

Pantallas interactivas  
Objeto Web Panel

*GeneXus® 16*

## Objeto Web Panel

- Implementa pantalla web sumamente flexible
- Ejemplo:

The screenshot shows the GeneXus Web Panel designer interface. The title bar reads 'EnterAttractionsFilter'. Below it is a tabbed menu with 'Web Form' selected and highlighted with a red box. Other tabs include 'Rules', 'Events', 'Conditions', and 'Variables'. The main area displays a form structure with a 'MainTable' container. Inside, there are three input fields: 'Country Id' with a dropdown arrow and variable '&CountryId', 'Attraction Name From' with variable '&AttractionNameFrom', and 'Attraction Name To' with variable '&AttractionNameTo'. A red bracket on the right side of these three fields points towards the text 'Variables: controles de entrada (no readonly)'. At the bottom of the form are two buttons: 'List Attractions By Country' and 'List Attractions By Name'.

Variables: controles de entrada  
(no readonly)

El web panel es el objeto más flexible que provee GeneXus.

Como ya hemos visto en algunos ejemplos que hemos mostrado, todo web panel ofrece un web form, que es una página web que nos permite diseñar y ofrecer variadas funcionalidades.

En este ejemplo habíamos visto que por el hecho de incluir variables en el web form éstas quedaban habilitadas para que el usuario les ingresara algún valor. Es decir, eran controles de entrada, o, dicho de otro modo, no readonly.

## Ejemplo: variables en Web panel

The screenshot displays the GeneXus IDE interface for a web panel named 'EnterAttractionsFilter'. The design view shows a form with a 'Country Id' dropdown, 'Attraction Name From' and 'To' text boxes, and two buttons: 'List Attractions By Country' and 'List Attractions By Name'. The 'List Attractions By Country' button is highlighted with a red box, and a red arrow points from it to the 'Country Id' dropdown. Another red arrow points from the 'Country Id' dropdown to the 'Item Values' property in the Properties window. The Properties window shows the 'Country Id' dropdown is a 'Dynamic Combo Box' with 'CountryId' as the 'Item Values' and 'CountryName' as the 'Item Descriptions'. A red arrow points from 'CountryName' to a database icon labeled 'Tabla Country'.

Event 'List Attractions By Country'

```
AttractionsList(&CountryId)
Endevent
```

En particular esta variable, de tipo combo dinámico, esperaba que el usuario eligiese un país de los cargados en el combo y al presionar el botón "List Attractions By Country", se ejecutaba el evento asociado, invocando al pdf que listaba las atracciones de ese país.

Aquí se accede a la base de datos únicamente para cargar los valores del combo.

Ejemplo: variables en Web panel

EnterAttractionsFilter \* X

Web Form Rules Events \* Conditions Variables

<No action group selected>

MainTable ListAttractionsByName

Country Id &CountryId

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

List Attractions By Country

List Attractions By Name

Event 'List Attractions By Name'

AttractionsByName( &AttractionNameFrom, &AttractionNameTo )

Endevent

AttractionsByName X

Source Layout Rules Conditions Variables

1 param( in: &NameFrom, in: &NameTo );

Y en estas otras variables el usuario ingresaba un rango de nombres de atracción para que, presionando este otro botón, se invocara al listado pdf que mostraba las atracciones dentro de ese rango recibido por parámetro.

Ejemplo:

