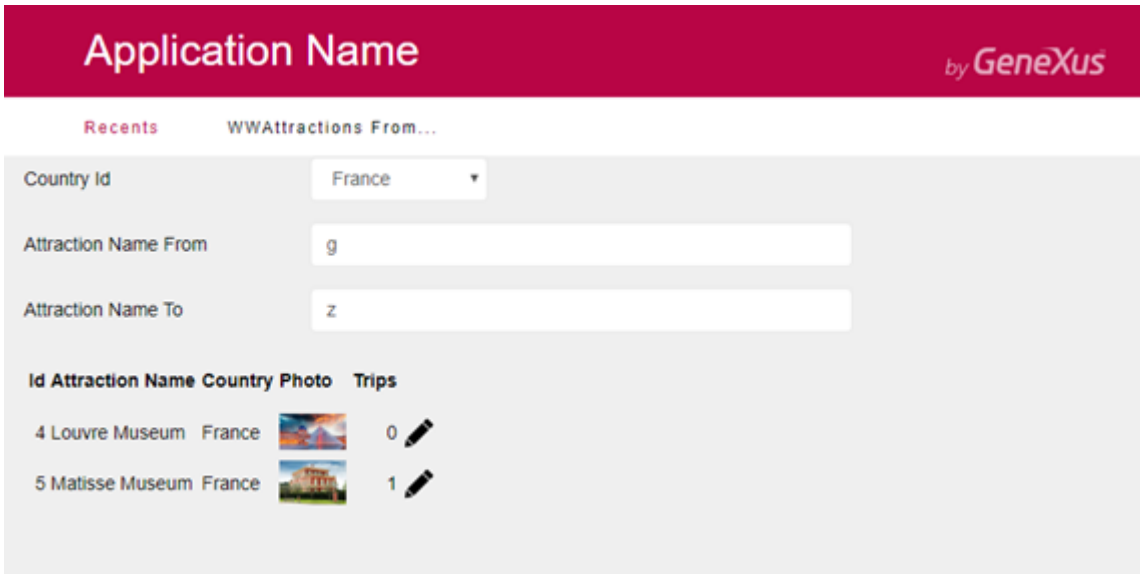
Pantallas interactivas: cómo guardar información de contexto

En este ejemplo veremos una forma de mantener datos en memoria, para evitar que los mismos se pierdan luego de llamar a otro objeto, y posteriormente volver a este.

Para ejemplificar esta situación, seguiremos con el ejemplo que venimos trabajando hasta el momento.

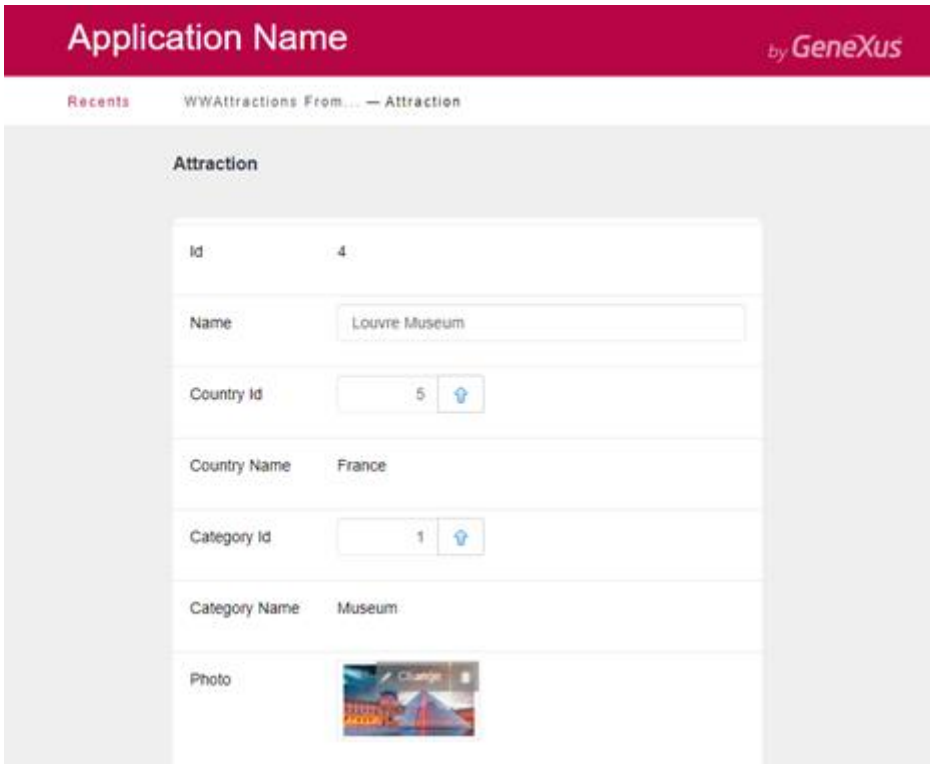
Hagamos un pequeño resumen.

En el video anterior, al ingresar valores en los filtros creados para este fin, nuestra grilla se refresca y mostrará solamente los datos que realmente nos interesan condicionada por dichos valores.

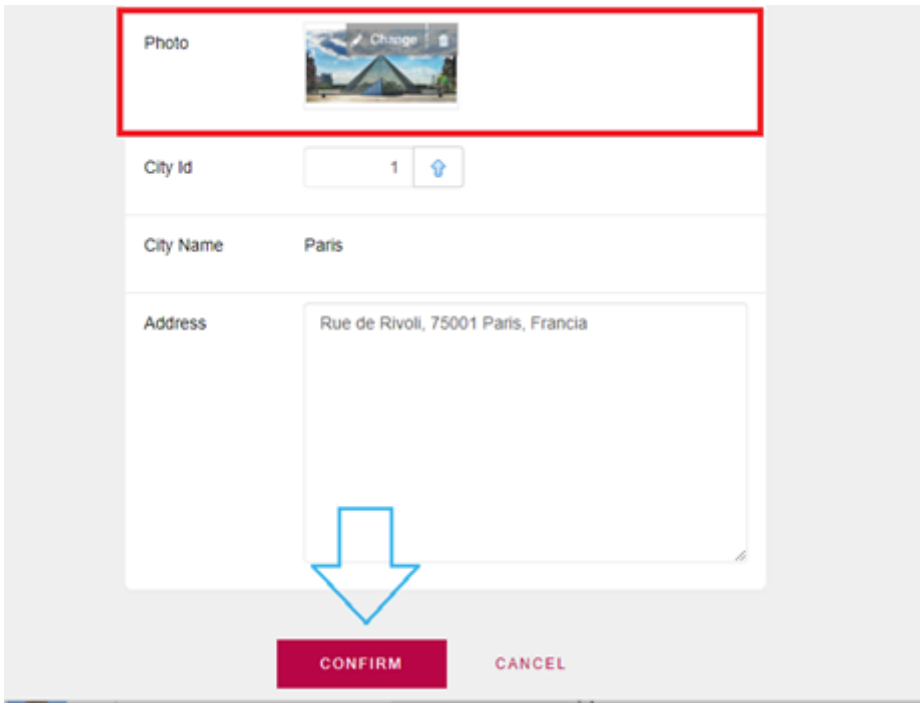


Si seleccionamos la acción de actualizar en una de las filas, como vimos, nos llevará a la transacción Attraction en modo Update. Donde nos permitirá editar los valores correspondientes a la atracción seleccionada.

