

Pantallas Interactivas

Más sobre Web Panels

GeneXus 16

Tablas base

Anteriormente vimos que con sólo colocar atributos en un grid, GeneXus entiende que debe ir a la base de datos a navegar la tabla correspondiente para recuperar la información, en forma análoga a un comando For Each.

Veamos los distintos casos en los que GeneXus puede determinar una tabla base a recorrer.

Habíamos visto

- Web panel sin grid o con un grid
¿Tabla base automática?
- Web panels con múltiples grids



Paralelos

The diagram shows a web panel with two separate grids. The top grid contains two input fields: 'Category Id' and 'Category Name'. The bottom grid contains four input fields: 'Attraction Id', 'Attraction Name', 'Country Name', and 'Attraction Photo' (represented by a picture icon).

Anidados

The diagram shows a web panel with a single grid containing two input fields: 'Category Id' and 'Category Name'. Below these is another grid containing four input fields: 'Attraction Id', 'Attraction Name', 'Country Name', and 'Attraction Photo' (represented by a picture icon).

¿tabla base?



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.

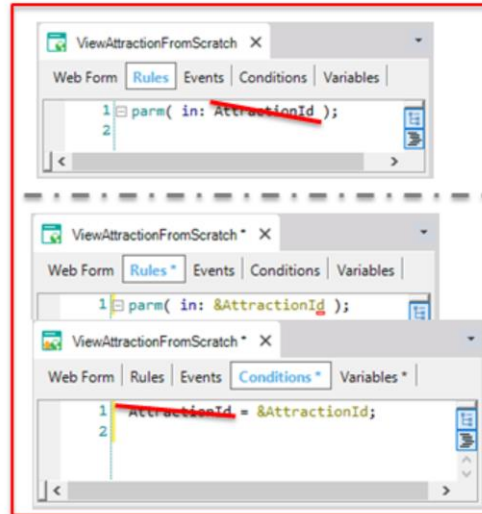
Web Panels grid



- 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.

Web Panel con un grid



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

The screenshot shows a web panel design with a grid. The grid has columns for Id (AttractionId), Attraction Name (AttractionName), Country (CountryName), Photo, and a group of controls for Trips (Trips, &trips, &newTrip). Below the grid is a control for Total Trips (&totalTrips). To the right, the configuration properties for 'Grid: Grid1' are shown:

Control Name	Grid1
Collection	
Base Trn	Attraction
Order	CountryId, AttractionNam...
Conditions	CountryId = &CountryId ...
Data Selector	(none)

- 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.

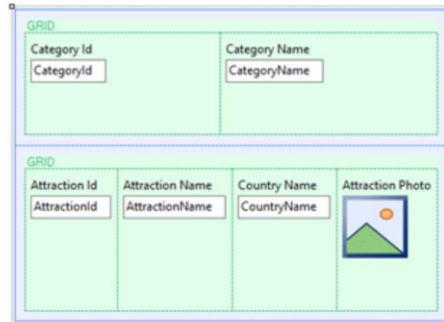


Tabla Base de cada grid:

- 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)



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.

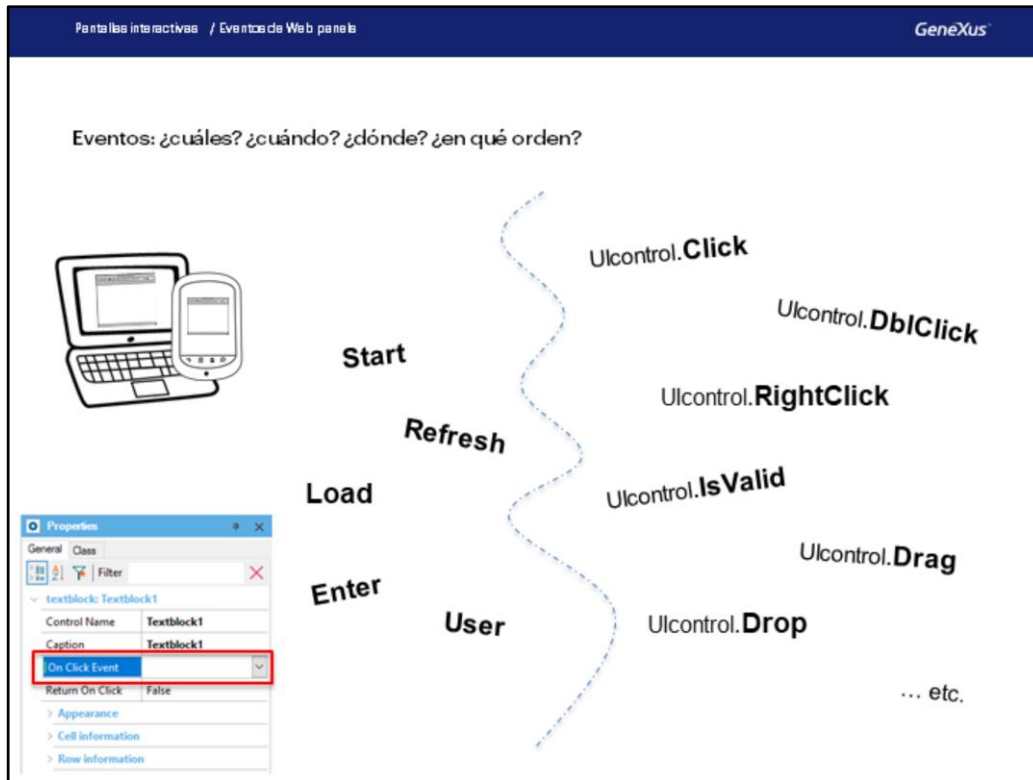
A diferencia del caso de un único grid, los atributos que estén fuera de for eachs en cualquier evento que no sea "su" Load, no participarán de la determinación de las tablas base. Pero deberán cumplir que pertenezcan a la tabla extendida de alguno de los grids. De no ser así, GeneXus lo advertirá en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

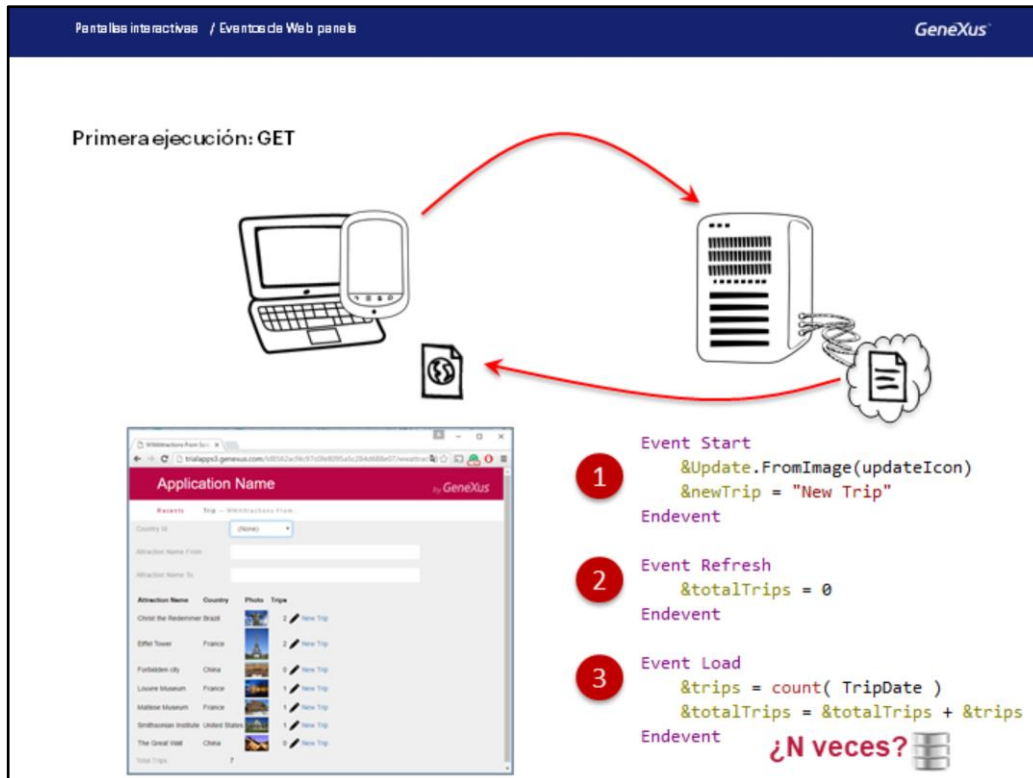
Eventos



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).

