Pantallas interactivas

Más sobre Web Panel

GeneXus 15

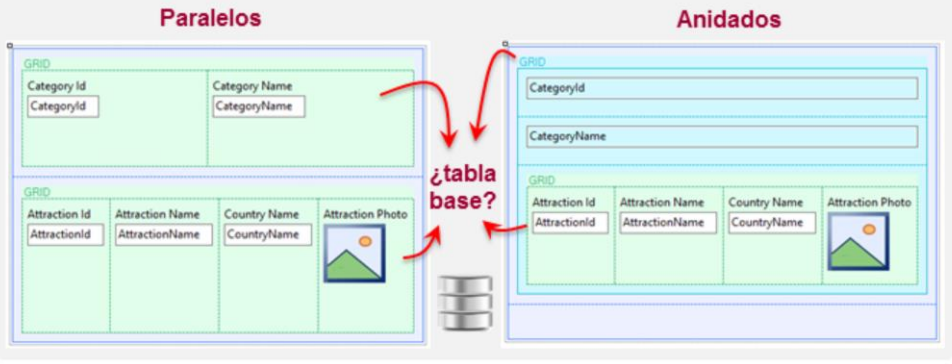
Tablas base

GeneXus

Habíamos visto

- Web panel **sin** grid o con **un** grid
  - ¿Tabla base automática?

- Web panels con múltiples grids



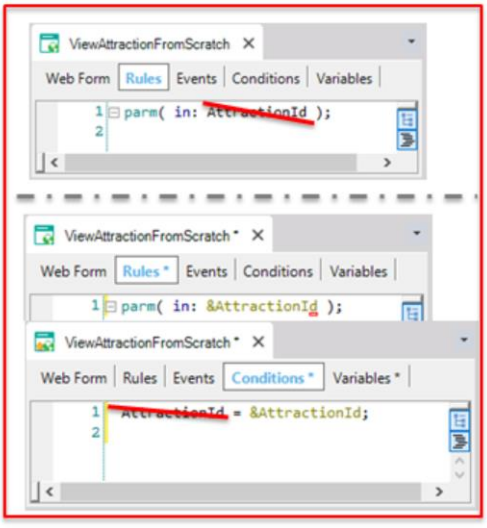
Habíamos visto el caso de Web Panel con atributos “sueltos” en el form, sin grid. También el de un Web Panel con un grid con atributos y también sin atributos.
Y habíamos dejado planteada la pregunta: cuando el web panel no tiene ningún grid o tiene uno, ¿dónde exactamente busca GeneXus la aparición de atributos para determinar si asocia una tabla base implícita o no al Web Panel? Y, de aparecer atributos en esos lugares, ¿qué criterios deben cumplir?

Por otro lado, habíamos visto que se pueden incluir múltiples grids en un Web Panel, tanto en forma paralela como anidada. Y otra vez, esos grids podrán cada uno tener o no tener tabla base asociada.

GeneXus

Web Panel **sin** grid

- No hay grid → no hay propiedad Base Transaction



- Atributos en el **form** (visibles u ocultos)
- Atributos en **eventos** (fuera de todo For each)

Tabla base: mínima tabla extendida

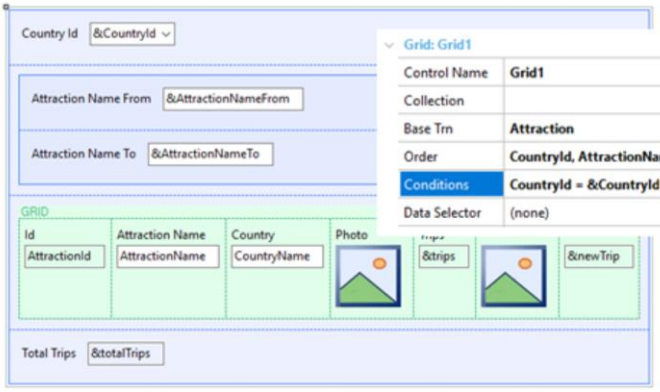
Cuando el web panel no tiene ningún grid, pero tiene atributos ya sea en el form (tanto visibles como ocultos) o en eventos (siempre y cuando estén fuera de un comando for each), el web panel tendrá tabla base.
¿Cómo se determina? De la misma manera que un for each al que no le especifiquemos transacción base: eligiendo la mínima tabla extendida que contenga a todos los atributos mencionados donde indicamos.

El o los atributos que se especifiquen en la regla parm, o en las Conditions generales (es decir, la solapa de Conditions) no serán tenidos en cuenta a la hora de determinar la tabla base. Se utilizarán como filtros LUEGO de que la tabla base se haya determinado.

GeneXus

Web Panel con **un** grid

- Hay un grid → tabla base del web panel = tabla base del grid



- Propiedad **Base Transaction**
- Atributos en el **form (grid y fuera del grid**, visibles u ocultos)
- **Order** del grid
- **Conditions** del grid
- **Data Selector** del grid
- Atributos en **eventos** (fuera de todo For each)

Tabla base: la asociada a la **Transacción Base** si hay definida
mínima tabla extendida en caso contrario

Cuando el web panel tiene un grid, entonces para determinar tabla base GeneXus observa lo que se indica arriba.

Si el grid tiene configurada Transacción Base, entonces su tabla base será la tabla base del grid/web panel. Los atributos mencionados en los lugares que se indica deberán pertenecer a su tabla extendida (exactamente igual que en el caso de un for each).

Si el grid no tiene configurada Transacción Base, entonces se determina la tabla base como la correspondiente a la mínima tabla extendida que contiene a todos los atributos de los lugares mencionados.

NO participan los atributos de las Conditions generales (las de la solapa Conditions) ni los de la regla Parm. Ni los atributos que se encuentren dentro de un for each.

Obsérvese que al grid se le puede aplicar un DataSelector para filtrar y ordenar su información, al igual que lo hacíamos en el For each (y también en un Grupo de Data Provider).

Web Panel con **varios** grids

- Hay dos grid paralelos → cada grid puede tener (o no) tabla base
- Existirán eventos Refresh y Load para cada grid. No existirá evento Load genérico.