Elegiremos una de las atracciones para modificar, lo haremos por ejemplo con el museo del Louvre



Cambiaremos la imagen correspondiente al atributo AttractionPhoto y seleccionamos confirmar.



Vemos que esta acción nos retorna a nuestro Web Panel.

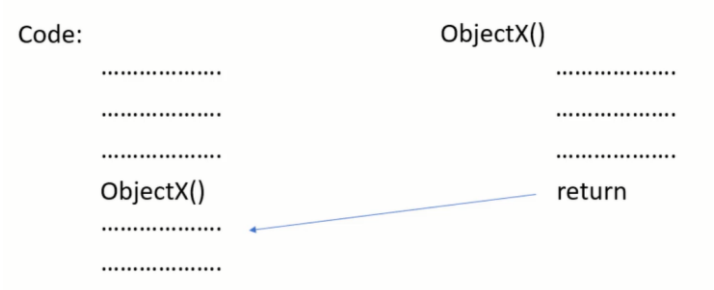
Esto es porque el pattern work with aplicado a la transacción Attraction, automáticamente agregó el comando Return, para de esta manera volver al objeto llamador. Esto lo podemos apreciar en la sección Eventos de la transacción Attraction, programado dentro del evento “After Trn”.

El evento “After Trn” se dispara cuando la transacción ha finalizado un ciclo, inmediatamente después del commit, es decir, luego de haberse grabado un cabezal con sus correspondientes líneas.

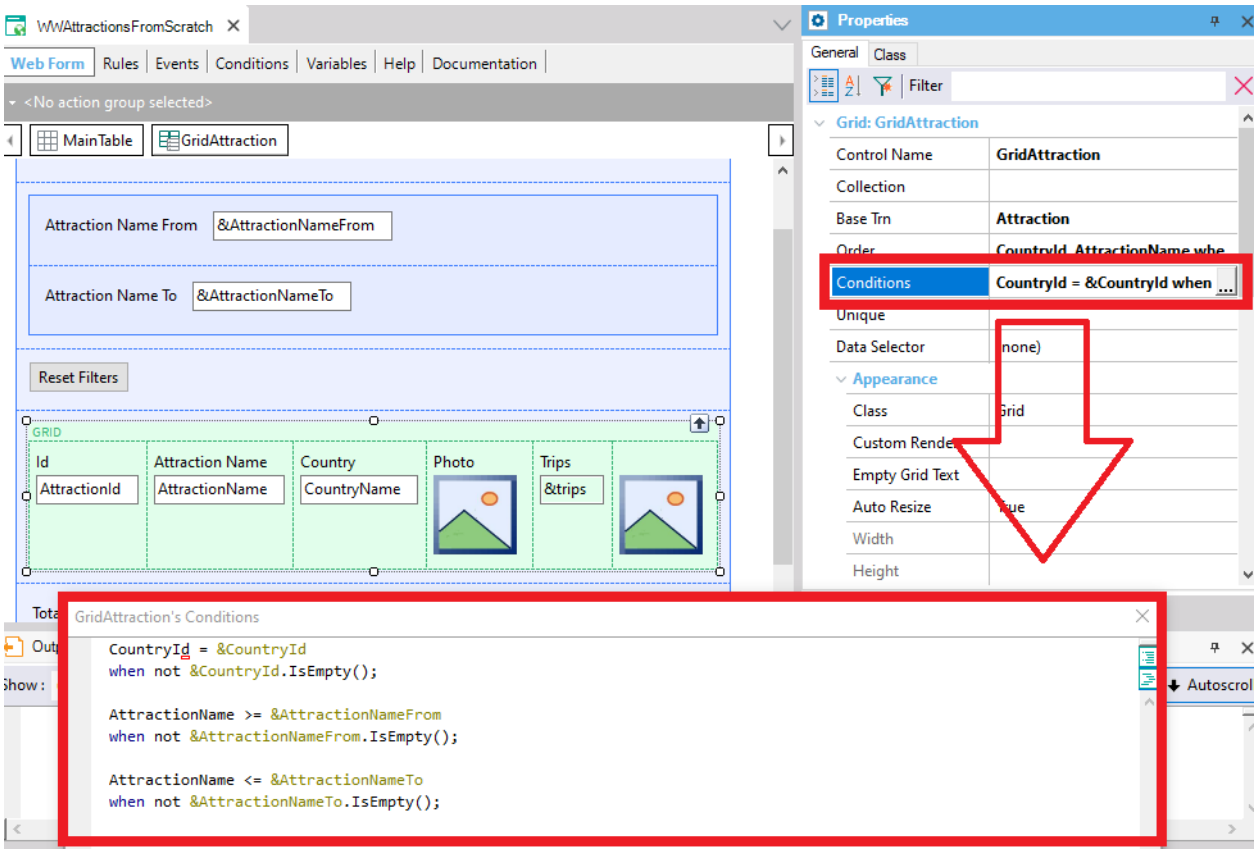
La función del comando return es finalizar la ejecución que se esté realizando en este objeto, y volver, retornar, al objeto llamador.