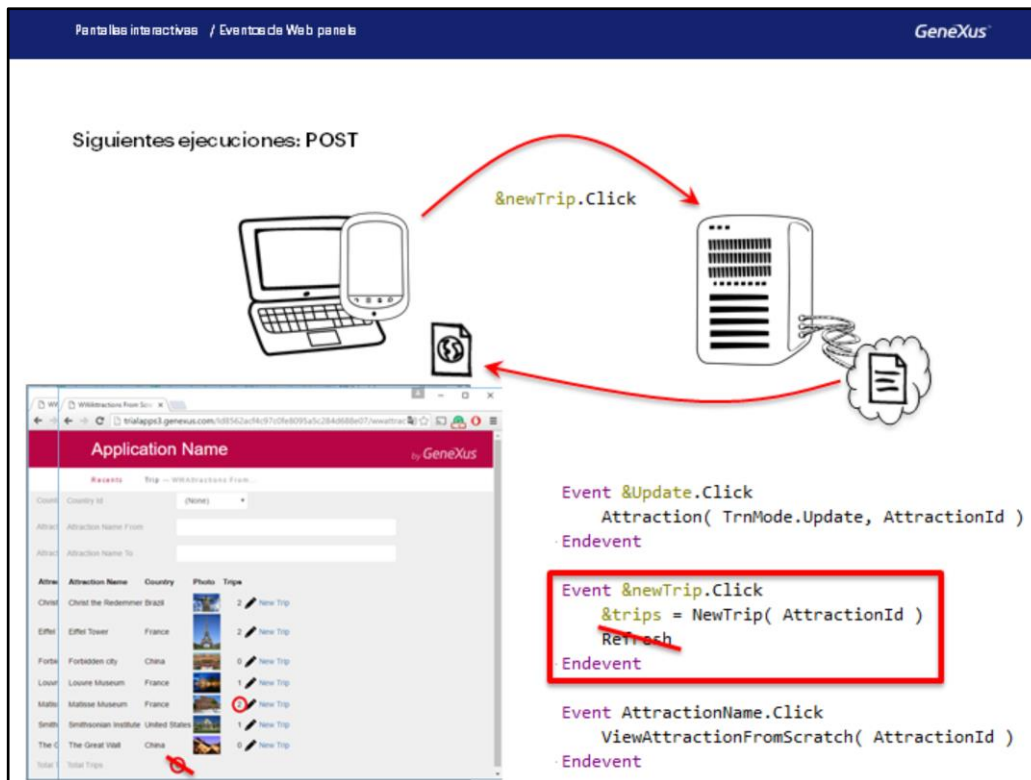
¿Qué sucede cuando invocamos al web panel desde el navegador **por primera vez**? Lo que se conoce como hacer un **GET**.

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 &AttractionNameFrom 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.

Pantallas interactivas / Eventos de Web panels
GeneXus

Web panels con varios grids: GET

1 Event Start
 &Attractions = "Attractions"
Endevent

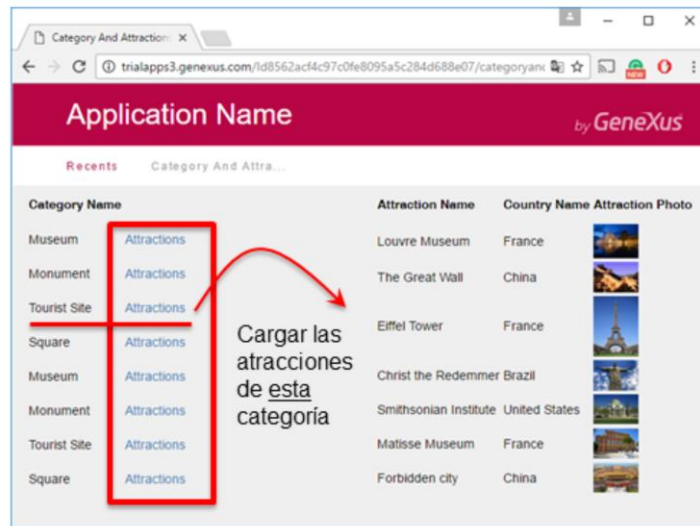
2 Event Refresh
Endevent

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.

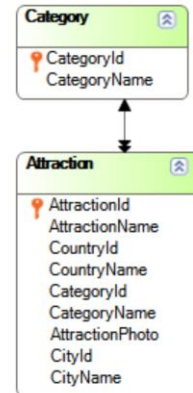
Dependiendo de si el grid tiene o no tabla base, el evento Load, como para el caso de un solo grid, se ejecutará N veces, una vez por cada registro de la tabla base a ser cargado como línea del grid; o una sola, si no tiene tabla base.

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).

Web panels con varios grids: GET



Si bien las tablas base de cada grid están relacionadas por una relación 1 a N, las cargas no se relacionan automáticamente.



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.

Pantallas interactivas / Eventos de Web panels GeneXus

Web panels con varios grids: POST

```

Event &Attractions.Click
  &CategoryId = CategoryId
Endevent
  
```

Properties window for Grid2:

Control Name	Grid2
Collection	
Base Trn	Attraction
Order	
Conditions	CategoryId = &CategoryId;
Data Selector	(none)

¿Cuándo se refresca?

Get

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.

Pantallas interactivas / Eventos de Web panels

GeneXus

Web panels con varios grids: POST

```
Event &Attractions.Click  
  &CategoryId = CategoryId  
  Grid2.Refresh()  
Endevent
```

CategoryAndAttractions X

Web Form Rules Events Conditions Variables

<No action group selected>

Events

Category And Attractions X

trialapps3.geneXus.com/ld8562ac4c97c0fe8095a5c284d688e07/categoryanc

Application Name by GeneXus

Recents Category And Attra...

Category Name	Attraction Name	Country Name	Attraction Photo	
Museum	Attractions	The Great Wall	China	
Monument	Attractions	Forbidden city	China	
Tourist Site	Attractions			
Square	Attractions			

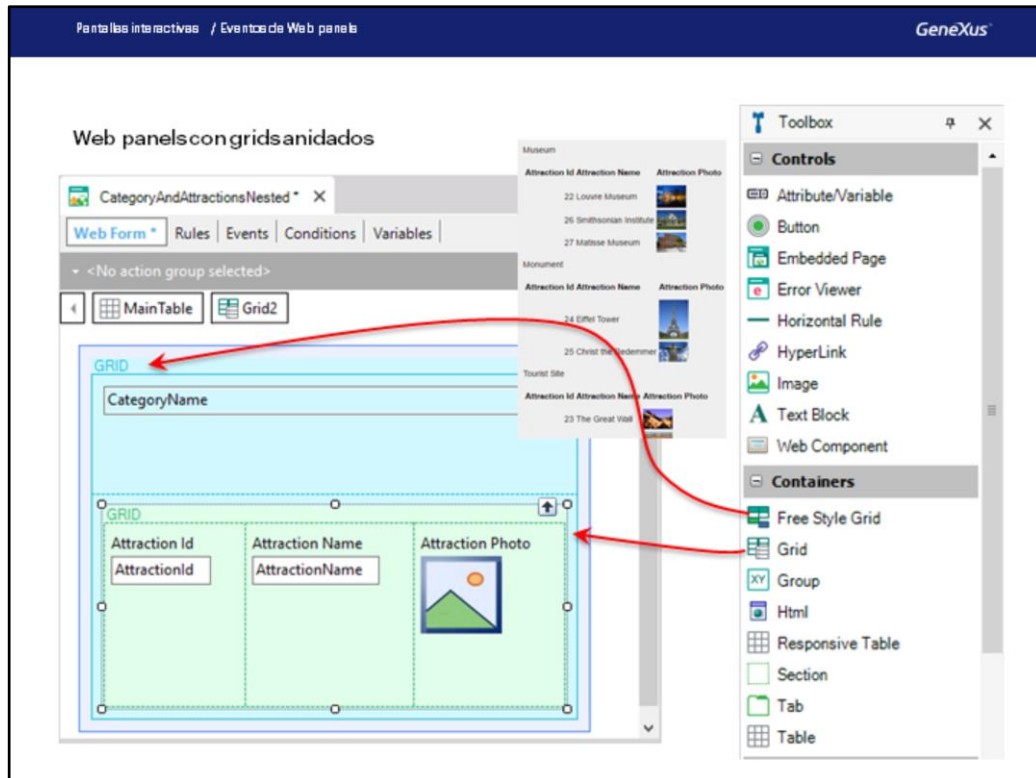
Properties

General Class

Grid: Grid2

Control Name	Grid2
Collection	
Base Trn	Attraction
Order	
Conditions	CategoryId = &CategoryId; ...
Data Selector	(none)

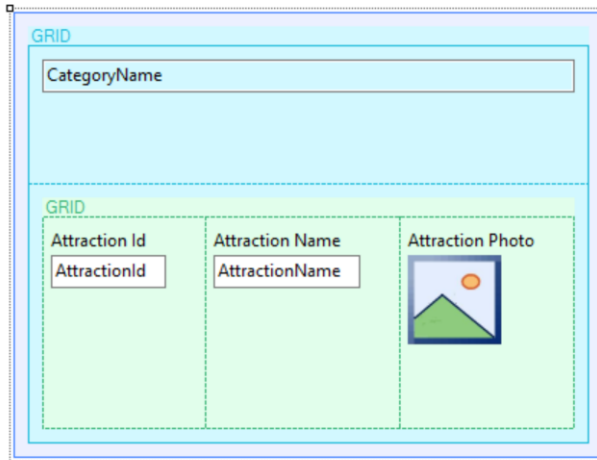
Y lo hacemos con el método Refresh de ese grid.



