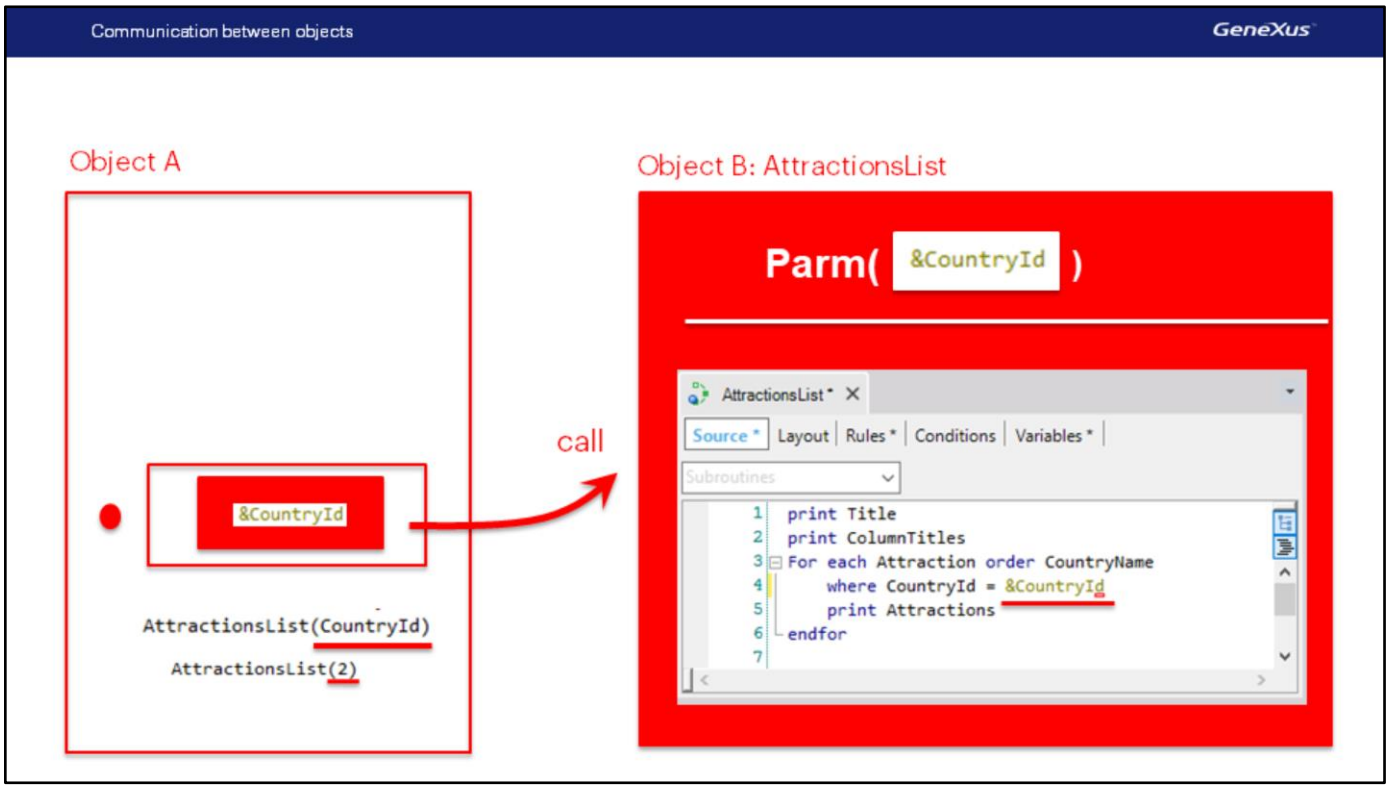
We change the Source:

```
print Title
print ColumnTitles
For each Attraction order CountryName
   where CountryId = 2
   print Attractions
endfor
```

We change it to ...

```
print Title
print ColumnTitles
For each Attraction order CountryName
   where CountryId = &CountryId
   print Attractions
endfor
```