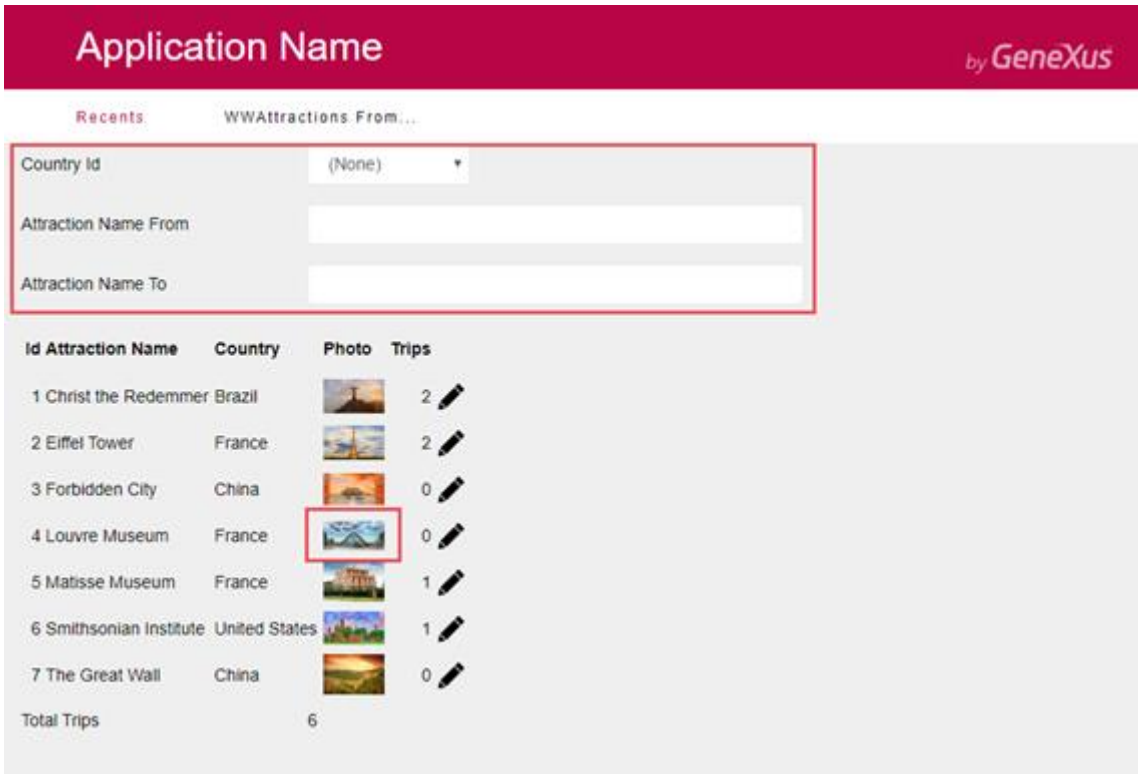
En nuestro ejemplo tanto el Web Panel WWAttractionsFromScratch como la transacción Attraction, o sea tanto el objeto llamador, como el llamado, son objetos con interfaz gráfica. Por lo que en este caso, el comando Return es equivalente a si hiciéramos una invocación al Web Panel por primera vez.

Distinto sería el caso si alguno de estos dos objetos, no tuviera interfaz gráfica, como es por ejemplo el caso de un procedimiento. Ahí el comando return del objeto que fue llamado, nos retornará a la línea siguiente inmediata a la invocación.



Volviendo a nuestro caso, al retornar se ejecutarán los eventos asociados a la carga del Web Panel. Primero el evento Start, seguido por el Refresh y después el Load, este último tantas veces como registros haya en el grid que cumplan con las condiciones declaradas en la propiedad conditions. En este caso como las condiciones no aplican si las variables están vacías, vemos en ejecución que se vuelven a cargar todas las atracciones.



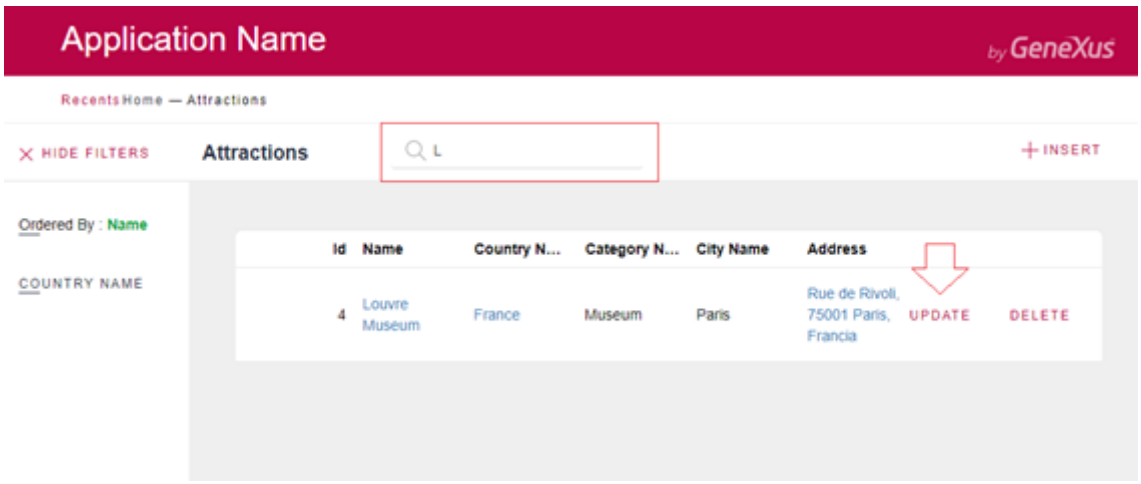


¿Por qué motivo pasa esto? Lo explicaremos en un momento.

Antes vayamos a ver el comportamiento del Web Panel creado automáticamente desde la sección patterns de la transacción Attraction.

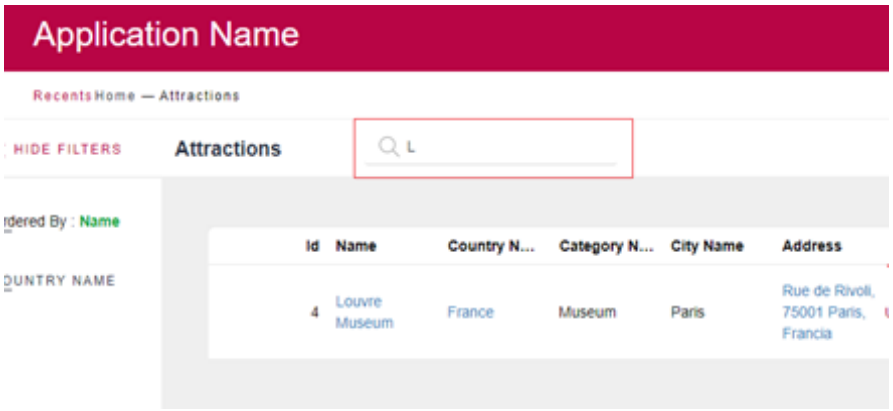
Filtremos por nombre de atracción ingresando la letra “L”, nos mostrará todas las atracciones que comiencen con esta letra, en este caso la única que hay ingresada con esta característica es la del Museo del Louvre.