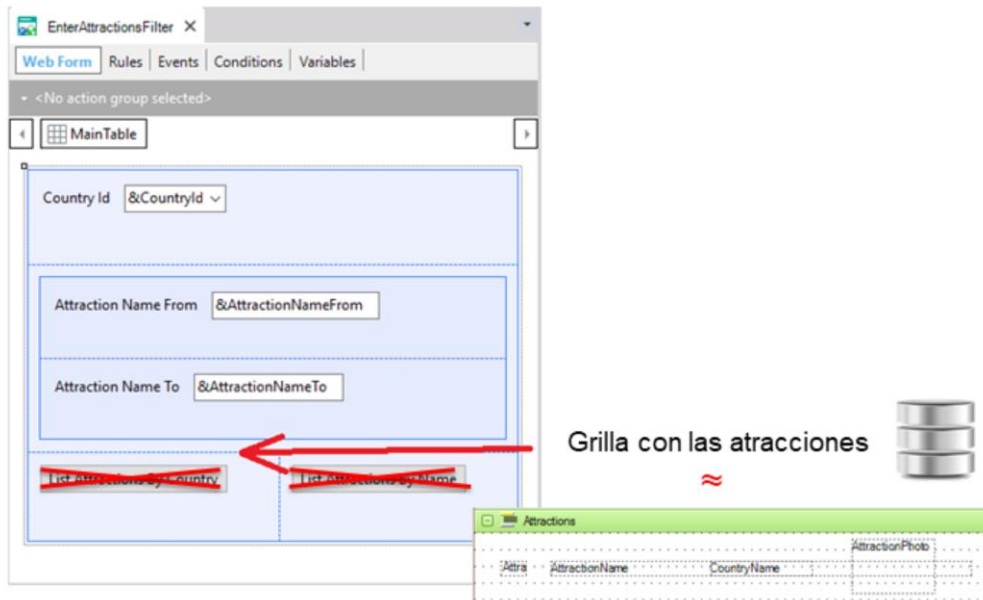
¿No podemos implementar este listado en la pantalla web anterior, que pedía los filtros al usuario?

```
1 print Title
2 print ColumnTitles
3 For each Attraction order CountryName
4   where AttractionName >= &NameFrom
5   where AttractionName <= &NameTo
6   print Attractions
7 endfor
```

Recordemos el Layout. En el Source programábamos la consulta a la base de datos con el for each, filtrando por nombre.

Pero ¿por qué definir estas consultas a través de listados pdf y no directamente en la propia pantalla en la que le pedíamos al usuario los datos para los filtros?

Modifiquemos el web panel para que pueda consultar la BD



¿Por qué no colocar aquí, en lugar de los botones, una grilla que muestre las atracciones deseadas?


Las filas de la grilla serán similares al printblock que imprime cada atracción.

## Web panels: consultas interactivas a la base de datos

Country Id

Attraction Name From

Attraction Name To

Id	Attraction Name	Country	Photo
AttractionId	AttractionName	CountryName	

Atributos en web panel son de salida: readonly



Los web panels, además de permitirnos definir variables para utilizarlas en acciones programadas en botones, nos permiten, -y es su objetivo fundamental- implementar consultas **interactivas** a la base de datos.

El término “**interactivas**” se refiere a que el usuario puede ingresar en la página del web panel una y otra vez distintos valores –**en variables**– y consultar a continuación datos de la base de datos que concuerden con esos valores ingresados, utilizándolos como filtros, como veremos ahora.

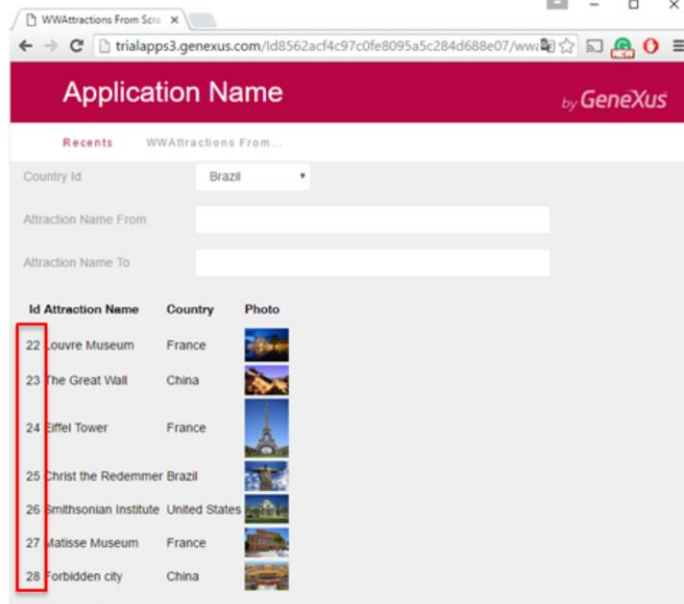
Grabemos este web panel con otro nombre. Como lo que vamos a implementar va a ser similar a un work with, llamémosle WWAttractionsFromScratch.

Eliminemos los botones, que ya no serán necesarios, así como los eventos asociados. Ahora insertemos debajo de las variables un control de tipo grilla (**grid** en inglés).

Se nos abre una pantalla para elegir los atributos y/o variables que serán las columnas de este grid. Como lo que queremos mostrar es lo mismo que mostrábamos en los listados pdf, elegimos los atributos AttractionId, AttractionName, AttractionPhoto y CountryName. Podemos cambiar los títulos de cada columna, editando las propiedades de cada uno de los atributos que conforman esas columnas del grid.