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.

Web panels con grids anidados (GET)

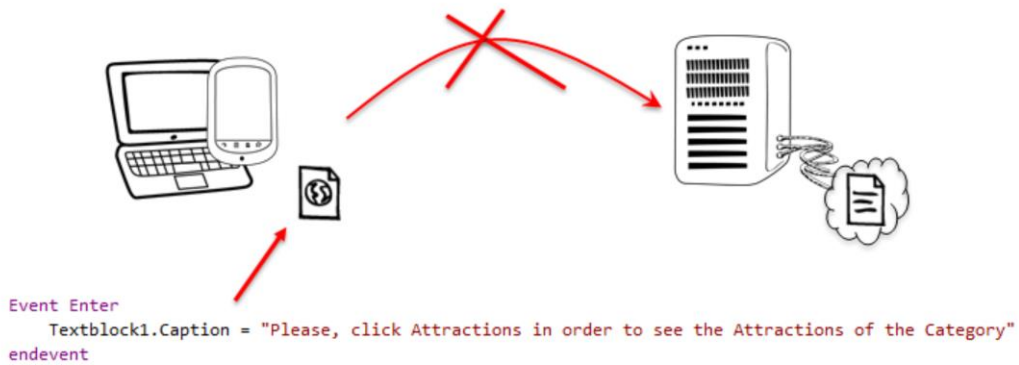


1. Start
2. Refresh
3. Grid1.Refresh
4. Grid1.Load
5. Grid2.Refresh
6. Grid2.Load

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.

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



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.

Selección múltiple en un grid

En muchas ocasiones debemos trabajar con un conjunto seleccionado de elementos, para hacer algo con ellos después.

Dado que un grid nos permite visualizar muchos elementos, es natural que queramos seleccionar varios elementos de un grid.

A continuación, veremos cómo podemos hacer para hacer una selección múltiple en un grid y cómo podemos recorrer los elementos para procesarlos.

Selección de una única fila de un grid






Application Name by GeneXus

Recents Attractions

× HIDE FILTERS Attractions + INSERT

Ordered By : Name

COUNTRY NAME

Id	Name	Country N...	Category N...	Photo	City Name	Address
4	Christ the Redeemer	Brazil	Monument		Rio de Janeiro	UPDATE DELETE
3	Eiffel Tower	France	Monument		Paris	UPDATE DELETE
7	Forbidden City	China	Tourist site		Beijing	UPDATE DELETE
1	Louvre Museum	France	Museum		Paris	UPDATE DELETE
6	Matisse Museum	France	Museum		Nice	UPDATE DELETE

Supongamos por ejemplo que tenemos una grilla con datos de atracciones.

Si queremos trabajar con una atracción en particular, se hace necesario marcar una fila del grid como seleccionada, de forma de poder acceder a los valores de cada columna para dicha fila.

Para seleccionar una única fila de un grid: AllowSelection=True

Behavior

Sortable	True
Allow Drop	False
Allow Drag	False
Notify Context Change	False
Allow Collapsing	False
Allow Selection	True
Allow Hovering	True



En una aplicación Web, si queremos marcar una fila del grid como seleccionada, vamos a las propiedades del grid y en la propiedad AllowSelection seleccionamos el valor True.

Con este valor, también se disponibiliza la propiedad Allow Hovering, que por defecto toma el valor True. Esto hace que al pasar el mouse sobre las filas, las mismas se van pintando de un color.

Con AllowSelection en True, al hacer clic sobre una atracción, la fila queda marcada con otro color. Estos colores pueden configurarse en el objeto Theme, en las clases asignadas a las propiedades Selected Row Class y Hover Row Class de la clase Grid.

Cómo seleccionar múltiples filas de un grid

The diagram illustrates the configuration for selecting multiple rows in a grid. It shows a visual representation of a grid on a smart device (left) and a web browser (right). The smart device grid has a single column labeled 'AttractionName'. The web grid has four columns: 'Selected', 'Id', 'Name', and 'Photo'. A red arrow points from the 'Selected' column in the web grid to the 'Selected' variable in the 'Variables' section of the 'Grid1' properties table.

Name	Type
Variables	
Standard Variables	
Selected	Boolean

Property	Value
Control Name	Grid1
Collection	
Default Action	<default>
Selection Type	Platform Default
Enable Multiple Selection	True
Pull To Refresh	False
Inverse Loading	False
Default Selected Item Layout	(none)
Show Selector	Always
Selection Flag	
Selection Flag Field Specifier	

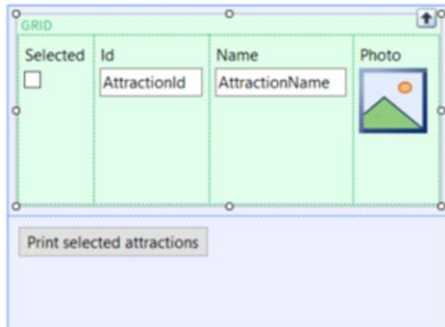
Supongamos que la agencia de viajes requiere que a partir de una grilla de atracciones, seleccionemos aquellas que son de nuestro interés e imprimamos un listado con las atracciones seleccionadas.

En una aplicación Web, podemos utilizar una propiedad del grid para seleccionar una única fila, como ya hemos visto, pero no tenemos propiedades que nos permitan indicar que queremos seleccionar varias filas a la vez.

En cambio en una aplicación para Smart Devices sí es posible, utilizando la propiedad **Enable Multiple Selection** del grid. Para poder hacer que aparezca el checkbox que nos permite seleccionar la fila, utilizamos la propiedad **Show Selector**.

Para lo que nos pide la agencia de viajes, como necesitamos una pantalla web, debemos resolver esto de otra manera. Una solución sería, agregar una variable del tipo boolean al grid, utilizarla para marcar la fila seleccionada y luego recorrer las filas del grid para procesar aquellas que fueron seleccionadas. Veamos cómo hacer esto.

Cómo seleccionar múltiples filas de un grid



Name	Type	Is Collection
Variables		
Standard Variables		
Selected	Boolean	<input type="checkbox"/>
SelectedAttractionsIds	Id	<input checked="" type="checkbox"/>

Primero vamos a crear un webpanel de nombre SelectedAttractions, al cual le insertamos un grid, con los atributos AttractionId, AttractionName y AttractionPhoto. Luego creamos una variable de nombre Selected, del tipo boolean y la agregamos al grid, en la columna de más a la izquierda.

Luego agregamos un botón y al evento lo llamamos "Print selected attractions". En este evento, vamos a invocar al objeto procedimiento que imprime las atracciones seleccionadas y para poder enviárselas, debemos guardarlas primero en una colección.

Para esto, agregamos una variable llamada SelectedAttractionsIds, ya que nos alcanza con guardar en la colección solamente los identificadores de las atracciones elegidas. Le asignamos el tipo de datos Id y la marcamos como colección.

Cómo seleccionar múltiples filas de un grid

Name	Type	Is Collection
Variables		
Standard Variables		
• JsonSelectedAttractions	LongVarChar(2M)	<input type="checkbox"/>
• Selected	Boolean	<input type="checkbox"/>
SelectedAttractionsIds	Id	<input checked="" type="checkbox"/>

```

1 Event 'Print selected attractions'
2   For each line in GridAttractions
3     if &Selected
4       &SelectedAttractionsIds.Add(AttractionId)
5     endif
6   endfor
7   if &SelectedAttractionsIds.Count > 0
8     &JsonSelectedAttractions = &SelectedAttractionsIds.ToJson()
9   endif
10  SelectedAttractions(&JsonSelectedAttractions)
11 Endevent
  
```

Para recorrer las fila de un grid, usamos la instrucción **For each line** ... in ... el nombre del grid, que en este caso es GridAttractions. Esta instrucción se posicionará en cada renglón del grid y nos permitirá recuperar los valores que van tomando sus columnas en cada fila. Algo importante a tomar en cuenta es que el For each line solamente recorrerá los registros cargados en el grid, si hay más registros en siguientes páginas, los mismos no serán incluidos. Solamente aquellos registros que se ven en los renglones del grid pueden recorrerse.