Seleccionemos la acción para actualizar esta atracción.



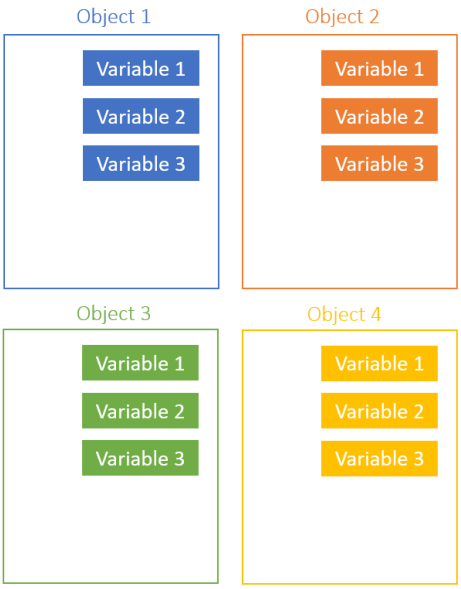
Observamos que nos llevará directamente a la transacción Attraction y nos mostrará la atracción seleccionada permitiendo su edición, exactamente igual que en el caso de nuestro web panel implementado a mano.

Cambiaremos la fotografía, confirmamos la actualización y nos retornará al Web Panel llamador.
Observamos que los filtros se mantienen. Esta es justamente la funcionalidad que queremos lograr.



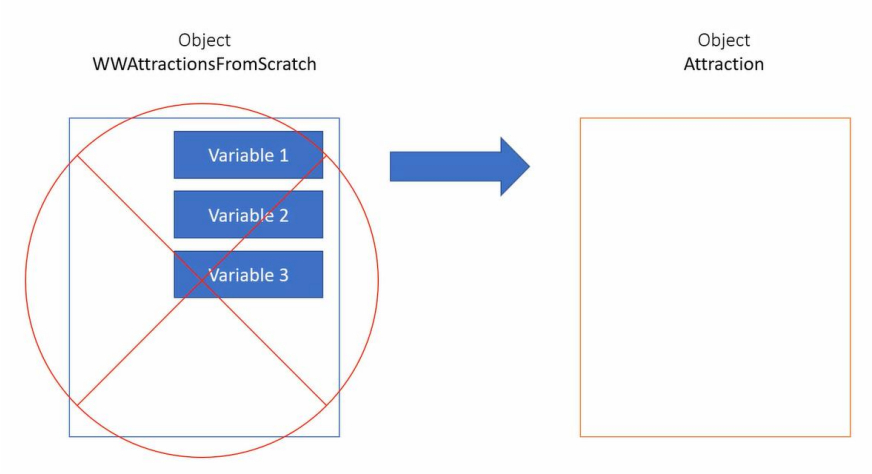
Programaremos en nuestro Web Panel una solución a nuestro modo, y luego repasaremos cómo lo hace el Pattern.

Recordemos que las variables declaradas en cada objeto, solamente podrán ser utilizadas dentro del mismo y mientras este esté activo.



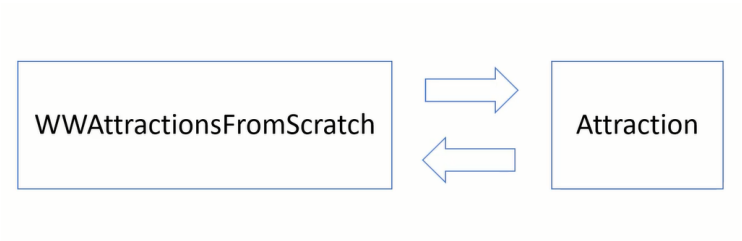
Por ejemplo, en nuestro caso, cuando invocamos a la transacción Attraction desde el Web Panel, en ese momento, nuestro objeto WWAttractionFromScratch se destruye, y consigo sus variables. Por lo que se perderán los valores que tuvieran guardados. Por contraparte el que pasará a tener un estado activo, es el objeto Attraction.

Luego, al volver de la transacción al Web Panel con el comando return, este último objeto y sus variables volverán a crearse.
Es por esto que ya no vemos los valores de nuestros filtros, porque en realidad éste es un **nuevo objeto.** Y las variables que utilizamos como filtros también fueron creadas nuevamente, las que teníamos antes de invocar al objeto Attraction fueron destruidas.



Esto contesta la pregunta que nos hacíamos hace un momento, de cuál era el motivo por el que los valores ingresados en nuestros filtros no se mantenían luego de actualizar un registro.

Lo que necesitamos es guardar la información de cada uno de nuestros filtros en una suerte de variables globales, de modo tal que no se pierda entre ejecuciones, o sea en el pasaje de un objeto a otro. En este caso entre el Web Panel y la transacción, y de la transacción nuevamente a nuestro Web Panel.



¿Cómo hacemos entonces para poder mantener en memoria el valor de una variable?

Tenemos desde GeneXus una manera de programar esta funcionalidad. Esto es mediante variables del tipo Web Session.

Estas variables nos permiten manejar una suerte de conjunto de variables globales, en las cuales podremos almacenar datos y acceder a ellos desde cualquier objeto, mientras la sesión esté activa.
Esto es justamente lo que estábamos buscando.

