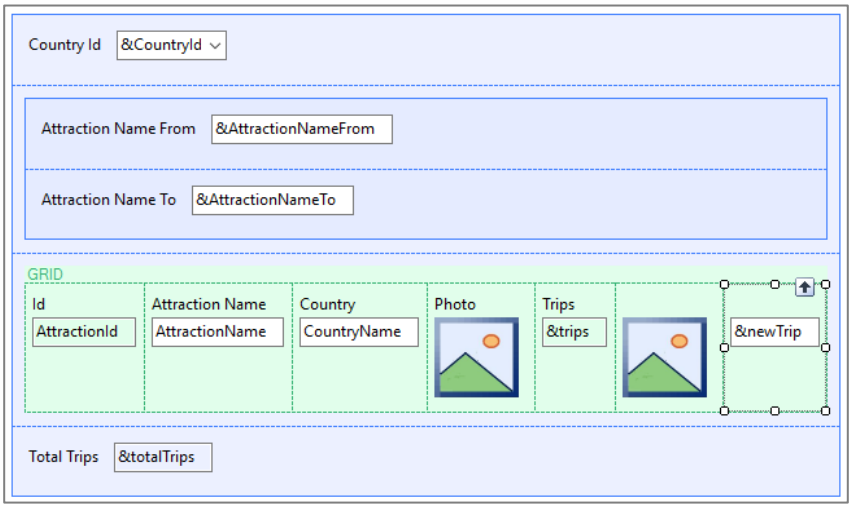
Pantallas interactivas: objeto Web Panel (continuación)

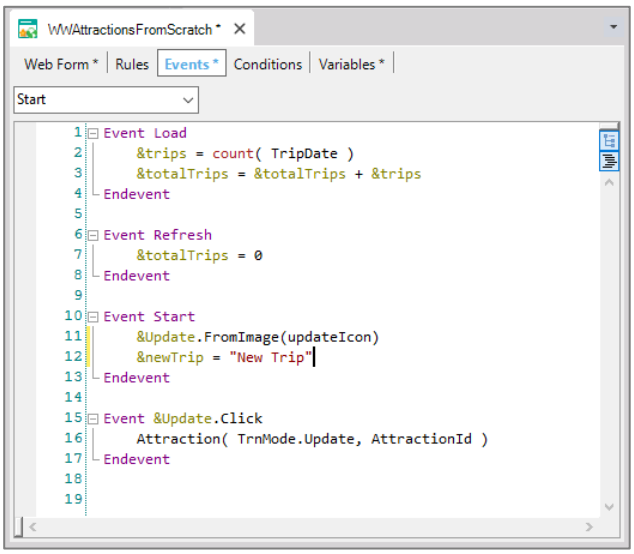
Estábamos construyendo nuestro web panel WWAttractionsFromScratch. Habíamos visto cómo condicionar los datos que se mostraban en el grid, cómo ordenarlos. El grid tenía tabla base.
Habíamos visto en ese caso cómo cargar una variable del grid para cada línea con el evento Load. Habíamos utilizado el evento Refresh, así como el Start, y también habíamos programado un evento a nivel de las líneas, para llamar a la transacción Attraction en modo update.

Ahora agreguemos otra acción a nivel de las líneas, pero una que no llame a otro objeto con interfaz, como era el caso de la invocación a la transacción Attraction.
Por ejemplo, imaginemos que queremos dar la posibilidad de que desde una línea (una atracción) se pueda crear un nuevo trip en la base de datos, con esa atracción.

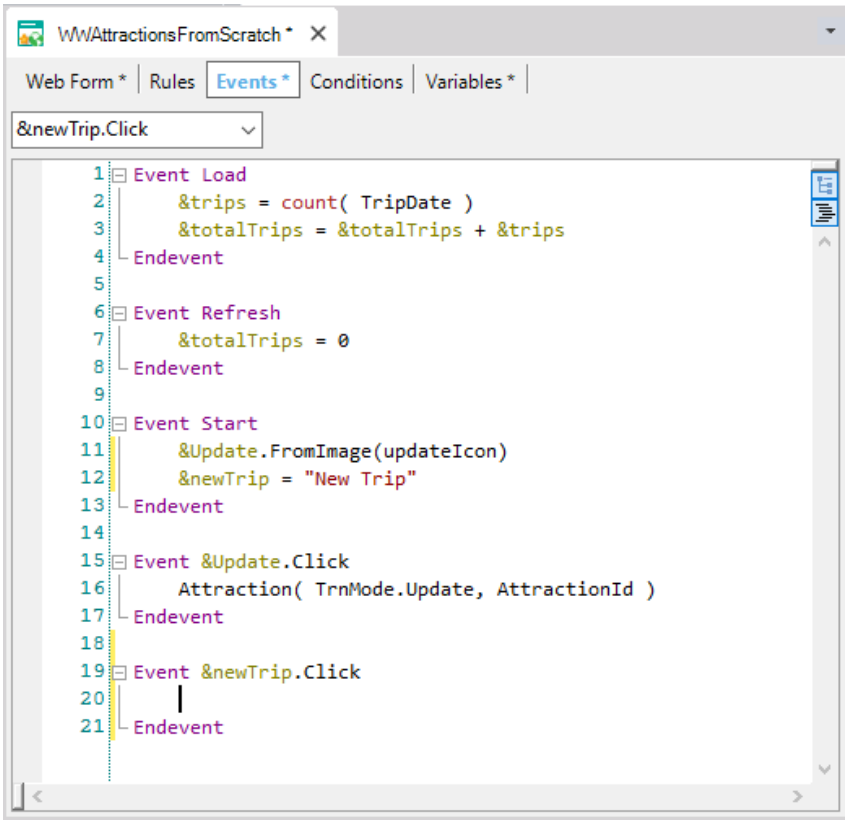
Agreguemos primeramente una nueva variable al grid, de nombre newTrip, character de 10



Cambiémosla a ReadOnly. Querremos que contenga el texto “New Trip”, así que se lo asignamos en el evento Start, puesto que no va a variar por línea:

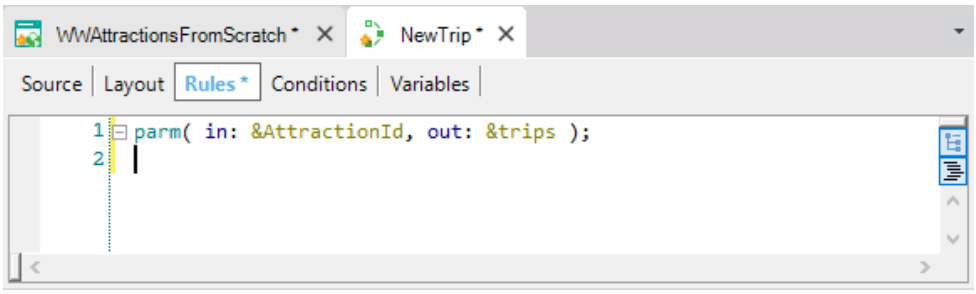


Y programemos para esa variable el evento click:

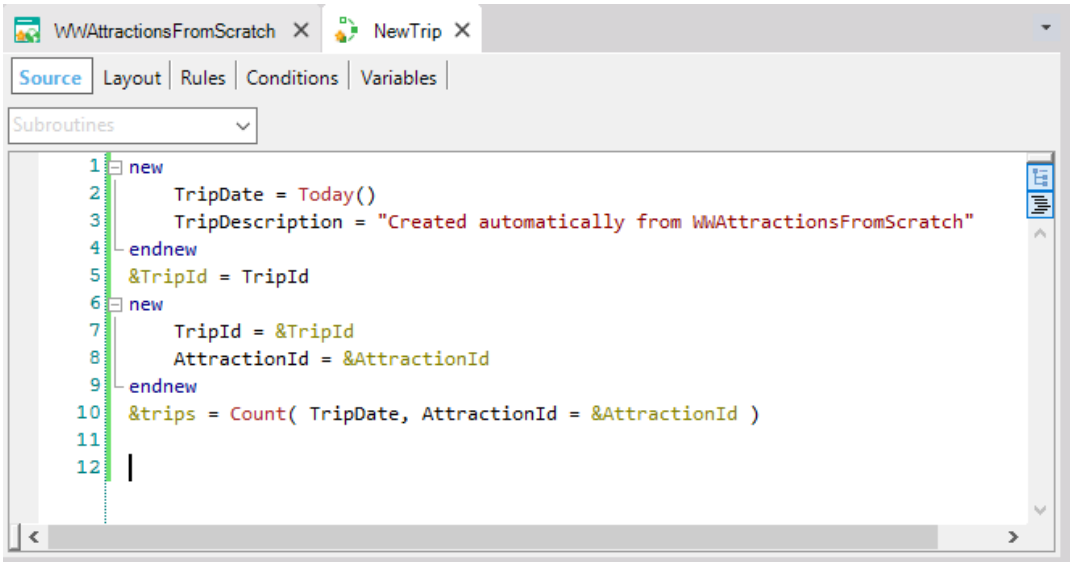


¿Qué es lo que queremos hacer cuando el usuario haga clic sobre New Trip?

Por ejemplo, llamar a un procedimiento al que le pasamos el identificador de la atracción de la línea:



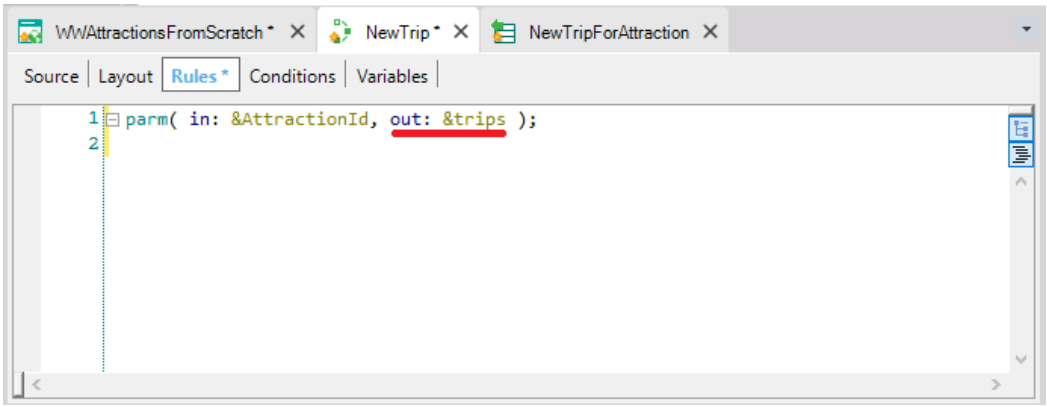
y crea el trip con esa atracción:



Observar que lo hemos implementado con el comando **new**, para crear directamente un registro en la tabla Trip y uno en la tabla TripAttraction. Usamos esta solución para mostrar el uso de estos comandos. Sin embargo la solución más aconsejable sería utilizar el business component de Trip pasa insertar. En este curso no vimos cómo cargar un business component de dos niveles, pero es muy sencillo.

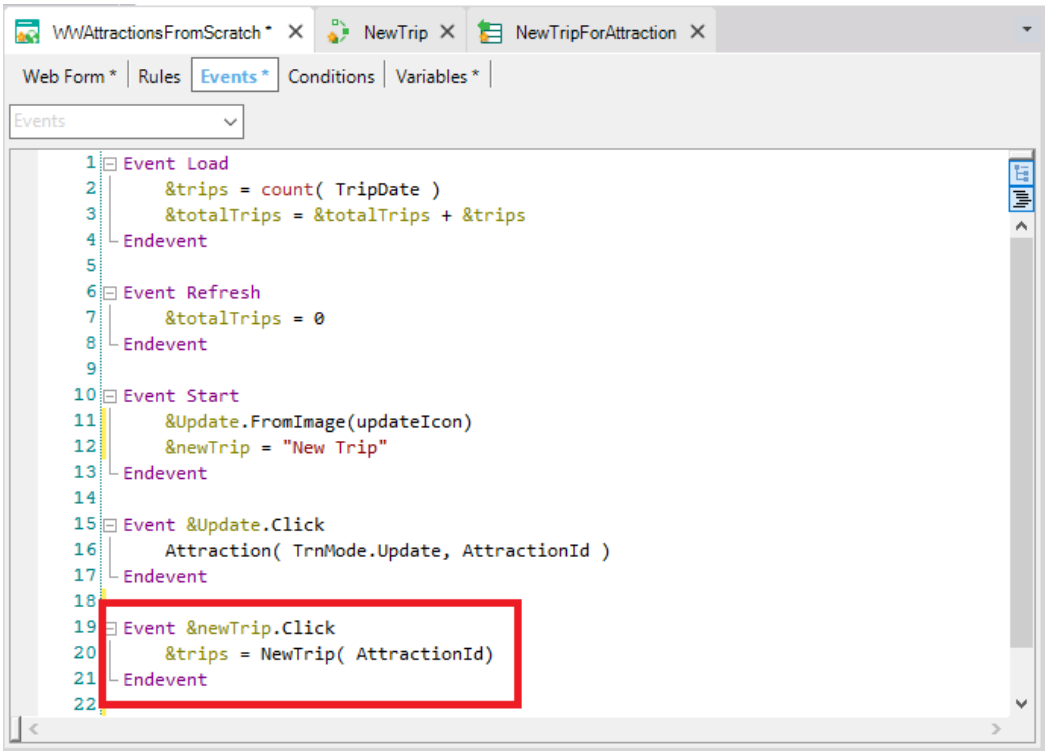
Veamos que luego de insertar cabezal y línea de Trip, calculamos la cantidad de trips en los que esa atracción se encuentra. Como la fórmula inline se está disparando sin que se esté posicionado en tabla alguna (está “suelta” en el código), tenemos que indicar la condición explícita de que queremos filtrar los trips de **la** atracción.