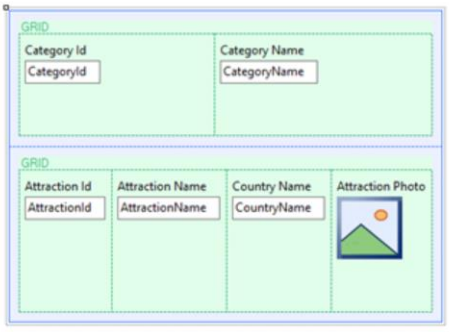


Tabla Base de cada grid:

- Propiedad **Base Transaction**
- Atributos en el **grid** (visibles u ocultos)
- **Order** del grid
- **Conditions** del grid
- **Data Selector** del grid

- Atributos en **evento Load** del grid (fuera de todo For each)

```
Event Grid1.Load
Endevent
```

```
Event Grid2.Load
Endevent
```

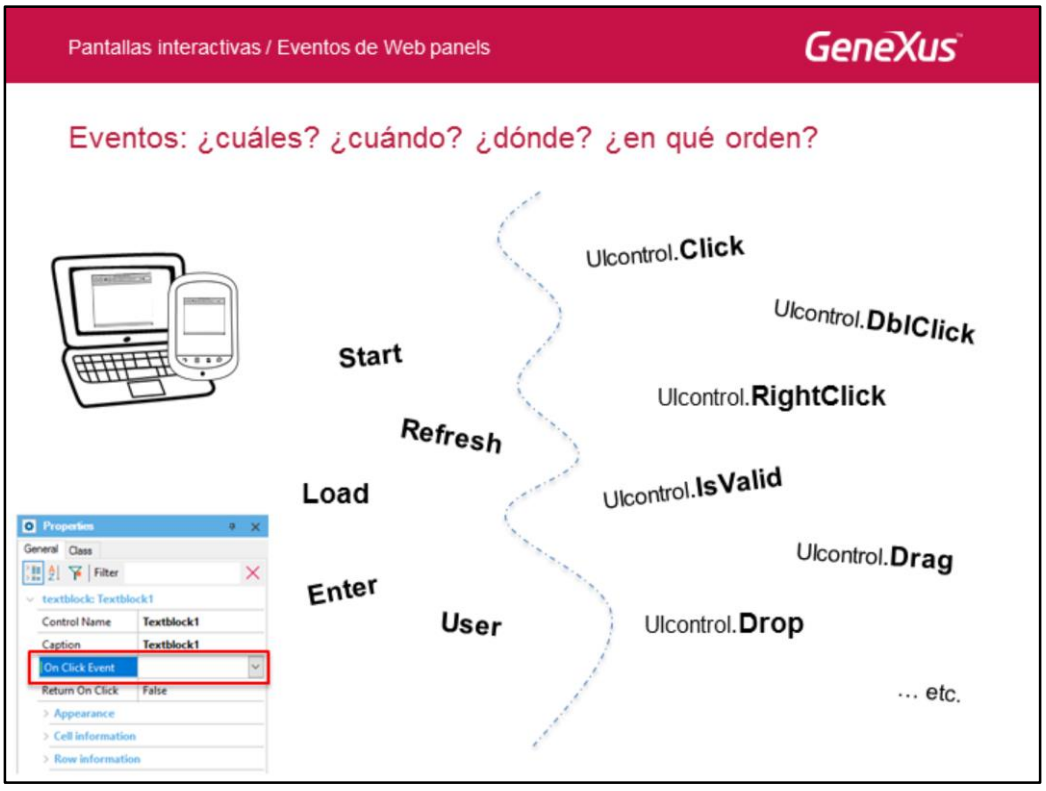
Cuando hay más de un grid en un web panel, **la tabla base de cada grid** se determina **considerando exclusivamente los atributos mencionados arriba**.

Cada grid tendrá su evento Load y la sintaxis se muestra arriba. No puede usarse el evento Load genérico, dado que no se sabría de cuál grid se trata.

A diferencia del caso de un único grid, los atributos que estén fuera de for eachs en cualquier evento que no sea "su" Load, no participarán de la determinación de las tablas base. Pero deberán cumplir que pertenezcan a la tabla extendida de alguno de los grids. De no ser así, GeneXus lo advertirá en el listado de navegación.

De existir atributos en la parte fija, la tabla base del primer grid en el form se determina tomando en cuenta los atributos sueltos, y los demás grids sin tomarlos en cuenta.
Si desea ver más sobre este caso especial acceda a nuestro wiki:
http://wiki.genexus.com/commwiki/servlet/wiki?6105,Determining+the+Base+Table+for+Each+Grid+in+a+Web+Panel