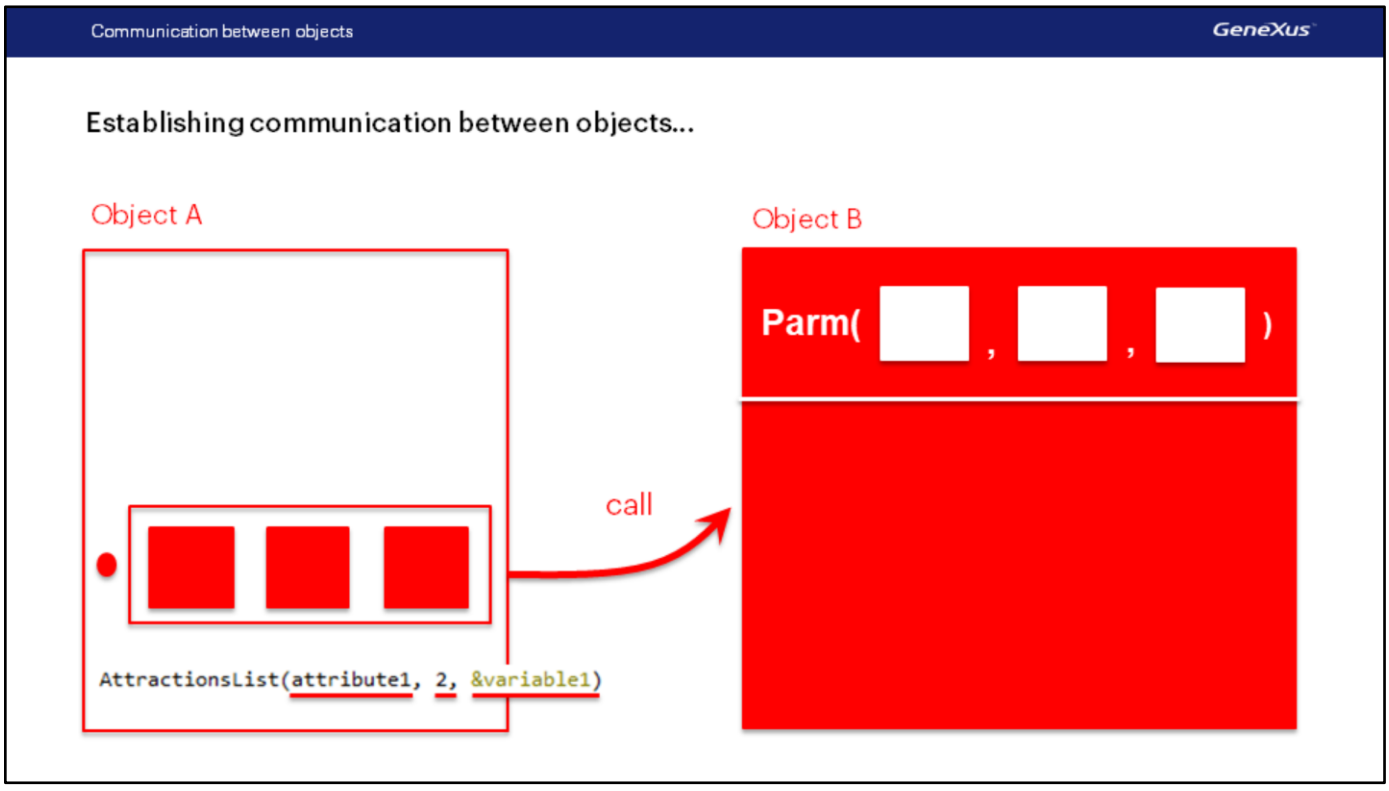
Now we only have to remove the fixed filter that we had (country value 2) in the For each command and change it for the variable whose value is received as a parameter.



Note that since the Parm rule has been stated in this way, from now on any object that calls the procedure will be able to (and will have to) send the country identifier value. It will no longer be possible to call this procedure without sending it a value of this type. That's why the AttractionsList procedure will no longer be displayed in the Developer Menu.

In the web panel case we had this value in a variable (that user entered in screen). But if we had the data in an attribute, we would include the corresponding attribute between the brackets.

We may also send a value.



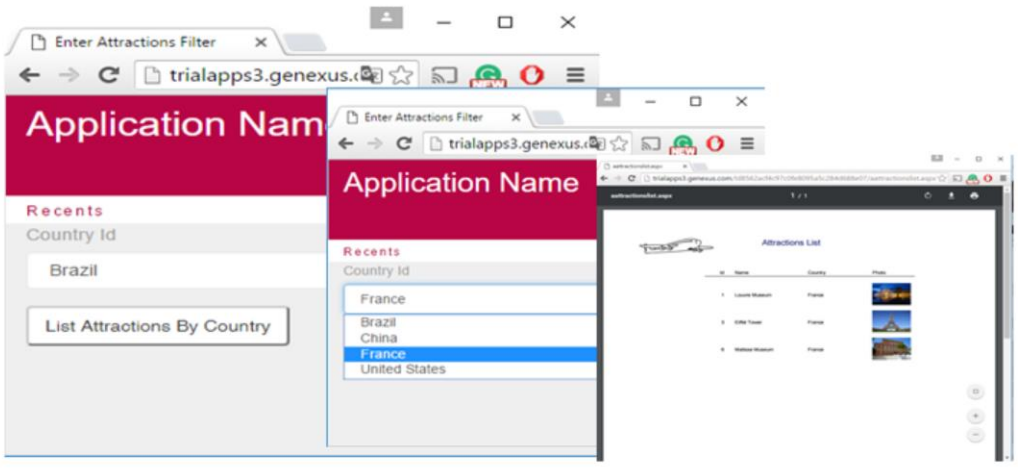
...Or, if we had to send two or more values, we would send several attributes, and/or explicit values, and/or variables separated by commas.