Y ese valor es el que se le devuelve a quien llama a este procedimiento:

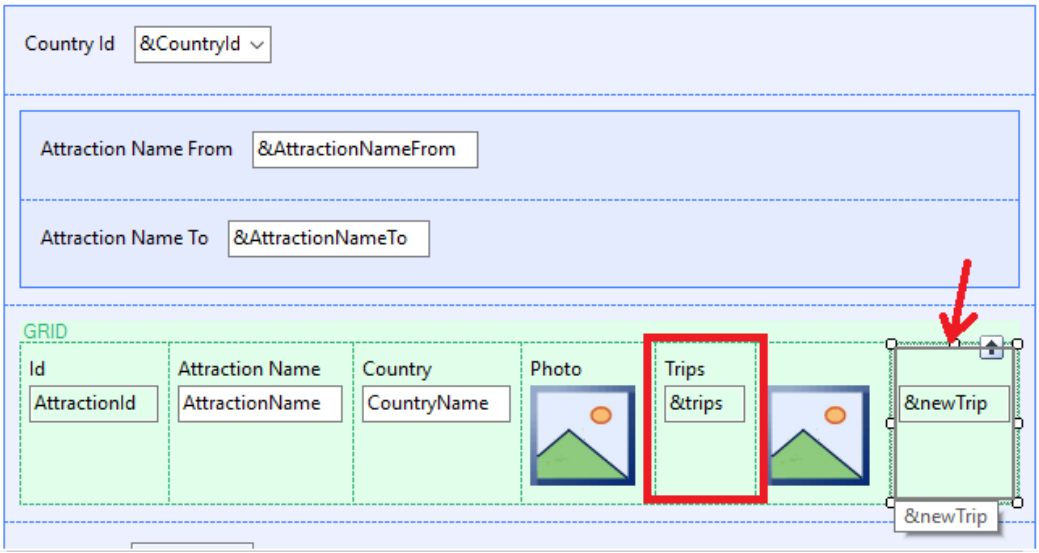


4

Entonces, desde el evento click del control (variable) &newTrip invocamos a este procedimiento, pasándole el AttractionId de la línea desde la que se hace el clic, y como el parámetro que devuelve es el que necesitamos mostrar en la columna &Trips, directamente se lo asignamos a la variable:



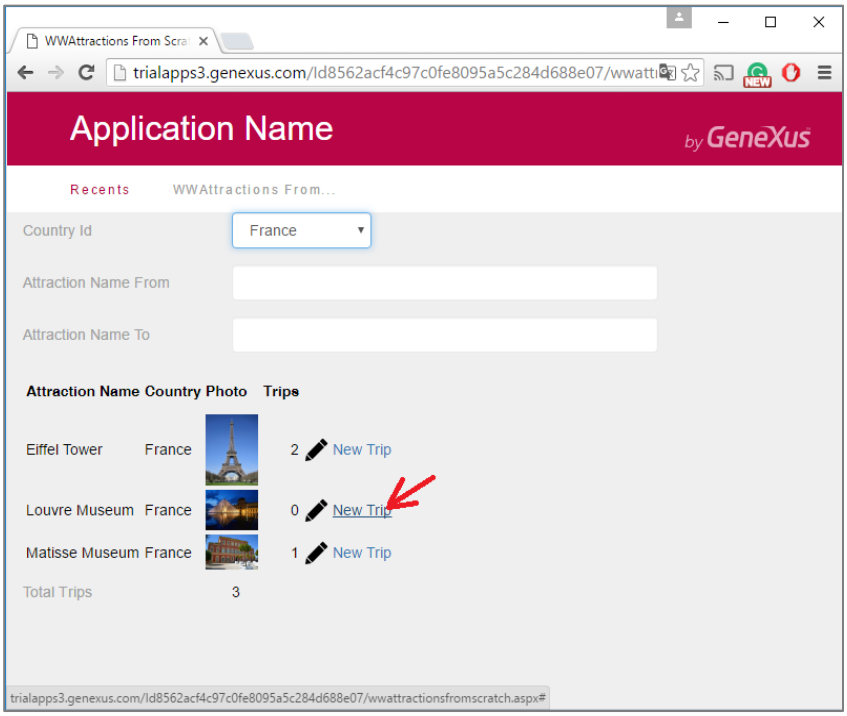
Lógicamente, cuando el usuario haga clic sobre New Trip:



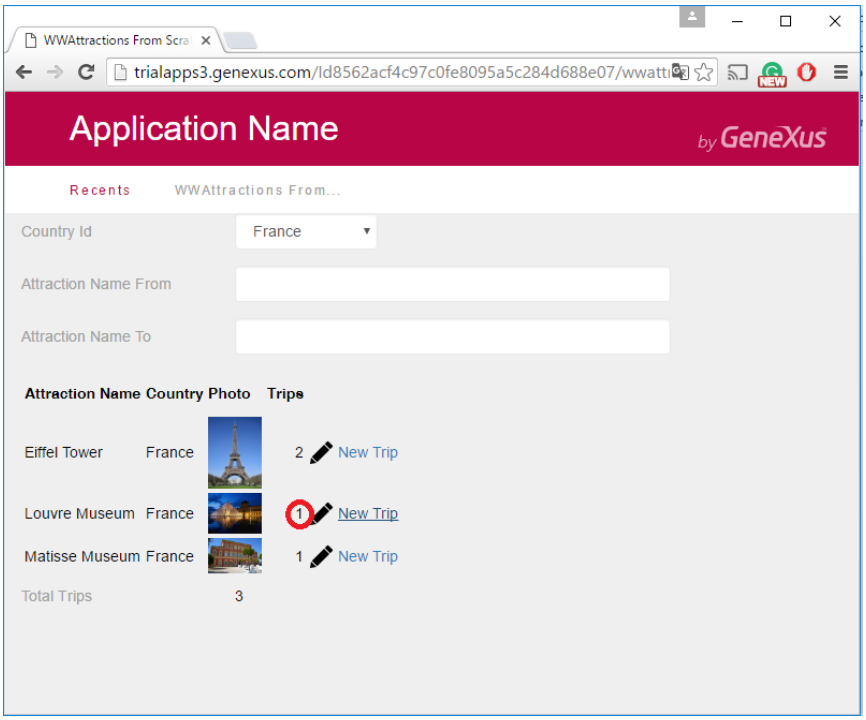
Deberá ver para esa línea, en la columna Trips, el valor que tenía antes del click, más uno. Es decir, deberá refrescarse la línea.

Ejecutemos para probar.

Por ejemplo, filtremos por Francia, y para la atracción museo Louvre, que por ahora no está en ningún trip, hagamos clic sobre New Trip:



Vemos que automáticamente se refrescó la línea:



Ahora muestra cantidad de trips del Louvre: 1.

Sin embargo lo que no se refrescó fue la cuenta del total, que debería decir 4, y sin embargo conserva el valor anterior. ¿Por qué?

Es que al ejecutar el evento click asociado a la variable &NewTrip, solamente se ejecutó su código, y como dentro del mismo se asignaba valor a una variable del grid, se refrescó la línea, **esa** línea, únicamente Sin ejecutar ningún otro evento. Ninguno, ni siquiera el Load.

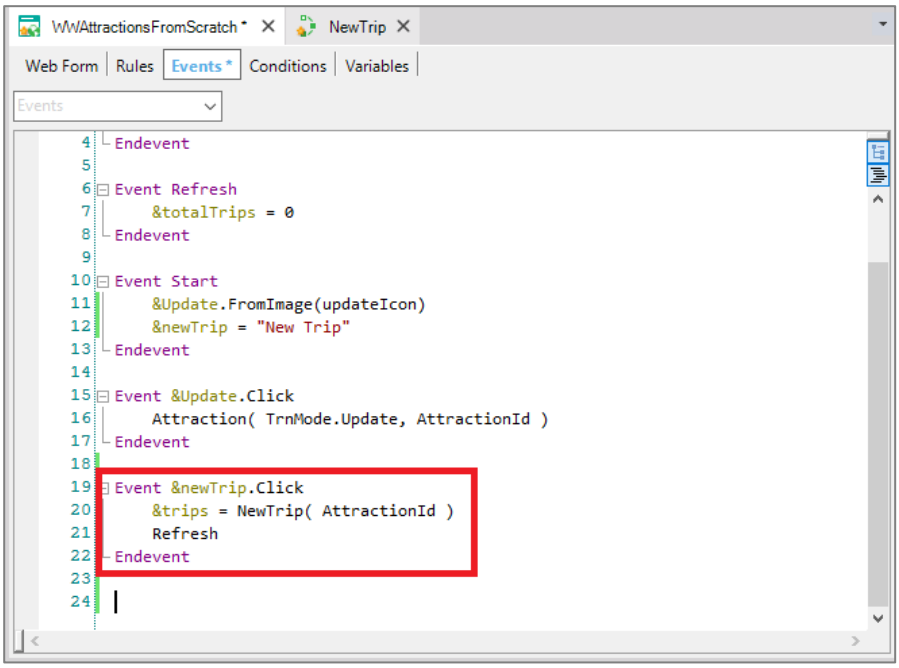
Por lo tanto, tenemos dos alternativas para mantener actualizado el Total de Trips cargados en el grid cuando se produce este evento.

Una posibilidad es a &totalTrips sumarle uno, puesto que el procedimiento solamente ha agregado un trip para esa atracción.

```
Event &newTrip.Click
    &trips = NewTrip( AttractionId )
    &totalTrips = &totalTrips + 1
Endevent
```

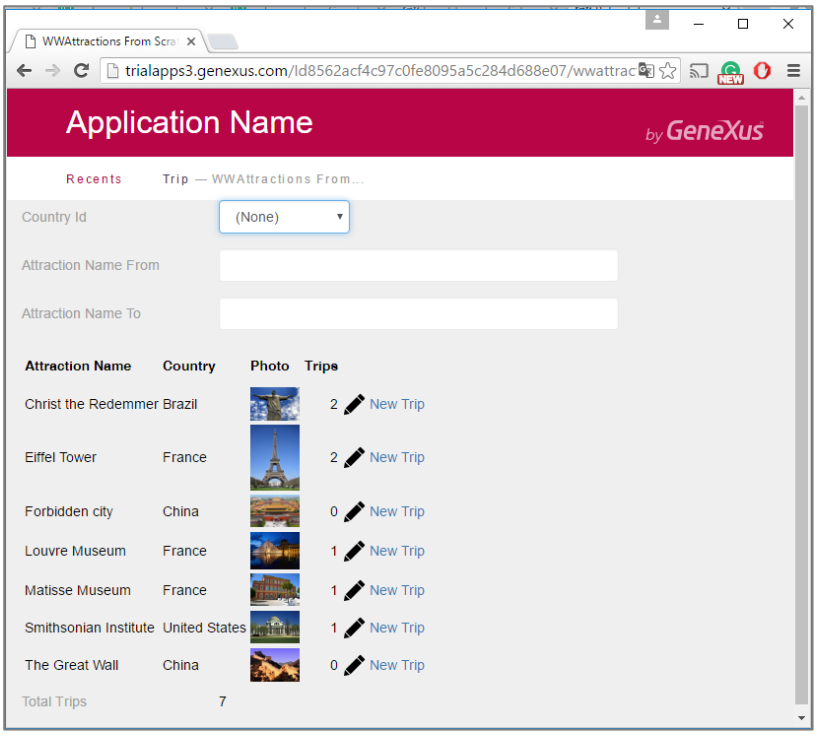
Pero si luego el procedimiento cambia y se agregan más trips, tendremos que recordar hacer el cambio en forma consistente en este evento.

Una mejor solución parecería ser poder pedirle al web panel que vuelva a ejecutar los eventos Refresh y Load. Para ello, contamos con el comando Refresh:

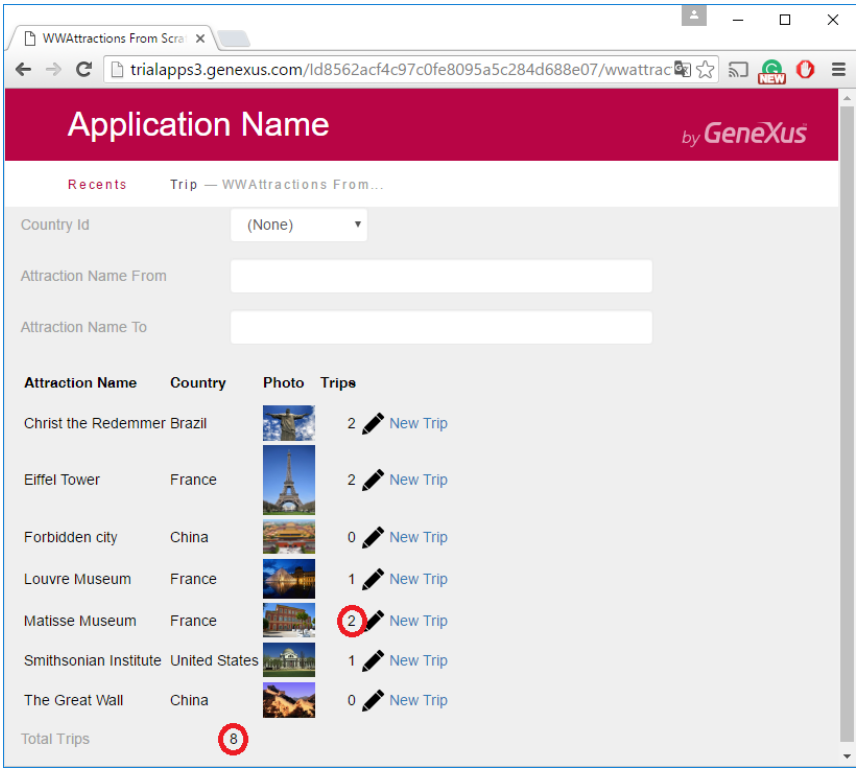


Al hacer esto, se volverá a ir a la base de datos a cargar el grid.

Ejecutemos para probar:



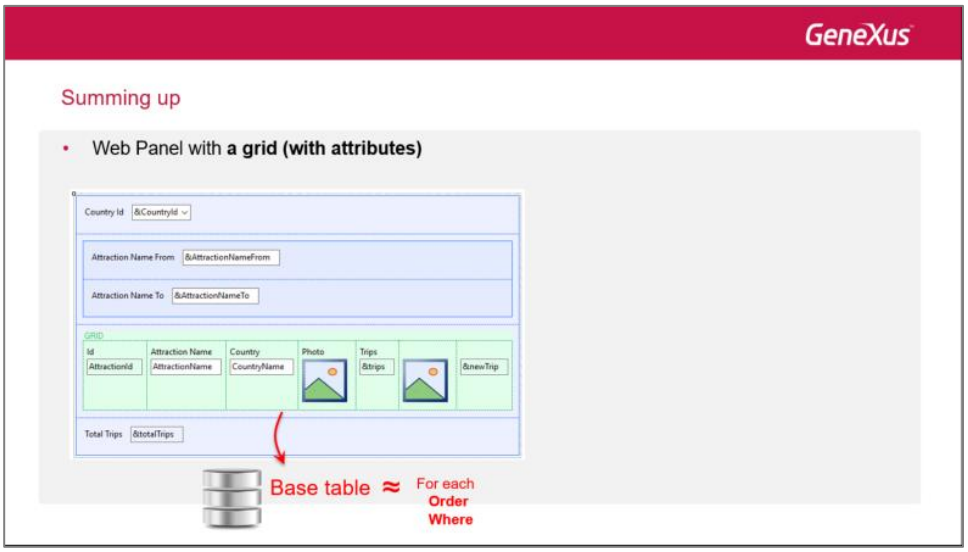
Agreguemos un trip para el museo Matisse:



Vemos que ahora actualizó todo. Es que volvió a ejecutar los eventos Refresh y Load. Este último se ejecutó 7 veces, una vez por cada línea a ser cargada.

Hagamos un repaso de todo lo visto.

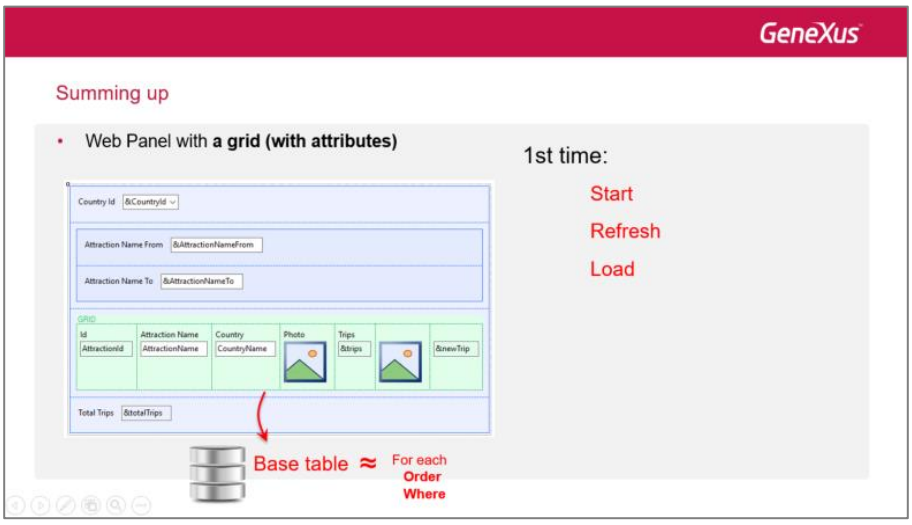
Para el caso de un web panel con un grid con atributos, GeneXus entiende que ese grid tiene una tabla base asociada, es decir una tabla que deberá navegar para cargar las líneas del grid. Cargará una línea por cada registro de esa tabla base. Para filtrar, tenemos la propiedad Conditions del grid, para ordenar, la propiedad Order. Un grid con tabla base es análogo a un for each.