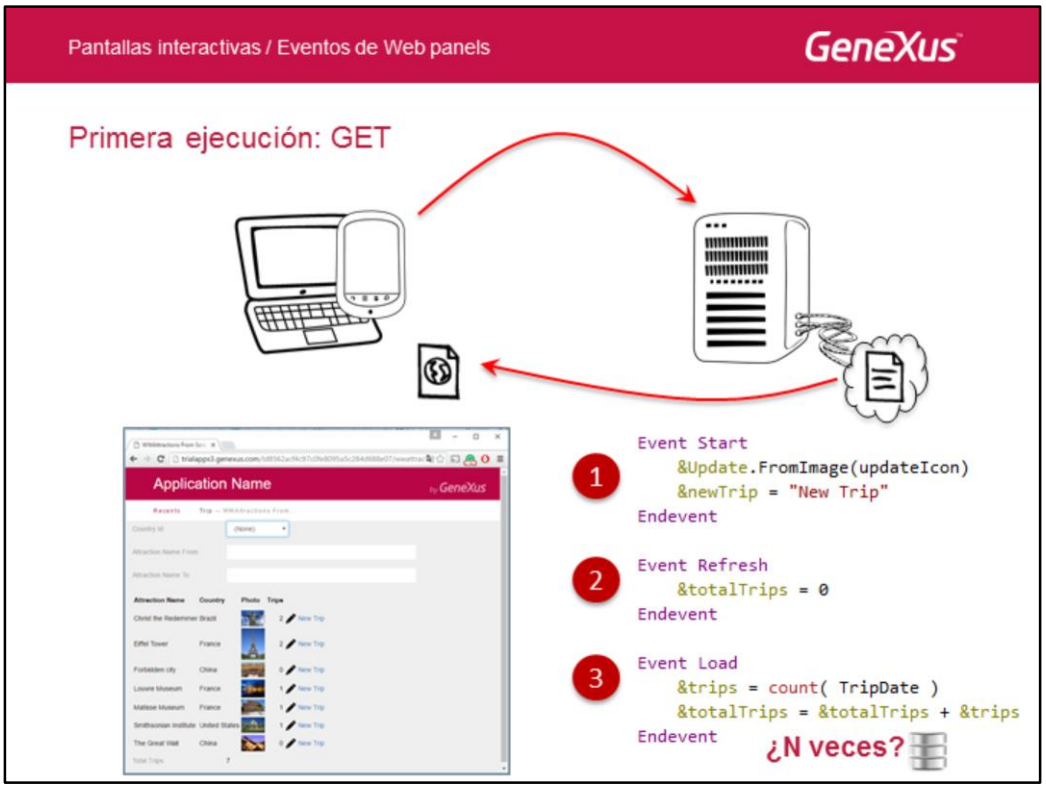
Eventos



Ya hemos visto y programado diferentes eventos en Web panels (por ejemplo el click, asociado a controles incluidos en el form, pero según el control se pueden programar también el doble click, botón derecho, etc.).

También hemos presentado y usado los eventos Start, Refresh y Load asociados al objeto web panel. El evento Enter es un evento del sistema que puede asociarse a cualquier control insertado en el form, y que también puede ejecutarse cuando el usuario presiona la tecla Enter. Hemos definido también para botones eventos de usuario, explícitamente. Es decir, le damos un nombre cualquiera a un evento y se lo asociamos a un botón o cualquier otro control (obsérvese la propiedad **On Click Event** que tienen los controles simples, en el form).

Veremos a continuación más detalladamente los eventos de los web panels, dónde se ejecuta cada uno (si en el servidor donde está instalada la aplicación o en el cliente –Browser–) y en qué orden.



En toda aplicación web tendremos una máquina –PC, notebook, o dispositivo inteligente– con conexión a internet con la que el usuario accederá a la aplicación a través de un navegador; y por otro lado el servidor, que será donde se encuentran todos los programas de la aplicación generados por GeneXus. Entre ellos, los correspondientes a los web panels (por ejemplo, el programa generado para el web panel que habíamos implementado desde cero).

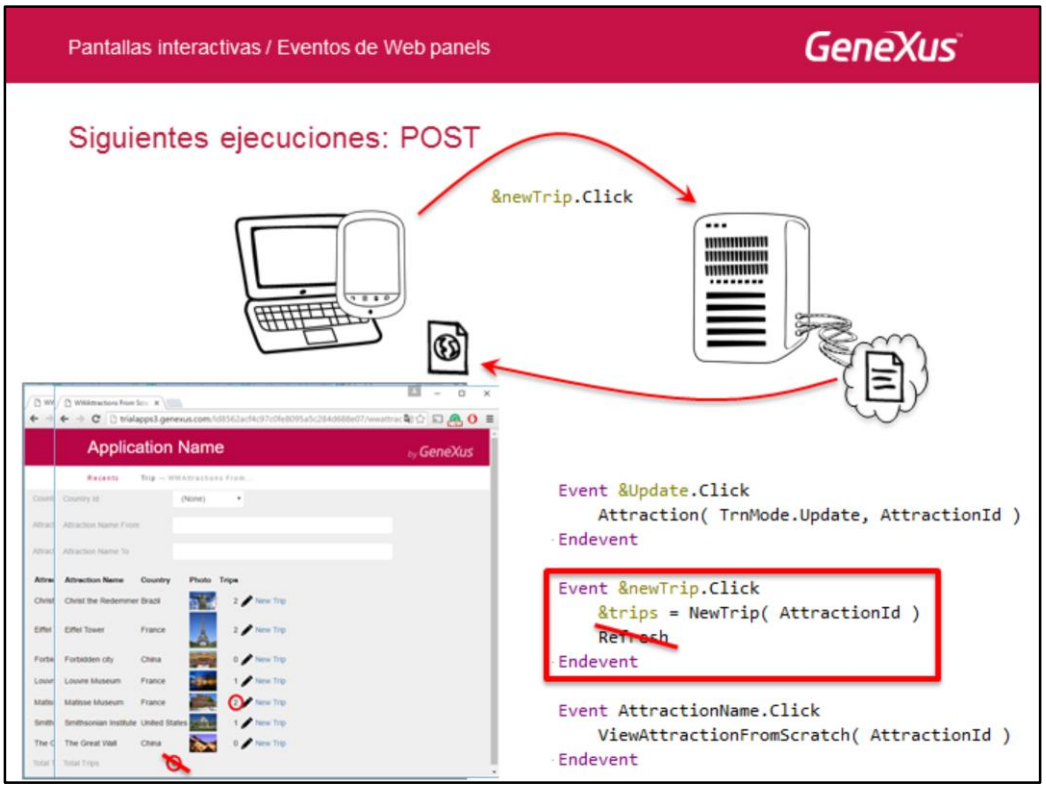
¿Qué sucede cuando invocamos al web panel desde el navegador **por primera vez**? Lo que se conoce como hacer un **GET**.

El cliente le pide al servidor que ejecute el programa asociado al web panel y le devuelva como resultado un archivo html que le indique al navegador cómo dibujar la pantalla (con qué datos, con qué formato, etc.).

¿Pero qué ejecuta el programa en el servidor para armar ese archivo html?

Los eventos Start, Refresh y Load, en ese orden. Si se trata de un web panel con tabla base, como es el caso, el evento Load se ejecutará N veces: una por cada registro que cumpla las condiciones.

Es importante tener claro que en esta primera ejecución del web panel (GET), el usuario no tiene tiempo de elegir algún valor en el Dynamic Combo Box, ni en las variables para filtrar por nombre de atracción. Es la primera llamada al objeto. Así que la variable &CountryId estará vacía… y la condition definida contempla que se filtre solamente si la variable &CountryId no está vacía (when not &CountryId.IsEmpty()). Lo mismo ocurrirá con las variables &AttractoinNameFrom y &AttractionNameTo. Por lo tanto, no se realizará ningún filtro y la página se mostrará con el grid cargado con todas las atracciones de todos los países.