Los **atributos** en el form de un web panel son, por defecto, **de salida**. Es decir, son **readonly**. Esto significa que GeneXus interpreta que debe ir a la base de datos a buscar su valor para mostrarlo al usuario.

## Web panels: consultas interactivas a la base de datos



Presionemos F5 para ejecutar nuestro nuevo web panel así como lo tenemos hasta ahora.

Vemos que salieron impresas todas las atracciones, con los datos que indicamos (el id, nombre, país y foto). Además vemos que salieron ordenadas por AttractionId.



Grid con atributos equivale a un for each

The screenshot displays the GeneXus IDE interface. On the left, a web form is shown with fields for 'Country Id' (a dropdown), 'Attraction Name From', and 'Attraction Name To'. Below these is a data grid with columns: 'Id', 'Attraction Name', 'Country', and 'Photo'. The grid's data source is set to 'Attraction'. In the center, a code window shows a 'For each Attraction' loop. The Properties window on the right shows the 'Base Trn' property of the grid set to 'Attraction'.

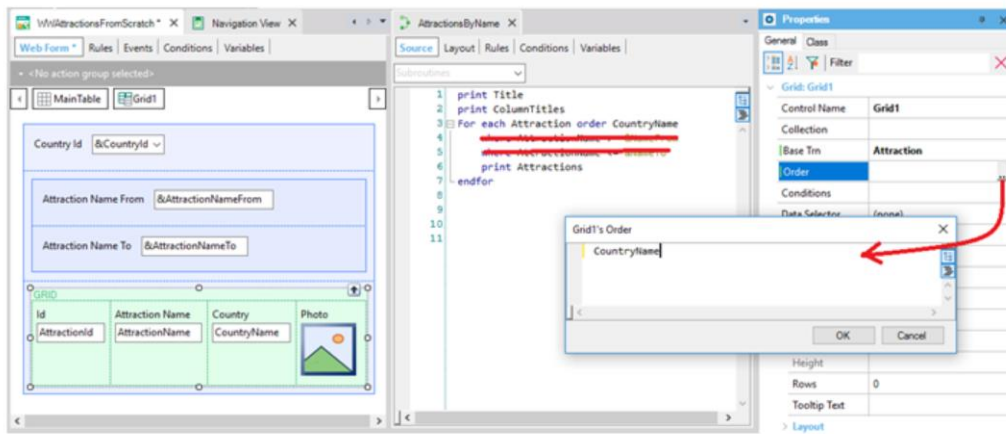
```
1 print Title
2 print ColumnTitles
3 For each Attraction endfor
4   print Attraction
5 endfor
```

Transacción base

Tan sólo con colocar un grid con esos atributos, GeneXus entendió que debía ir a la base de datos a navegar la tabla Attraction, accediendo a Country para traer el país de la atracción, tal como lo hacíamos con el comando for each (por ahora sin las cláusulas order y where).

Si observamos las propiedades del grid, vemos que de hecho tiene una de nombre **Base Trn**, que es análoga a la transacción base del for each. De hecho, para asegurarnos de que para el grid se elija la tabla base Attraction, que es la que deseamos, es una buena práctica indicar la transacción base, igual que para el for each.

Grid con atributos equivale a un for each



Order

Por otro lado, observemos que tenemos una propiedad **Order** para el grid. Esta propiedad se corresponde con la cláusula Order del for each.

Así, si quisiéramos ordenar por nombre de país, como en el listado, en la propiedad Order colocaríamos el atributo CountryName.

Grid con atributos equivale a un for each

The screenshot shows a web application on the left and a development tool on the right. The web application displays a grid of attractions with columns for Id, Attraction Name, Country, and Photo. The Country column is highlighted with a red box. The development tool shows a 'Web Panel WWAttractionsFromScratch Navigation Report' with a 'Warnings' section and an 'Event Load' section. The 'Warnings' section contains a message about a missing index for 'CountryName' and a 'Fill' statement. The 'Event Load' section shows a 'Navigation' event with a 'filters' section containing a 'CountryName' filter. A red arrow points to the 'CountryName' filter with the text 'Base table of the Grid'.

**Warnings**

spc0038 There is no index for order CountryName; poor performance may be noticed in grid 'Grid1'.