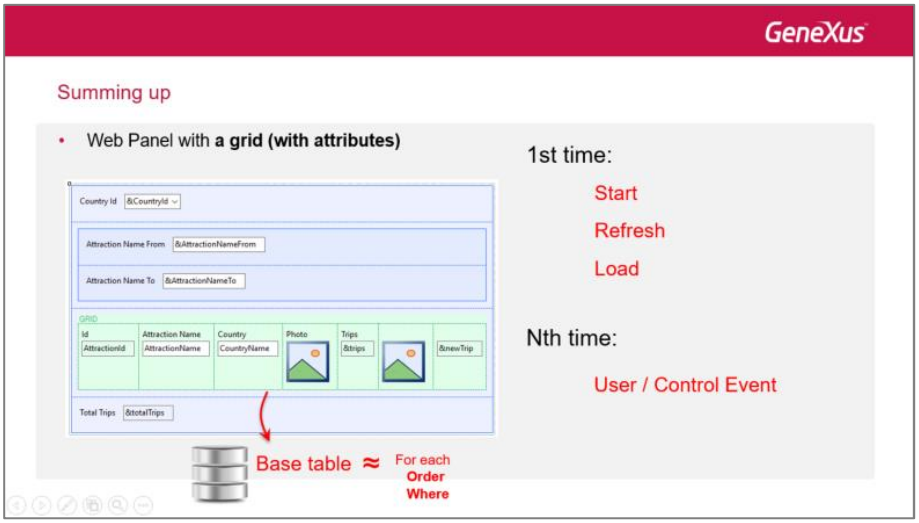
En los web panels se producen eventos del sistema a los que les podemos programar código para que sea ejecutado en el momento en que se disparan. Vimos tres de estos eventos, que se disparan siempre que se abre un web panel es decir, en su primera ejecución:

- El **Start** se dispara esta única vez. Allí pueden inicializarse variables, por ejemplo.
- El **Refresh** se dispara antes de cargarse la información de la pantalla. Tras este evento se producirá el acceso a la base de datos para traer los datos de la tabla base y su extendida.
- Por cada registro de la tabla base que está por ser cargado en el grid, se produce un evento **Load**. Por tanto aquí será el momento de programar todas las acciones que queramos que se ejecuten antes de que la línea sea efectivamente cargada en el grid. Los datos que se cargan en el grid son exclusivamente los de las columnas visibles o invisibles que se hayan incluido en él.

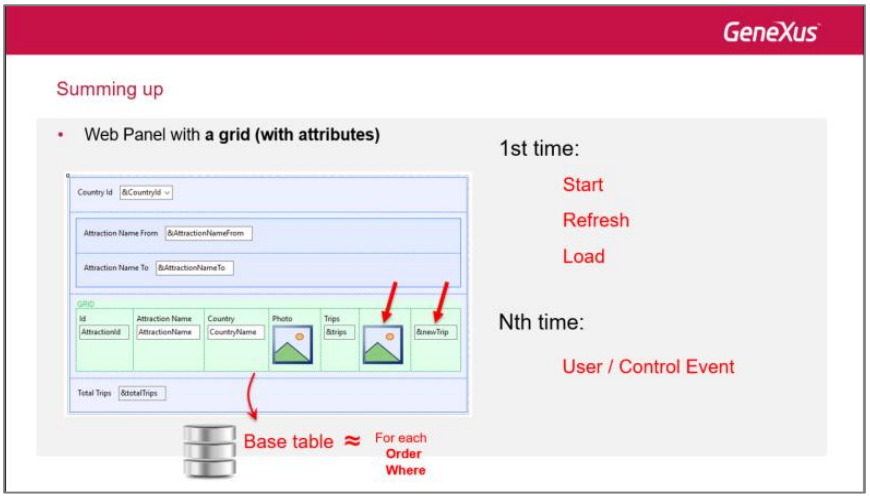
Después de esto, el web panel estará cargado con la información que extrajo de la base de datos y se desconecta de ella.



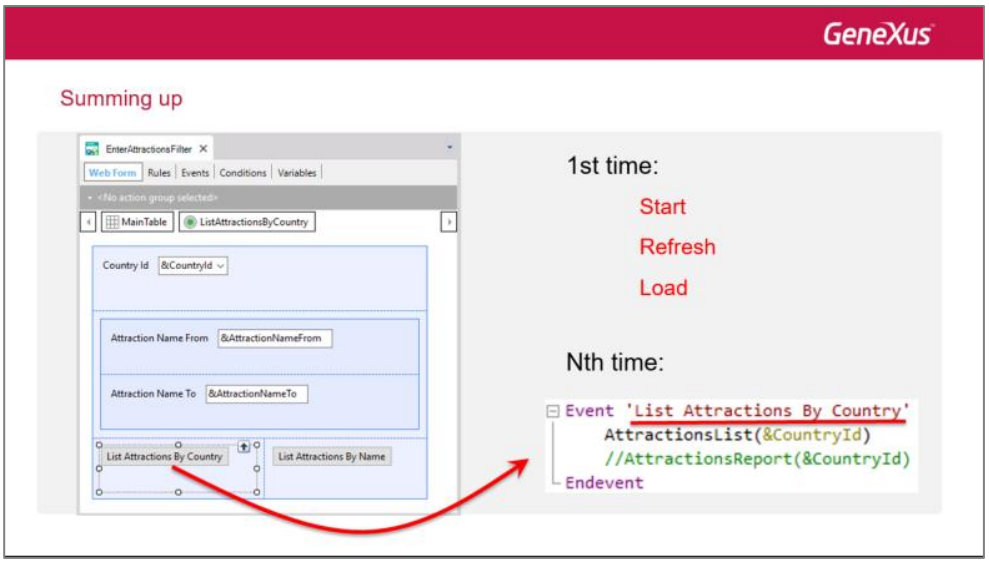
Pero los web panels también permiten definir otros eventos, que se dispararán luego de la primera vez, es decir, una vez que el web panel ya ha sido cargado, y siempre a partir de la acción del usuario.



Por ejemplo, el evento Click que programamos sobre esta imagen o el Click sobre New Trip,

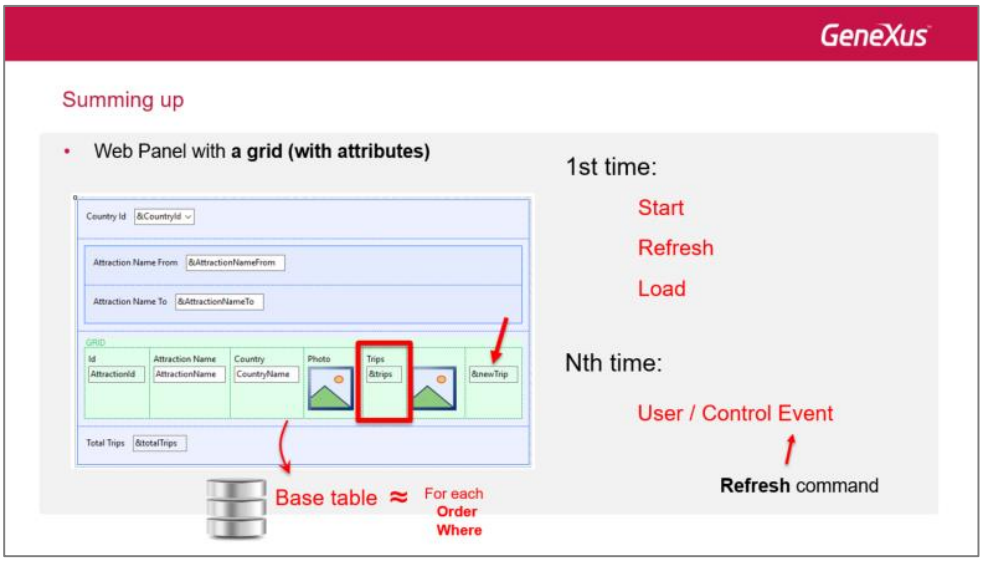


o incluso en el web panel que habíamos definido muchas clases atrás, el evento de usuario que asociamos al botón, al que dimos este nombre:

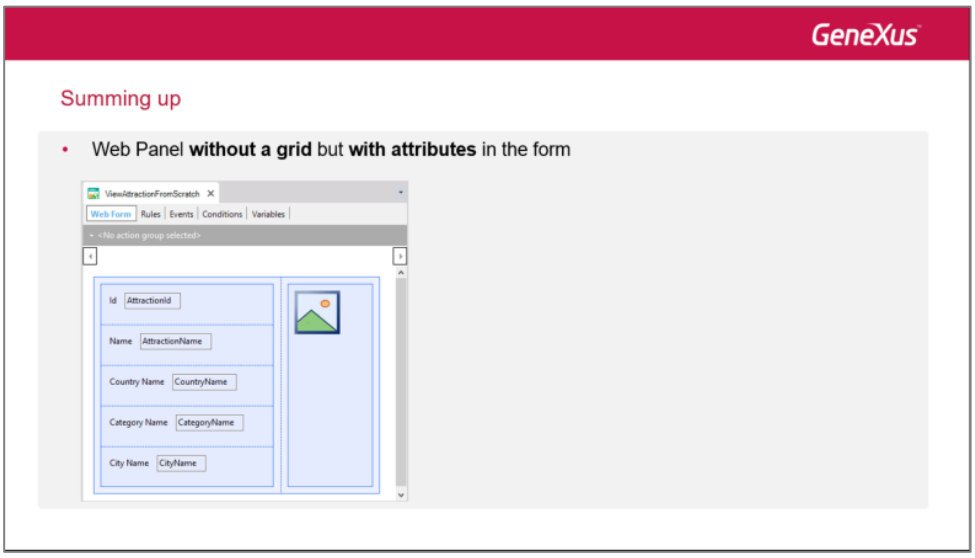


Estos eventos son conocidos como eventos de usuario, o eventos de los controles (como el Click que vimos). Cuando el usuario provoca un evento de estos, únicamente se ejecuta su código, sin refrescarse la pantalla. La única excepción es cuando el evento se produce a nivel de una línea del grid, como el caso que vimos de &newTrip, y dentro de su código se asigna valor a una variable del mismo grid, que en nuestro caso era &Trips. En ese caso, el valor de la variable se refresca en la pantalla, para esa línea, para que se muestre actualizada.

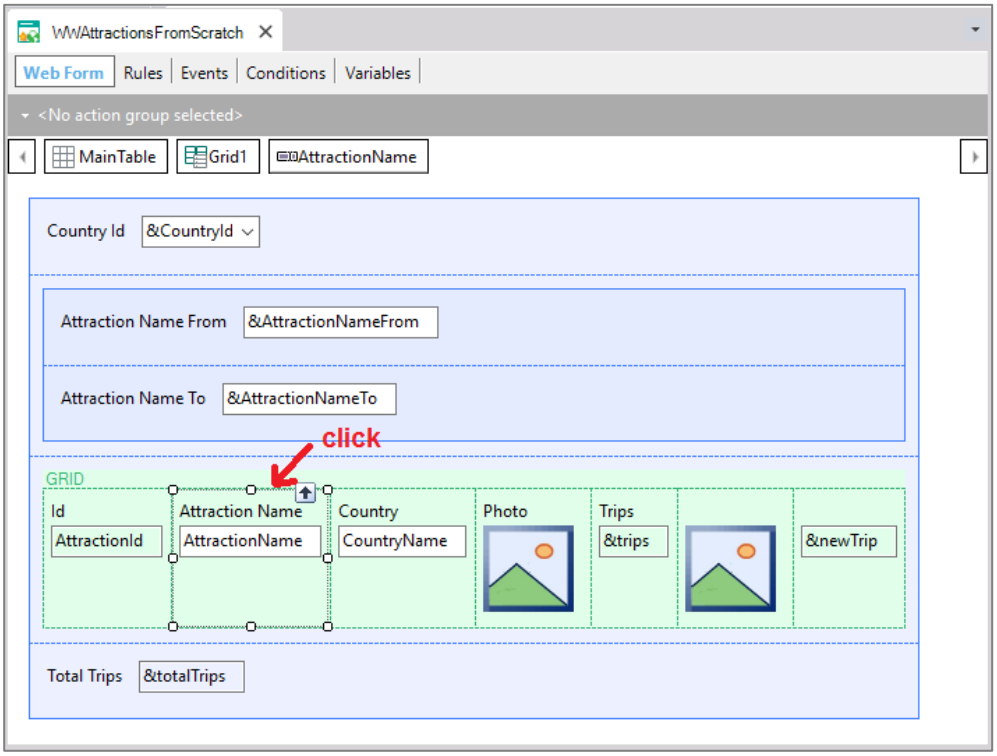
Si necesitamos que se vuelva a ejecutar el Refresh y se vuelvan a cargar las líneas en el grid a partir de la base de datos (por ejemplo para actualizar el total de las líneas), entonces podemos escribir el comando Refresh en el evento.



Estudiamos el caso de un web panel con un grid con atributos. Pero, ¿qué pasa si lo que tenemos es un web panel sin grid pero con atributos en el form?