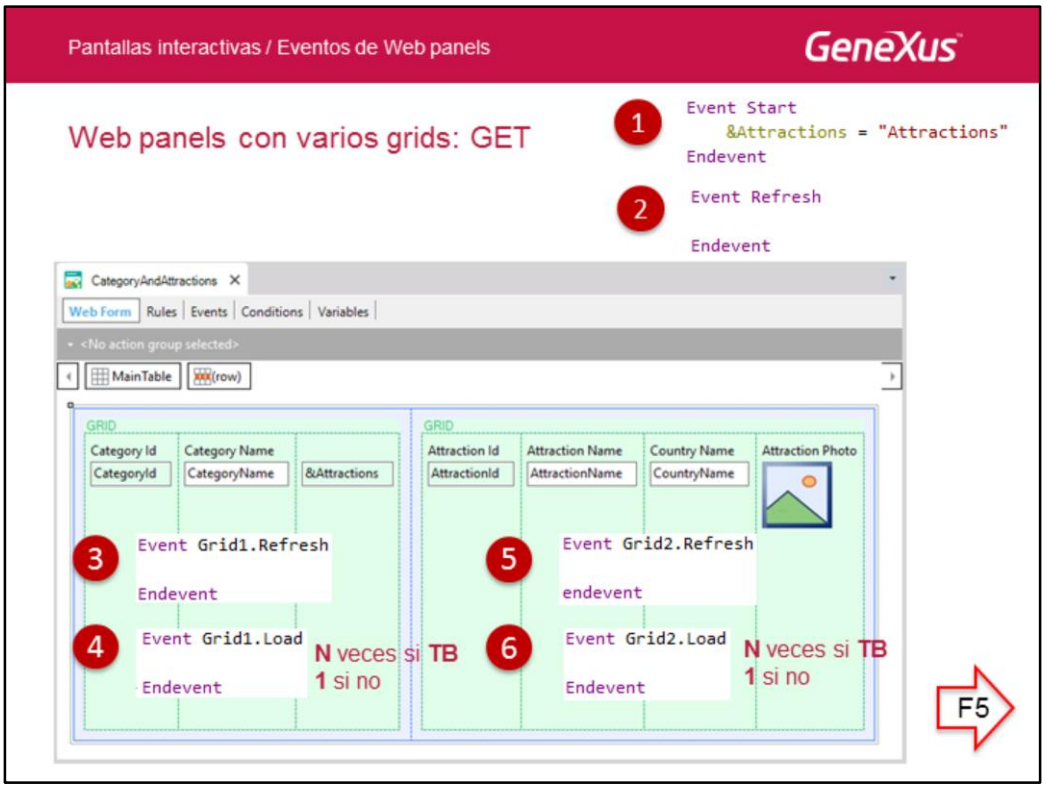
En forma general, un **POST** se produce cada vez que se efectúa alguna acción en el cliente, que requiere volver al servidor a ejecutar.

Acciones de este tipo pueden ser presionar la tecla Enter o algún botón o control asociado a un evento.

Cuando se ejecuta un POST, sucede lo siguiente:
- Se lee la información en pantalla
- Se ejecuta el evento de usuario que provocó el Post.

Por ejemplo, si el usuario hace clic sobre la opción del grid New Trip, se envía al servidor el AttractionId de la línea sobre la que se hizo clic. En el server se invoca al procedimiento NewTrip pasándole como parámetro ese valor de AttractionId. Como el valor devuelto por el procedimiento se carga en la variable del grid, &trips, esto provoca que automáticamente se vuelva a cargar la línea en el html resultante.

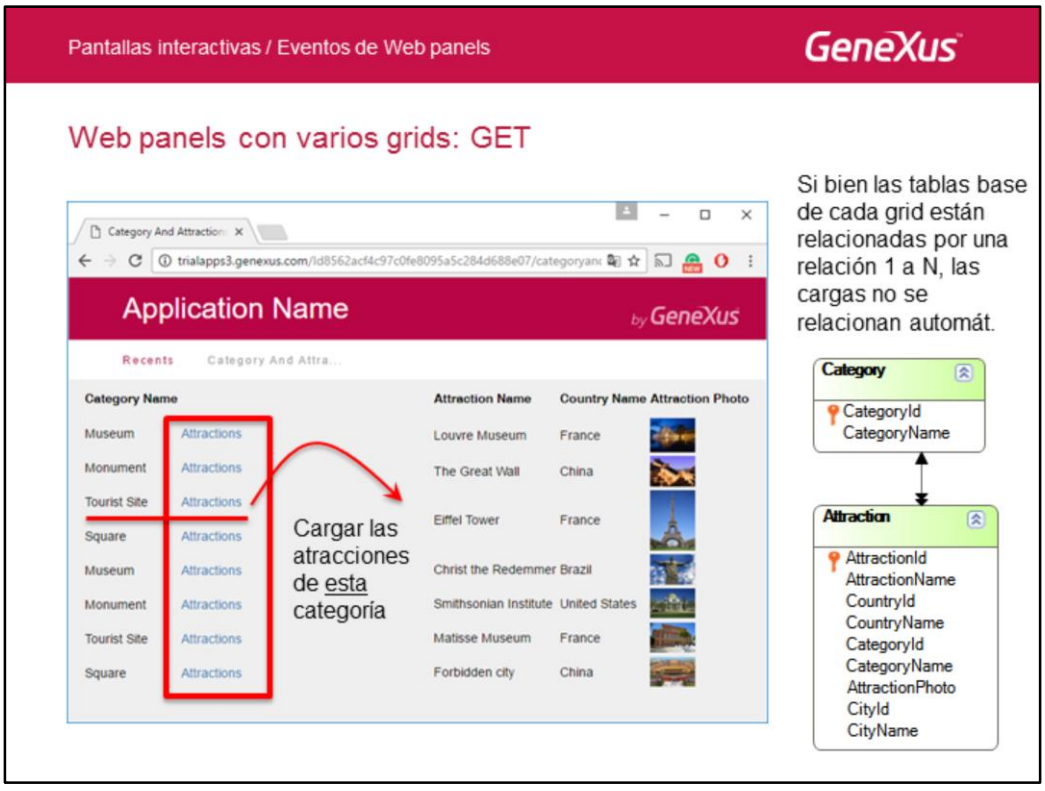
Si necesitamos que se vuelva a cargar todo (por ejemplo para que la variable &totalTrips se muestre con el valor que corresponde) entonces colocando el comando Refresh en el evento de usuario se vuelven a disparar Refresh y Load, como habíamos visto clases atrás. Esto se ejecuta en el Server, antes de armar el html que se devuelve al cliente.