Para averiguar si la atracción fue seleccionada o no, escribimos If &Selected y en caso de que esto se cumpla agregamos el AttractionId a la colección, por lo que escribimos &SelectedAttractionsId.Add(AttractionId), y cerramos el if. Ahora cerramos el for each line con un endfor. Si estuviéramos en una aplicación para Smart Devices, en lugar de la variable booleana utilizamos la propiedad **Show Selector** del grid y para recorrer las filas disponemos del comando **For each selected line**.

Volviendo a nuestro webpanel, cuando se recorran todas las filas del grid, pasaremos la colección de las atracciones seleccionadas al listado. Pero para poder hacerlo, no podemos pasar la variable colección como parámetro, sino que debemos serializar su contenido, es decir, generar un archivo de texto con un formato estructurado, por ejemplo un JSON o un XML.

Vamos a crear una variable para guardar el JSON, que llamamos JJsonSelectedAttractions del tipo LongVarChar.... y la cargamos a partir de la colección, utilizando el método ToJson.

Ahora sí, invocamos al procedimiento SelectedAttractions y le pasamos por parámetro el JSON que obtuvimos previamente.

Cómo seleccionar múltiples filas de un grid

Source View:

```

1 print Titles
2
3 &SelectedAttractionsIds.FromJson(&JsonSelectedAttractions)
4
5 For &AttractionId in &SelectedAttractionsIds
6   For each Attraction
7     Where AttractionId = &AttractionId
8     &AttractionName = AttractionName
9     &AttractionPhoto = AttractionPhoto
10  Endfor
11  print Attractions
12 Endfor

```

Layout View:

Titles

List of selected attractions

Id	Name	Photo
&AttractionId	&AttractionName	&AttractionPhoto

Attractions

Variables View:

Name	Type
Variables	
Standard Variables	
Autodefinited Variables	
AttractionId	Attribute:AttractionId
AttractionName	Attribute:AttractionName
AttractionPhoto	Attribute:AttractionPhoto
JsonSelectedAttractions	LongVarChar(2M)
SelectedAttractionsIds	Id

Procedure:

```

1 Parm(in:&JsonSelectedAttractions);
2 Output_file('SelectedAttractions', 'PDF');

```

Creamos un objeto de tipo procedimiento, al que llamamos SelectedAttractions y definimos las variables AttractionId, AttractionName y AttractionPhoto. Luego creamos una variable llamada JSonSelectedAttractions, del tipo LongVarChar y una variable SelectedAttractionsIds del tipo Id, a la cual marcamos como colección. Ahora vamos a las reglas y escribimos una regla Parm, que recibe como entrada a la variable JSonSelectedAttractions. Aprovechamos y ya incluimos a la regla Output_file...y marcamos al procedimiento como Main y el Call Protocol en HTTP.

En Layout, renombramos el printblock con el nombre Titles y arrastramos un Text Block al que le ponemos el texto "List of selected attractions". Agregamos también etiquetas para las columnas del listado, una con el texto "Id", otra "Name" y otra "Photo". Y por último ponemos una línea debajo de estas etiquetas de las columnas.

Ahora insertamos otro printblock, lo llamamos Attractions y arrastramos a las variables AttractionId, AttractionName y AttractionPhoto.

En el source escribimos print Titles.... Y ahora cargamos la variable colección SelectedAttractionsId con el contenido de la variable JSonSelectedAttractions, usando el método FromJson.

Con esto, los Ids de las atracciones seleccionadas quedaron guardados en la colección, por lo cual debemos recorrer la misma para accederlos, así que escribimos: For &AttractionId in &SelectedAttractionsIds, y a continuación con cada AttractionId accedido recuperaremos el nombre de la atracción y su foto, así que escribimos For each Attraction, Where AttractionId = &AttractionId y cargamos las variables &AttractionName y &AttractionPhoto a partir de los atributos correspondientes.





Cerramos el for each, imprimimos el printblock con los datos de las atracciones y finalmente cerramos el For in.

Ahora sí estamos en condiciones de probar lo que habíamos programado, así que damos F5.

Cómo seleccionar múltiples filas de un grid



Application Name

Recents Select Attractions

Selected	Id	Name	Photo
<input checked="" type="checkbox"/>	1	Louvre Museum	
<input type="checkbox"/>	2	Great Wall	
<input type="checkbox"/>	3	Eiffel Tower	
<input checked="" type="checkbox"/>	4	Christ Redeemer	

Print selected attractions

List of selected attractions

Id	Name	Photo
1	Louvre Museum	
4	Christ Redeemer	

En el Developer Menu hacemos clic en SelectAttractions y vemos que se abre el webpanel, mostrando la lista de atracciones. Vemos que el control asociado a la variable booleana Selected, es un check box, que nos permite seleccionar las atracciones.

Vamos a elegir al Museo de Louvre y a la Torre Eiffel... y presionamos Print selected attractions. Vemos que se abre el listado, mostrando a las atracciones que habíamos seleccionado.

En resumen...

- Para trabajar con una fila del grid, usamos la propiedad AllowSelection
- En un grid de un objeto SD, disponemos de la propiedad Enable Multiple Selection, pero en un grid de un objeto Web no.
- Para recorrer las filas de un grid usamos For each line en ambiente Web y For each selected line en una app SD
- Nota: No es posible pasar una colección por parámetro a un reporte PDF, debe convertirse a JSON/XML y pasar el string

Repasemos los conceptos vistos hasta ahora.

Si queremos seleccionar a una sola fila de un grid, ponemos en True a la propiedad AllowSelection, con lo cual podremos acceder a los valores de cada columna del grid, para la fila seleccionada.

Si queremos seleccionar más de una fila al mismo tiempo, si estamos en una aplicación para Smart Devices el grid cuenta con la propiedad Enable Multiple Selección. Al asignarle el valor True, es posible seleccionar múltiples filas para procesarlas luego.

En un grid de una aplicación web, no disponemos de esa propiedad, por lo cual debemos utilizar una variable booleana en el grid, que nos permitirá guardar la selección de la fila.

Para recorrer las filas de un grid en una aplicación Web disponemos del comando For each line. Este comando iterará solamente por las filas que estén cargadas en el grid en un momento dado y en cada iteración dispondremos de los valores de las columnas del grid para la fila recorrida. Para saber si una fila fue seleccionada, utilizamos el valor de una variable booleana que agregamos para ese fin.

En el caso de una aplicación SD se utilizamos la propiedad Show Selector y el comando For each selected line.

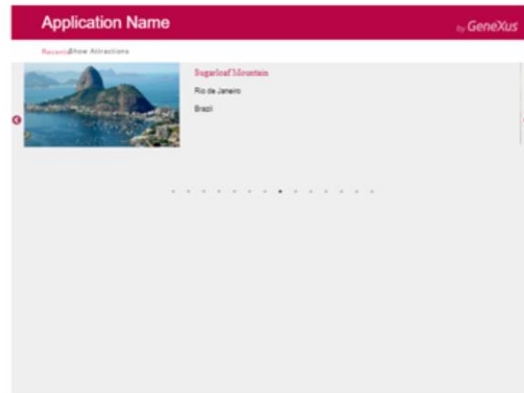
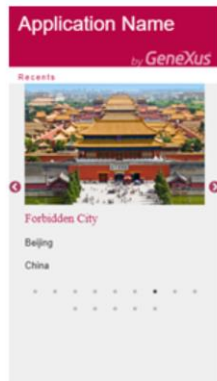
No es posible pasar una variable colección por parámetro a un objeto web, como un webpanel o un procedimiento con protocolo HTTP. Para enviar la información de las atracciones seleccionadas, usamos una colección pero después serializamos esta información generando un archivo de texto estructurado, por ejemplo usando el formato JSON o XML.

Cambiando el comportamiento de un grid

Cambiando el valor de algunas propiedades del grid, podemos lograr que el mismo cambie la forma de mostrar la información, por ejemplo mostrar los datos horizontalmente, con scroll infinito, o en forma flexible, personalizando el despliegue en función de su contenido.

Veamos a continuación algunos casos.