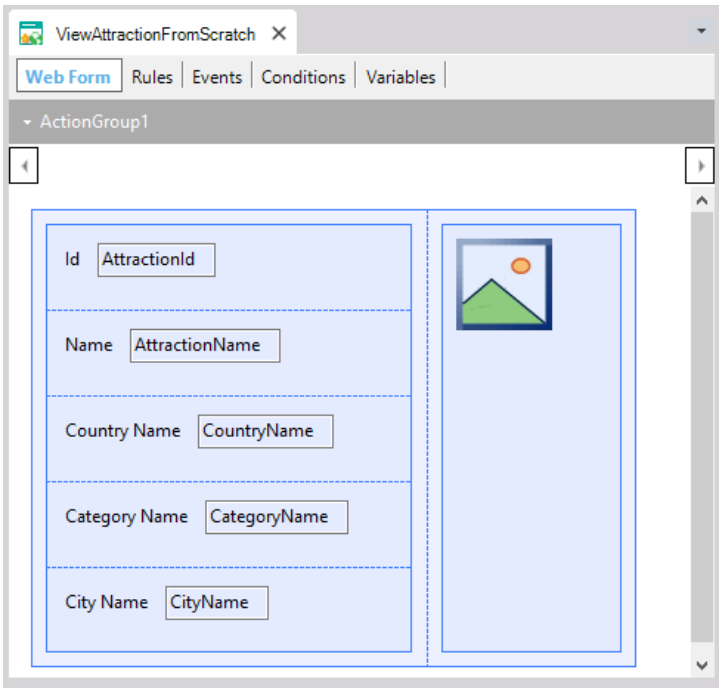


Supongamos que queremos que, haciendo clic sobre el nombre de atracción,…



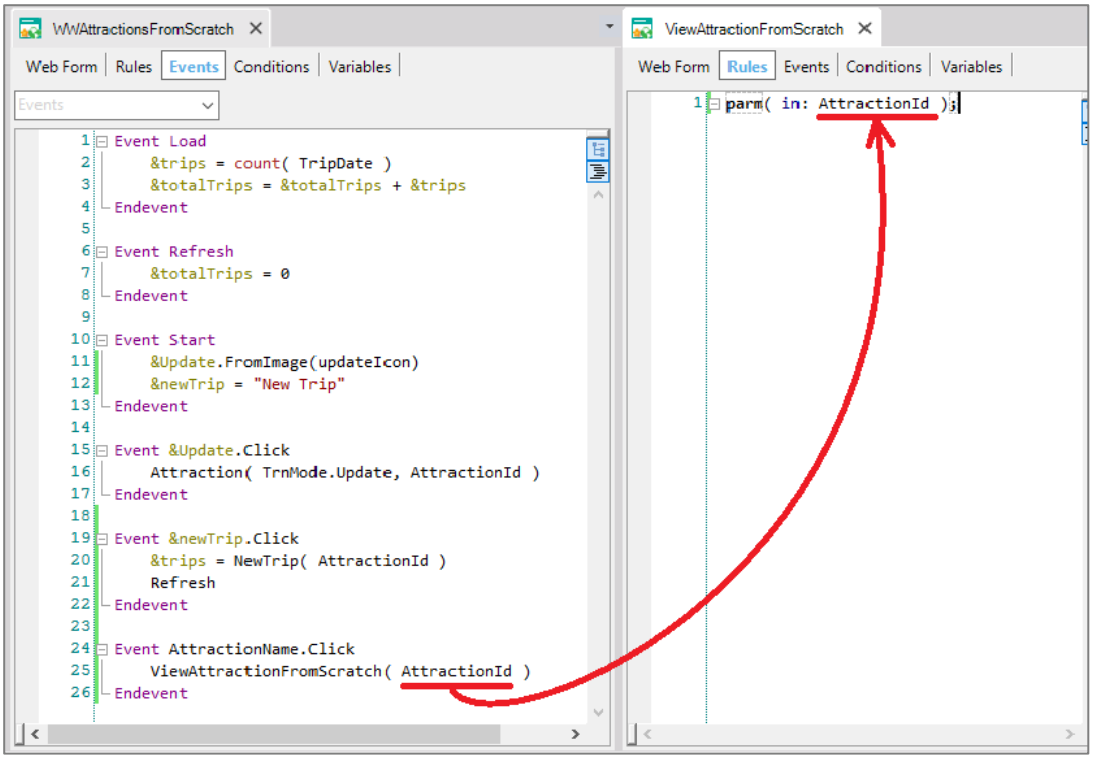
… se llame a un web panel que muestre todos los datos de la atracción.

Para ello, ya hemos implementado este web panel…



… en cuyo form hemos insertado los atributos de una atracción que nos interesa mostrar.
Además, hemos escrito una regla parm, para recibir un parámetro. Veamos que en vez de recibir en una variable, elegimos recibir en el atributo AttractionId.

Lo que deseamos es que cuando desde el evento click de AttractionName en nuestro otro web panel…

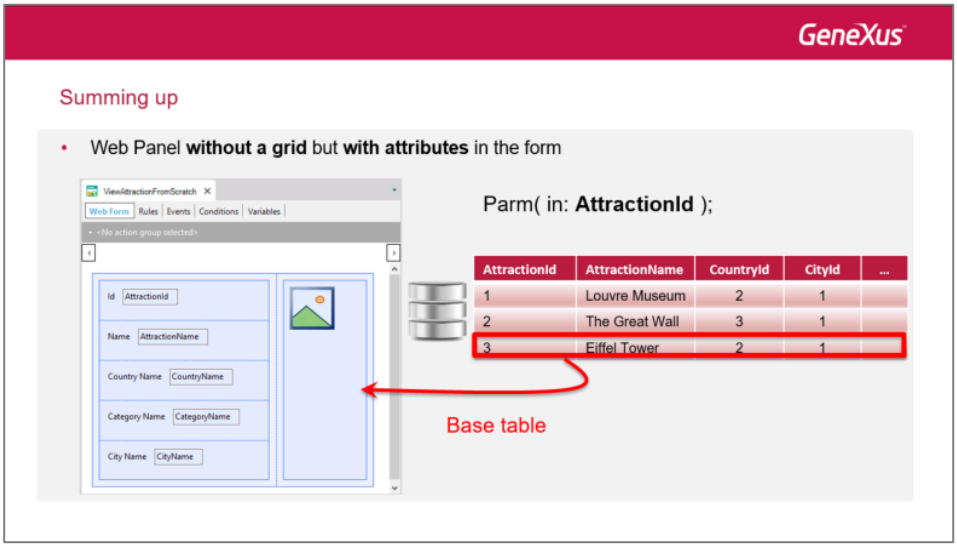


… se llame a este Web panel y se le pase el id de atracción de la línea del grid, automáticamente GeneXus vaya a la tabla de atracciones, encuentre la atracción con ese id y para esa atracción, muestre la información de los atributos que se colocaron en el form.

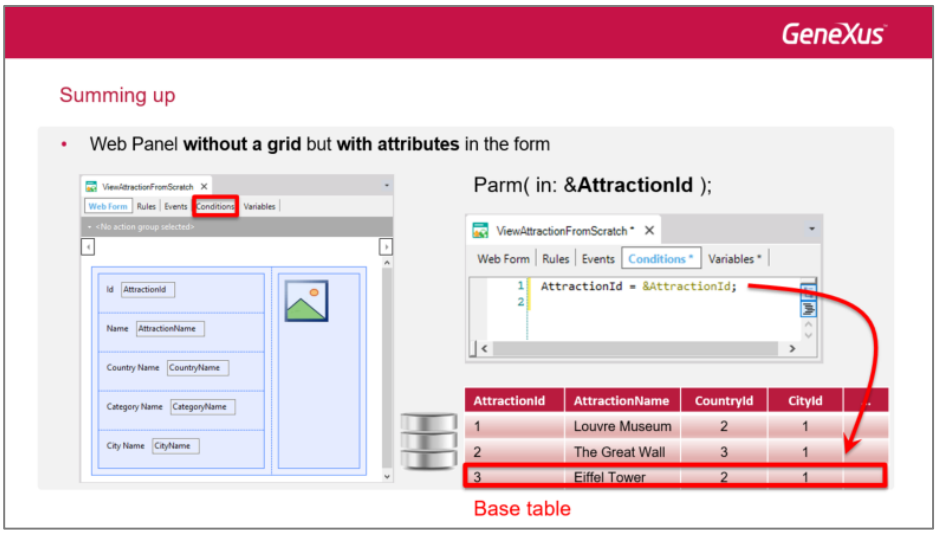
En definitiva, un web panel que no tiene ningún grid pero que tiene atributos en el form también tendrá tabla base. ¿Cómo la determina GeneXus, siendo que ahora no tenemos transacción base para indicar? No lo veremos en este curso, pero es análogo al caso de un for each en el que no se indica Transacción Base.

Pero en este caso, como no tenemos un grid, se tendrá que cargar únicamente UN registro de esa tabla base y de su extendida. ¿Dónde indicamos el filtro que permita devolver ese registro?

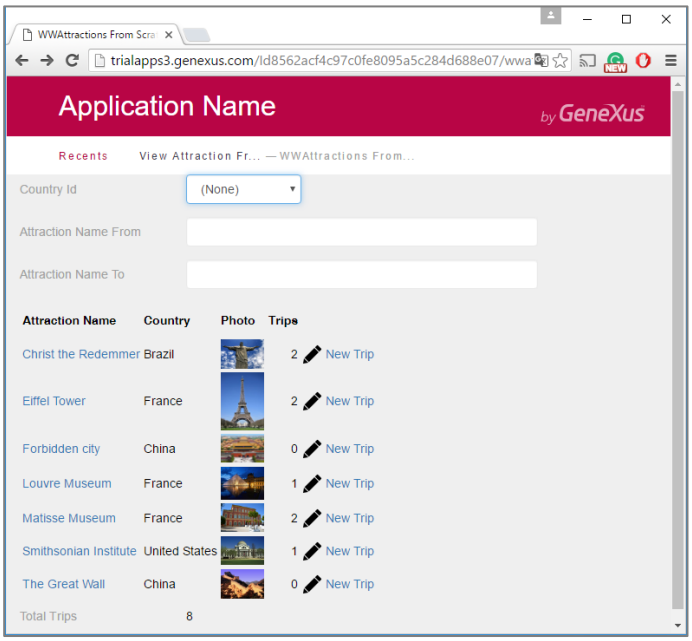
En nuestro caso fue recibiendo en el atributo AttractionId. Allí estamos especificando el filtro automático, para que sepa con cuál de todos los registros de la tabla base debe quedarse.

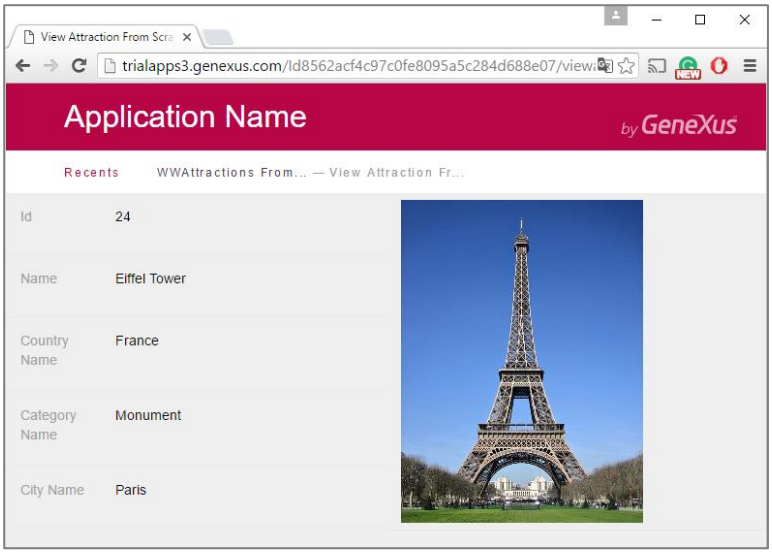


Si necesitamos establecer otro tipo de condición, o, si por ejemplo hubiéramos recibido en variable, en lugar de en atributo, tenemos la solapa de Conditions para establecer el filtro:



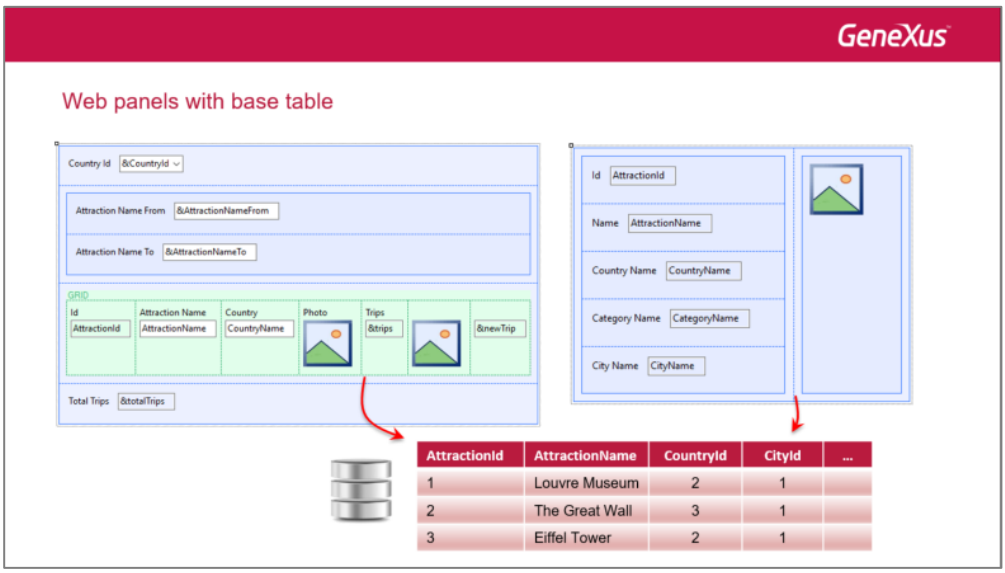
Ejecutemos para verlo.





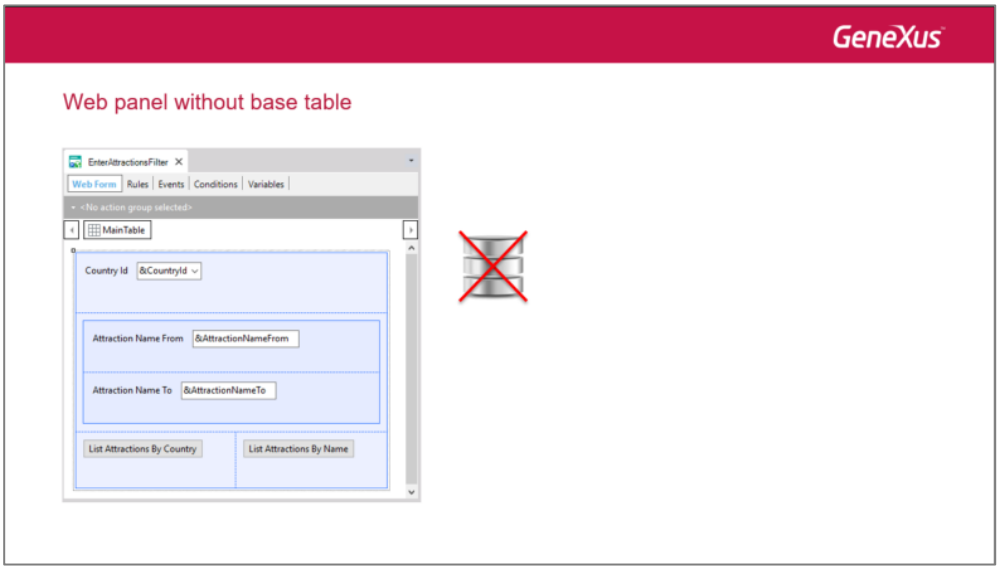
Hemos visto hasta aquí dos web panels que tienen tabla base

- el primero con un grid Allí la tabla base del grid es la tabla base del web panel, es decir, la tabla a la que el web panel automáticamente decide ir a navegar para traer la información a cargar en la pantalla.
- el segundo sin grid, pero con atributos. Allí la tabla base del web panel se encuentra a partir de esos atributos.



Ahora veremos el caso de web panels sin tabla base, es decir, web panels que no tienen programada **en forma automática** una consulta a la base de datos.

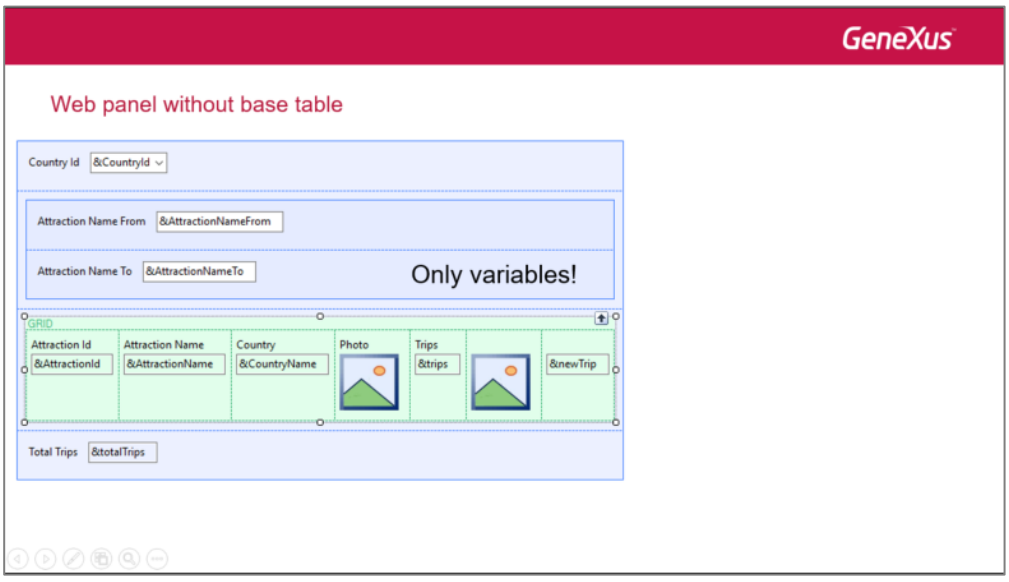
El caso más obvio es cuando no se consulta en absoluto a la base de datos como fue el caso de nuestro web panel de partida, el que únicamente pedía datos al usuario y llamaba a otros objetos.