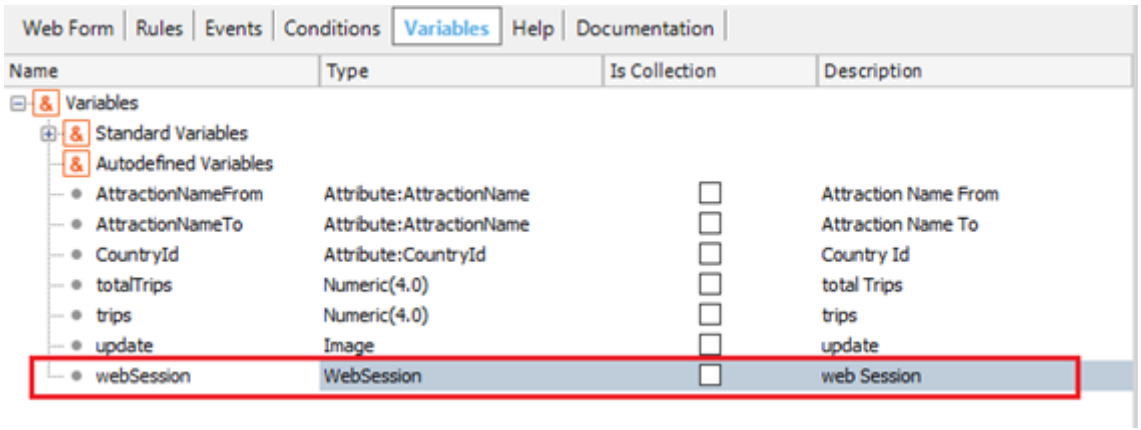
Una gran ventaja de estas variables, es que nos permiten guardar un conjunto de datos del tipo clave-valor. Por lo que solo necesitaremos declarar una variable del tipo Web Session, y en ella guardar todos los filtros que tengamos, cada uno de ellos con una clave única.

Key	Value
‘Key1’	‘Value1’
‘Key2’	‘Value2’
‘Key3’	‘Value3’
‘Key4’	‘Value4’
….	….
….	….

Así, tendemos una clave y un valor para el filtro CountryId, otra clave y valor para AttractionNameFrom y por último para AttractionNameTo, los tres filtros que necesitamos conservar entre ejecuciones de nuestro panel

Esto cumple con nuestra necesidad, el poder guardar temporalmente la información ingresada en los filtros, para recuperarla luego. Programemos esto.

Crearemos primero que nada una variable a la cual en este caso le pondremos de nombre “webSession”.



Al ponerle este nombre, ya GeneXus le asigna el tipo de datos WebSession, entendiendo que seguramente sea lo que nos interesa, lo cual en este caso efectivamente es así. En caso que esto no se cumpla, siempre podemos cambiar el tipo de datos que asigna automáticamente por el que queramos,

Luego, deberemos evaluar en qué momento queremos que esta variable guarde el o los valores deseados.

Recordemos los eventos principales que tenemos al momento programados:

- El evento Start, que se ejecutará una única vez cuando carga la página por primera vez.

```
Event Start
    &update.FromImage(updateIcon)
Endevent
```

- El evento Refresh, que se disparará luego del evento Start y cada vez que se cambie algún filtro que esté dentro de las conditions de la grilla o se refresque la página desde el navegador.

```
Event Refresh
    &totalTrips = 0
Endevent
```

- Y el evento Load, que se ejecutará luego del evento Refresh, tantas veces como datos sean cargados en nuestra grilla.
Este evento se ejecutará N veces por tener tabla base asociada.

```
Event Load
    &trips = Count(TripDate)
    &totalTrips = &totalTrips + &trips
Endevent
```

¿Dentro de estas tres opciones, cuál consideras la mejor para guardar estos datos?

De estas opciones claramente la más conveniente sería dentro del evento Refresh, ya que cada vez que cambiemos alguno de los valores de los filtros, necesariamente se disparará este evento, y es allí cuando éstos se guardarían en nuestra variable de sesión, de modo que podremos recuperarlos al volver de la transacción,

Pero tenemos también otro evento, que es el evento Click de la variable update

```
Event &update.Click
    Attraction(trnMode.Update, AttractionId)
Endevent
```

Este evento se disparará inmediatamente después de hacer click en la acción actualizar. Lo cual sería también una buena opción para aquí guardar los valores de los filtros.

Consideramos que es una buena opción, ya que en este evento es donde invocaremos a la transacción Attraction, y como vimos es ese el momento exacto donde se destruye al objeto llamador y consigo sus variables. Por lo que necesitamos justamente antes de esto guardar los valores de las variables de filtro.

Si lo hacemos en el evento Refresh, guardaremos los valores de los filtros cada vez que se cambie un dato en alguno de ellos. Ya que por estar en las conditions del grid, cada cambio dispara este evento. Pero esto no es necesario, ya que puede que en la ejecución actual no necesitemos en absoluto actualizar las atracciones filtradas. Y en

ese caso, ¿para qué guardaríamos esos filtros en la sesión, si las variables no van a ser destruídas?

Sin embargo, si lo hacemos en el evento relacionado a la acción de actualizar, guardaremos estos valores **solo una vez,** cuando es imprescindible hacerlo, inmediatamente antes de que se destruya el objeto y sus variables.

Bien, vamos a hacerlo en el evento click de la variable.

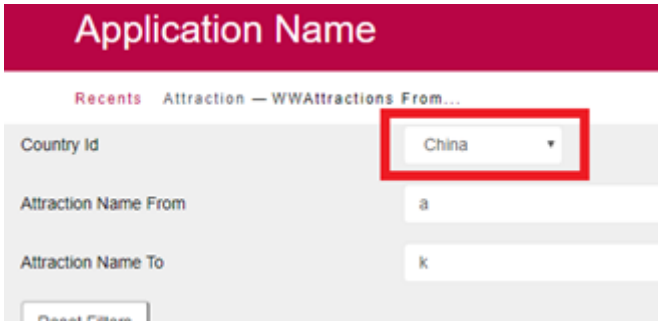
```
Event &update.Click
    Attraction(trnMode.Update, AttractionId)
Endevent
```

Escribimos la variable webSession, le aplicaremos el método “set” para guardar valores en ella, y como primer parámetro deberemos ingresar una clave (Key), la cual deberá ser un valor del tipo character, por lo que debemos ingresarla entre comillas.

En este caso le asignaremos el nombre CountryId. Esta clave única es la que nos servirá luego para recuperar el valor que guardemos.

Luego debemos asignarle un valor (value), que será el dato que queremos almacenar en memoria. En este caso nos interesa guardar el valor de la variable CountryId, que será el valor del primer filtro que tenemos en pantalla.

```
Event &update.Click
    &webSession.Set('CountryId', &CountryId.ToString())
    Attraction(trnMode.Update, AttractionId)
Endevent
```



Tener en cuenta que el valor que ingresemos también deberá ser del tipo Character, por lo que al ser la variable CountryId del tipo numérico, debemos convertirla al tipo Character. Esto lo logramos aplicándole el método ToString a la variable.

```
Event &update.Click
    &webSession.Set('CountryId', &CountryId.ToString())
    Attraction(trnMode.Update, AttractionId)
Endevent
```

Luego, hacemos lo mismo para los otros dos filtros, AttractionNameFrom y AttractionNameTo.

```
Event &update.Click
    &webSession.Set('CountryId', &CountryId.ToString())
    &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
    &webSession.Set('AttractionNameTo', &AttractionNameTo)
    Attraction(trnMode.Update, AttractionId)
Endevent
```

Volvamos a ejecutar la aplicación.