FILL &CountryId with CountryId, CountryName in

=Country ( CountryId) INTO CountryId CountryName

Order CountryName

**Event Load**

Order: CountryName

No index

Navigation Start FirstRecord

filters: from: NotEndOfTable

Loop while: NotEndOfTable

Join location: Server

=Attraction ( AttractionId)

=Country ( CountryId)

Base table of the Grid

Presionamos F5. Y vemos que ahora el grid sale ordenado por nombre de país.

Observemos el listado de navegación del web panel.

Aquí nos está indicando la navegación que tiene que realizar para cargar el combo box de la variable &CountryId, que por ahora no estamos usando para nada.

Y aquí está indicando la navegación que tendrá que realizar para cargar (Load) el grid. Vemos que lo que lista para esta carga es idéntico a lo que lista para un for each. Vemos que eligió la tabla Attraction, recorriéndola por CountryName, el atributo de la propiedad Order. Recorrerá toda la tabla. Y por cada registro de Attraction a ser cargado, accederá a la tabla Country para mostrar el CountryName de la atracción.

## Filtrar los datos del grid

The screenshot shows the GeneXus IDE interface. On the left, a web form is visible with a dropdown for 'Country Id' and a grid below it. The grid has columns for 'Id', 'Attraction Name', 'Country', and 'Photo'. In the center, the 'Source' tab shows a code snippet: 

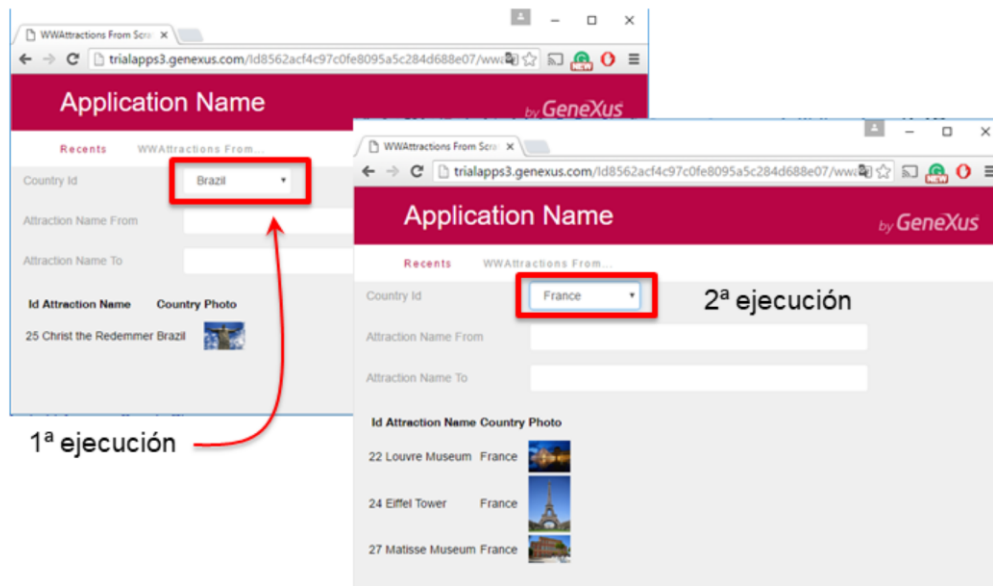
```
1 print Title
2 print ColumnTitles
3 For each Attraction order CountryName
4   where CountryId = &CountryId
5   print Attractions
6 endfor
```

 On the right, the 'Properties' panel for 'Grid: Grid1' is shown. The 'Control Name' is 'Grid1'. The 'Base Trm' is 'Attraction'. The 'Order' is 'CountryName'. The 'Conditions' property is highlighted with a red box and contains the text 'CountryId = &CountryId;'. A red arrow points from this property to the 'where' clause in the code snippet. The 'Data Selector' is '(none)'. The 'Appearance' section shows 'Class' as 'Grid', 'Auto Resize' as 'True', and 'Rows' as '0'.

**Condiciones de filtro**

Hasta ahora no hicimos nada con las variables. Pero queríamos utilizarlas para filtrar los datos que se muestran en el grid, tanto por país como por nombre de atracción. En AttractionsList filtrábamos por país. ¿Cómo indicamos este filtro para el grid? A través de la propiedad **Conditions**.

