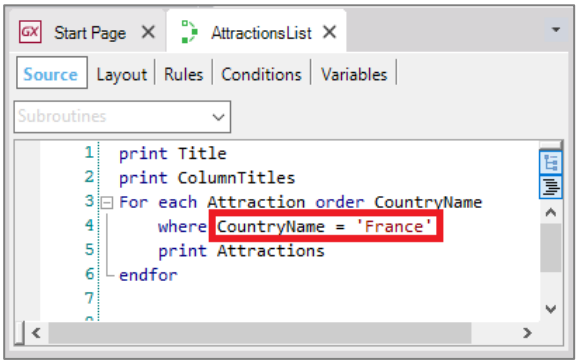
Comunicación entre objetos

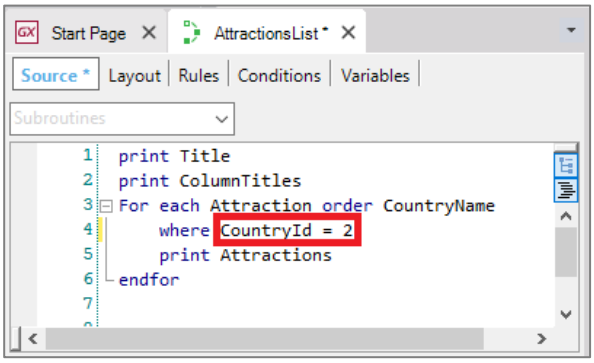
En situaciones anteriores nos hemos encontrado con la necesidad de llamar a un objeto desde otro.

Por ejemplo, cuando implementamos el objeto procedimiento AttractionsList, necesitamos filtrar las atracciones que tienen como nombre de país “France”:



o, lo que era similar, identificador de país igual a 2 (que correspondía a “France”)

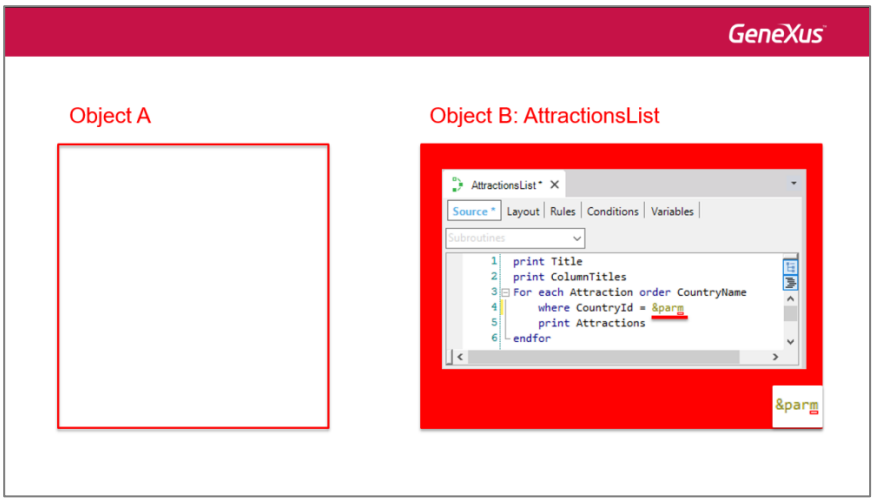
…y para lograrlo, utilizamos valores fijos en el código:



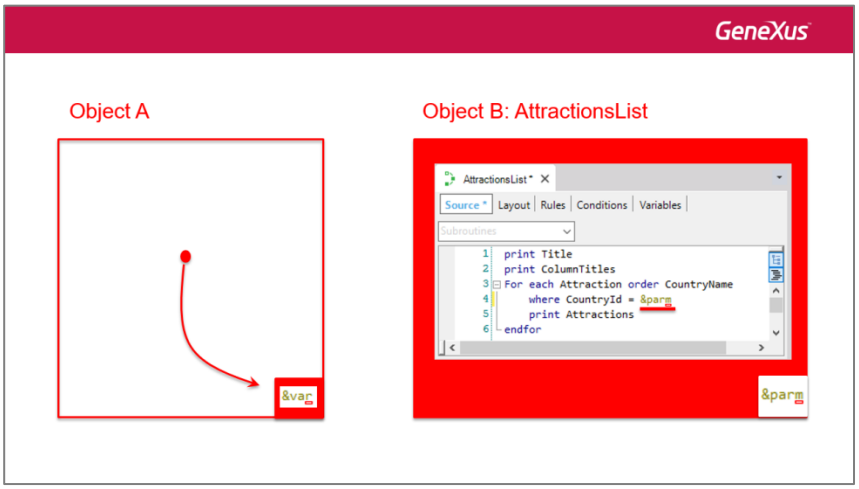
Pero esto implica que si quisiéramos filtrar las atracciones de un país diferente a Francia, deberíamos modificar

el código del procedimiento ¡cada vez!

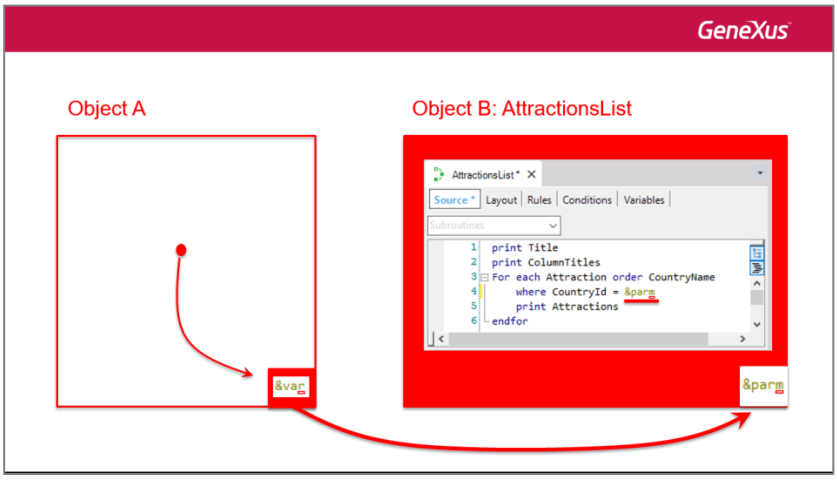
Lo ideal sería que pudiéramos “recibir” de alguna manera en este objeto ese valor por el que queremos filtrar.



Dicho de otro modo, que otro objeto GeneXus pueda permitir al usuario elegir ese valor:

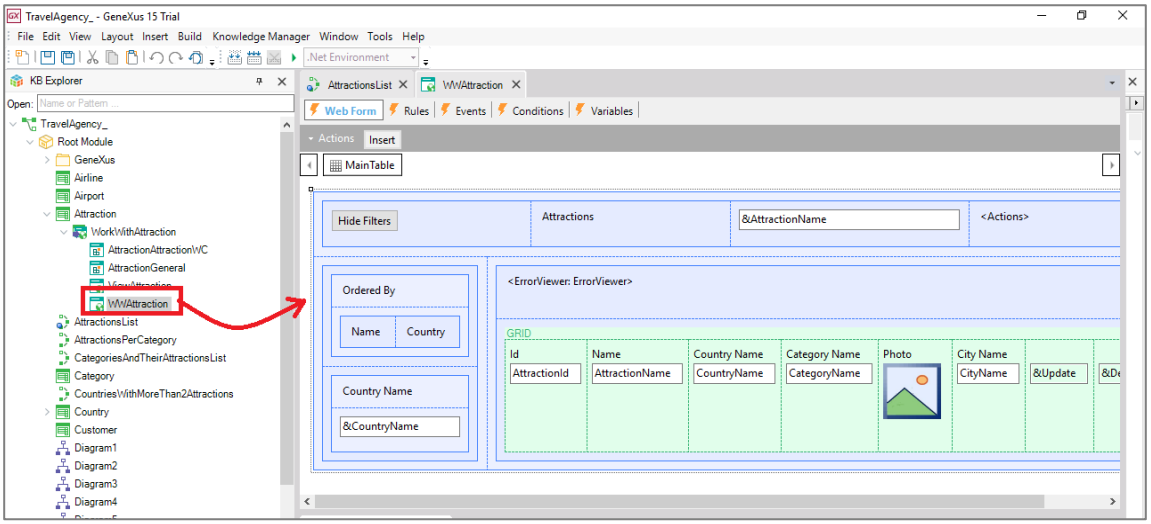


… y luego lo envíe a este objeto procedimiento para que liste las atracciones en base a ese país recibido.

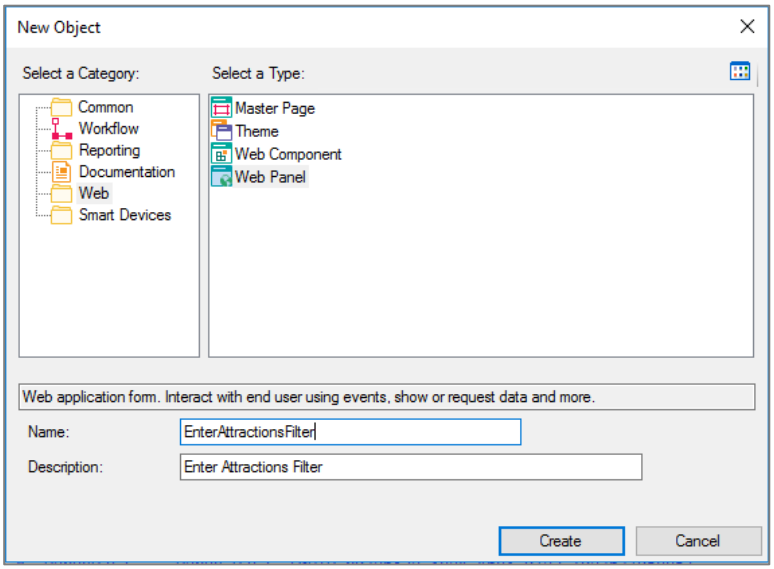


Veremos a continuación, mediante este ejemplo, cómo implementar la comunicación entre objetos GeneXus.