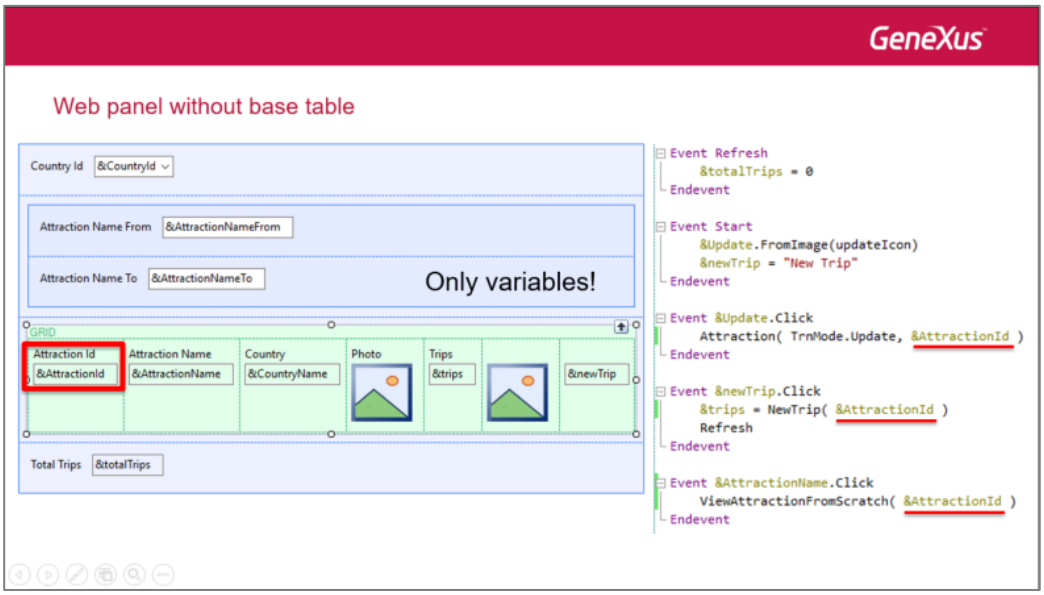
Pero también podemos tener un web panel que sí consulte la base de datos, solo que esa consulta y carga en la pantalla esté completamente en manos del desarrollador.

Los web panels que habíamos implementado con tabla base, bien podrían haberse implementado de este otro modo.

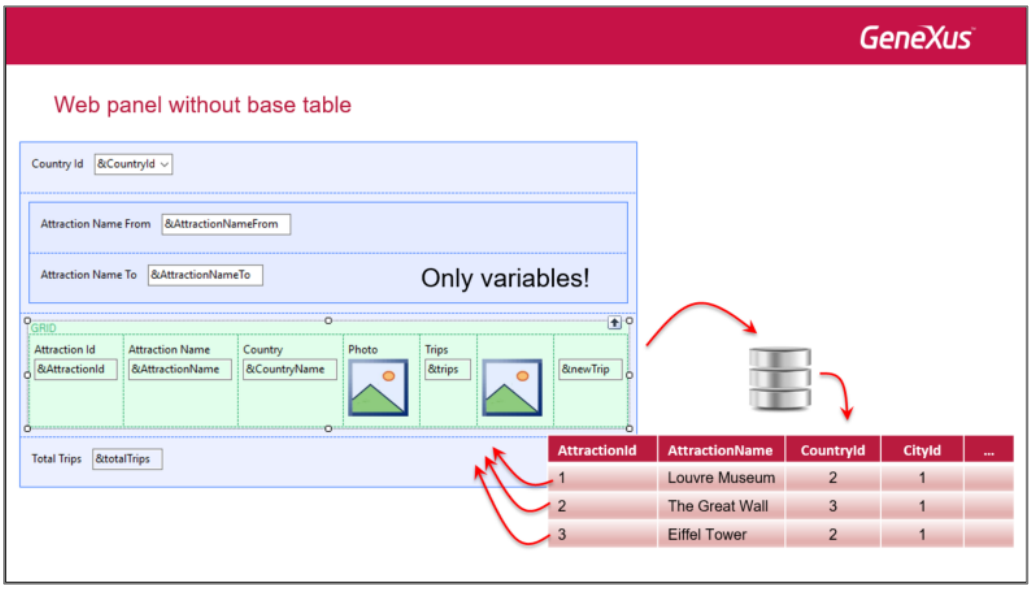
Veamos el caso del web panel que muestra en el grid las atracciones. Pero ahora en vez de tener atributos en el grid, tenemos variables.



Así, en los eventos tenemos que cambiar las invocaciones que teníamos, donde pasábamos el atributo AttracionId, por la variable &AttractionId.

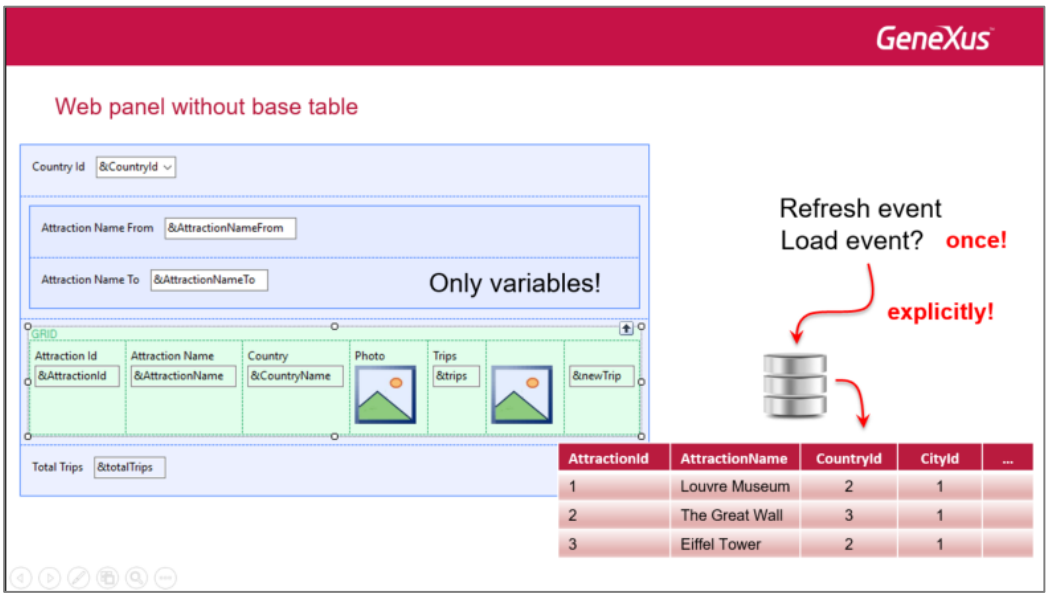


Ahora bien, si ahora sólo tenemos variables, ¿cómo sabe GeneXus que tiene que ir a navegar la tabla Attraction y su extendida para cargar una línea del grid por registro?

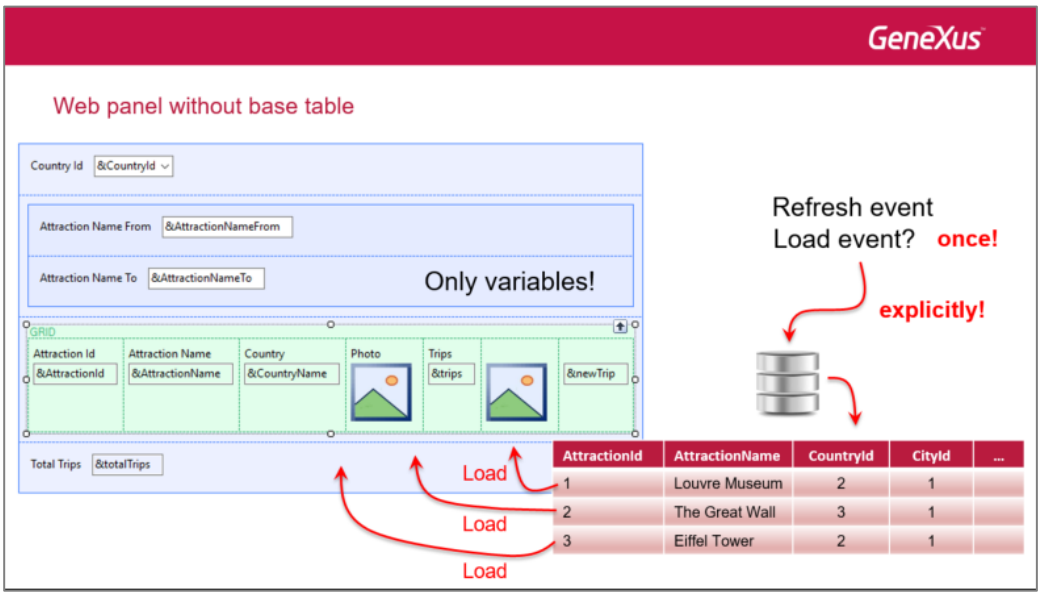


No lo sabe. La carga de este grid no será automática, como en el caso en que colocamos atributos.
Es decir, el evento Load no se ejecutará cuando ya se haya ido a navegar una tabla, por cada registro en el que se está posicionado en un momento dado. Es que ¡no se está posicionado en lugar alguno!

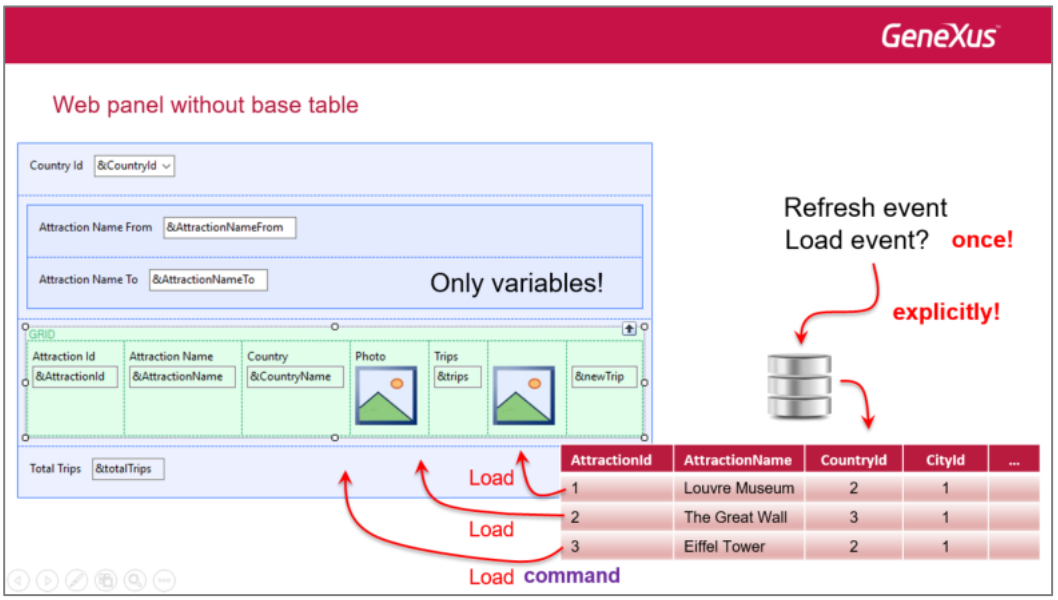
Sin embargo, el evento Load se va a disparar, sí. Solo que lo hará una única vez después del Refresh y sin estar en la base de datos en absoluto.
Es en ese disparo, en esa ejecución, en la que vamos a tener que programar la carga del grid en forma manual.



Es decir, vamos a tener que pedir explícitamente que se acceda a la base de datos (en nuestro caso, yendo a la tabla Attraction), y decirle cada vez, que cargue cada línea.

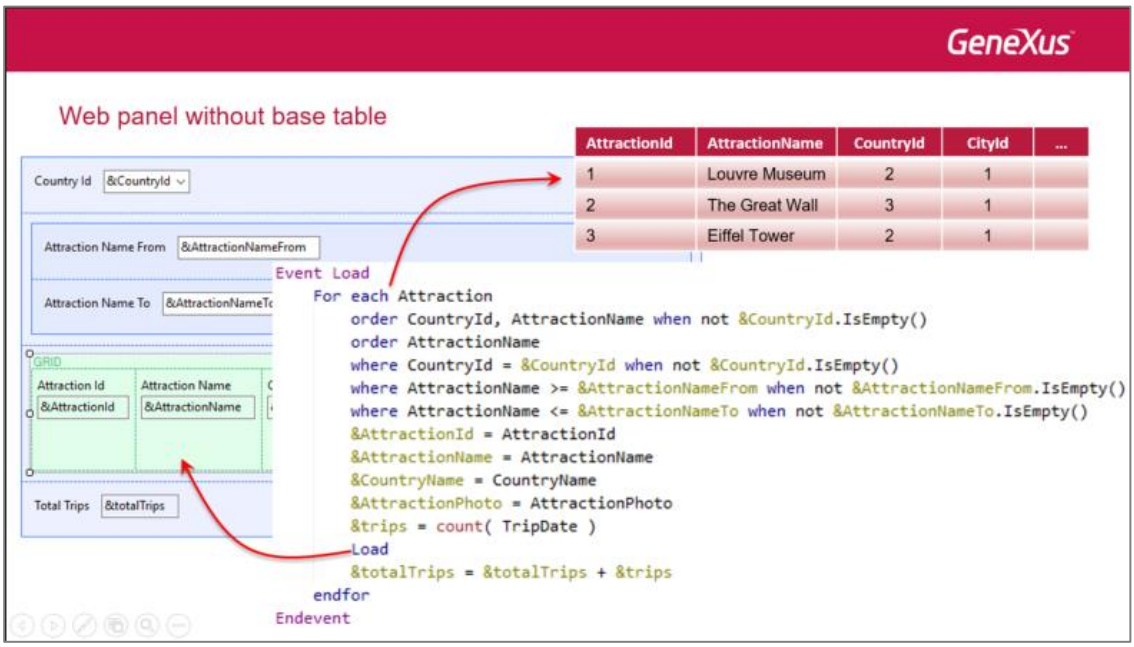


Para decirle que inserte una nueva línea en el grid, con los valores que en ese momento tengan las variables correspondientes a las columnas, se cuenta con el **comando Load**. El comando, **no** el evento. Toda vez que dentro del **evento Load**, se encuentre un **comando Load**, único lugar donde este comando está permitido, se insertará una línea en el grid.