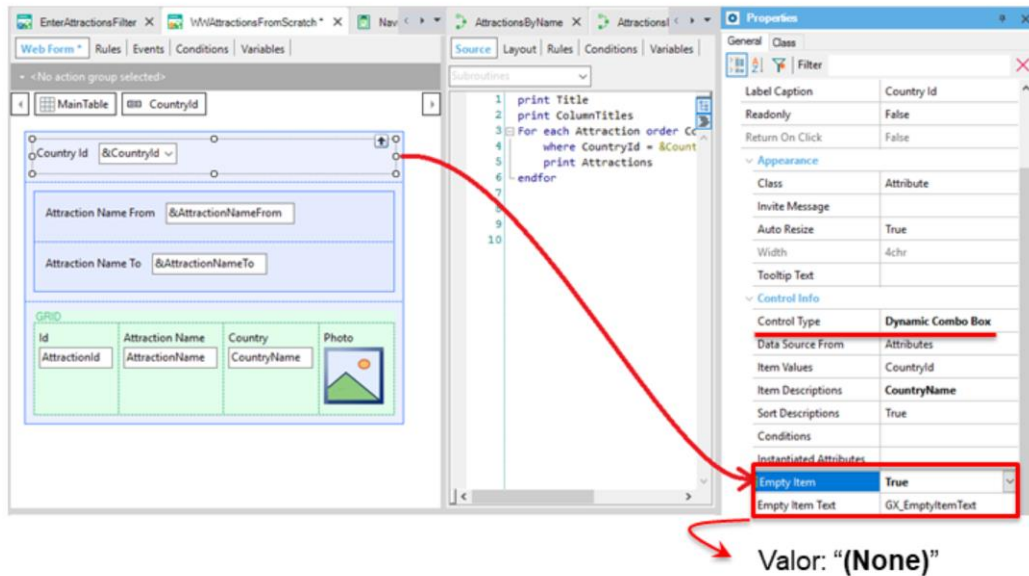
Filtrar los datos del grid



Observemos que por defecto el combo toma el valor Brazil, y que en el grid sólo vemos la atracción de Brazil.

Si elegimos Francia vemos que se refresca la pantalla, volviéndose a cargar el grid, ahora con las atracciones de Francia.

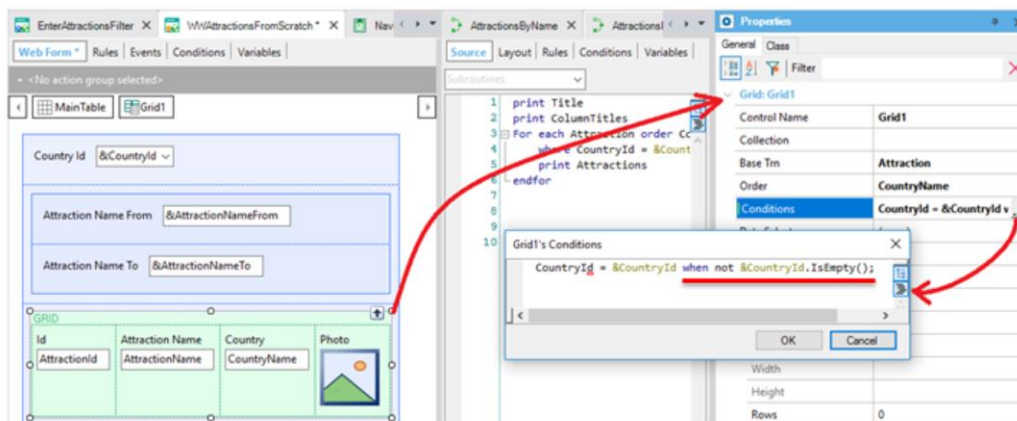
## Filtrar los datos del grid en forma condicional



Probablemente querremos que el combo aparezca sin valor elegido la primera vez, y que en ese caso se muestren las atracciones de todos los países.

Para ello editemos las propiedades del dynamic combo box... y pasemos a **True** la de nombre Empty Item. Esto hará que se agregue una opción "(None)" al combo. Corresponderá al valor empty, es decir, vacío, cero.