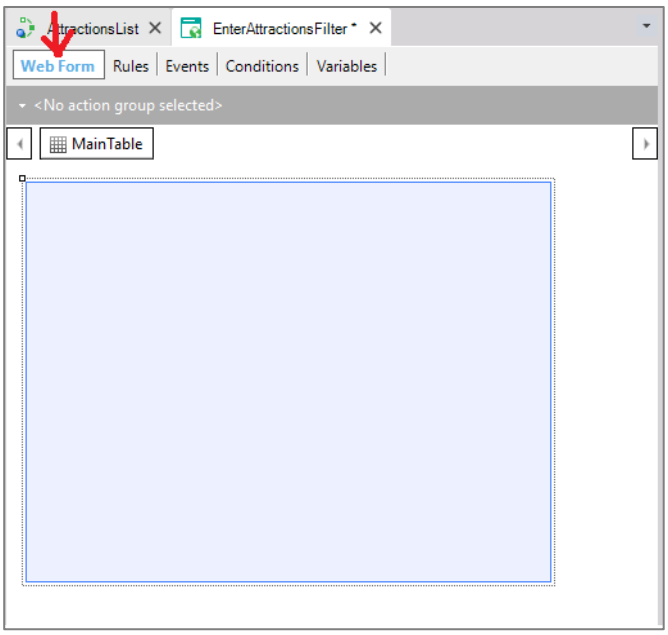
Para empezar, debemos crear un objeto que sea capaz de ofrecernos una pantalla para pedir valores al usuario y hacer algo con esos valores. El objeto que permite esto es el **web panel**, que estudiaremos en detalle más adelante. Por ahora digamos que se trata de un panel visual muy flexible que permite pedir datos al usuario, mostrar información de la base de datos o de otras fuentes, entre otras cosas. Por ejemplo, el Work With de atracciones fue implementado automáticamente por GeneXus como un Web Panel.



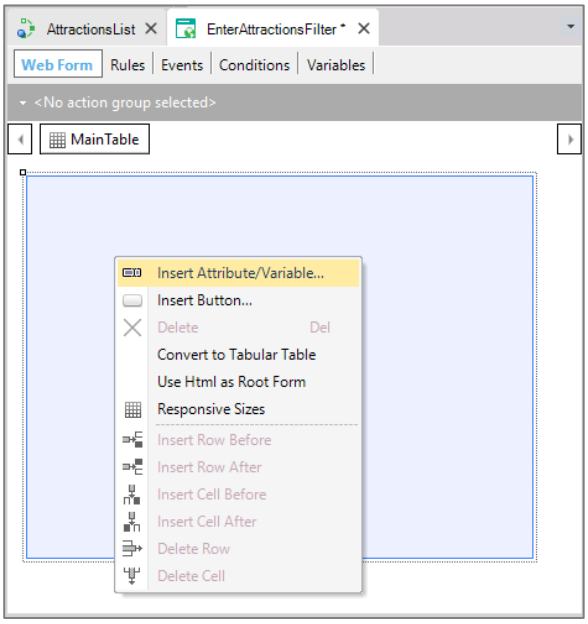
Entonces, vamos a crear un objeto de este tipo. Lo llamaremos EnterAttractionsFilter:



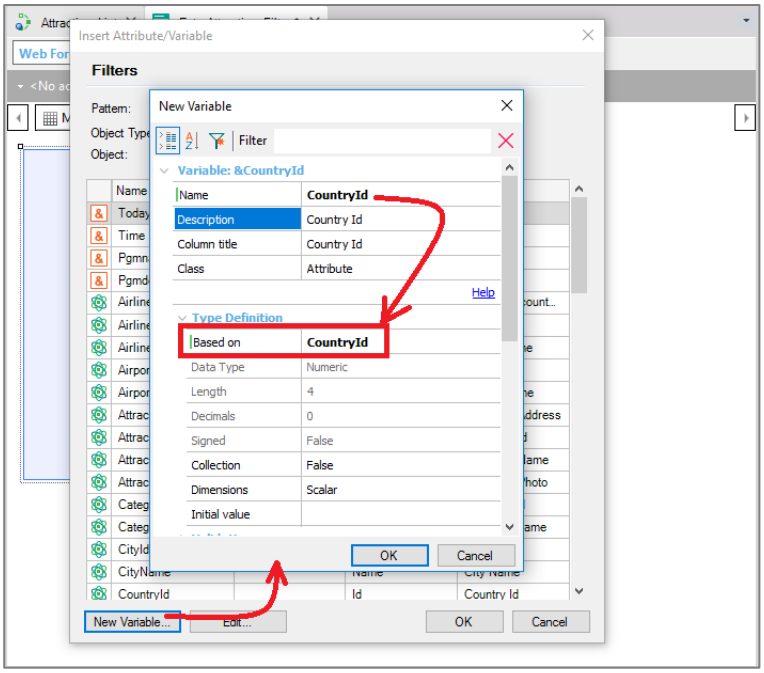
Veamos que se crea un Form Web, que será la pantalla del objeto. Contiene únicamente una tabla.



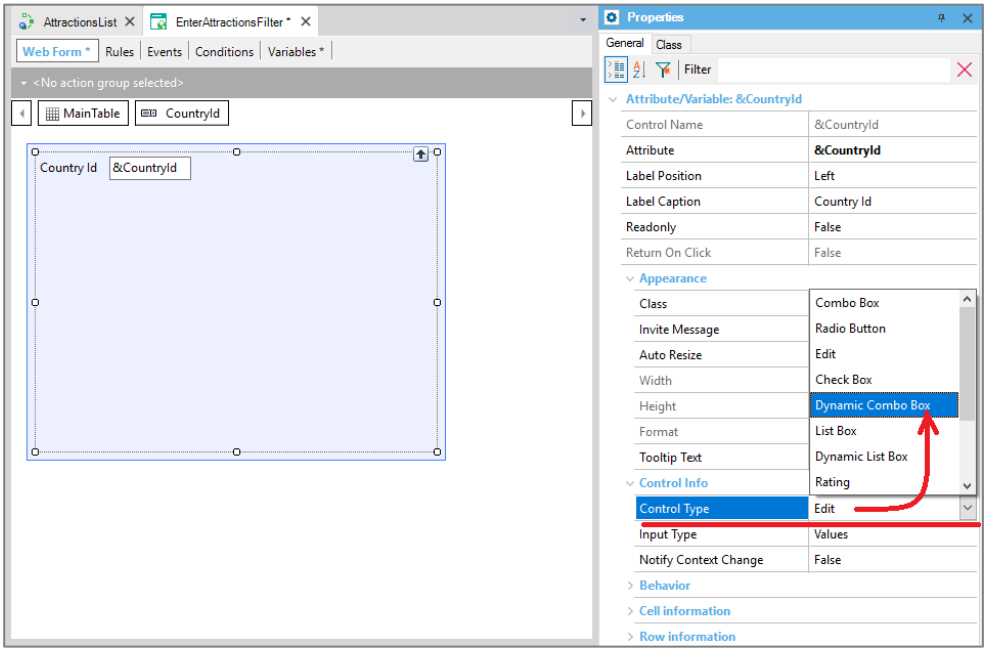
Le agregamos una variable CountryId



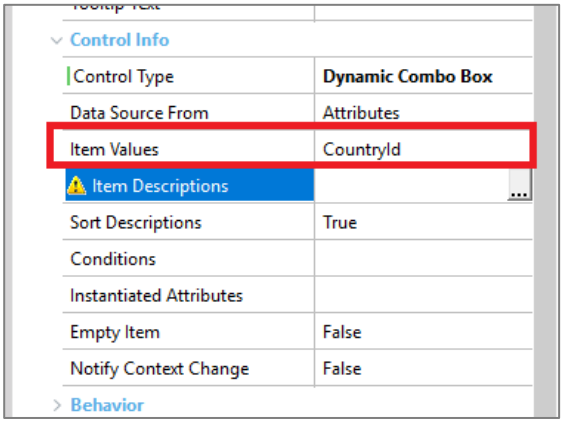
la que por nombrar igual que al atributo queda basada en él, y, por lo tanto, asume su mismo tipo de datos. De este modo si le cambiamos el tipo de datos al atributo, por ejemplo, por numérico de 10 en vez de 4, la variable automáticamente cambiará tomando ese nuevo valor.



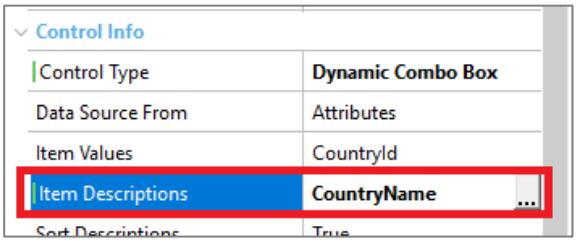
Ahora editamos las propiedades de la variable y veamos que su propiedad Control Type asume el valor Edit. Esto significa que cuando se ejecute el web panel este campo esperará que el usuario digite un valor numérico, pero no brindará ninguna ayuda para elegir qué valores existen en la base de datos y a qué país corresponde. Vamos a cambiar el tipo de control por Dynamic Combo Box:



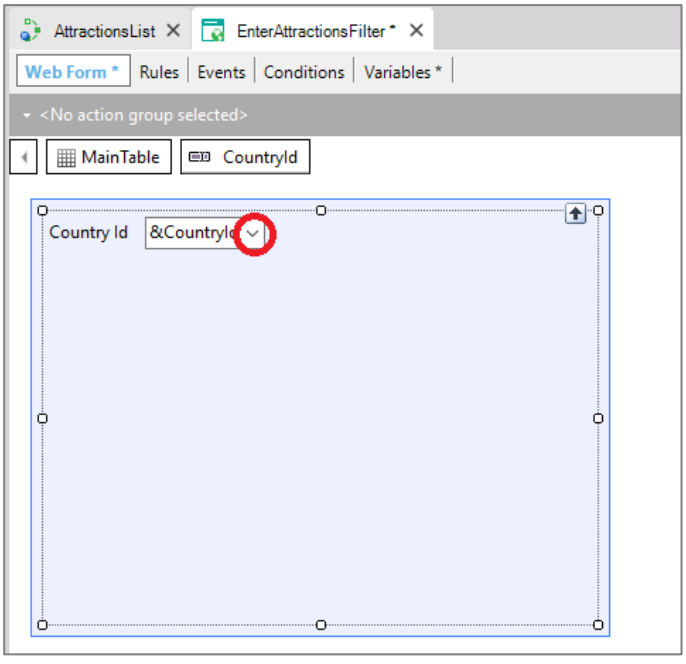
De esta manera se le va a ofrecer al usuario una serie de valores extraídos de la base de datos, para que él elija el que desea. ¿Qué valores? Los del atributo CountryId:



Es decir, se va a recorrer la tabla Country y cargar en el combo los CountryIds existentes. Pero como los identificadores no suelen decirnos nada, si bien la variable va a almacenar un identificador de país, lo que se le va a mostrar al usuario en la pantalla es el contenido del atributo que indiquemos aquí… Elegimos mostrar el nombre de país:

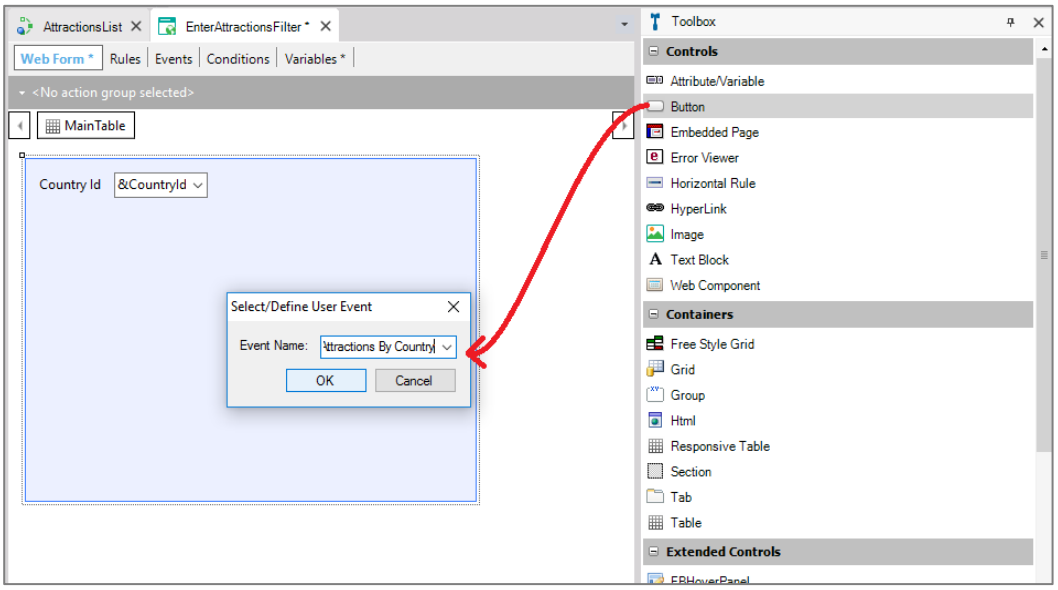


Y vemos cómo aparece la flechita indicadora de combo:

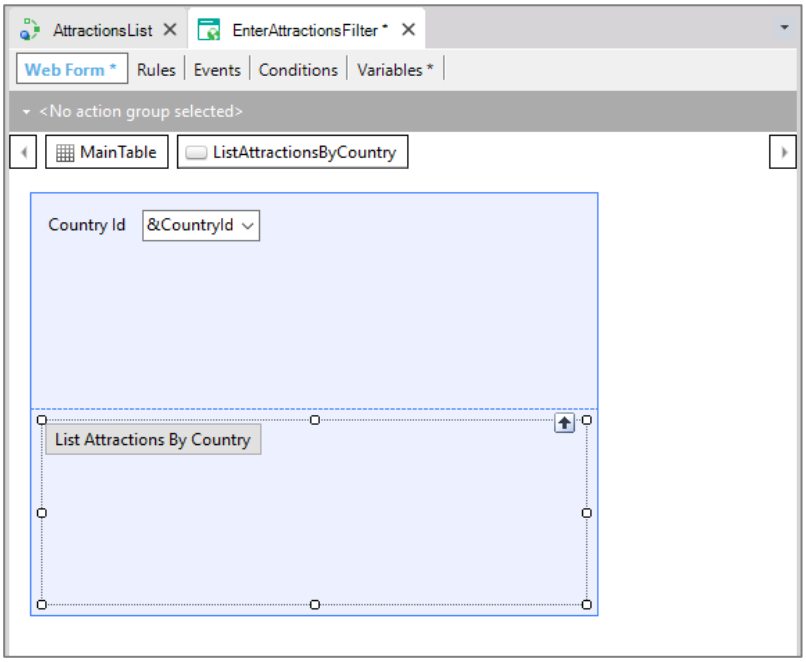


Resumiendo, esto nos ofrecerá en ejecución un combo que nos presentará la lista de países de la base de datos para elegir el que nos interesa.

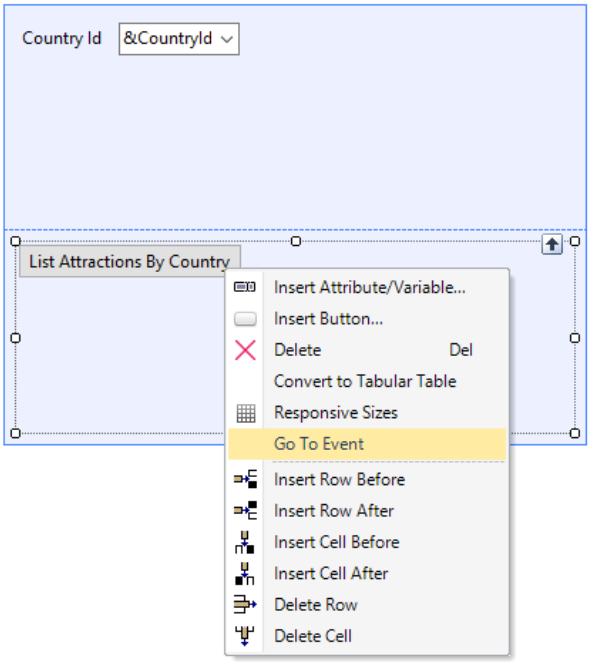
Agregamos además un botón. Nos pide que indiquemos el nombre del evento que ese botón tendrá asociado. Lo llamamos: “List Attractions By Country”:



Y vemos que el texto del botón asume el mismo nombre por defecto:



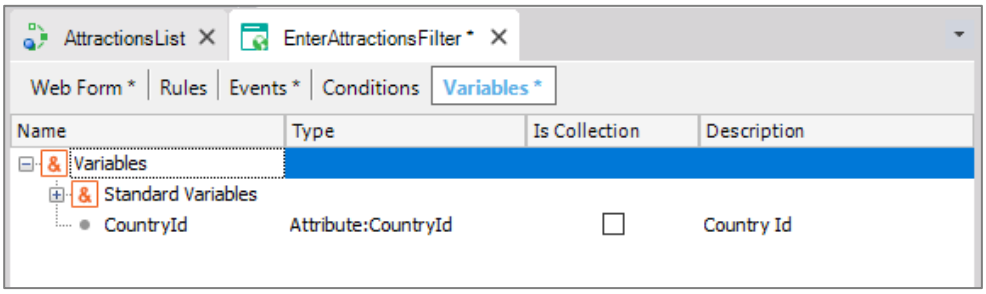
Si nos posicionamos sobre él, presionamos botón derecho y elegimos Go to Event…



…vemos que se creó un evento con ese nombre, y se pasó de la solapa Web Form a la Events en forma automática, y el cursor está esperando a que ingresemos el código que se ejecutará cuando este evento se dispare. Es decir, cuando el usuario presione el botón asociado.

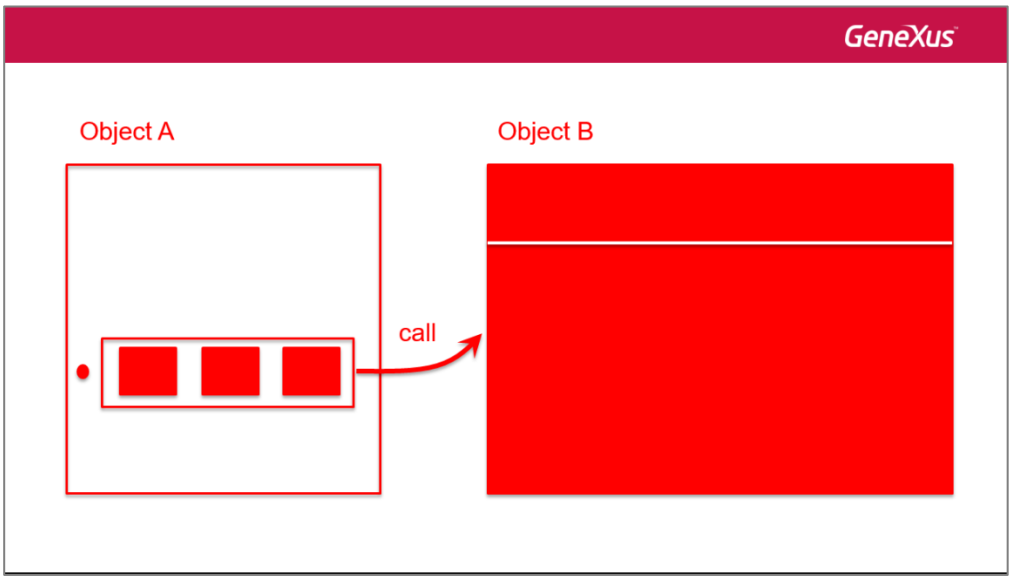
Lo que necesitamos que se haga en ese momento es llamar al objeto procedimiento AttractionsList que lista las atracciones y enviarle el país que queremos que utilice para filtrarlas:

Notemos que al momento de apretar el botón y ejecutar este código, la variable &CountryId contendrá el identificador de país del país que el usuario haya elegido en el combo box en la pantalla. Anteriormente vimos que una variable es una porción de memoria a la cual le damos un nombre y nos sirve para guardar un dato en forma temporal, y que cada objeto tiene su sección de variables,

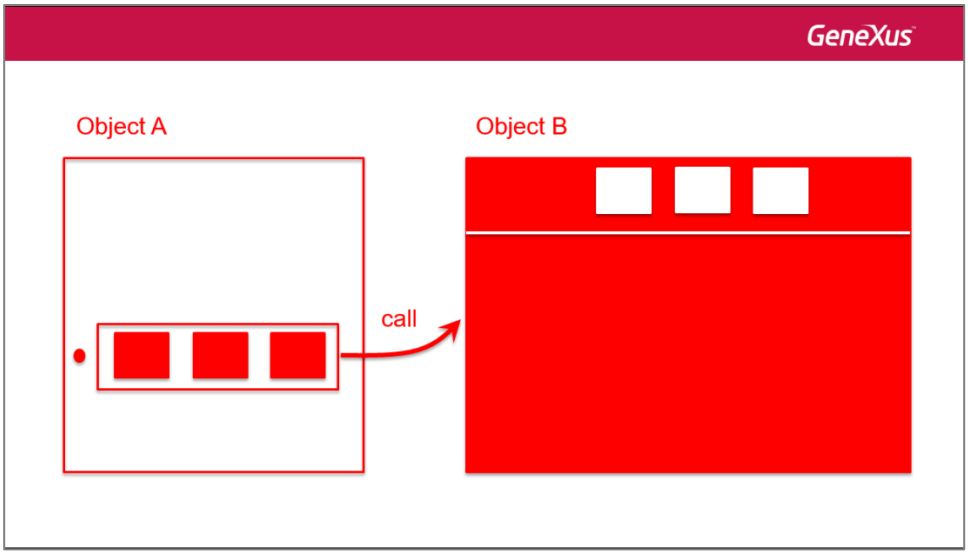


o sea que las variables que se definen en un objeto son conocidas solamente dentro de ese objeto. Así, si dos objetos tienen una variable CountryId definida, aunque se llamen igual, se tratará de dos variables distintas.

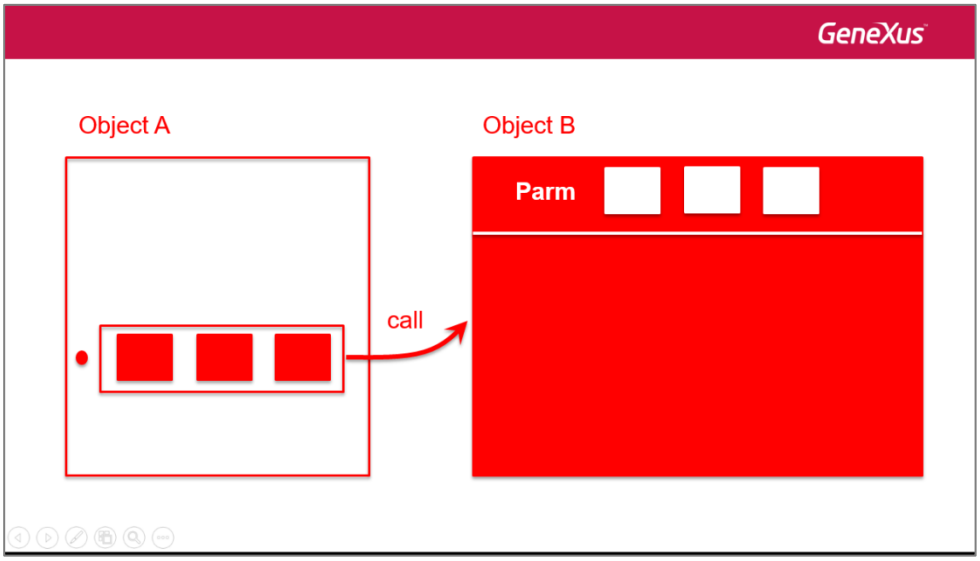
Entonces, ¿cómo hacemos para que un objeto A pueda invocar a otro objeto B en un momento dado, pasándole valores?:



y que ese otro objeto B pueda cargar en sus variables internas los valores que se le enviaron, para poder hacer algo con esa información:

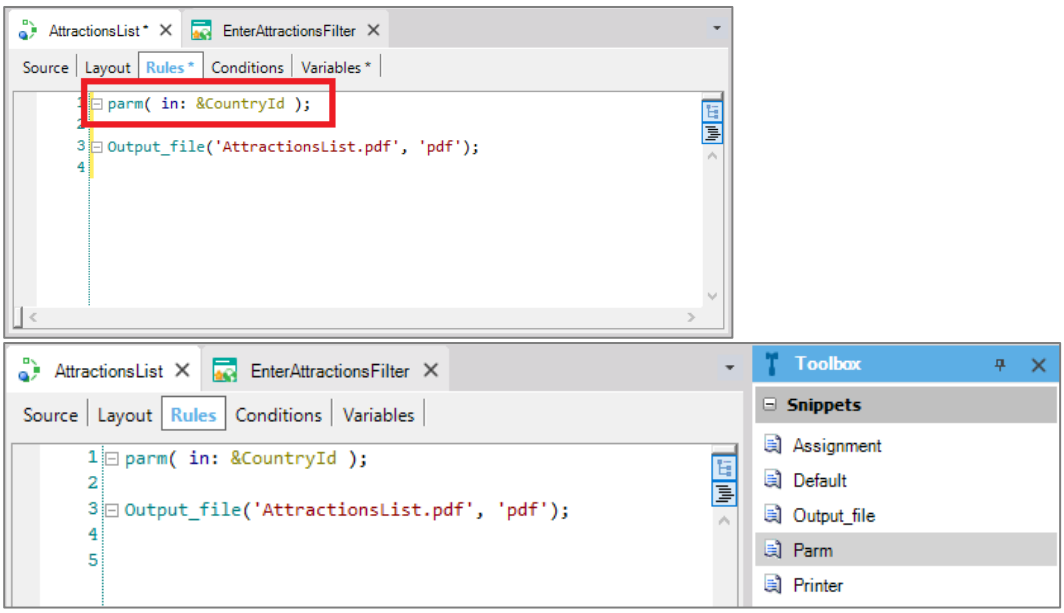


Para que un objeto pueda recibir valores (a los que llamamos **parámetros**), debemos ir a su sección Rules y escribir una regla **Parm**. Esa regla **Parm** declara los parámetros que el objeto puede recibir y/o devolver a quien lo llame.

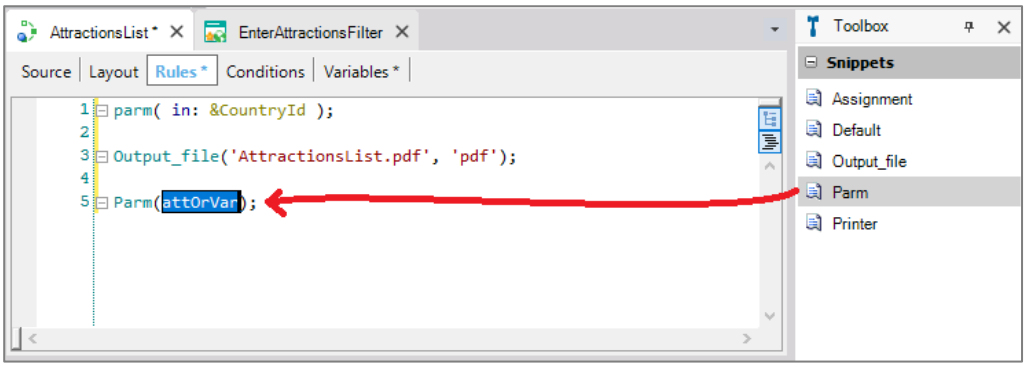


Como en nuestro ejemplo quien va a recibir los valores es el objeto procedimiento AttractionsList, abrimos el objeto y vamos a su sección de reglas.

Escribimos:



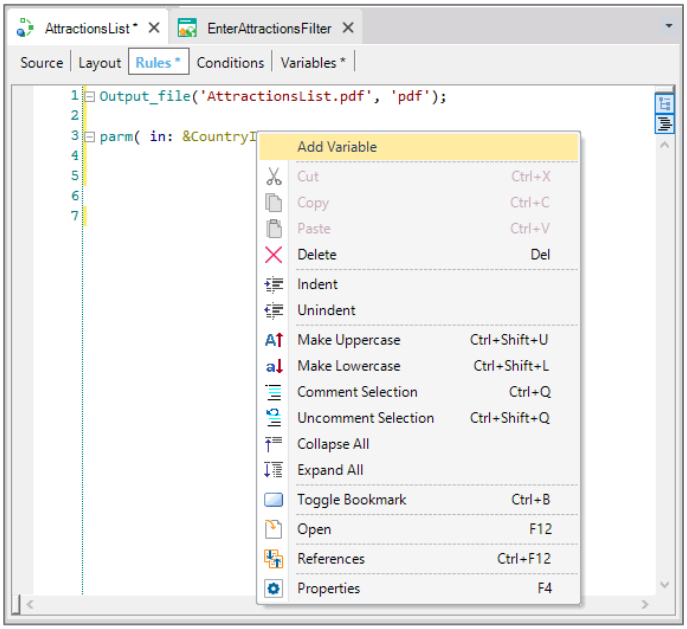
Observemos que en la Toolbox se nos ofrecen todas las reglas que podríamos escribir en un objeto de este tipo. Entre ellas, la parm. Podríamos haberla arrastrado desde allí.



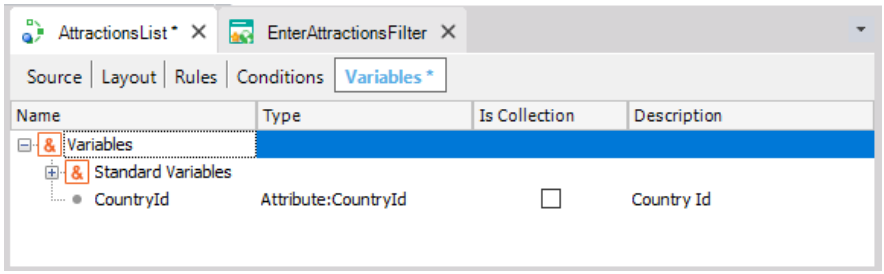
Vemos también nos informa que tenemos que sustituir esto por un atributo o variable. Luego veremos el caso de los atributos. Por ahora solamente vemos el caso de las variables.