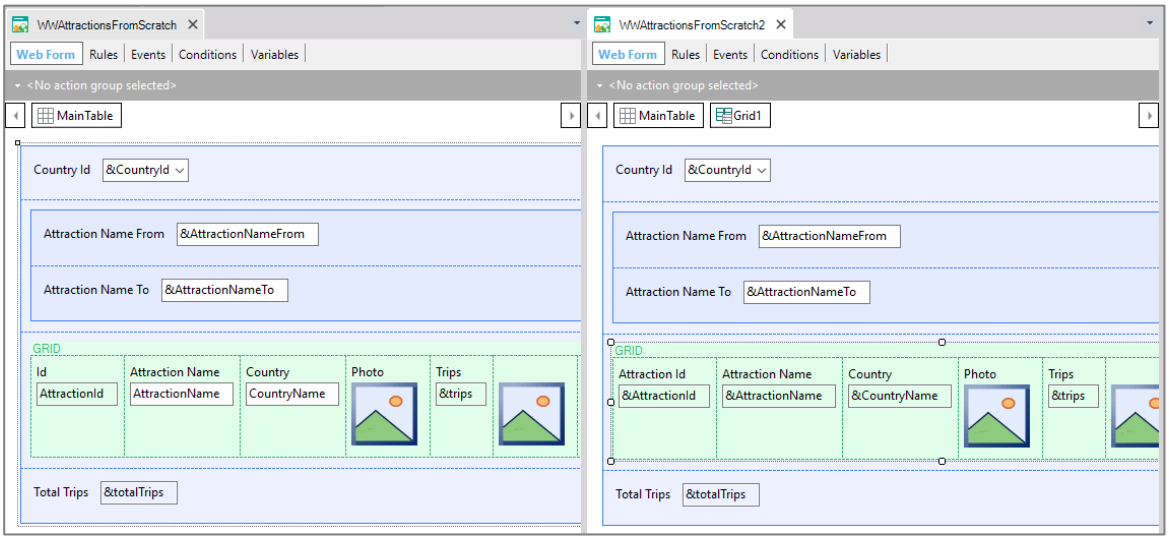


Lo que vamos a tener que hacer es programar, entonces, dentro del **evento Load**, el acceso a la tabla Attraction con un for each En él especificaremos las cláusulas order y en las cláusulas where las condiciones que deben cumplir los datos. Y por cada registro que cumpla con esos filtros, cargaremos las variables del grid con los valores de esa atracción.

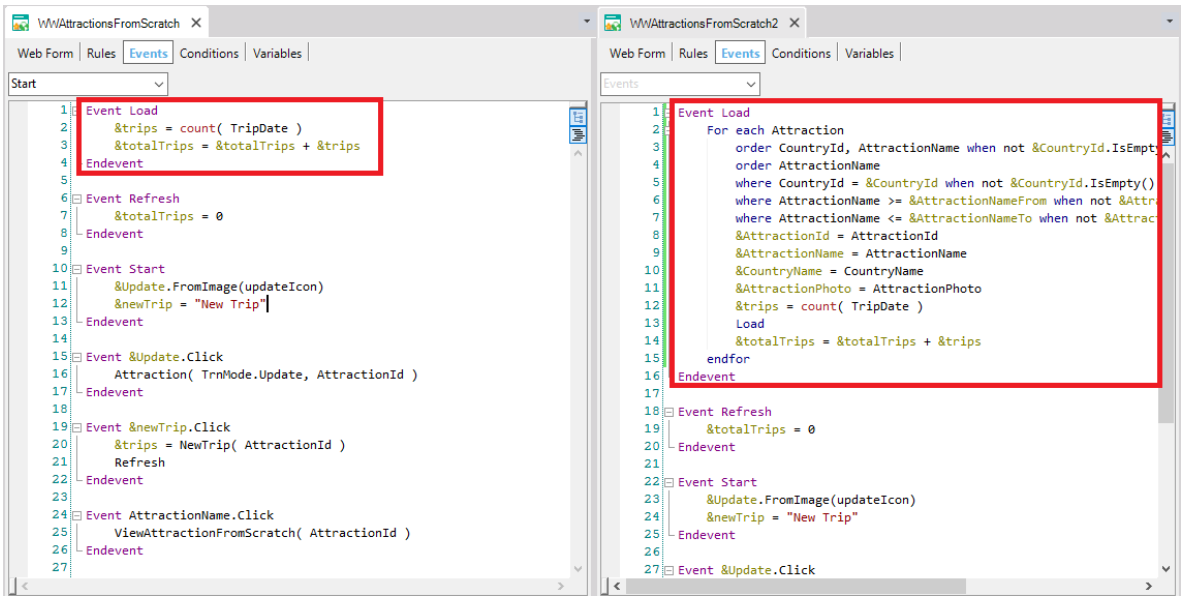
Y cuando tenemos todas las variables cargadas, entonces escribimos el comando Load para que se agregue una línea en el grid con esos valores.



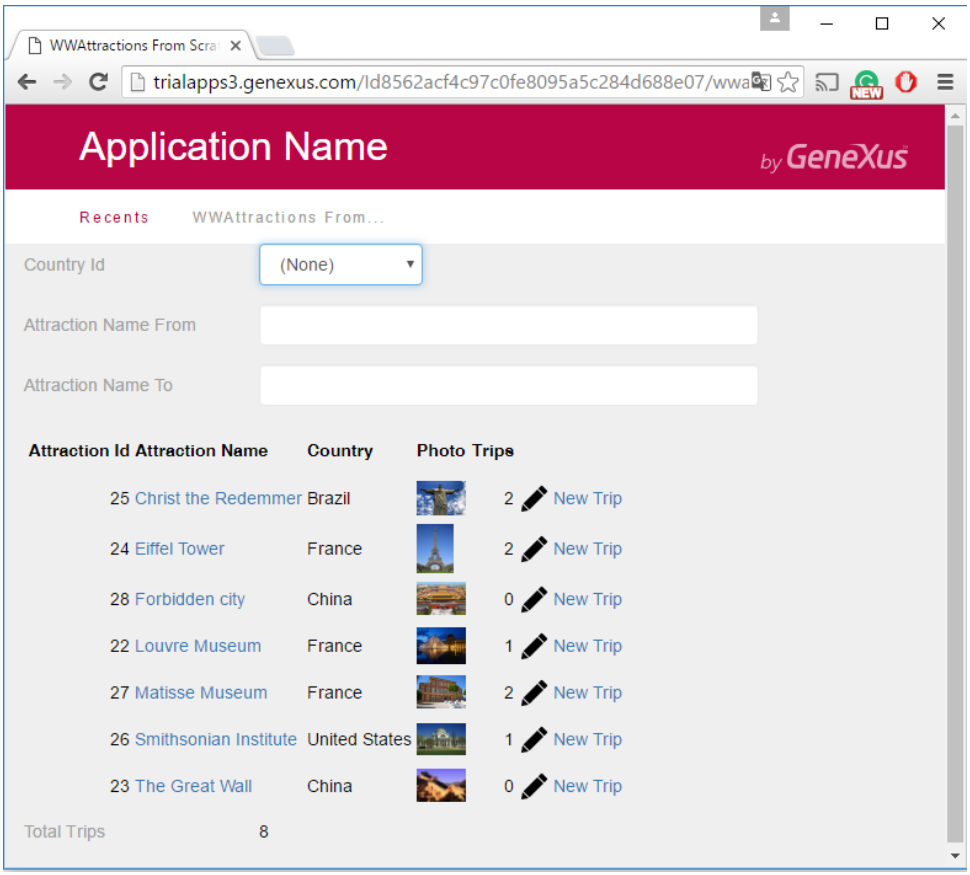
Aquí tenemos este web panel ya implementado. Hemos hecho un save as del que teníamos con atributos, y lo cambiamos por variables:



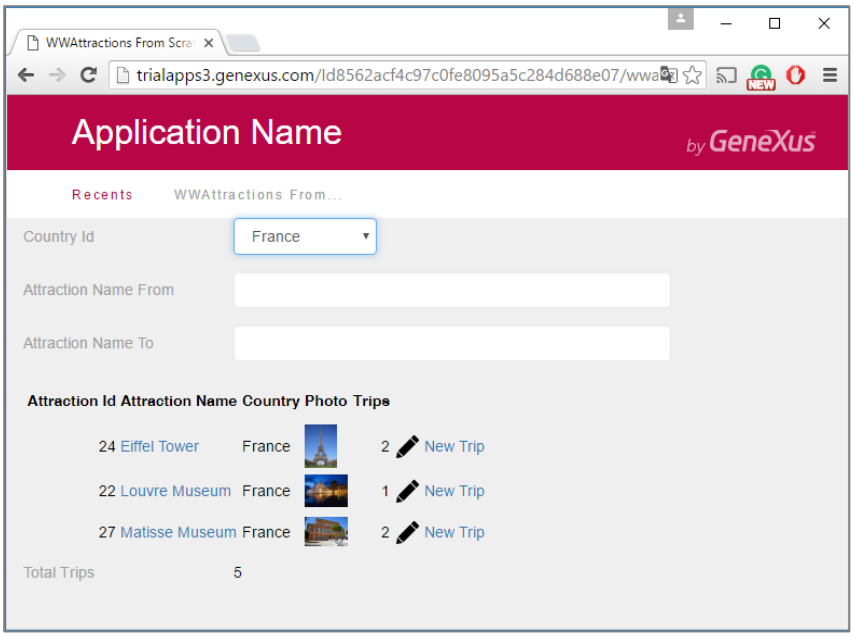
Y también cambiamos los eventos, tal como estudiamos recién.



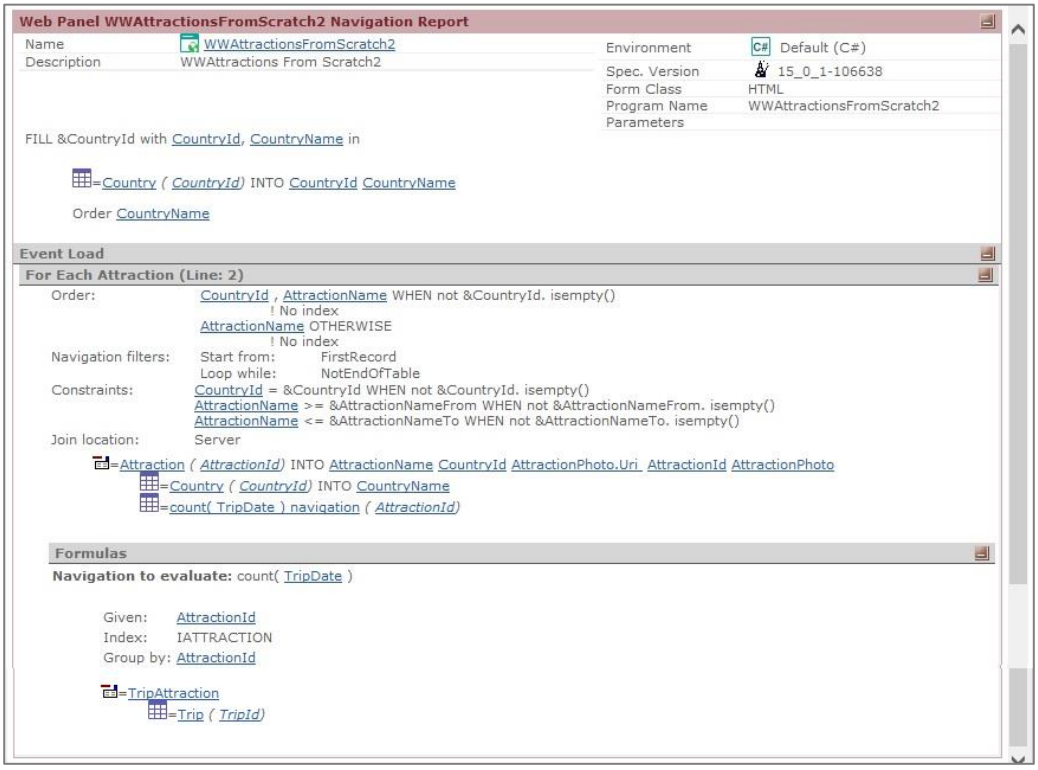
Probemos lo hecho para ver cómo en ejecución no notamos diferencia alguna entre el web panel con tabla base y el web panel sin.



Filtremos por ejemplo por Francia.



Ahora, si observamos el listado de navegación del web panel sin tabla base:

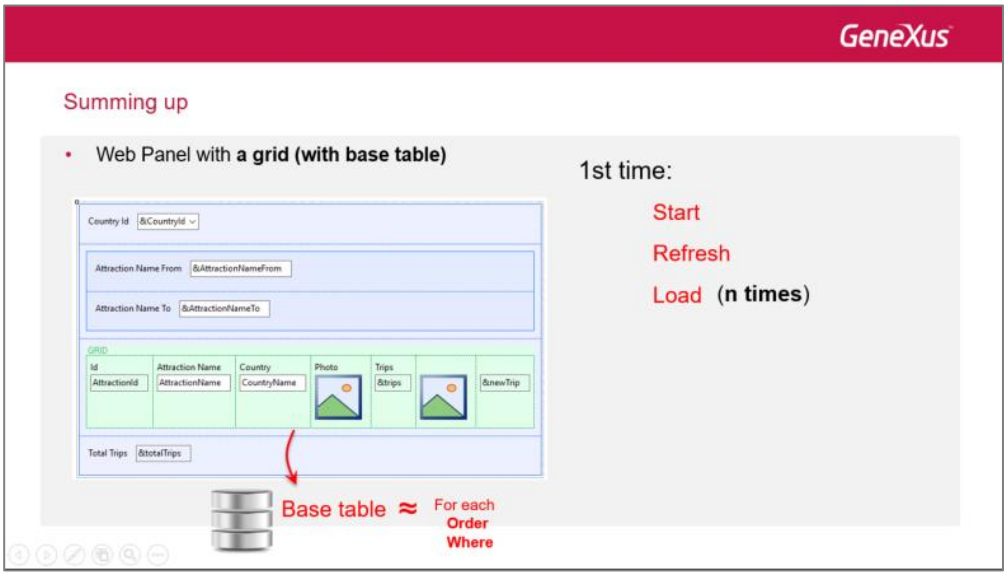


Vemos que aparece el for each en el Load, indicando la navegación.

Resumiendo:

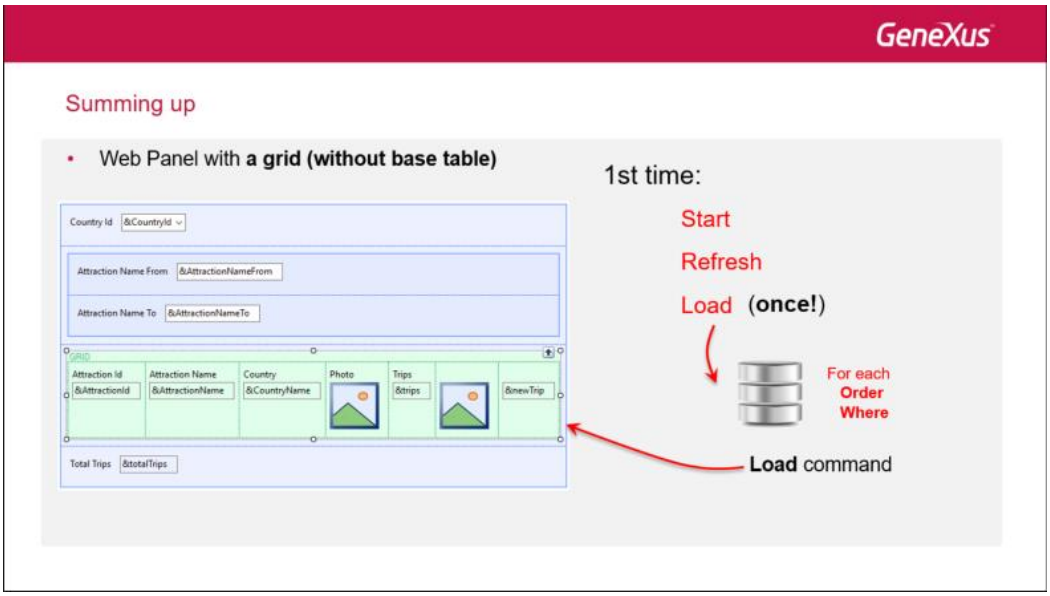
Cuando teníamos el web panel con un grid con tabla base, al abrir el web panel se ejecutaba el evento Start, inmediatamente el Refresh, a continuación del cual se accedía a la tabla base en forma automática, filtrando de acuerdo a las conditions del grid y ordenando de acuerdo a los atributos indicados en la propiedad Order del grid. Decimos que existe una suerte de for each implícito, que GeneXus coloca internamente, sin que el desarrollador deba hacer nada.