## Filtrar los datos del grid en forma condicional

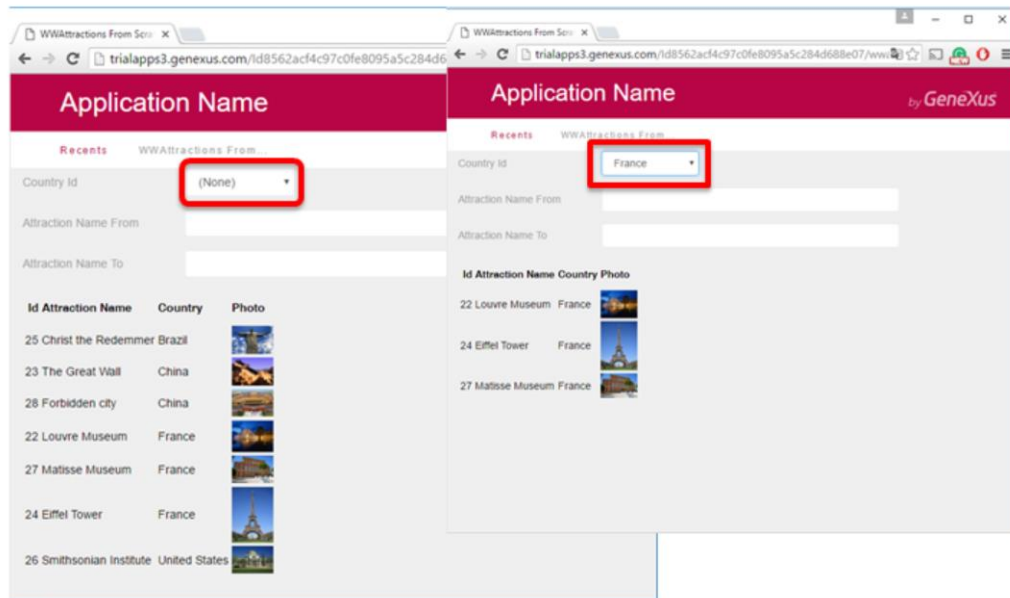


Condiciones condicionadas

F5

Entonces vayamos a la propiedad **Conditions** y especifiquemos que queremos que se aplique esta condición solamente cuando el combo no tiene el valor vacío. Cuando sí lo está, que no la aplique.

Filtrar los datos del grid en forma condicionada



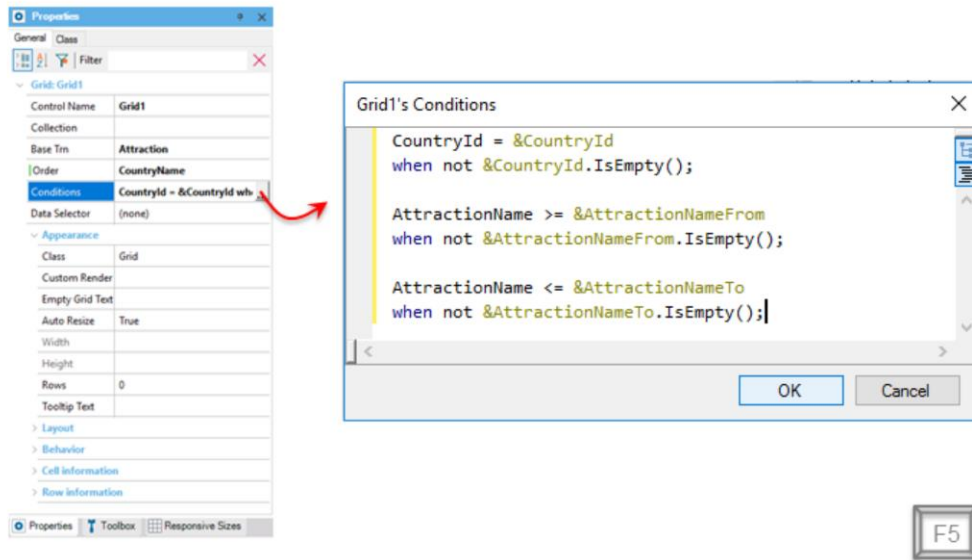
Ejecutemos...

Vemos cómo aparece el valor (None) en el combo, y además cómo para este caso, no se están filtrando las atracciones. Se muestran todas.

Si ahora elegimos, por ejemplo, Francia, como la variable no tiene el valor vacío, sí se aplica el filtro, y se muestran las atracciones de Francia.