Ingresaremos valores en nuestros filtros.
Recordemos que cada vez que cambiamos el valor de alguno de los filtros se dispara el evento Refresh, y luego el Load por cada registro que se carga en la grilla.

Luego seleccionaremos en la acción actualizar de una atracción.
En ese momento será disparado el evento &Update.Click, en el cual programamos guardar los valores de nuestros filtros en la variable &webSession. Para posteriormente realizar la invocación de la transacción Attraction. Que como vimos, será el momento donde se destruye nuestro objeto Web Panel y sus variables.

Al cambiar algún dato de esta atracción y confirmar. Aplicará el comando return, y como en este caso es el equivalente a llamar al Web Panel por primera vez, volverá a ejecutar los eventos Start, Refresh y Load por cada valor a ser cargado en el grid.

Bien, ahora lo que necesitaremos es poder recuperar estos valores que guardamos en la variable webSession.

Cual consideras sería el mejor momento para poder recuperar estos valores?
Repasemos nuevamente los eventos que tenemos y evaluemos:

¿En el evento Start?
¿En el evento Refresh?
¿En el evento Load?
¿El evento &Update.Click?

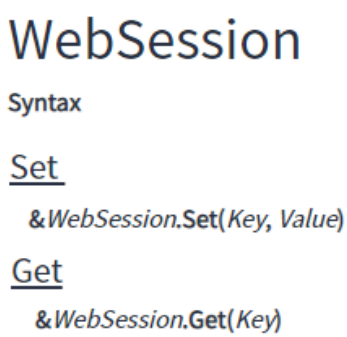
Claramente el único que nos servirá aquí será el evento Start. Ya que como vimos, este evento es ejecutado una sola vez cuando carga el Web Panel por primera vez, y como

cuando retornamos de la transacción, es el equivalente a llamar por primera vez nuestro Web Panel, es allí justamente cuando necesitaremos recuperar esos valores,

```
Event Start
    &update.FromImage(updateIcon)
Endevent
```

Como vimos, para poder asignarle una clave y valor a la variable webSession, lo hacemos con el método set. Ahora necesitamos saber cómo recuperar el valor allí

guardado.
Esto lo hacemos mediante el método Get, al cual le deberemos indicar la clave (key) del valor que deseamos recuperar.



Entonces, a cada variable que utilizamos como filtro, le aplicaremos el método Get, pasándole por parámetro la clave correspondiente a cada uno.

Lo haremos primero para la variable CountryId paso a paso,

Ingresamos la variable CountryId, y le asignamos el valor del método Get de la variable webSession, pasándole por parámetro la clave, o sea 'CountryId'.

```
Event Start
    &update.FromImage(updateIcon)
    &CountryId = &webSession.Get('CountryId').ToNumeric()
Endevent
```

Recordemos lo que comentamos anteriormente, las variables del tipo Web Session guardan solamente datos del tipo Character. Al ser la variable CountryId del tipo numérico, para asignarle el valor de la variable WebSession, que será del tipo Character, deberemos convertir esa información en numérico, esto lo conseguimos con el método ToNumeric().

```
Event Start
    &update.FromImage(updateIcon)
    &CountryId  = &webSession.Get('CountryId').ToNumeric()
Endevent
```

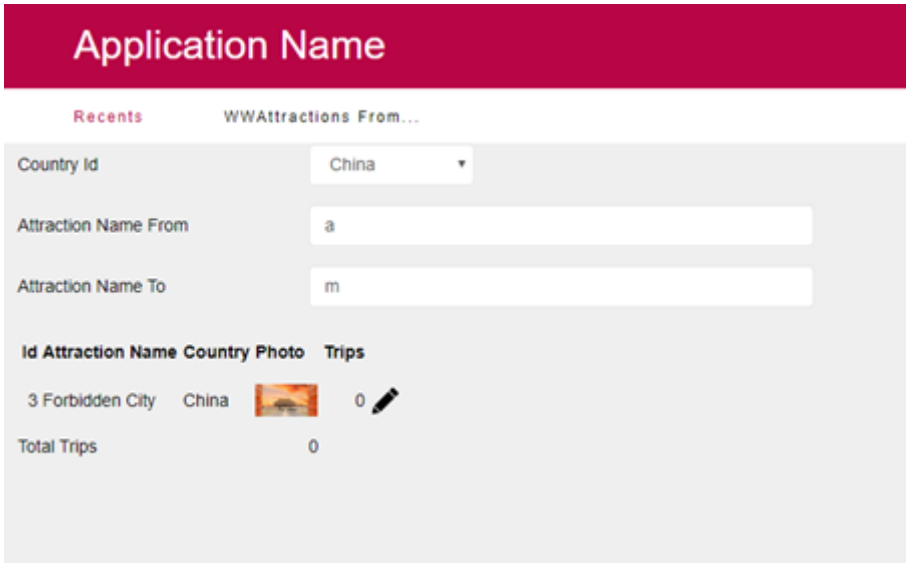
Luego hacemos el mismo procedimiento para las variables AttractionNameFrom y AttractionNameTo.

```
Event Start
    &update.FromImage(updateIcon)

    &CountryId          = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo   = &webSession.Get('AttractionNameTo')
Endevent
```

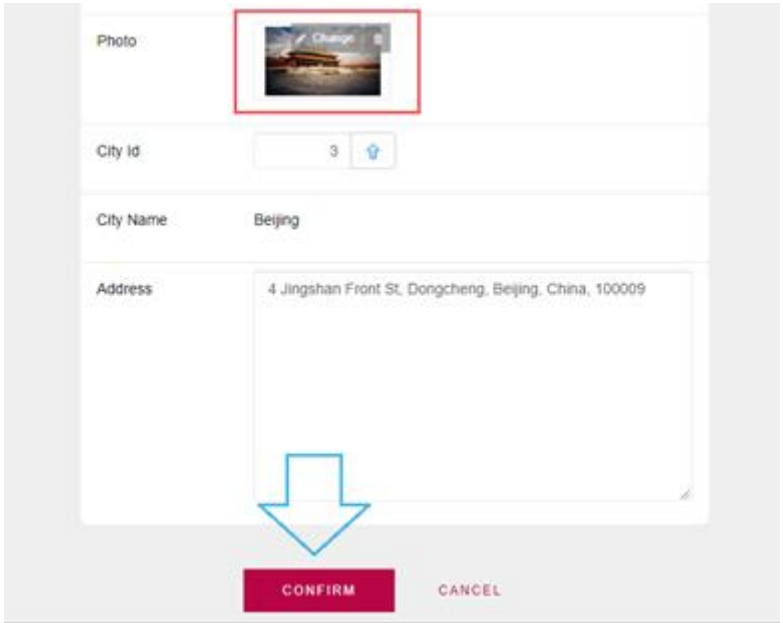
Volvamos a ejecutar la aplicación y probemos ahora su comportamiento.