Grid horizontal



Un grid horizontal es un grid que nos permite mostrar la información como si fuera un carrousel, es decir, que podemos paginar desde la derecha y los elementos se irán desplazando en forma horizontal, hacia la izquierda.

Grid horizontal

Grid: Grid1

Control Name	Grid1
Collection	
Base Trn	
Order	
Conditions	
Unique	
Data Selector	(none)
Appearance	
Class	Grid
Custom Render	HorizontalGrid
Empty Grid Text	
Auto Resize	True

HorizontalGrid

Paged	True
Show Page Controller	True
Page Controller Class	GridPageController
Show Arrows	True
Infinite	False
Auto Play	False
Variable Width	False
> Rows per page	1
Layout	
Cell Padding	1
Cell Spacing	2

Para hacer que un grid muestre su contenido en forma horizontal, es decir en formato carousel, debemos asignar la propiedad Custom Render en el valor HorizontalGrid.

Al asignar este valor, se habilitarán una serie de propiedades mediante las cuales podremos configurar cómo se verá el grid. Por ejemplo, la propiedad Show Page Controller determina si se verá o no el paginador y la propiedad Page Controller Class nos permitirá definir la **clase** en la que está basada este paginador.

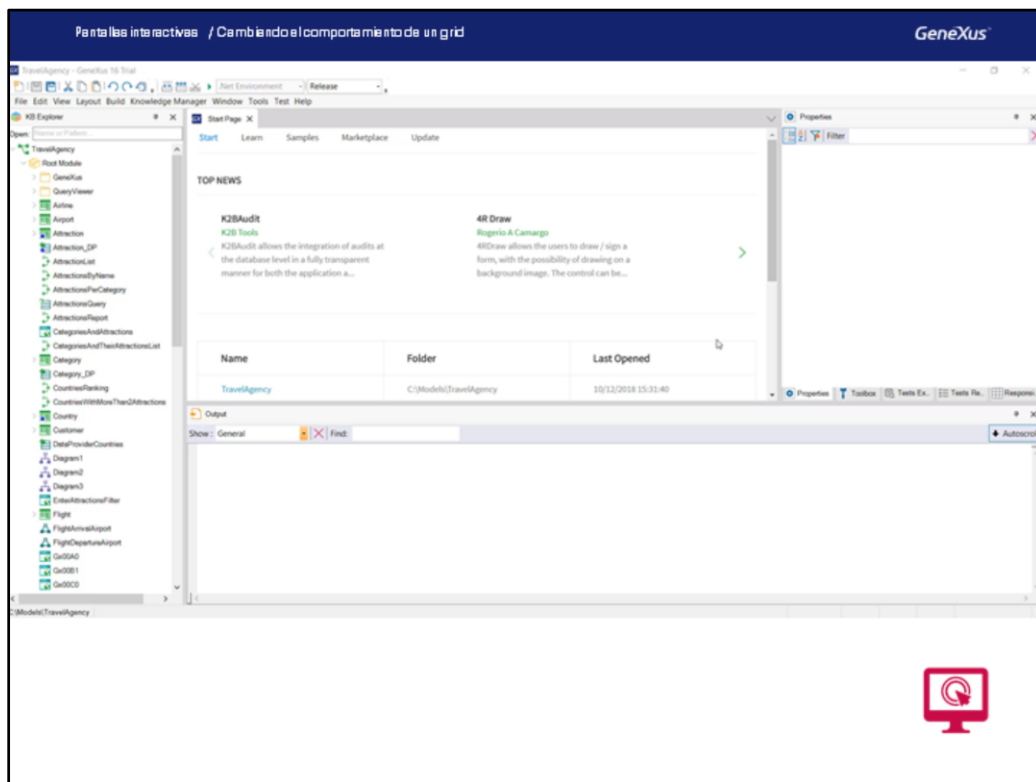
Esta clase es como un template al que le podemos cambiar propiedades y todos los controles que se basen en la clase, heredarán dichos valores.

Hay un cierto paralelismo con el concepto de dominio que vimos antes. Cambiando los valores de las propiedades de una clase, podremos personalizar todos los controles que se basan en ella.

Estas clases, que definen la apariencia y comportamiento de cada control en la pantalla, se agrupan bajo el **objeto Theme**, que estudiaremos más adelante.

Volviendo a las propiedades que nos permiten personalizar nuestra grilla horizontal, vemos que podemos decidir si queremos ver o no las flechas del grid que usamos para cambiar de registro, definir si el ancho del grid es variable, cómo será su paginado, etc.

Veamos un ejemplo de un grid de este tipo.



[Demo: <https://youtu.be/b76ee3lu3hQ>]

Veamos una demo de cómo configurar un grid horizontal mediante la propiedad **Custom Render**.

Para probar lo anterior, tenemos un webpanel ShowAttractions con un grid. En particular es un FreeStyleGrid, pero el ejemplo también es válido para un grid estándar.

Al grid agregamos los atributos AttractionName, AttractionPhoto, CityName y CountryName y una tabla responsive para mejorar la alineación.

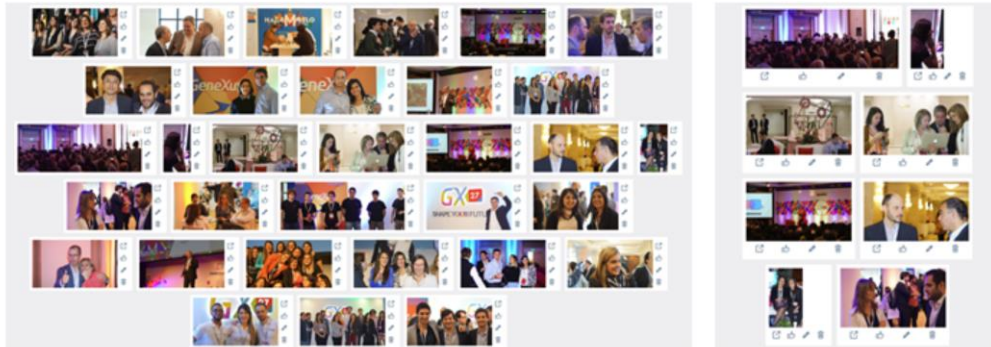
Si ejecutamos este webpanel... vemos el grid con su comportamiento habitual. Ahora asignamos el valor Horizontal grid a la propiedad **Custom Render** y ejecutamos.

Vemos que con sólo hacer esto, se despliega la información en forma horizontal, como en un carrousel, mostrándose una atracción por vez.

También vemos el control del paginado del carrousel y las flechas a la derecha e izquierda de la página. Como dijimos, todo esto es configurable desde el objeto Theme.

Como estamos usando Chrome, si presionamos F12 podemos elegir el tamaño de la pantalla. Si elegimos otros dispositivos vemos que la información se ajusta de forma responsiva, incluso si cambiamos la orientación del mismo.

Grid flexible



Un grid flexible nos permite visualizar la información de una manera adaptable a nuestras necesidades y nos permiten una personalización mayor en la visualización del contenido, que el que puede permitir el comportamiento responsivo.

Cuando agregamos fotos por ejemplo, nunca sabemos el largo o ancho de cada elemento que debemos mostrar. Estos elementos deberán reordenarse automáticamente de la mejor manera dependiendo del tamaño de la pantalla. Por ejemplo en la figura de la izquierda vemos que las fotos tienen las acciones a la derecha y en la figura de la izquierda, con una pantalla más chica, los elementos se ajustaron al ancho y los controles de las fotos se movieron automáticamente debajo de las fotos.

Grid flexible

Grid: Grid1

Control Name	Grid1
Collection	
Base Trn	
Order	
Conditions	
Unique	
Data Selector	(none)

Appearance

Class	Grid
Custom Render	FlexGrid
Empty Grid Text	
Auto Resize	True

FlexGrid

Flex Direction	Row
Flex Wrap	No Wrap
Justify Content	Flex Start
Align Items	Stretch
Align Content	Stretch

Para hacer que un grid muestre su contenido en forma flexible, debemos asignar la propiedad **Custom Render** en el valor FlexGrid.

Al asignar este valor, se habilitarán una serie de propiedades mediante las cuales podremos configurar cómo se verá el grid.

Grid flexible

FlexGrid

Flex Direction	Row
Flex Wrap	No Wrap
Justify Content	Flex Start
Align Items	Stretch
Align Content	Stretch

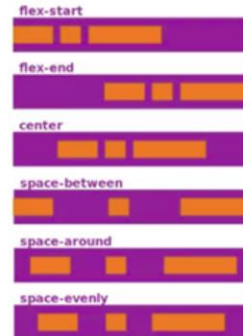
Flex Direction



Flex Wrap



Justify Content



La propiedad **Flex Direction** nos permite cambiar la dirección en que se muestran los elementos en la pantalla, si es por fila o fila invertida, por columna o columna invertida.