These parameters are also declared in the parm rule in an ordered manner, separated by commas.

Obviously, an object that doesn't receive parameters must not declare the Parm rule.

DEMO

- We try what we have done in GeneXus: F5
  Note the AttractionsList procedure is no longer displayed in the Developer Menu

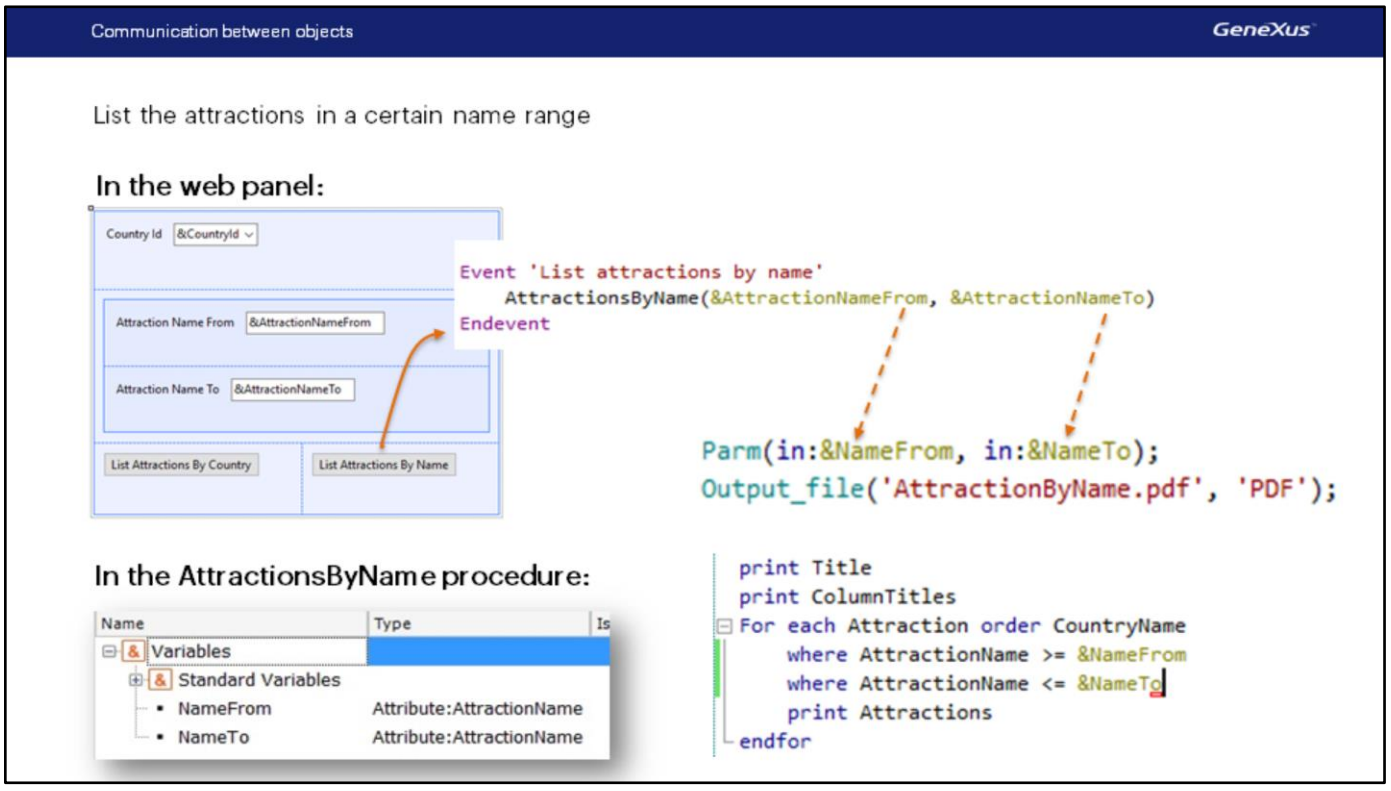


We try what we have done by pressing F5. We see that the AttractionsList procedure is no longer displayed. Now we can only call it through the web panel...

In the country combo, we choose France... and press the button.

By choosing the value France from the dynamic combo, the identifier value of France was internally selected (in this case, value 2); that value is sent to the AttractionslList procedure.

We see that the report is run, showing only the attractions of the country France.