Filtraremos todas las atracciones de China comprendidas entre la A y la M.



En este caso solo nos aparecerá una atracción.
Seleccionaremos la acción de actualizar sobre la misma, y recordemos que en ese momento, previo a invocar a la transacción, guardaremos los datos de nuestras variables de filtros en memoria.

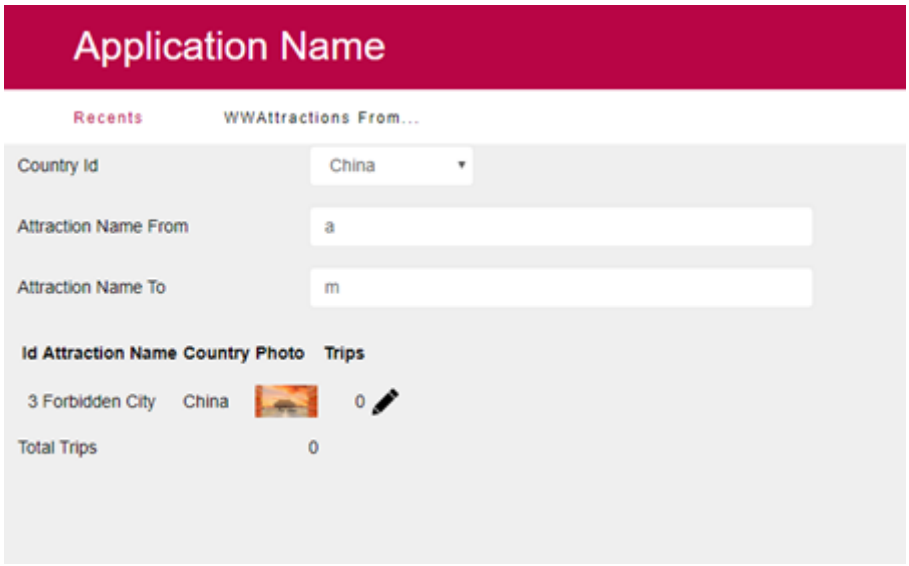
Cambiaremos la foto, y confirmamos la acción.



Al retornar al Web Panel, como vimos, el primer evento que se ejecutará será el evento Start. Donde programamos recuperar la información que guardamos en nuestra variable de sesión, asignandole esos valores a nuestras variables de filtro.

```
Event Start
    &update.FromImage(updateIcon)
    &CountryId          = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo   = &webSession.Get('AttractionNameTo')
Endevent
```

Observamos que ahora sí, se mantiene los valores ingresados en los filtros como era deseado.



Y además, a continuación del Start se habrá disparado el Refresh, que a su vez, habrá disparado la carga del grid de acuerdo a los filtros. Por eso vemos la grilla nuevamente filtrada, con la foto modificada.

Pero una vez que programamos esto, ¿qué va a pasar cuando el usuario abra en su navegador por primera vez este web panel? Porque en ese caso no tendremos nada guardado, y por ende nada que recuperar.

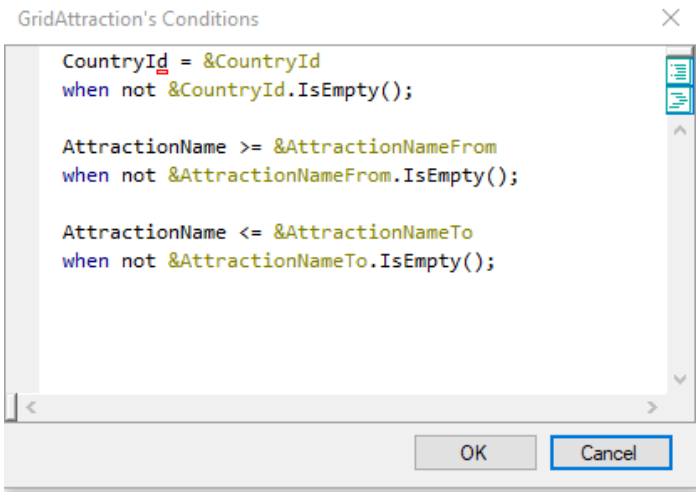
Cerremos toda sesión que tengamos activa en nuestro navegador, y ejecutemos desde Genexus nuevamente la aplicación haciendo el ciclo completo.

Ingresamos al Web Panel creado por nosotros.
Y al ser en esta sesión la primera vez que se ejecuta el Web Panel, como sabemos lo primero que se dispara es el evento Start.

```
Event Start
    &update.FromImage(updateIcon)
    &CountryId = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo = &webSession.Get('AttractionNameTo')
Endevent
```

La variable de sesión buscará por las claves ingresadas si hay información para recuperar, no la habrá, ya que aún no se guardó ningún valor en la variable

&webSession. Por lo que en este momento las variables utilizadas para nuestros filtros seguirán vacías.
Y como en las conditions del grid declaramos que no aplique ningún filtro si éstas variables están vacías, es que se muestran todas las atracciones.