Flex Wrap establece cómo se van a acomodar los elementos cuando ocupan más que el largo de una fila y no pueden mostrarse todos. El valor Wrap los acomoda en la siguiente línea.

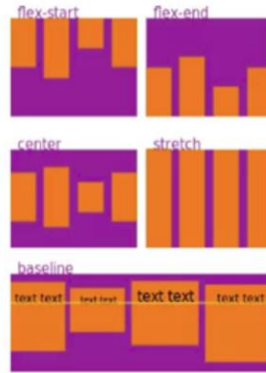
Justify Content nos permite establecer cómo se va a alinear el contenido, si de izquierda a derecha, de derecha a izquierda, centrado, que se ajuste la separación para ocupar todo el ancho disponible o si se agregan espacios entre los elementos.

Grid flexible

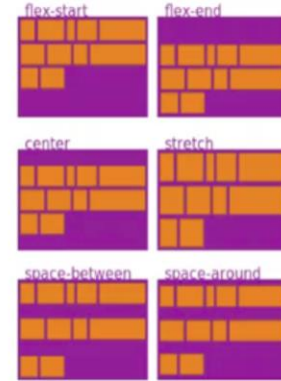
FlexGrid

Flex Direction	Row
Flex Wrap	No Wrap
Justify Content	Flex Start
Align Items	Stretch
Align Content	Stretch

Align Items



Align Content



La propiedad **Align Items** nos permite seleccionar como se alinean los ítems en las filas.

Align Content nos permite alinear la fila cuando hay espacio extra en el contenedor.

Pantallas interactivas / Cambiando el comportamiento de un grid

GeneXus

Grid con scroll infinito

Application Name

by GeneXus

Recents Attractions

X HIDE FILTERS





Attractions

Q Name

+ INSERT

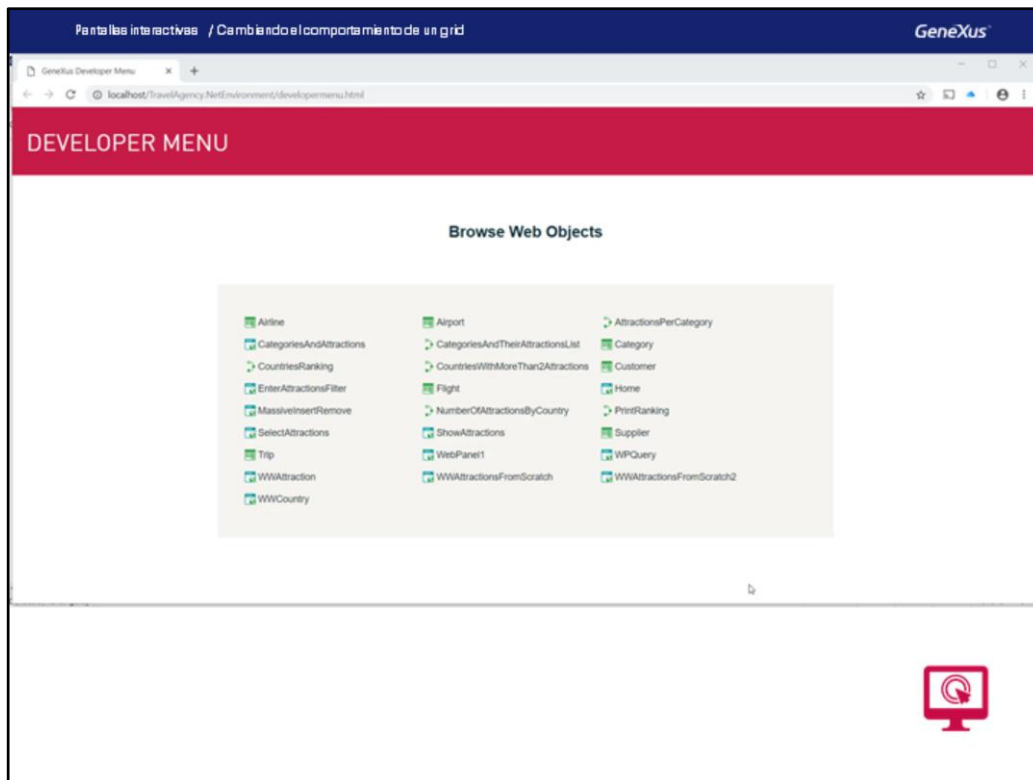
Ordered By : Name

COUNTRY NAME

Id	Name	Country Name	Category Name	Photo	City Name	Address
10	Copacabana Beach	Brazil	Tourist site		Rio de Janeiro	UPDATE DELETE
3	Eiffel Tower	France	Monument		Paris	UPDATE DELETE
7	Forbidden City	China	Tourist site		Beijing	UPDATE DELETE
1	Louvre Museum	France	Museum		Paris	UPDATE DELETE

Loading...

Una característica que podemos necesitar es que, en vez de que el grid nos muestre la información paginada, los renglones se vayan cargando a medida que hacemos scroll, en la misma página.



[Demo: <https://youtu.be/jWNAkRXwGw0>]

A los efectos de esta demo, hemos agregado algunas atracciones. Si vamos al “trabajar con Atracciones”, vemos que hay 10 atracciones que se muestran en la página y que al final de la misma contamos con control de paginado. Si presionamos el botón de siguiente, veremos 4 atracciones más.

Si quisiéramos cambiar el valor por defecto de 10 a 5 registros por página, abrimos la instancia del pattern WorkWithAttraction, hacemos clic sobre el nodo Selection, en la propiedad **Rows per page** elegimos el valor **<custom>** y en la propiedad **Custom Rows**, asignamos el valor 5. Presionamos F5....Y vemos que las atracciones quedan separadas en 3 páginas.

Si necesitamos revisar información de las atracciones y en vez de 14 atracciones tuviéramos 1000, aún dejando el valor por defecto de 10 registros por página, son muchas páginas a revisar. Sería mucho más sencillo si pudiéramos hacer que la grilla fuera cargando los registros en forma automática, a medida que hacemos scroll hacia abajo, sin tener que paginar a mano. Esto es lo que se conoce como **scroll infinito**.

Para obtener este comportamiento en el grid del patrón WorkWith, vamos al nodo Selection y en la propiedad **Paging mode** seleccionamos el valor **<infinite scrolling>**. Presionamos F5 y vemos que aparece la barra de scroll en el grid y desaparece el control de paginado. Al mover el scroll, aparece el mensaje "Loading..." indicando que se están cargando los registros de la Base de Datos y podemos hacer scroll indefinidamente hasta ver todas las atracciones que tenemos cargadas en la tabla.

Si queremos cambiar la cantidad de registros por página en un grid estándar que no esté en un patrón Work With, cambiaríamos la propiedad **Rows** del valor 0 que es el valor por defecto, al valor que deseamos. El valor 0 equivale a “unlimited” y hace

que se vean todos los registros de la grilla de una vez.

Presionamos F5...Y vemos que desaparece el control de paginado y no aparece la barra de scroll en la grilla. Sí aparece el scroll en la página, porque la grilla queda muy larga para verla entera, y la página nos hace scrollear. (Nota: Para ver la imagen un en tamaño mayor al por defecto, edite la columna AttractionPhoto, ponga su propiedad Auto Resize en False y cambie los valores de las propiedades Height y Width).

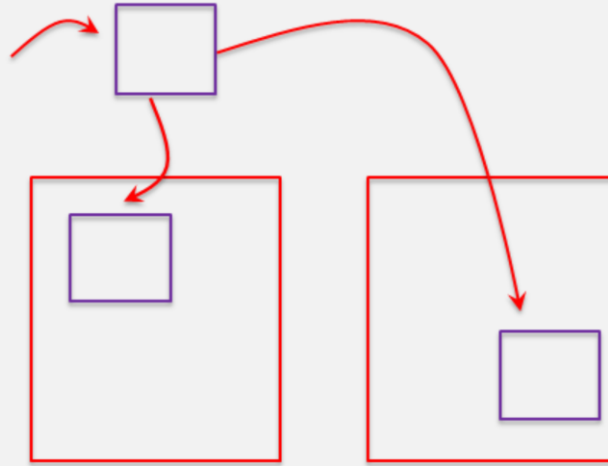
Para obtener el scroll infinito en un grid estándar que no esté en un patrón Work With, asignamos la propiedad **Rows** con un valor distinto de 0 y cambiamos la propiedad **Paging** al valor **Infinite Scrolling**.