Y por cada registro a ser cargado como línea del grid, antes de hacerlo se ejecuta el evento Load. Es por ello que el evento **Load** se va a disparar, en este caso, **tantas veces como líneas vayan a cargarse en el grid**.

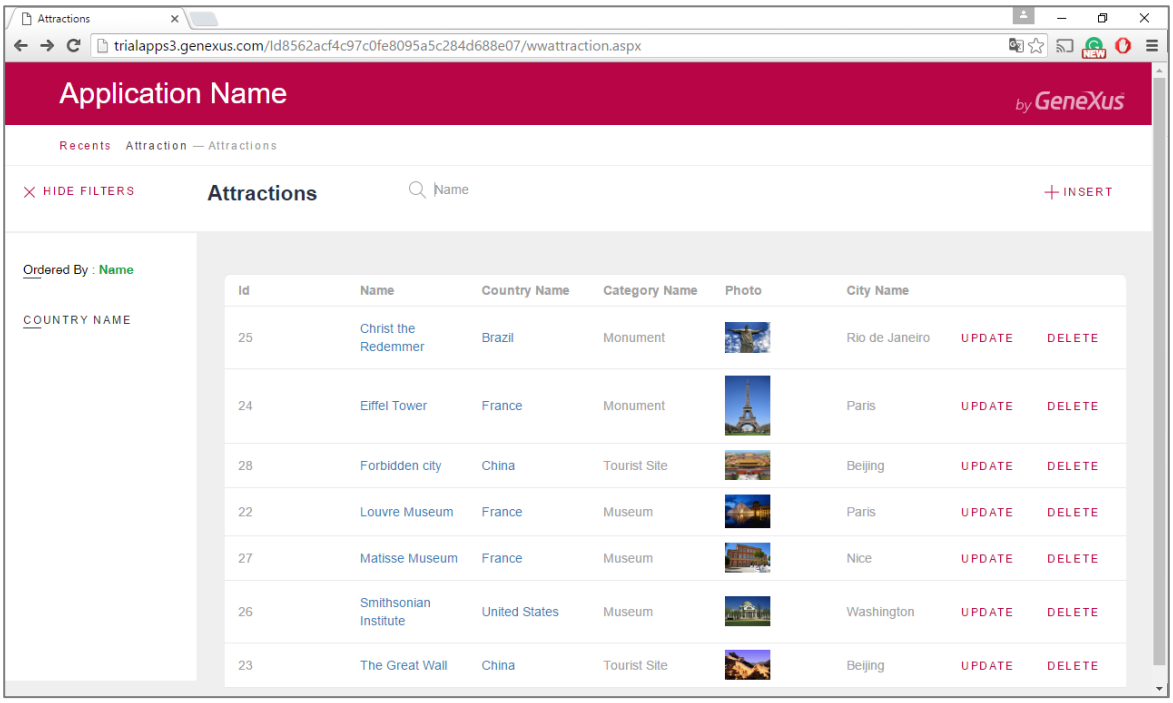


En cambio, cuando el web panel no tiene tabla base, en el grid hay sólo variables y no hay ningún for each implícito.

Al abrir el web panel se ejecuta el evento Start igual que antes, el Refresh igual que antes, y el Load una única vez. Si se necesita ir a la base de datos para recuperar información, hay que hacerlo en forma explícita dentro de este evento. Dicho de otro modo: debemos escribir el for each y allí utilizar las cláusulas Order y Where. Para poder cargar cada línea, hay que hacerlo explícitamente con el **comando Load**, por cada una.

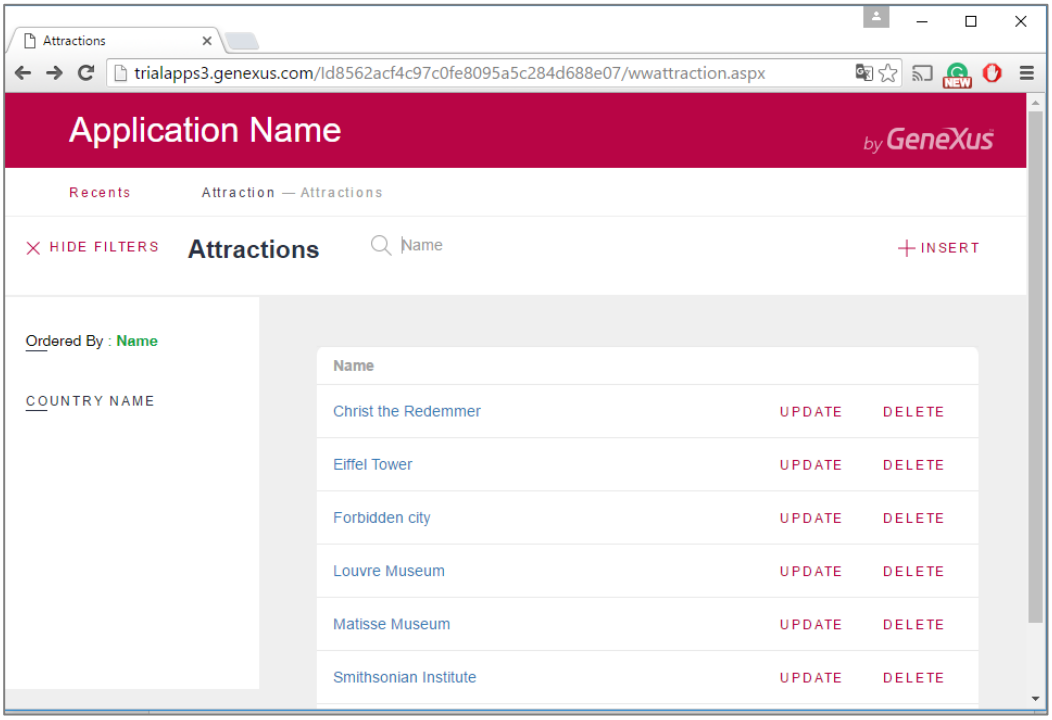


Para finalizar, observemos que las aplicaciones diseñadas por GeneXus autoajustan la información que presentan en la pantalla de acuerdo al tamaño de esa pantalla. Por ejemplo, ejecutemos el Work With Attractions que creó el pattern. Estamos ejecutando en un navegador que está ocupando toda la pantalla de una notebook.

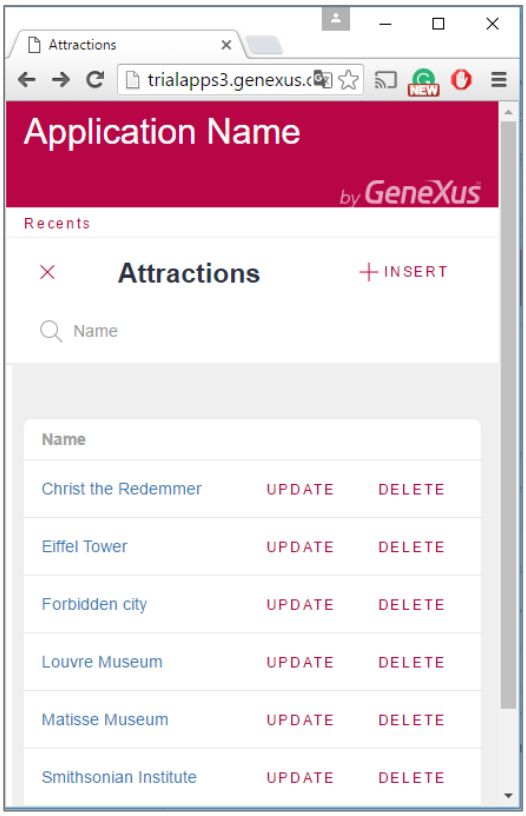


Pero observemos qué sucede si hacemos más pequeña la pantalla del navegador:

Si la vamos achicando desde la derecha…

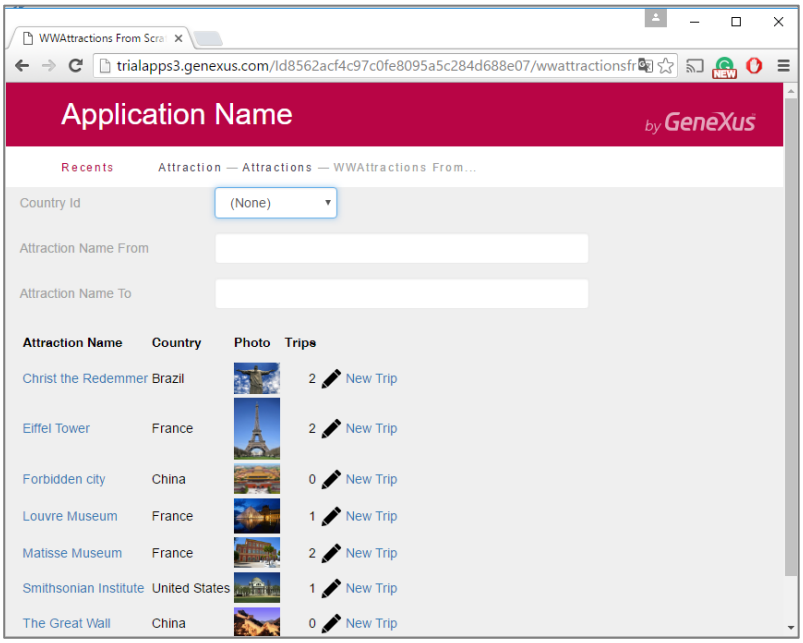


Vemos que en un momento el grid empieza a mostrar únicamente el nombre de la atracción y las acciones. Y si la seguimos achicando… ya ni siquiera vemos la columna izquierda:

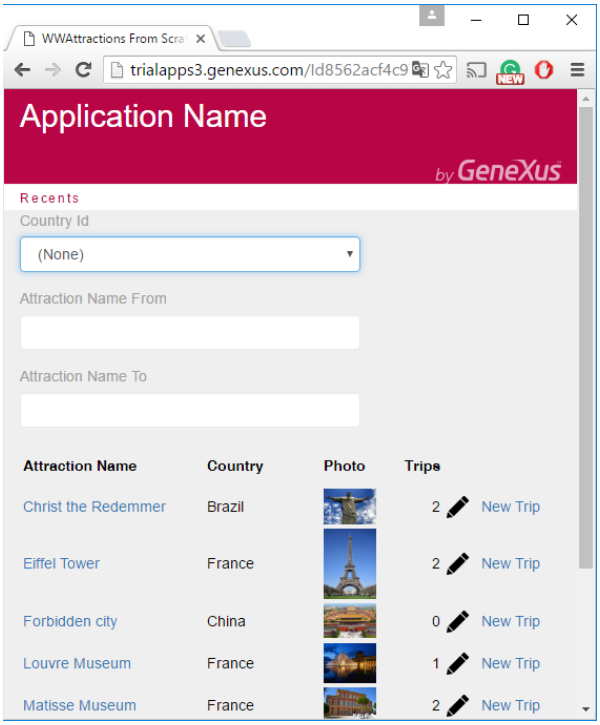


Así saldrá en un teléfono móvil que ejecute la aplicación desde su navegador.

Si ahora vamos a nuestro Work With, el que desarrollamos de cero, vemos que aunque no hicimos nada, tiene cierto autoajuste mínimo:

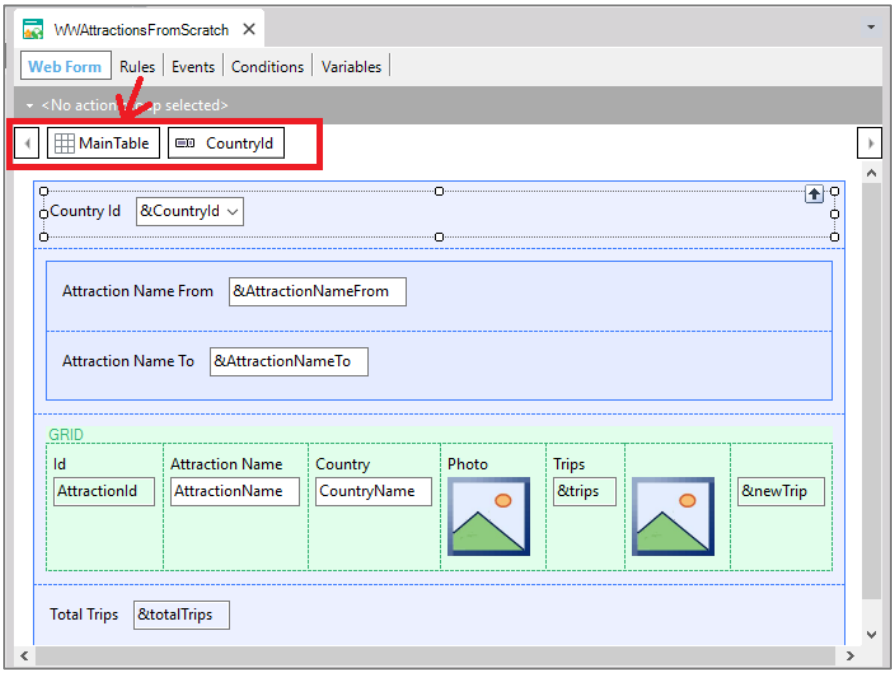


Por ejemplo, prestemos atención a las variables de arriba, que pasarán de tener la etiqueta a la izquierda y ocupando un ancho determinado antes de que empiece el propio control variable,

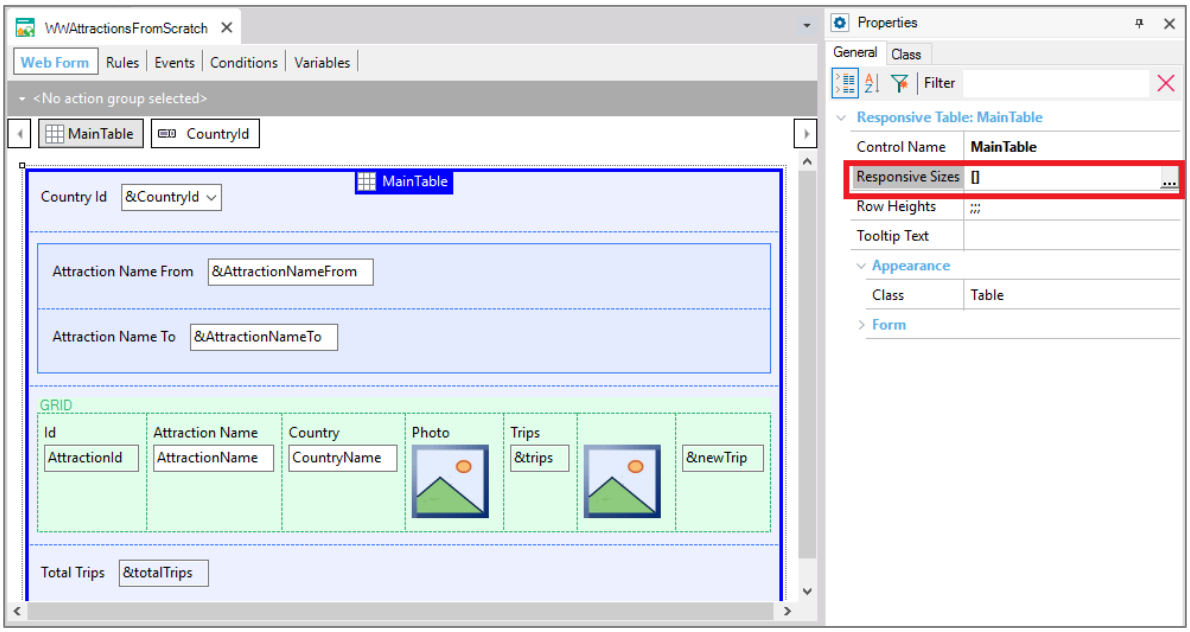


A tener la etiqueta arriba, ocupando el 100% del ancho.