Now, let's suppose that we want to list all the attractions whose names match a range of values selected by the user. For example, between “A” and “D”.

To do so, to the web panel that we previously created we will add the possibility for the user to enter a start name and an end name. In this way, pressing a button will call a list to show all the tourist attractions whose names are within that range.

We open the EnterAttractionsFilter web panel and add a table with two variables:

- &AttractionNameFrom... is based on the AttractionName attribute,
- And &AttractionNameTo, which is also based on the definition of AttractionName.
  As we've said before, this means that the variable definition is linked to the attribute definition, and if we change the attribute data type, the variable will be automatically changed accordingly.

Next, we add an event button called “List Attractions By Name”. We click on the button we've just added, right-click and select Go to event. From here we need to call the procedure that will print the tourist attractions within the selected range.

We already had the AttractionsList report... but it received in a parameter the country identifier, not the name range. We will save it with another name, AttractionsByName, and change its Parm rule, so that now it receives two input parameters: The &NameFrom variable and &NameTo variable

We define both variables based on the AttractionName attribute.

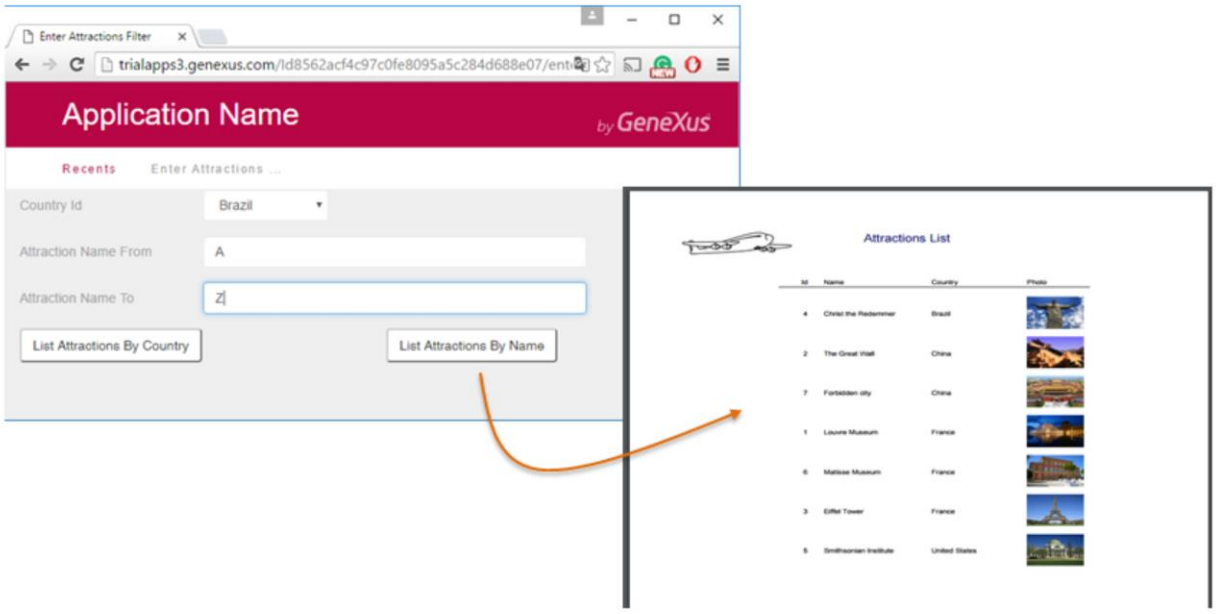
Note that for the variables we have used different names than those of the variables we created in the web panel. What's most important is that the **data types sent and receive** must match.

The first parameter we wrote in the call will be loaded in the first parameter defined in the Parm rule of the called object, and the second parameter of the call will be loaded in the second parameter of the invoked object. We must pay attention to the order used in the invocation and definition of the Parm rule. It's good

practice to use related names as we've done here, in order to better understand the code.

With this the procedure is ready. Let's press F5 to run it...

At runtime...


Enter Attractions Filter
trialapps3.genexus.com/Id8562acf4c97c0fe8095a5c284d688e07/ent
Application Name
by GeneXus
Recents
Enter Attractions ...
Country Id
Brazil
Attraction Name From
A
Attraction Name To
Z
List Attractions By Country
List Attractions By Name
Attractions List
Id
Name
Country
Photo
4 Christ the Redeemer Brazil
2 The Great Wall China
7 Forbidden city China
1 Louvre Museum France
6 Matisse Museum France
3 Eiffel Tower France
5 Smithsonian Institute United States

GeneXus™

The power of doing.

Videos	training.genexus.com
Documentation	wiki.genexus.com
Certifications	training.genexus.com/certifications