Con “in” estamos indicando que la variable &CountryId será un parámetro de entrada. Esto significa que solamente será utilizado para recibir un valor de quien lo llame. No para devolver. Podemos omitir esta información y dejar que GeneXus la infiera.

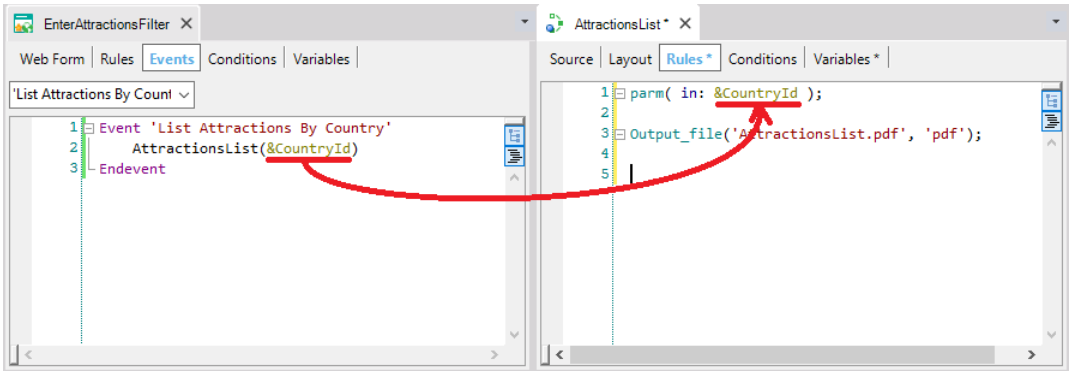
Hemos escrito el **nombre** de la variable, pero no la hemos ingresado como variable en el objeto. Para hacerlo, una de las formas es posicionarnos sobre el nombre, dar botón derecho y elegir “Add variable”:



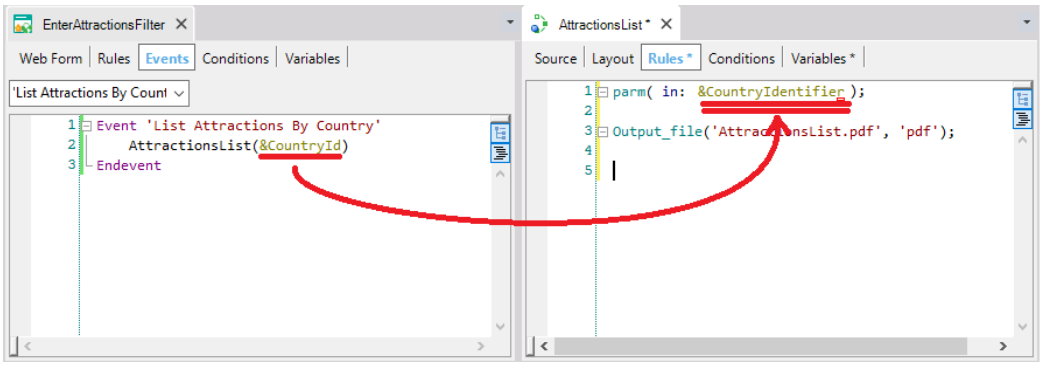
Si ahora vamos a la solapa de variables vemos que se la ha definido, basada, por defecto, en el tipo del atributo CountryId. Esto se debe a que la llamamos igual que a un atributo.



En este objeto hemos definido la variable con el mismo nombre y tipo de datos que la que usamos en el web panel para que el usuario ingresara el país:



Sin embargo, como dijimos, son dos variables distintas. Una válida solamente en el web panel y la otra en el procedimiento. Podríamos haberlas llamado distinto en ambos objetos, pero para que la comunicación y el pasaje de información sea correcta, el tipo de datos debe coincidir entre quien llama y quien es llamado.

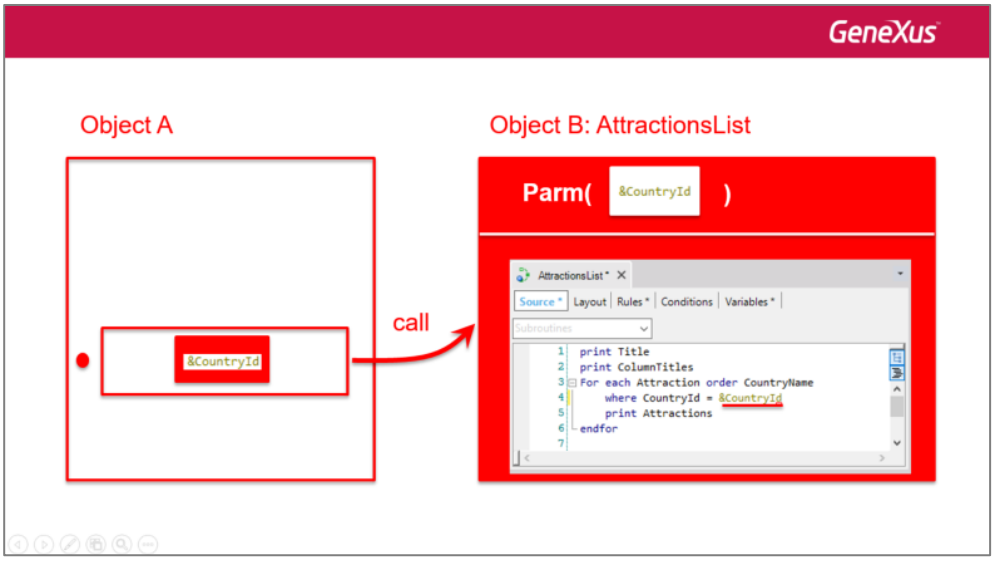


Ahora, nuestro objeto procedimiento está preparado para recibir a un identificador de país, en este caso desde el webpanel EnterAttractionsFilter.

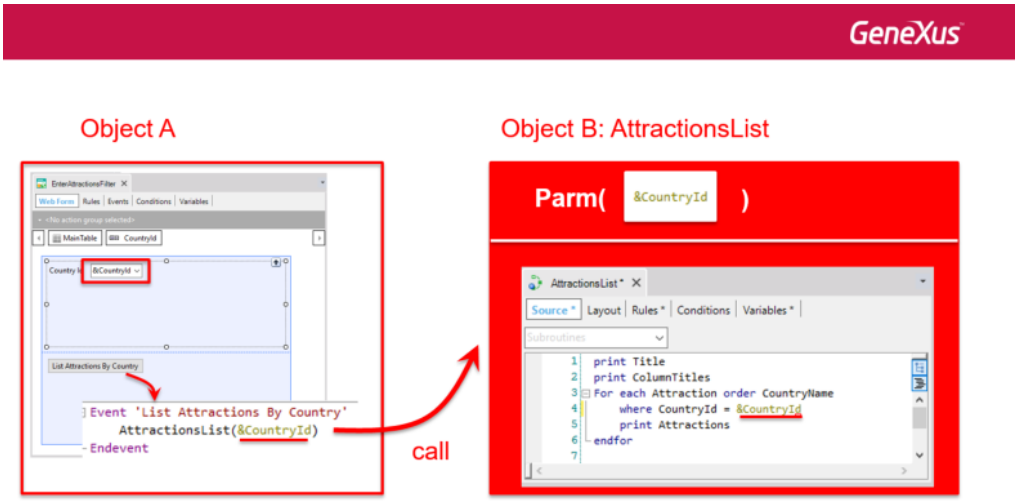
Sólo nos resta quitar el filtro fijo que teníamos (el valor 2 de país) en el For each, y cambiarlo por la variable cuyo valor es recibido como parámetro.

Observemos que al haber definido la regla Parm de esta manera, de ahora en adelante cualquier objeto que llame al procedimiento podrá (y deberá) pasarle el valor del identificador del país.

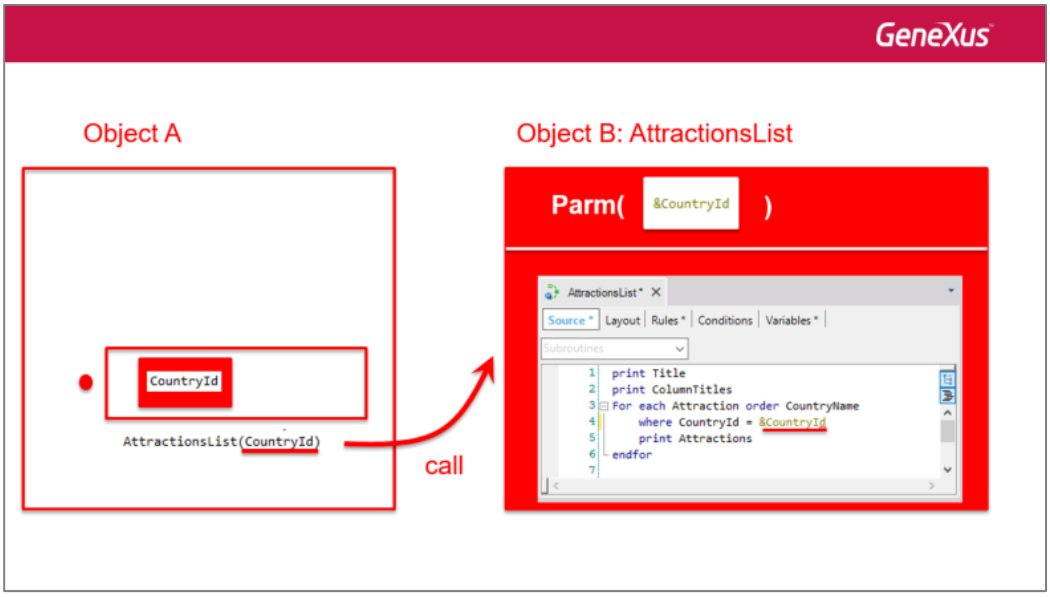
Ya no podrá invocarse a este procedimiento sin enviarle un valor de este tipo. Es por esta razón que el procedimiento AttractionsList ya no va a aparecer en el Developer Menu.