En un grid Free Style, el valor por defecto de la propiedad Rows es <unlimited> y este valor no puede cambiarse.

Web components

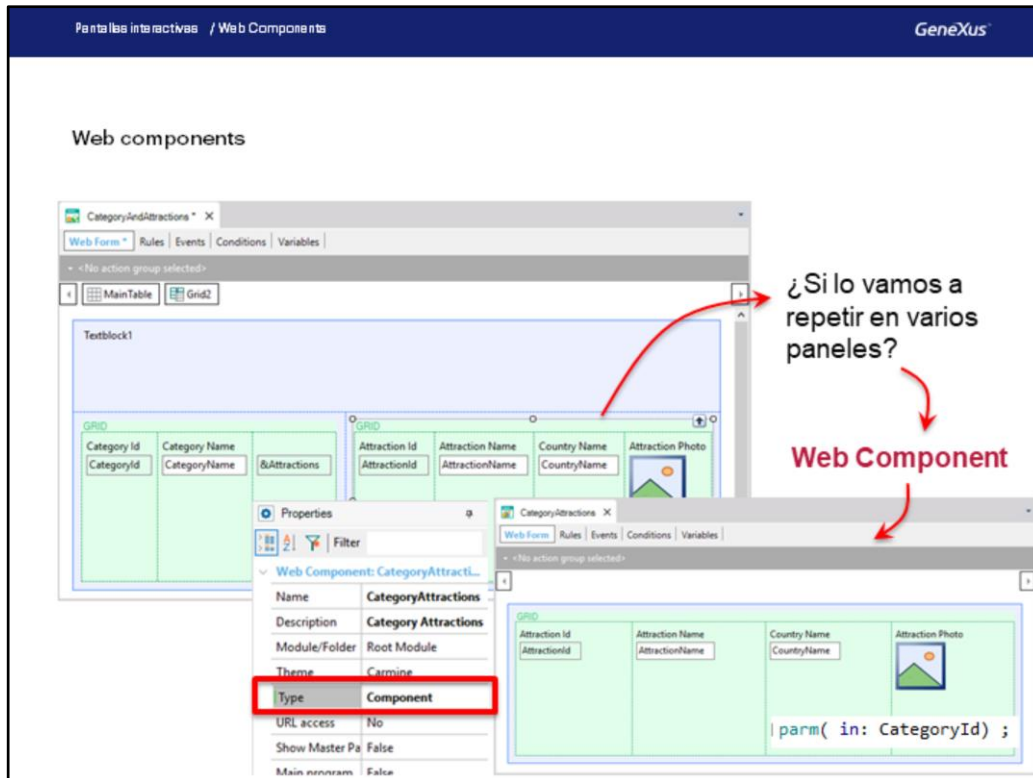
Habíamos visto

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



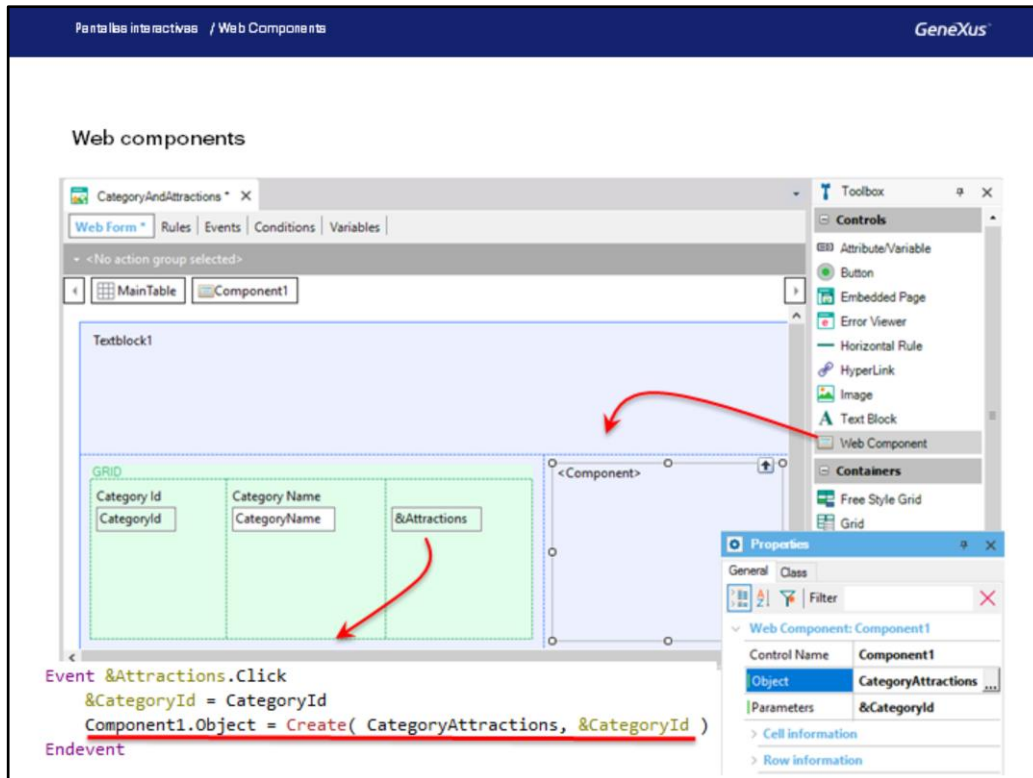
En otra clase habíamos visto que existen tres tipos de web panels.

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...



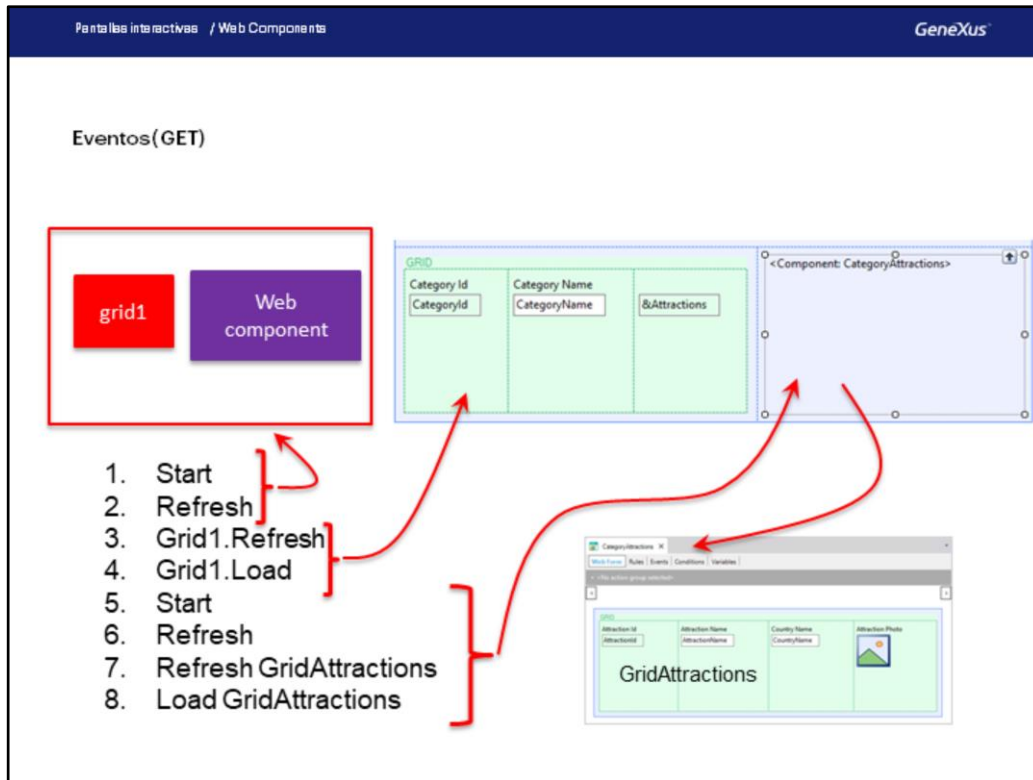
En el web panel donde teníamos inicialmente el grid de atracción, colocamos en su lugar un control Web Component.

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 parámetro 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.