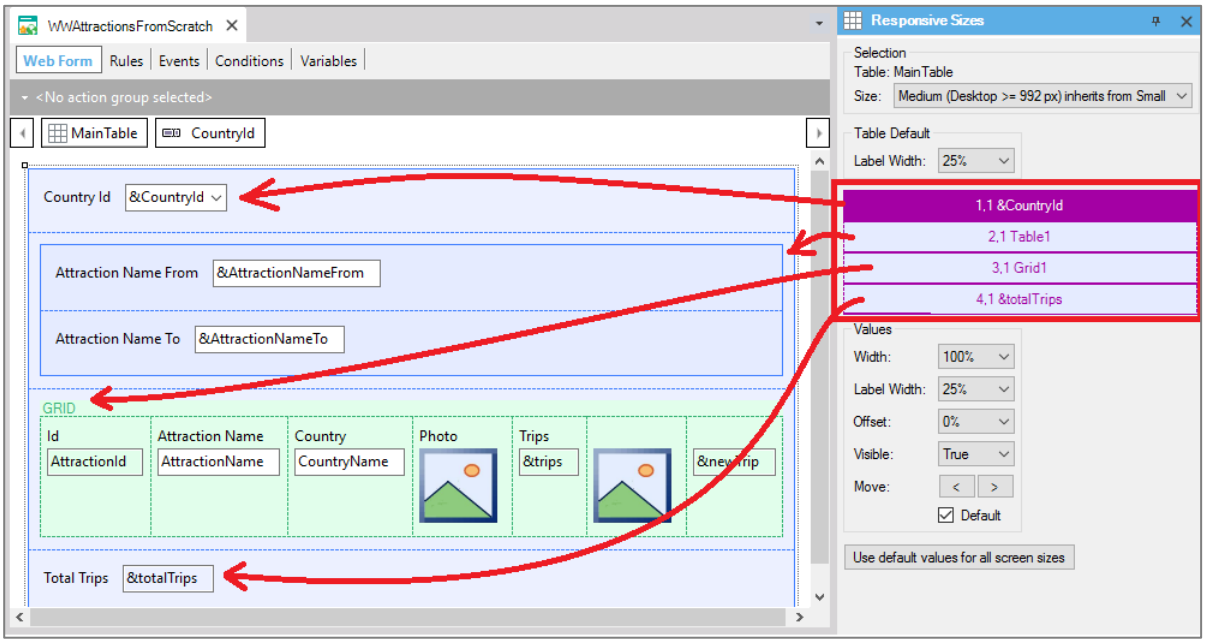
Para lograr este comportamiento los objetos web utilizan las tablas responsivas: responsive tables.
No lo estudiaremos en este curso, pero si vamos a nuestro web panel y nos posicionamos dentro del form en el control Country Id, vemos que aquí arriba nos muestra el control tabla dentro del que está insertado:



Es la tabla principal. Si cliqueamos allí:

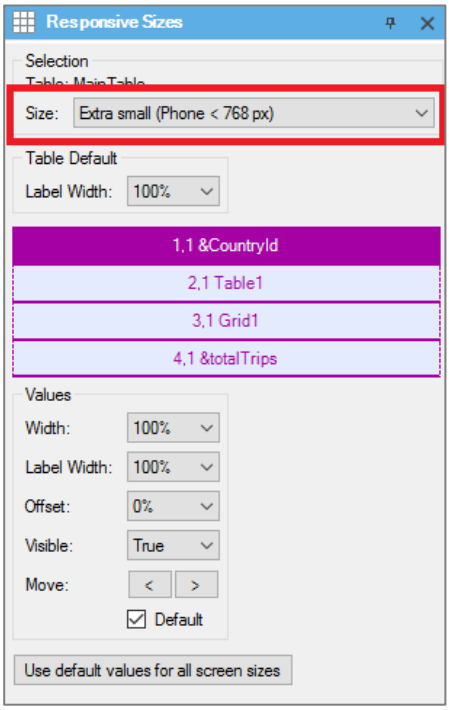


Vemos que entre las propiedades de esta tabla está la **Responsive Sizes**, es decir, tamaños de respuesta o "responsivos". Y si cliqueamos allí, se nos edita esta pantalla:



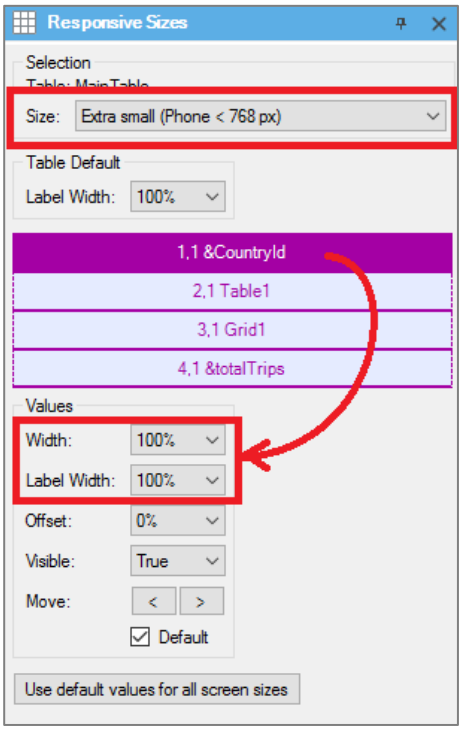
Que si vemos tiene los controles que están insertos dentro de esa tabla principal. Algunos, como esta tabla o como el grid, tienen a su vez controles dentro.

Pero además tenemos un combo que nos permite seleccionar los tamaños de pantalla. Estamos en Medium. Pero si ahora elegimos Small o Extra Small



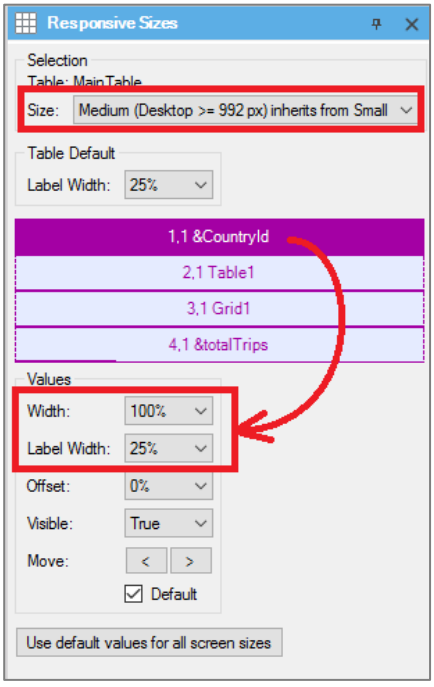
Vemos que los valores cambian.

En particular, observemos que para &CountryId en pantalla Extra small dice que la etiqueta va a ocupar el 100% del ancho, y el propio control variable ocupará el 100% restante.



Y es por eso que en ejecución veíamos la etiqueta sobre la variable.

Si ahora elegimos el tamaño Medium, vemos que la etiqueta ocupa el 25% del ancho, y el control variable propiamente dicho el tamaño restante hasta completar el 100%.

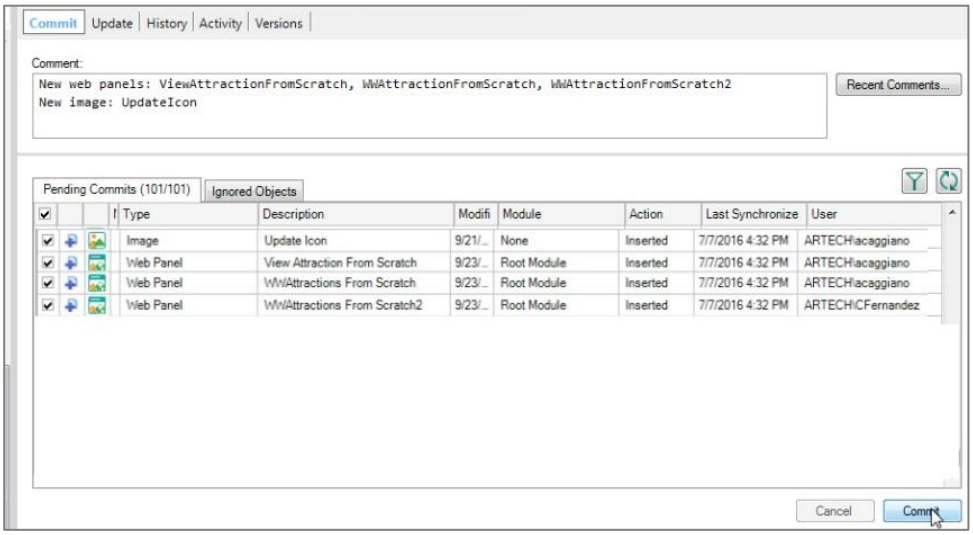


Esto coincide con lo que veíamos en ejecución. Además veamos que tenemos la propiedad Visible para poder ocultar algún control para un tamaño de pantalla determinado.

Así jugaríamos con los controles para que se pueda autoajustar lo que se muestra en ejecución de acuerdo al tamaño de pantalla en la que se despliegue el form en cada oportunidad.

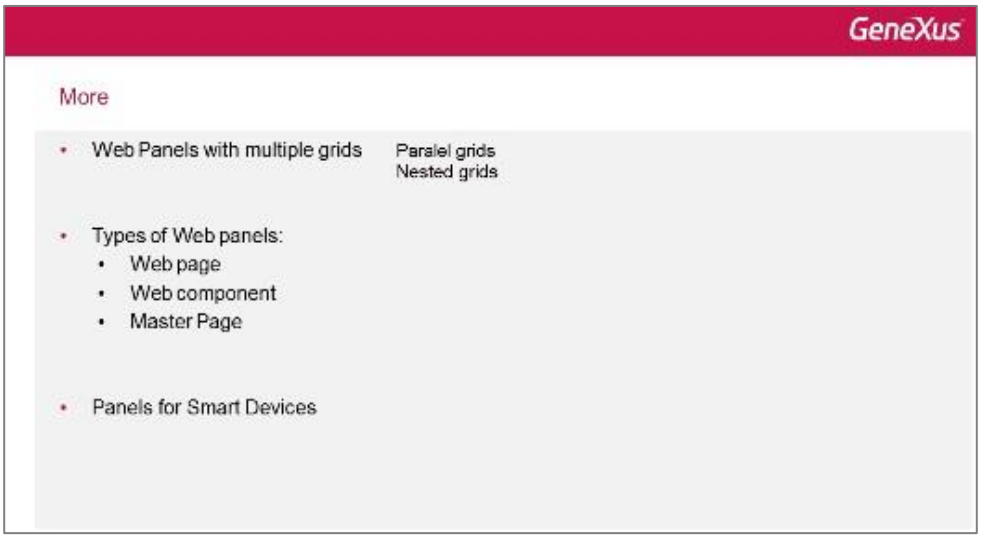
Esto se conoce como Responsive Web Design o Diseño Web Responsivo. Aquí solamente lo dejamos introducido.

Subamos todo lo hecho a GeneXus Server.



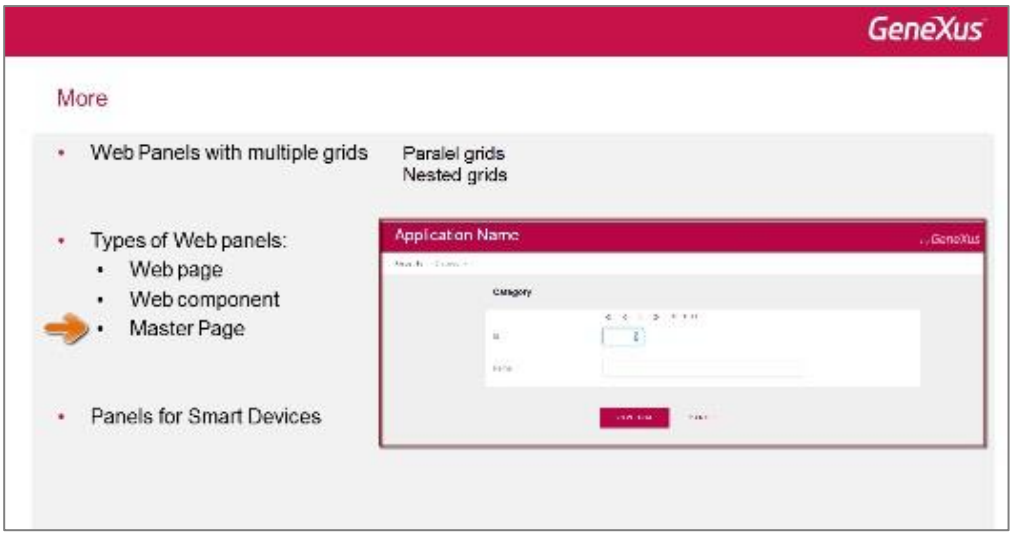
Hay mucho más para saber sobre web panels, por ejemplo cuando el web panel no tiene ningún grid o tiene uno, ¿dónde exactamente busca GeneXus la aparición de atributos para saber si asocia una tabla base implícita o no? Y de aparecer atributos en esos lugares ¿qué criterio deben cumplir? Aunque podemos imaginar que serán los mismos que para un For each: pertenecer a la tabla extendida.

Es posible implementar web panels con múltiples grids: así sean paralelos o anidados, es análogo a tener for each paralelos o anidados.



Esxisten tres tipos de web panels. En estos videos hemos visto únicamente el tipo web page, pero existen web panels que se pueden definir como componentes para el caso que una parte de una página se necesite repetir en múltiples web panels. Al definirla como componente se puede insertar en otros objetos con pantalla web.

Por otro lado tenemos web panels que se definen como master pages, es decir páginas maestras. Toda KB se crea con páginas maestras, que son el marco dentro del cual se cargan todas las páginas que se ejecutan. Aquí podemos ver esta parte de arriba de la transacción que es parte de la master page de la aplicación. La pantalla de la transacción se abre en el área destinada para ello en la master page.



Por último mencionemos que para el desarrollo de aplicaciones móviles para dispositivos inteligentes, smart devices, contamos también con paneles. Se llaman Panels for Smart Devices. En otro video veremos una introducción a este tipo de aplicaciones.