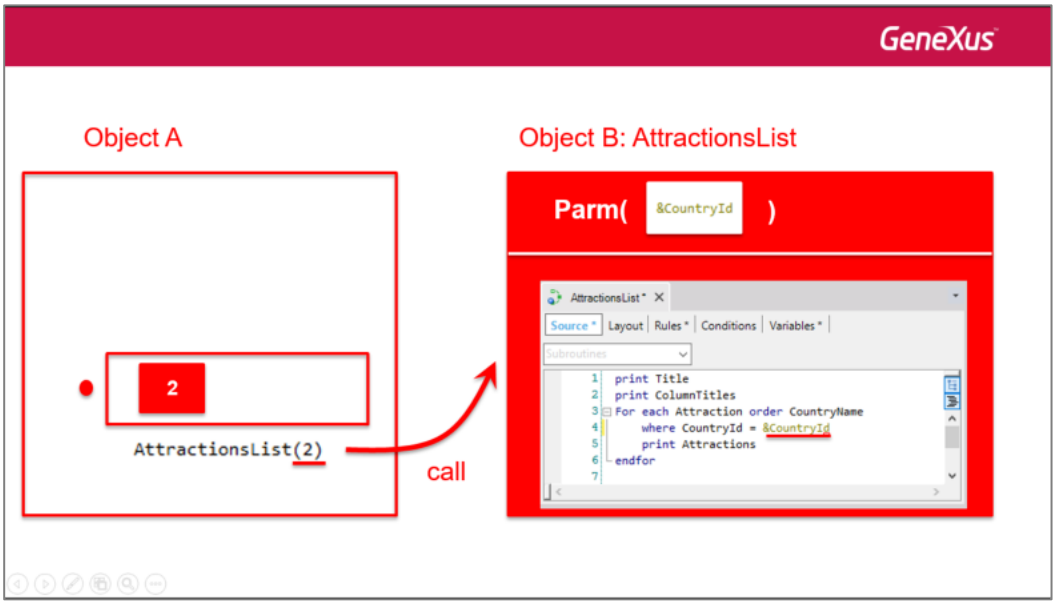
En el caso del webpanel teníamos ese valor en una variable:



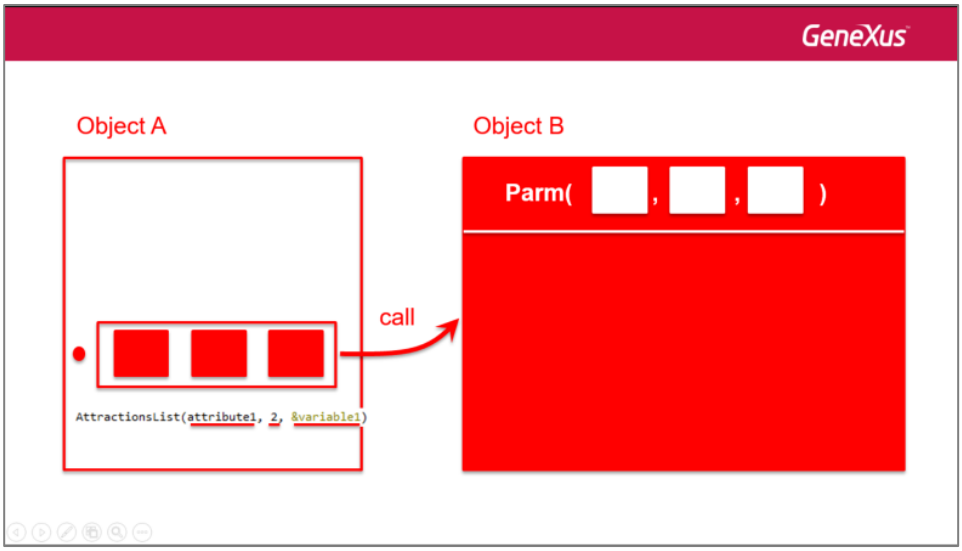
Pero si tuviéramos el dato en un atributo, incluiríamos dentro del paréntesis al atributo que corresponda,



También podríamos pasar directamente un valor:



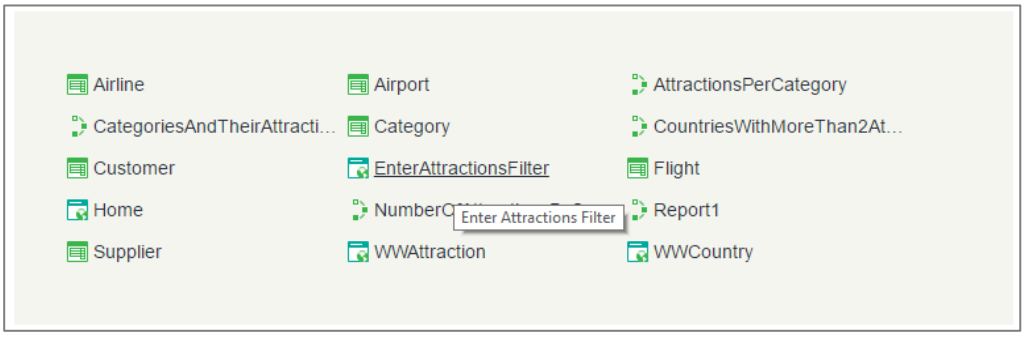
o en caso de tener que pasar dos o más valores, enviaríamos varios atributos, y/o valores explícitos, y/o variables, separados por coma:



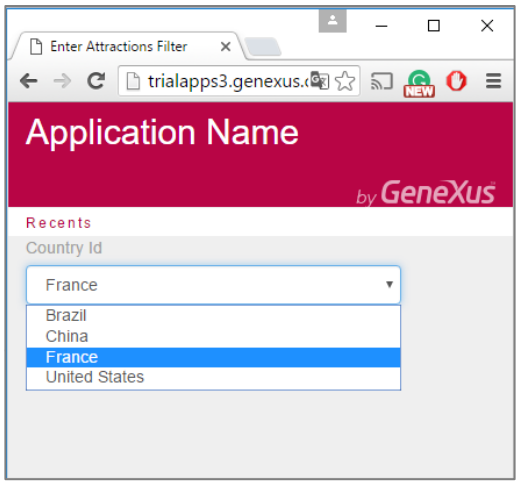
Esos parámetros también se declaran en la regla parm en forma ordenada y separados por comas.

Evidentemente, un objeto que no recibe parámetros no debe declarar regla Parm.

Vamos a probar lo que hemos hecho: F5. Vemos que ya no aparece el procedimiento AttractionsList. Ahora solamente podemos invocarlo a través del web panel…

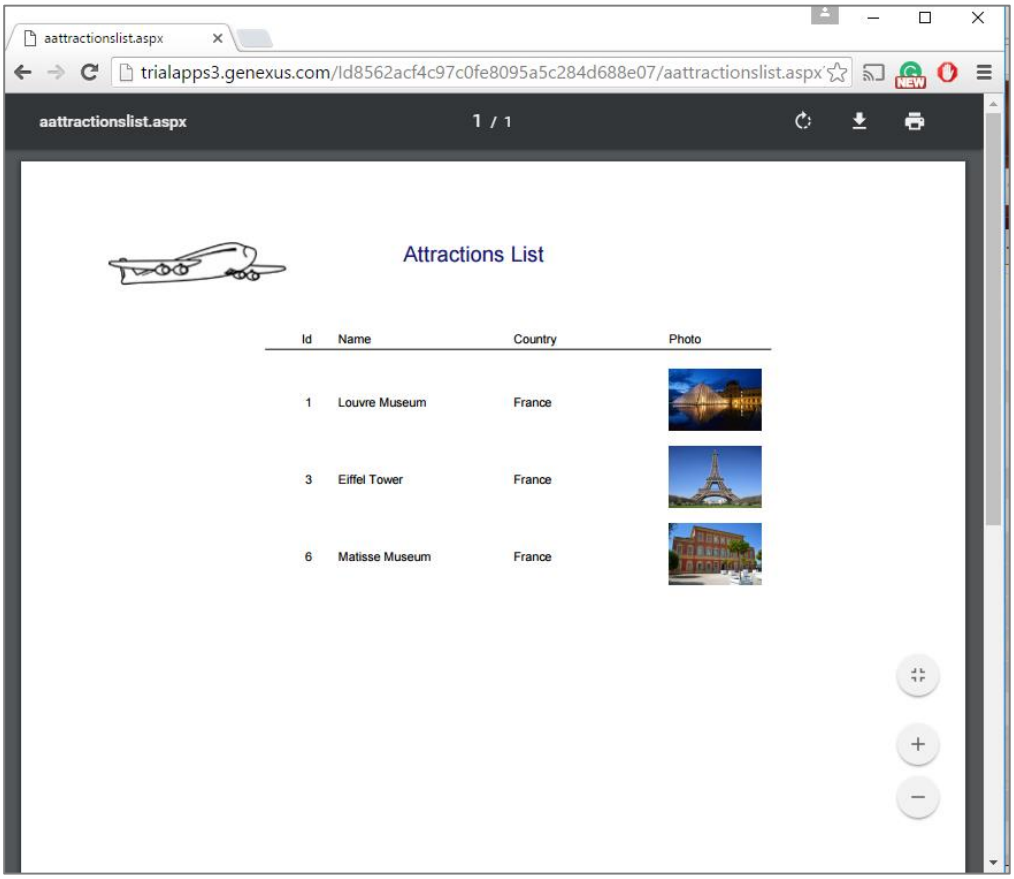


En el combo del país, elegimos Francia…



Al haber elegido el valor France del dynamic combo, internamente se seleccionó el valor del identificador de Francia (en este caso el valor 2) y ese valor es el que se le envía al procedimiento AttractionslList.