Para Web panels con varios grids, el evento Refresh que ahora será genérico (del web panel como un todo), lanzará el evento Refresh y el Load de cada grid.

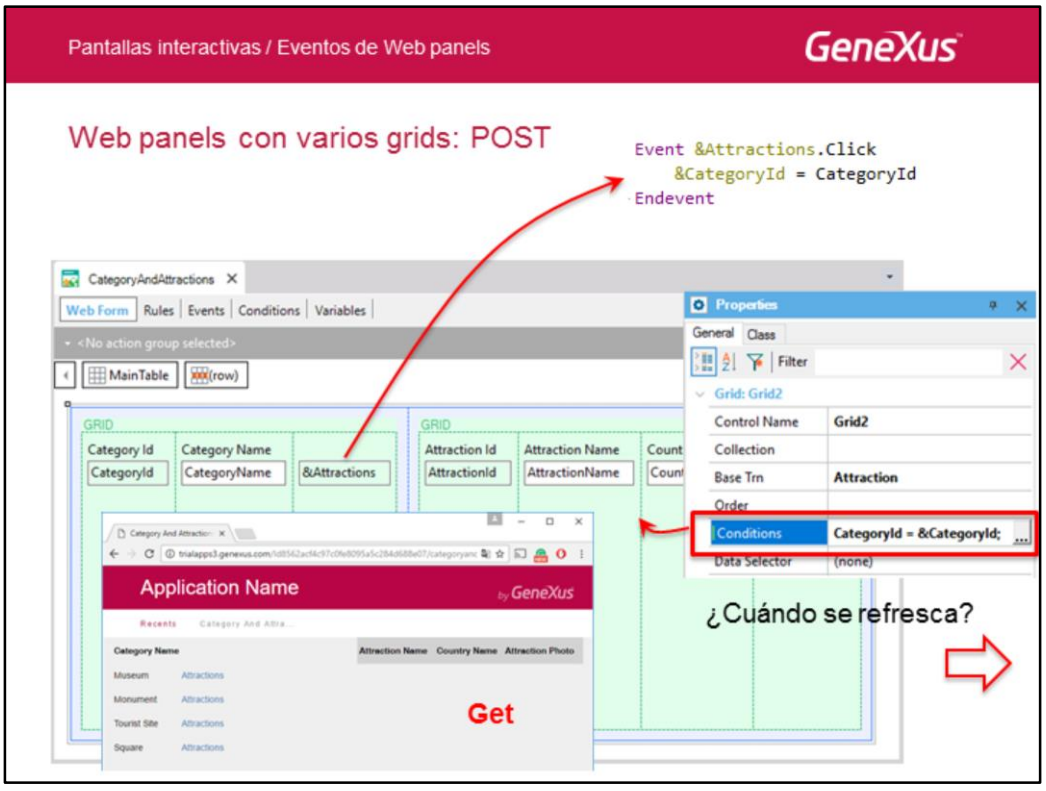
Dependiendo de si el grid tiene o no tabla base, el evento Load, como para el caso de un solo grid, se ejecutará N veces, una vez por cada registro de la tabla base a ser cargado como línea del grid; o una sola, si no tiene tabla base.

El orden de disparo de los eventos es el que presentamos arriba. Va a depender del orden en el que se encuentren los grids en la pantalla (de izquierda a derecha, de arriba abajo).



Para que en el grid de atracciones se muestren únicamente las de la categoría elegida en el grid de la izquierda, tendremos que especificar el filtro explícitamente.

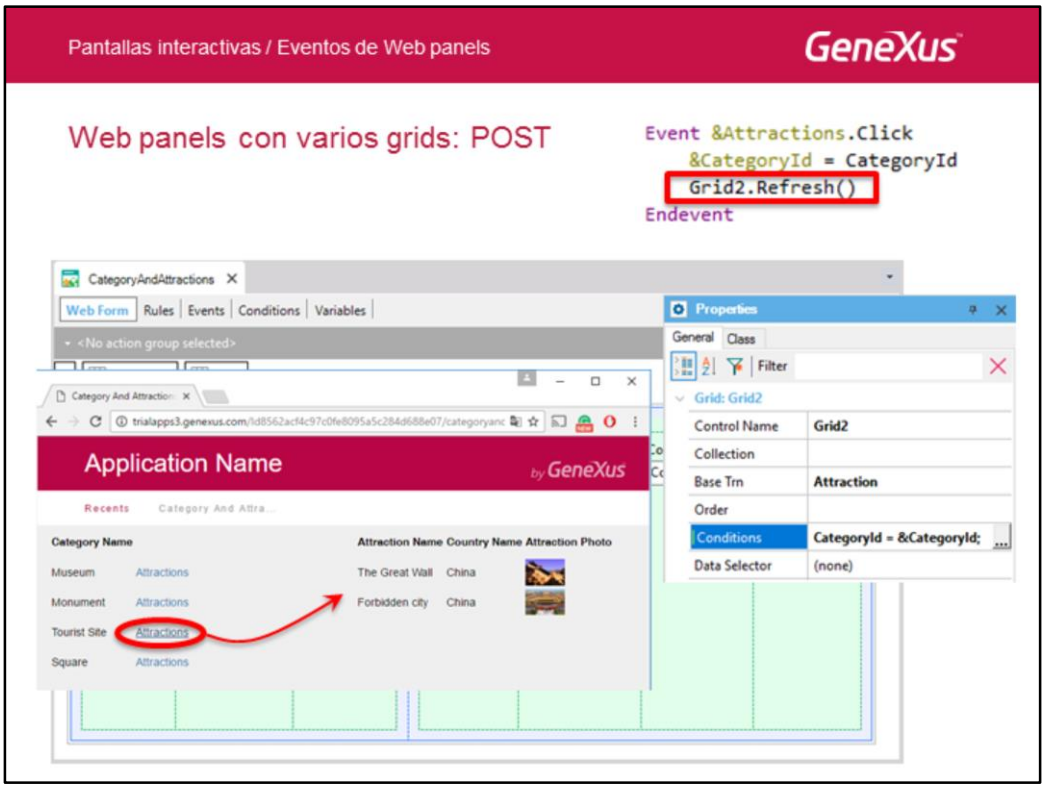
Para ello hemos agregado una columna con la variable &Attractions, de tipo character(15), con el texto Attractions (lo hemos especificado así en el evento Start, porque nos alcanza con que se defina cuando se abre el web panel. No cambiará luego). Así, cuando el usuario haga click sobre esa opción, debemos pedir que cargue (nuevamente) el grid de atracciones.