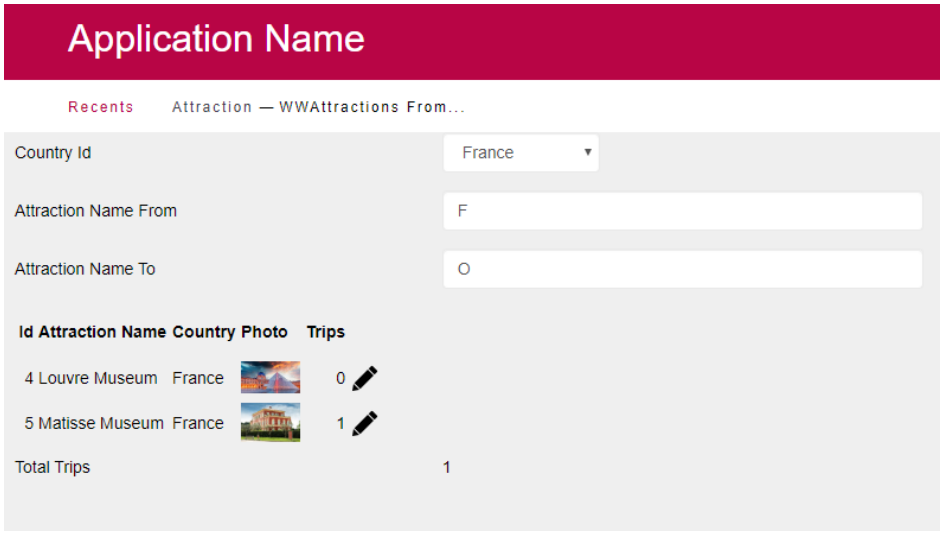
Luego del Start se ejecutará el evento Refresh, e inmediatamente después el Load. Como nuestro grid tiene tabla base, el evento Load se ejecutará N veces, tantas como registros sean cargados en nuestra grilla.

```
Event Refresh
    &totalTrips = 0
Endevent
```

```
Event Load
    &trips = Count(TripDate)
    &totalTrips = &totalTrips + &trips
Endevent
```

Seleccionaremos por ejemplo para filtrar por el país Francia, y vemos que ya en ese momento nuestra grilla aplica el filtro, mostrando solamente las atracciones que tengan a Francia como país. Lo que hizo nuestra aplicación una vez que elegimos el filtro de país, fue disparar inmediatamente el Evento Refresh, y como consecuencia se dispara luego el evento Load, en este caso tres veces, porque se encontraron tres registros para mostrar con esta condición.

Como filtro Attraction Name From, ingresaremos la letra F, y en ese momento se vuelve a disparar el evento Refresh, seguido por el Load. En este caso el Load se va a ejecutar dos veces, ya que son dos registros los que cumplen con estas condiciones y por ende los que se cargarán en la grilla.
Por último en el filtro Attraction Name To ingresaremos la letra O. Se volverá a ejecutar el evento Refresh, y luego el load en dos oportunidades.

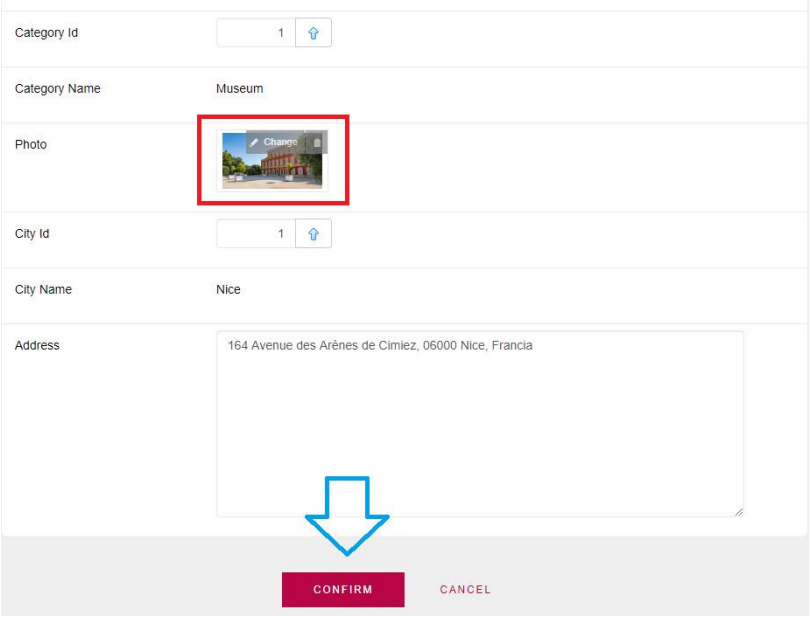


Seleccionamos la acción de actualizar sobre la atracción museo Matisse.
En ese momento se dispara el evento &Update.Click, en el cual programamos que previo a invocar a la transacción Attraction, guarde los datos de las variables de filtro en nuestra variable de sesión &webSession.
Para esto utilizamos el método Set de la variable de sesión, pasándole por parámetro la clave y el valor que deseamos guardar. En nuestro caso son tres valores los que nos interesa mantener en memoria. Que son las variables &CountryId, &AttractionNameFrom y &AttractionNameTo.

```
Event &update.Click
    &webSession.Set('CountryId', &CountryId.ToString())
    &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
    &webSession.Set('AttractionNameTo', &AttractionNameTo)
    Attraction(trnMode.Update, AttractionId)
Endevent
```

En este momento se invoca al objeto Attraction, y éste pasa a quedar activo. Y nuestro objeto Web Panel es destruido junto a sus variables.

Modificamos algún dato de la atracción, por ejemplo modificaremos su fotografía, y confirmamos la acción.



En ese momento nos llevará nuevamente hacia el objeto llamador, o sea nuestro Web Panel. Esto lo hace porque como vimos el pattern work with aplicado a la transacción Attraction, agregó el comando Return dentro del evento After Trn.

En este momento se vuelven a ejecutar los Eventos Start, Refresh y Load por cada registro a ser cargado en el grid. Esto como lo explicamos, es porque el comando return en este caso es equivalente a si hiciéramos una invocación de nuestro Web Panel por primera vez.

En el evento start, se cargarán los valores guardados en la variable de sesión. Esto mediante el método Get, y pasándole por parámetro la clave con la que guardamos cada valor en el evento &Update.Click utilizando el método Set.

Ese dato se lo asignaremos a cada variable correspondiente, en nuestro caso a las tres variables que utilizamos para los filtros. &CountryId, &AttractionNameFrom y &AttractionNameTo..

```
Event Start
    &update.FromImage(updateIcon)
    &CountryId          = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo   = &webSession.Get('AttractionNameTo')
Endevent
```

Luego de esto se ejecutará el evento Refresh, y posteriormente el evento Load por cada registro a ser cargado en el grid.
En este caso las atracciones a ser cargadas en la grilla, que cumplen con estas condiciones son dos. Por lo que el Load se ejecutará dos veces.

Y lo que observamos en pantalla es que los valores que habíamos ingresado en nuestros filtros, se mantienen luego de actualizar un registro.

En el próximo video veremos como programa esta funcionalidad automáticamente el Work With de la transacción Attraction, y lo compararemos con nuestra solución.