Vemos que se ejecuta el reporte, mostrándonos solamente las atracciones del país France:

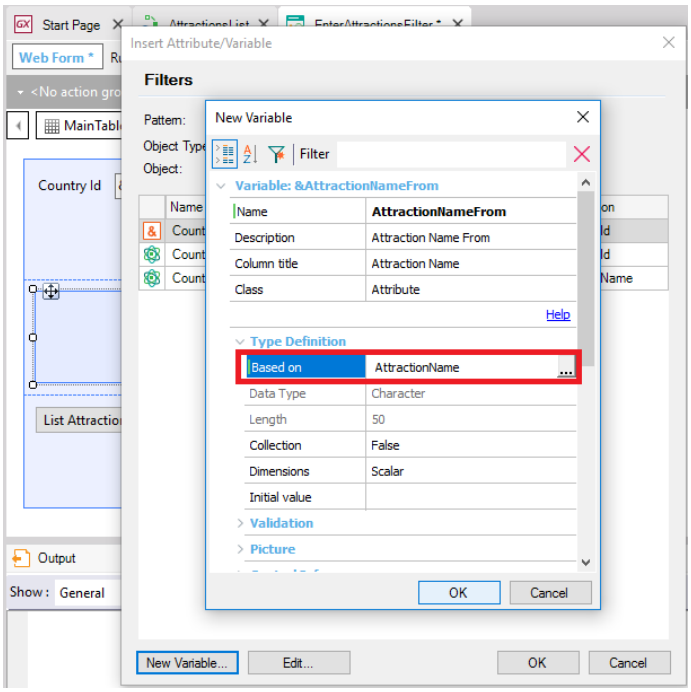


Vamos a suponer ahora que queremos listar todas las atracciones cuyos nombres se encuentren entre dos valores elegidos por el usuario. Por ejemplo, entre la “A” y la “D”.

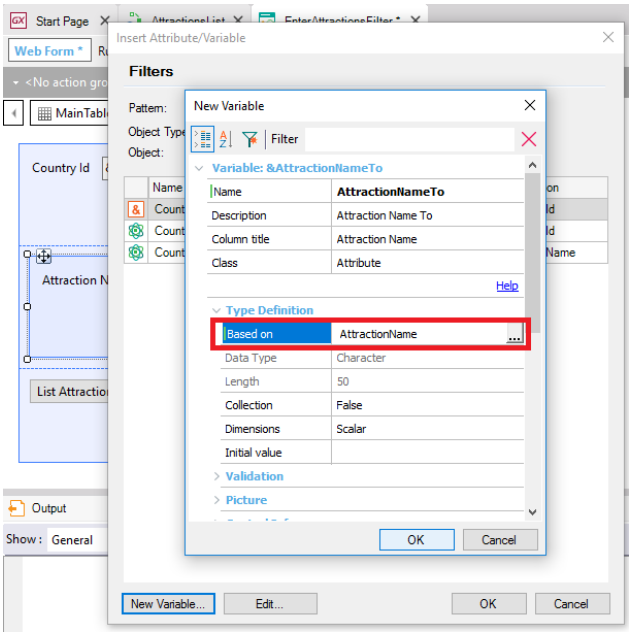
Para eso, vamos a agregarle al webpanel que definimos previamente la posibilidad de que el usuario ingrese un nombre inicial y un nombre final para que, presionando un botón, se invoque a un listado que muestre todas las atracciones turísticas cuyos nombres se encuentren dentro de ese rango.

Vamos al webpanel EnterAttractionsFilter y agregamos una tabla con dos variables.

- &AttractionNameFrom… basada en la definición del atributo AttractionName:

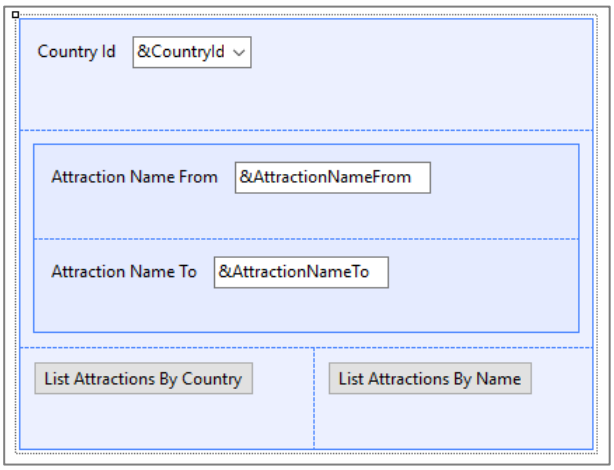


- y &AttractionNameTo, también basada en la definición de AttractionName:



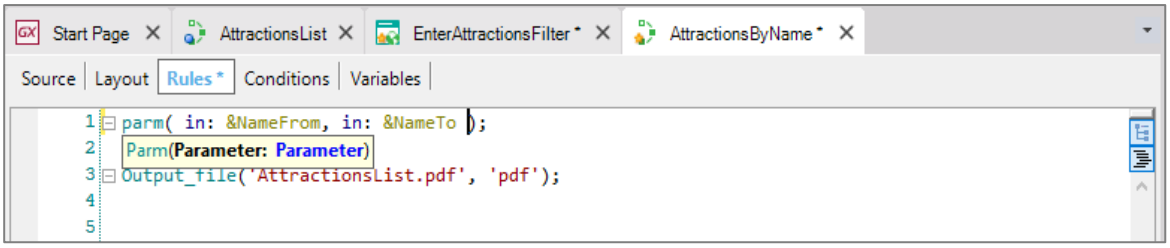
Como ya dijimos, esto significa que la definición de la variable está enlazada con la definición del atributo y si cambiamos el tipo de datos del atributo, automáticamente se cambiará la variable en forma acorde.

Luego agregamos un botón de evento “List Attractions By Name”:

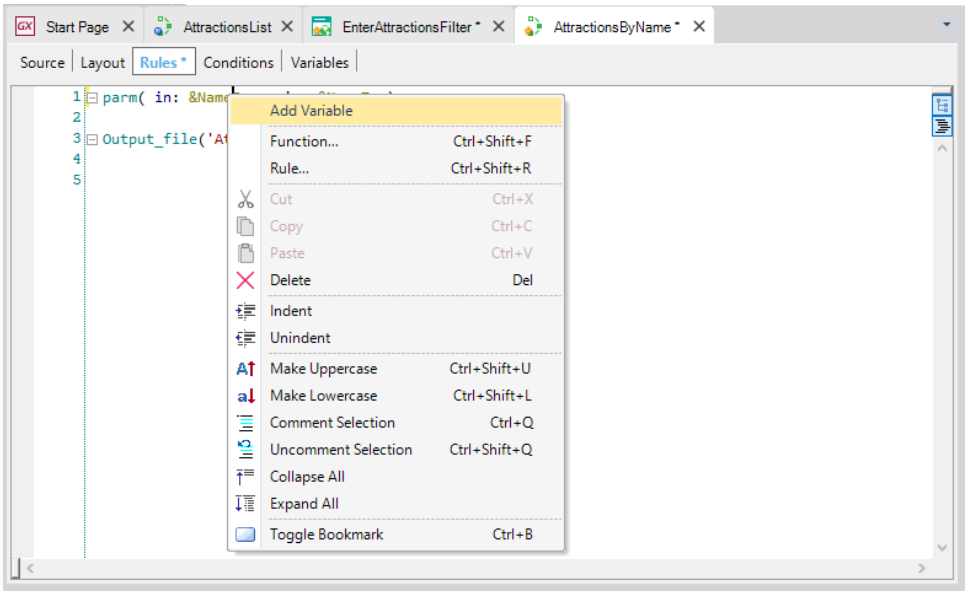


Nos posicionamos en el botón que acabamos de agregar, presionamos botón derecho y elegimos Go to event. Tenemos que invocar aquí dentro al procedimiento que imprimirá las atracciones turísticas del rango.

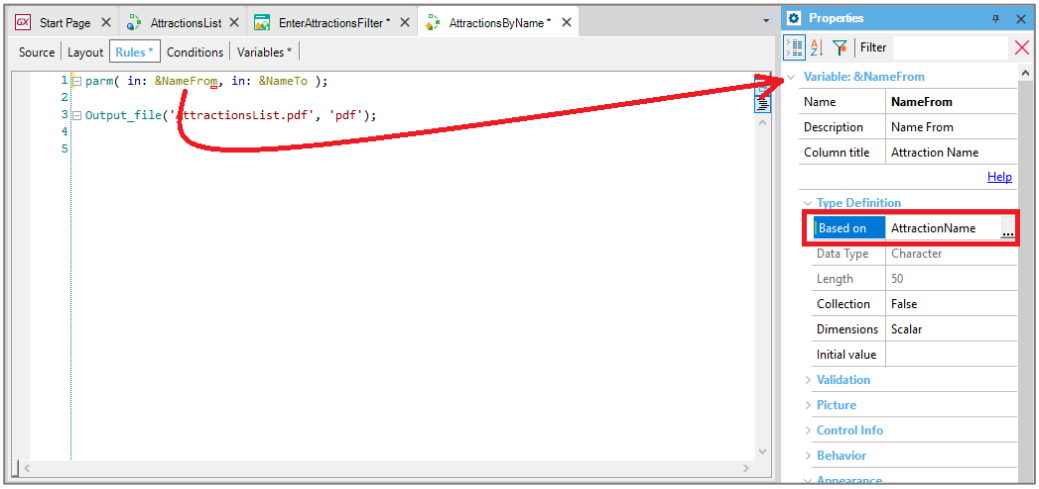
Ya teníamos el reporte AttractionsList… pero recibía por parámetro el identificador de país, no el rango de nombres. Vamos a grabarlo con otro nombre, AttractionsByName, y modificar su regla parm, para que ahora reciba dos parámetros de entrada: la variable &NameFrom, y la variable &NameTo:



Tenemos que definirlas como variables, y especificarle el tipo de datos, así que hacemos botón derecho sobre la primera y presionamos “Add Variable”



y editando sus propiedades, la basamos en el atributo AttractionName:



Hacemos lo mismo con &NameTo.