Para programar la carga del grid de atracciones a partir de la categoría elegida por el usuario en el otro grid, creamos una variable &CategoryId a la que le asignaremos valor cada vez que el usuario presione sobre "Attractions" para la línea deseada. Observemos que hemos especificado como condición para que una atracción sea cargada en el grid (Grid2) que su CategoryId coincida con el valor de la variable.

De este modo, en la primera ejecución ninguna atracción cumplirá la condición, porque para ese caso la variable &CategoryId estará vacía.

Luego, cuando el usuario haga clic sobre &Attractions se le da valor a la variable a partir del CategoryId de esa línea. Pero si no hacemos nada más, nunca se refrescará el Grid2. Tenemos que pedirle al grid2 que se refresque.



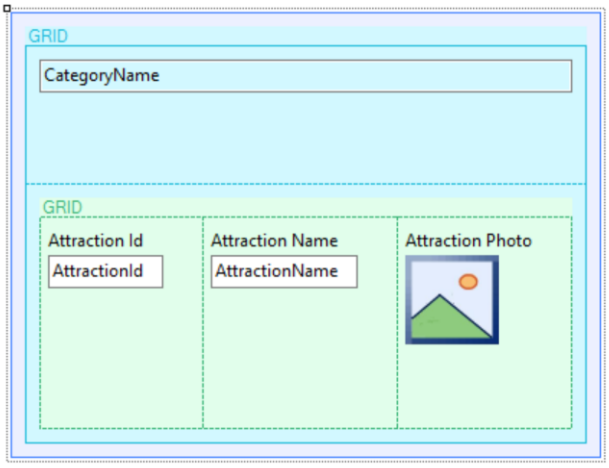
Y lo hacemos con el método Refresh de ese grid.

Para anidar grids se utilizan los grids Freestyle. Por ejemplo, en lugar de ver todas las categorías y cliqueando sobre una ver sus atracciones, podemos verlo todo anidado, es decir, por cada categoría, sus atracciones. Como si se tratara de un caso de for eachs anidados. Es análogo.

Aquí las navegaciones sí se relacionan.

GeneXus

Web panels con grids anidados (GET)

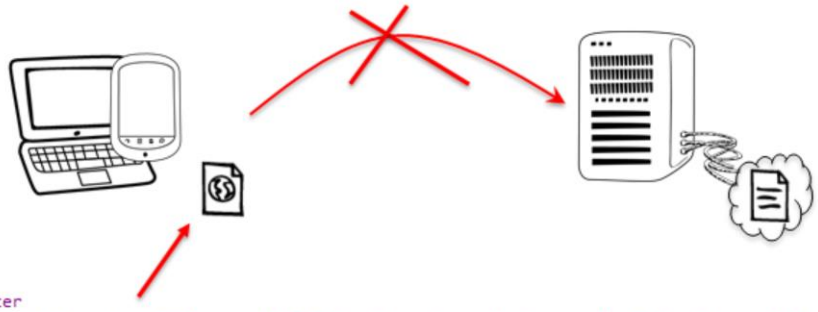


1. Start
2. Refresh
3. Grid1.Refresh
4. Grid1.Load
    5. Grid2.Refresh
    6. Grid2.Load

El orden de disparo de los eventos es el esperado. Primero Start del web panel, luego Refresh y luego Refresh del grid y Load. Si tiene tabla base, el Load se disparará N veces, una por cada línea. Si no, se disparará sólo una.
En cualquier caso, luego del Load del grid freestyle, se disparará Refresh y Load del Grid anidado. Otra vez, este segundo Load podrá ejecutarse una sola vez, o N, dependiendo de si tiene tabla base o no.

GeneXus

Algunos eventos pueden resolverse en el cliente, sin POST al Server



```
Event Enter
    Textblock1.Caption = "Please, click Attractions in order to see the Attractions of the Category"
endevent
```

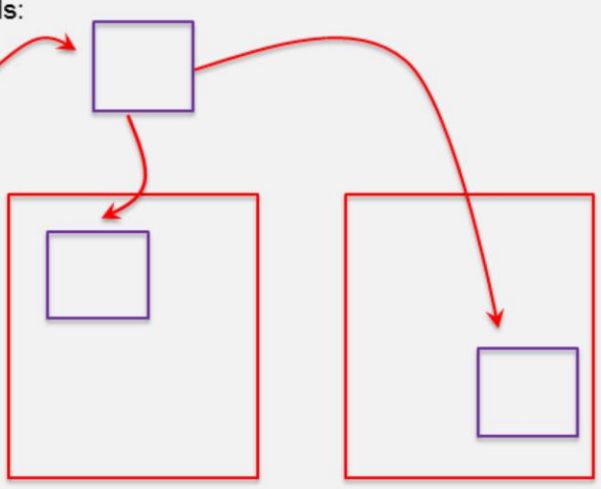
No toda acción efectuada por el usuario y asociada a un evento producirá un POST al servidor. Algunas se pueden resolver en el propio cliente.

Por ejemplo, si en un evento enter, de usuario o asociado a un control, se pone invisible un control, o se le cambia el Caption, el color, etc., eso se resuelve en el propio cliente.