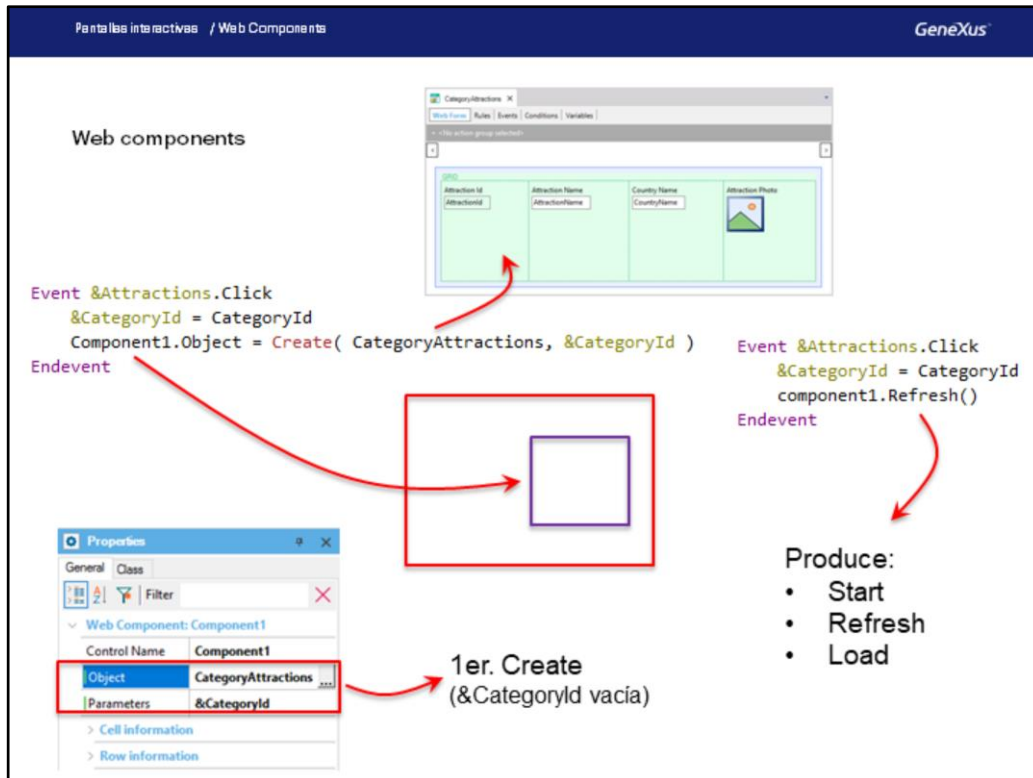
Existe el método Refresh() a nivel del componente. Lo veremos luego de ver los eventos que se ejecutan en el GET.



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 ejecuta 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 Refresh 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.

Eventos Globales

Pantallas Interactivas / Eventos Globales
GeneXus

Eventos Globales

- > Main Programs
- > Root Module
 - > GeneXus
 - > Common
 - GlobalEvents
 - > SD
 - > Web
 - > References
 - > Customization
 - > Documentation

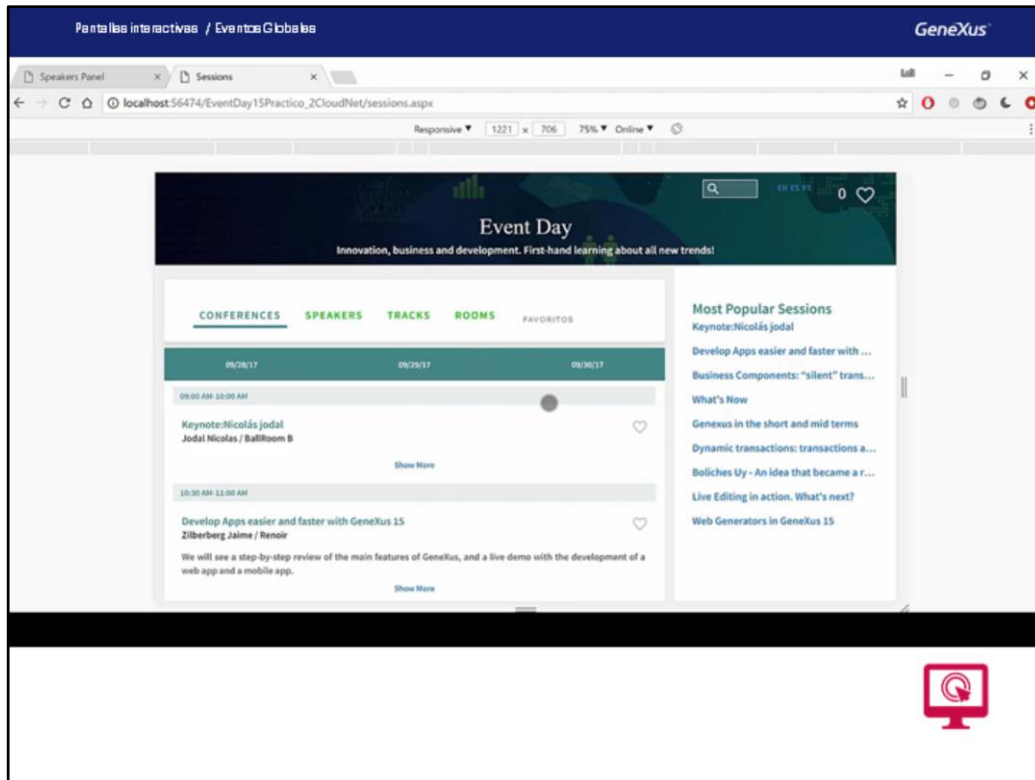
Structure	Type	Is Collection
GlobalEvents		
Properties		
Methods		
Events		
CustomSingleEvent	None	<input type="checkbox"/>
CustomParmEvent	None	<input type="checkbox"/>
Parm1	Numeric(8.0)	<input type="checkbox"/>
Parm2	Character(20)	<input type="checkbox"/>

Muchas veces necesitamos implementar una solución donde una acción en un componente en pantalla, causa una reacción en otro componente de la pantalla.

Por ejemplo, disparar una acción en un webcomponent cuando el usuario selecciona un elemento de un menú, que se encuentra en otro componente.

Dado que la comunicación entre webcomponents es limitada – la única comunicación viable es de padres a hijos – los eventos globales nos permiten establecer comunicación entre webcomponents que no se encuentran relacionados.

Mas Información: <https://wiki.genexus.com/commwiki/servlet/wiki?31167>



[DEMO: <https://youtu.be/D8akim-gRSQ>]

Supongamos que tenemos una aplicación Web para un evento. Si vamos al componente que despliega las conferencias, al hacer clic en el botón de Favorito, se está refrescando un componente que muestra el total de favoritos, que se encuentra en el master page y que no tiene una relación directa con el componente que lo está disparando.

Vamos a ver como logramos esto en GeneXus

Para ello tenemos un nuevo objeto llamado GlobalEvents. Podemos definir tantos objetos Global Events como queramos, básicamente esto es un objeto externo, y podríamos definir GlobalEventsBackend, GlobalEventfrontend, etc. dependiendo cómo queramos agruparlos, es importante recordar que son globales y por lo tanto es importante utilizarlos con cierto criterio y ordenadamente para que no generen demasiado ruido.

Nosotros tenemos aquí nuestro GlobalEvents que básicamente lo que dice es que se actualice la cantidad de favoritos.

En nuestro caso, nuestro Evento no recibe parámetros, pero es posible implementar eventos que sí, si fuese necesario pasar parámetros seguirían definiendo los mismos, asociados al evento, con sus tipos de datos etc. y en el momento que se invoca es responsabilidad del objeto que dispara el evento definir cuáles son los parámetros y luego todos los componentes que implementan el evento o que escuchan por este evento van a recibir los parámetros con este formato.

¿Cómo disparamos este evento global? En nuestro caso lo estamos haciendo a partir del componente de sessions que básicamente cuando hace clic en la imagen de favoritos, se hace el toggle que muestra la imagen en gris o en amarillo y además se dispara el GlobalEvents.

¿Quién recibe/ejecuta este GlobalEvents? Eso es una decisión nuestra, y lo puede hacer cualquier componente que tenga sentido.

En nuestro caso, lo está haciendo el componente que muestra la cantidad de eventos favoritos, básicamente este componente implementa GlobalEvents.UpdateFavoritesQuantity y básicamente aquí va a hacer la llamada que necesite para actualizar la información, que podría ser un refresh etc.

En definitiva si tuviéramos más de un componente que necesitan suscribirse a este evento, es posible. Por ejemplo, podríamos querer estar mostrando información de los favoritos en distintas partes de las pantallas y cuando alguien agrega un evento o una charla a sus favoritos, podríamos estar refrescando diferentes componentes cada uno de ellos implementando el Events.UpdateFavoritesQuantity.

Mas Información: <https://wiki.genexus.com/commwiki/servlet/wiki?31167>

GeneXus™

The power of doing.

Videos

Documentation

Certifications

training.genexus.com

wiki.genexus.com

training.genexus.com/certifications