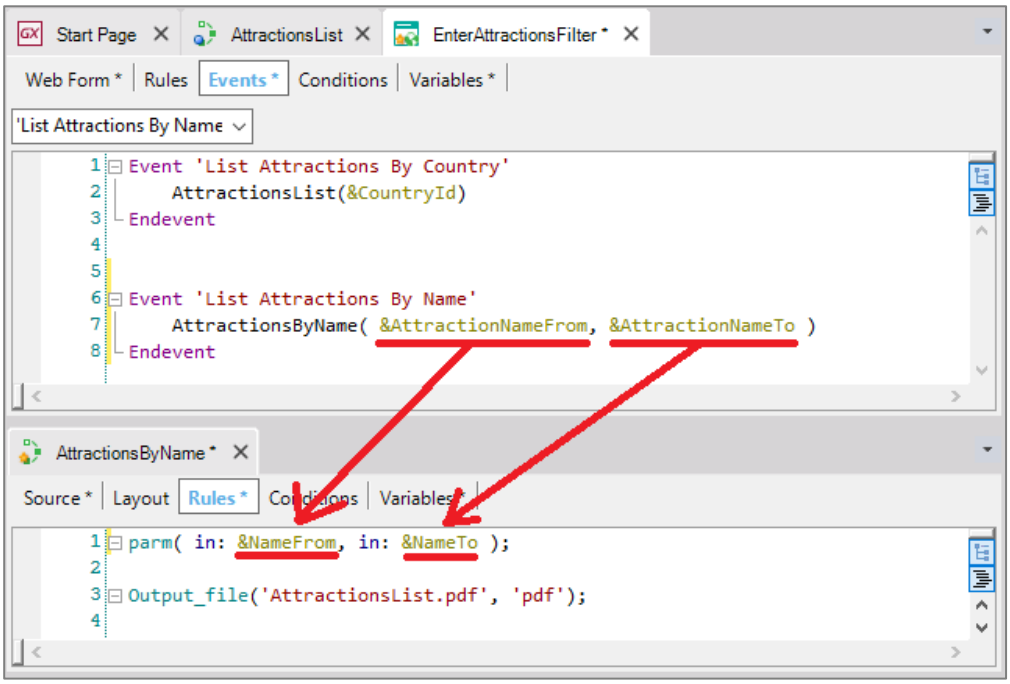
Ahora vamos a utilizar estas variables/parámetros en el For each…, para quedarnos con las atracciones que cumplan que su AttractionName sea mayor o igual al valor de la variable &NameFrom, así como menor o igual al valor de la variable &NameTo:

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

Con esto el procedimiento está listo y solamente nos resta invocarlo desde el web panel.

Vamos al webanel y agregamos la invocación. Arrastramos el reporte AttractionsByName de esta ventana para no tener que digitarlo y entre paréntesis escribimos los parámetros separados por coma, en este caso las variables &AttractionNameFrom y &AttractionNameTo que se le ofrecen al usuario en la pantalla:

Observemos que el primer parámetro que escribimos en la invocación se cargará en el primer parámetro definido en la regla Parm del objeto llamado y el segundo parámetro de la llamada se cargará en el segundo parámetro del objeto invocado.

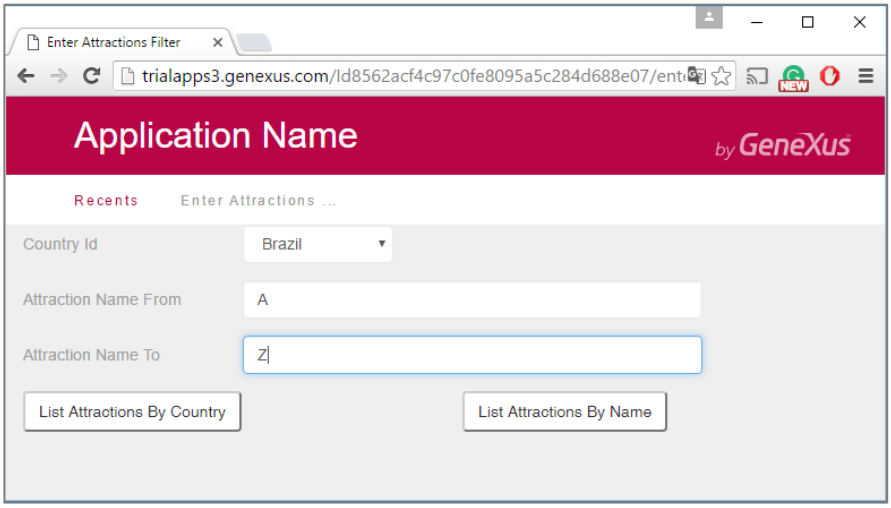


Debemos ser cuidadosos de respetar el orden en la invocación y en la definición de la regla Parm. Es buena práctica usar nombres relacionados como hicimos aquí, a efectos de entender mejor el código.

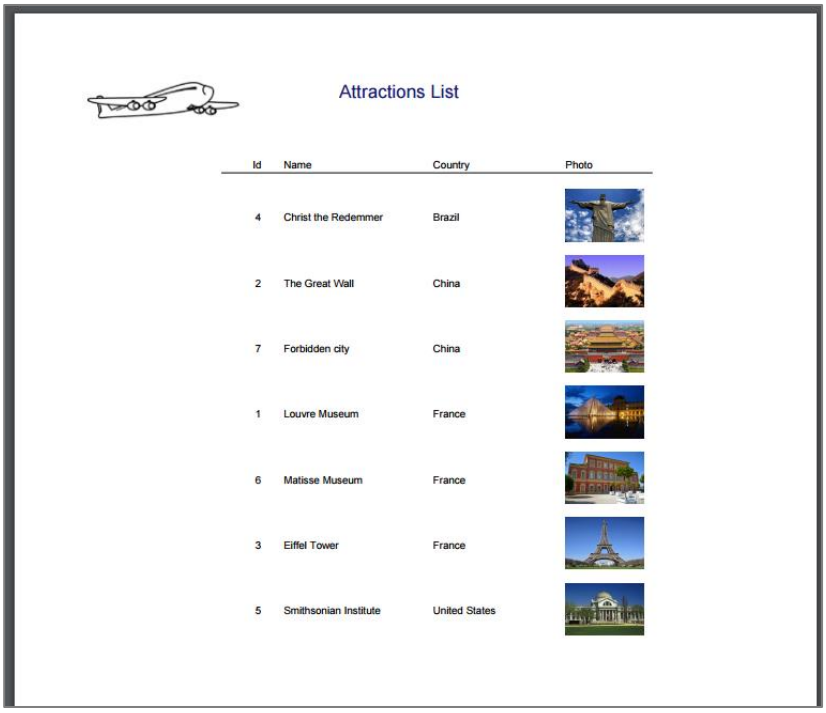
Notemos que los nombres que hemos elegido para las variables del web panel y las del procedimiento son distintas. Como ya dijimos, lo importante es que los tipos de datos enviados y recibidos, coincidan.

Presionemos F5 para ejecutar….

Abrimos el webpanel y para comenzar queremos ver todas las atracciones entre la “A” y la “Z”:



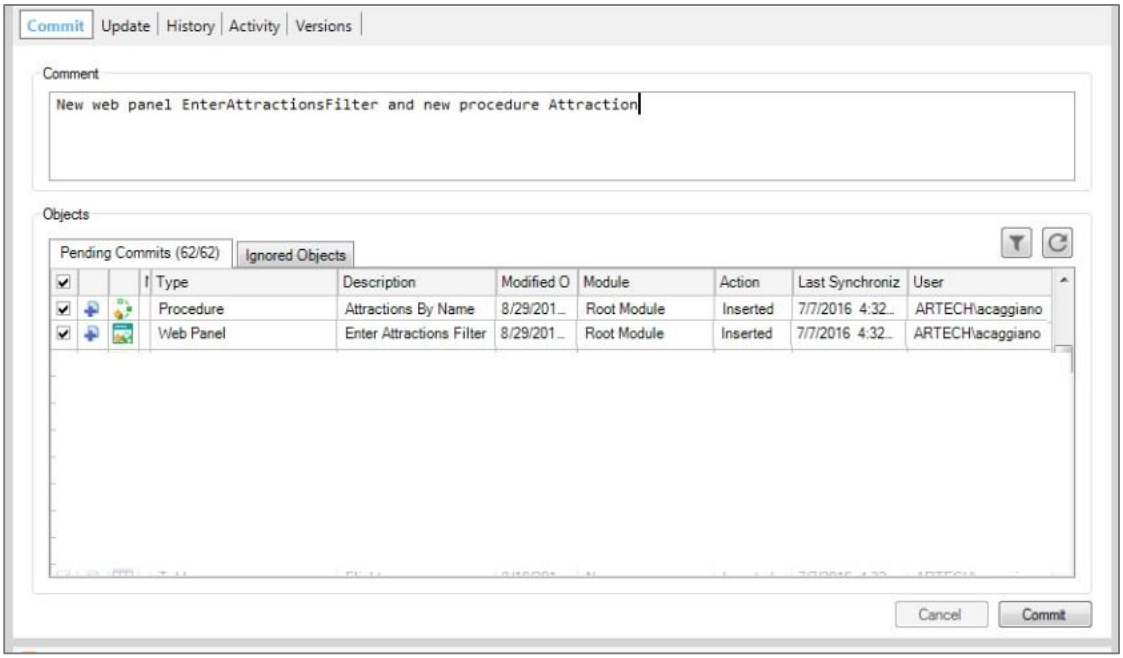
Presionamos el botón “List Attractions By Name”… y vemos que se listan todas las atracciones.



Ahora acotamos un poco más el rango, ponemos entre la ‘A’ y la ‘F’:

Y vemos cómo funcionó el filtro:

Para finalizar enviemos todo lo que hicimos a GeneXus Server…



En el siguiente video veremos otras formas de enviar y recibir parámetros, incluyendo los efectos de colocar un atributo en la regla Parm, en lugar de una variable.