Web components

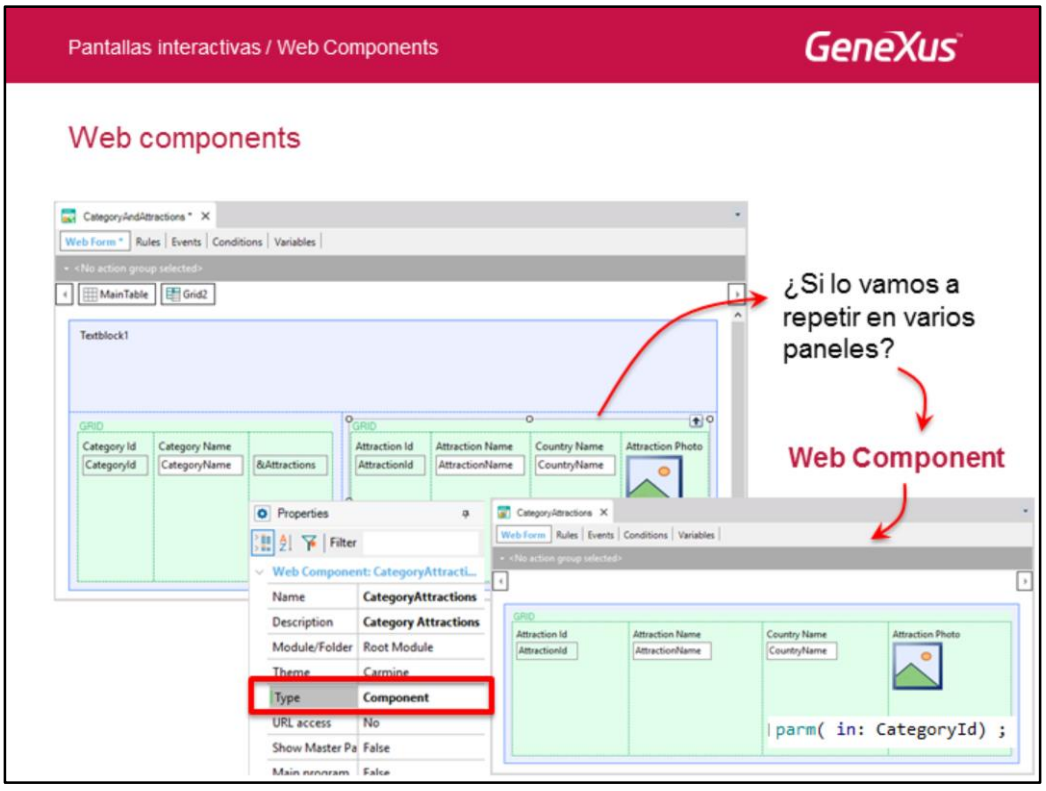
Habíamos visto

- Tipos de Web Panels:
  - Web page
  - Component
  - Master Page



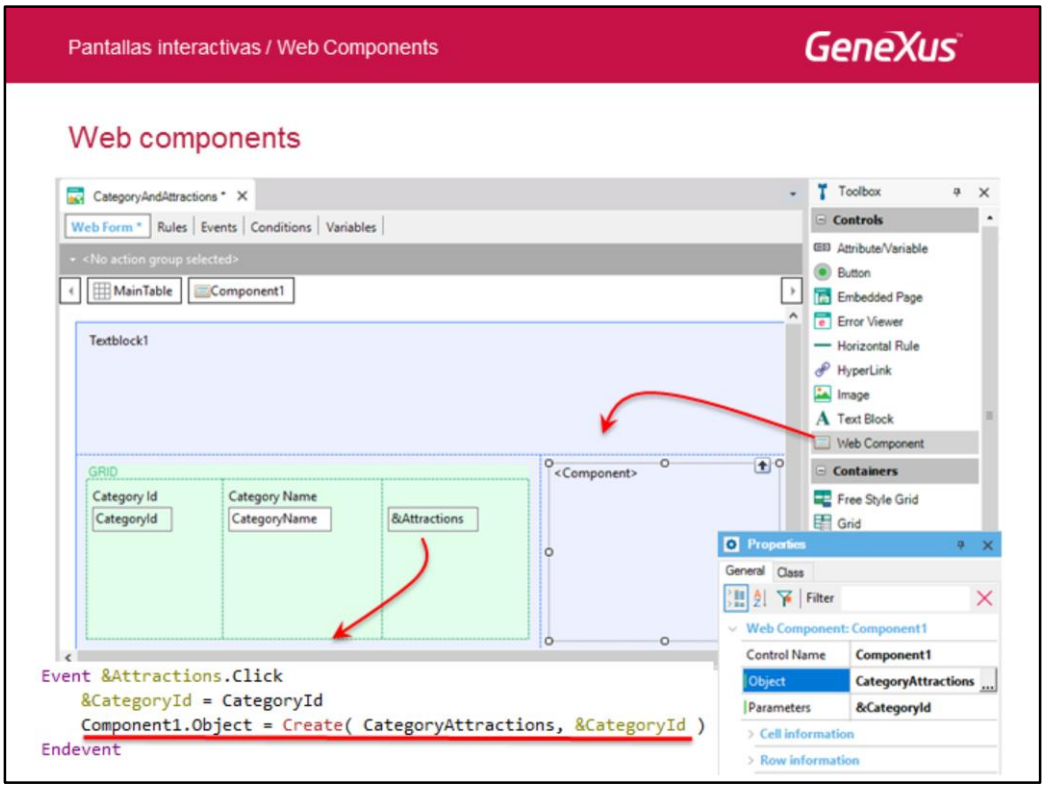
En otra clase habíamos visto que existen tres tipos de web panels.

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



Por ejemplo, supongamos que el grid que muestra las atracciones de la categoría elegida se repite en varios web panels.
Entonces será mejor tener un web panel de tipo Component con él y toda su programación (por ejemplo, hemos sustituido la condición que teníamos a nivel del grid, por la regla parm recibiendo en el atributo CategoryId, que como sabemos realiza un filtro automático).

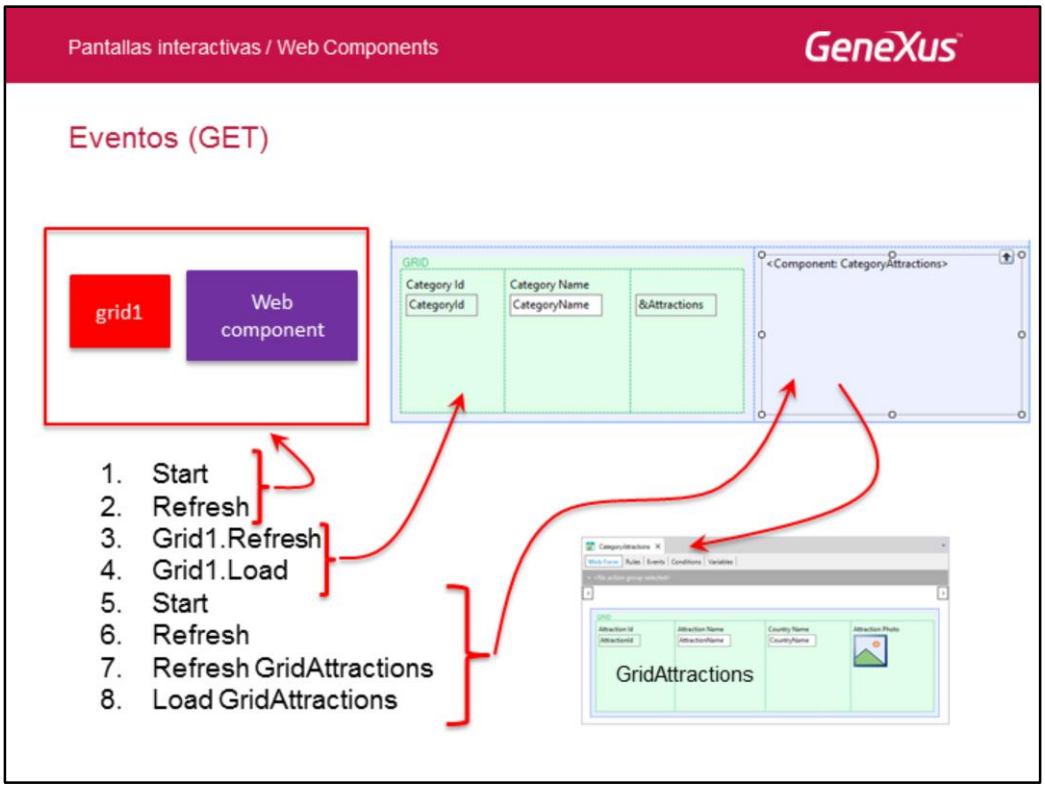
Luego…



En el web panel donde teníamos inicialmente el grid de atracción, colocamos en su lugar un control Web Component.
Sólo nos resta decirle qué objeto de tipo Web Component se cargará allí dentro.
Podemos hacerlo tanto por programación, como a través de las propiedades del control Component, como vemos arriba.
Allí le indicamos el objeto que se cargará, y el prámetro que le debemos enviar. En este caso, el valor de la variable &CategoryId.

Pero ahora debemos modificar también la programación del click del grid de categorías. Una vez que a la variable &CategoryId le damos el valor de la categoría de la línea, tenemos que lograr que el web component se vuelva a crear dentro del control, para que pueda recibir el valor de la variable por parámetro. Para ello usaremos la propiedad Object a nivel de programación, como se ve.

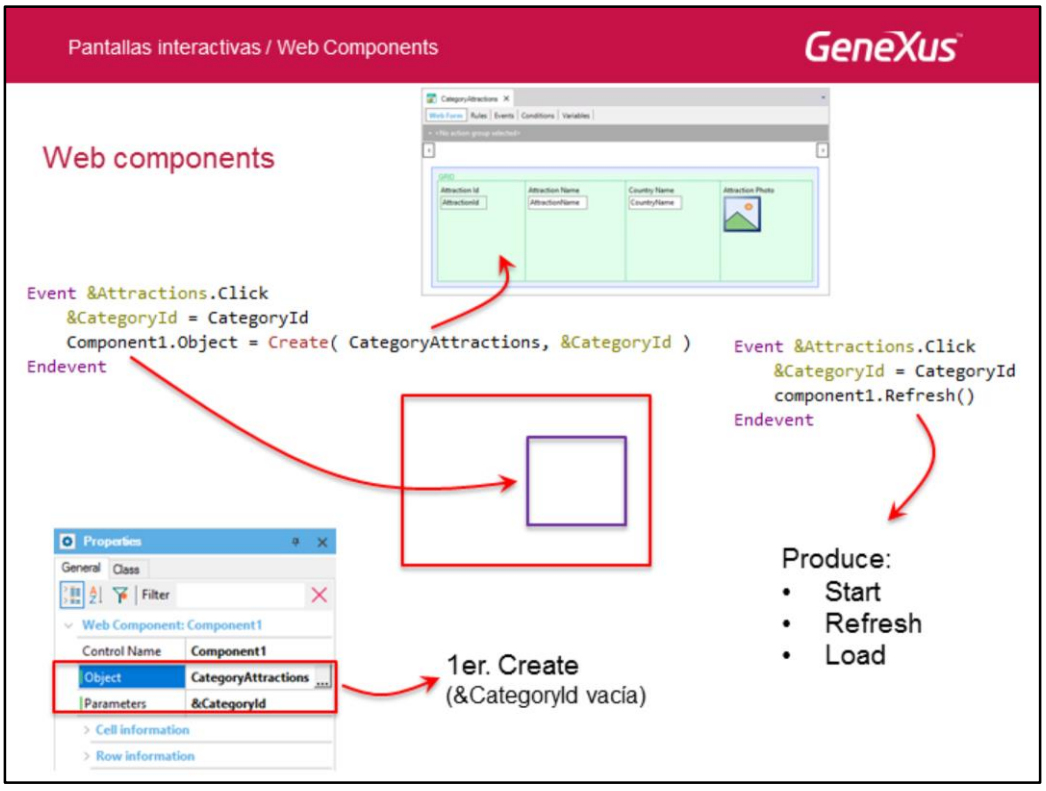
Existe el método Refresh() a nivel del componente. Lo veremos luego de ver los eventos que se ejecutan en el GET.



Si hay un web component dentro de un web panel, los eventos que se disparan y el orden son los esperables. Se dispara:

1. Evento Start del web panel
2. Evento Refresh general del web panel
3. Evento Refresh del grid1
4. Evento Load del grid1
5. Eventos del componente (Start, Refresh, Refresh y Load de cada grid que se encuentre)

Luego, si hay un evento de usuario o de un control a nivel del componente, se ejectua ese evento y se refresca únicamente la parte del componente que corresponda. Aquí podrá ver más información: http://wiki.genexus.com/commwiki/servlet/wiki?22472,Event+Execution+Scheme,



El método Refesh a nivel del Web component disparará los eventos Start, Refresh y Load del web component, y volverá a cargar toda su área de pantalla. ¿Por qué no lo usamos? Por el parámetro que queremos enviarle al objeto que creamos allí dentro. Es que el método Refresh enviará por parámetro el valor que tenía &CategoryId al momento del Create (que en este caso habrá sido cuando se hizo el GET, es decir, cuando se dibujó por primera vez la pantalla). En ese momento, la variable estaba vacía.

No entraremos en detalles aquí sobre esto.

GeneXus™

Videos	training.genexus.com
Documentación	wiki.genexus.com
Certificaciones	training.genexus.com/certificaciones