Filtrar los datos del grid por múltiples condiciones

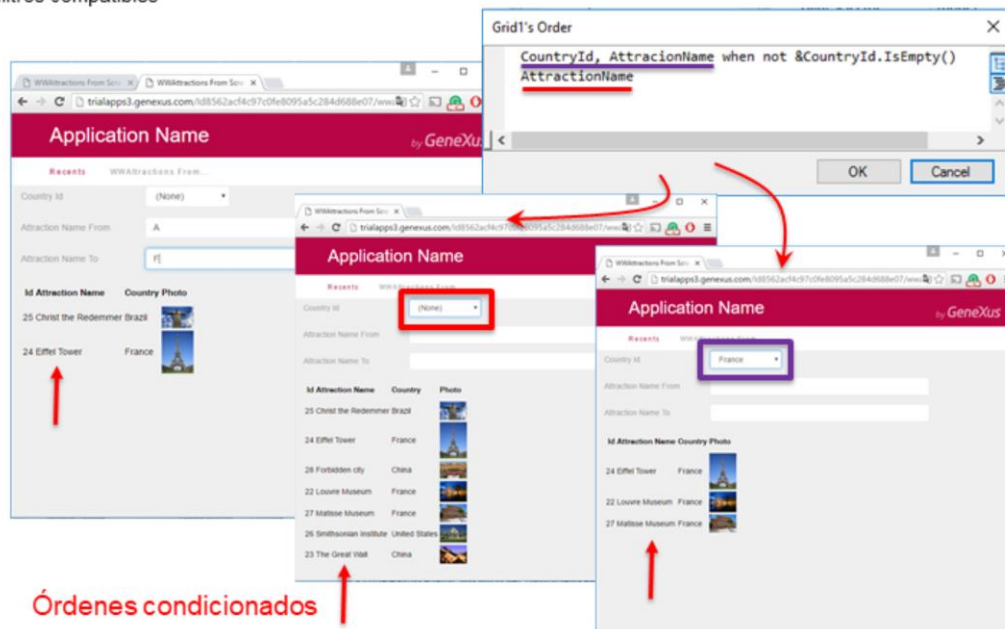


También tendríamos que agregar los filtros por nombre de atracción, que queremos que se sumen al otro filtro. Entonces... si en el listado filtrábamos en el for each con estas dos cláusulas where... en el grid las agregaremos como condiciones.

AttractionName mayor o igual al valor de la variable &AttractionNameFrom del form, que el usuario habrá ingresado, de deseirlo. Otra vez, si el usuario no ingresa valor en la variable, no queremos que este filtro se aplique. Entonces usamos la cláusula when. Esta cláusula también se puede utilizar en la cláusula where del for each, en forma completamente análoga.

Y agregamos el otro filtro.

## Órdenes y filtros compatibles



Ejecutemos nuevamente. Y elijamos ver las atracciones entre la A y la F.

Podemos pedirle al web panel que si el usuario elige un país, entonces ordene la información por CountryId y dentro de CountryId por AttractionName, y que si no, la ordene por AttractionName. Esto pensando en optimizar la búsqueda de los registros de la tabla.

Para ello, editamos del grid la propiedad Order y escribimos primero el orden, condicionado a que el usuario haya elegido un país en el combo. En ese caso se filtrará por ese país, pero además las atracciones saldrán listadas alfabéticamente para ese país. Y si el usuario dejó el combo con el valor "(none)", es decir, vacío, entonces se elegirá el orden siguiente, que es por AttractionName.

No nos detendremos en esto aquí. Lo dejamos apuntado únicamente para mostrar que también es posible condicionar la forma en que se quiere ordenar la información. Esto es idéntico en un for each.

Ejecutemos... Aquí ordenó por AttractionName. Y si elegimos Francia, ordenará por el Id de Francia y dentro de él por AttractionName.

En definitiva, siempre vemos las atracciones ordenadas alfabéticamente.

Si dentro de las atracciones de Francia, queremos las que se encuentran entre la A y la F, veremos que el grid se cargará filtrando por las tres condiciones que habíamos escrito.

## Resumiendo

Properties

General Class

Grid: Grid1

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

Tabla base

≈ For each Order Where

Id Attraction Name Country Photo

AttractionId AttractionName CountryName

≈ For each Tabla extendida

Hemos implementado un web panel en el que incluimos algunas variables para que el usuario les ingrese valor, y en el que insertamos un control Grid con atributos.

Los atributos corresponden a información de la base de datos, por lo que GeneXus entiende que debe ir a buscar esa información. Un grid con atributos es como un for each. Por ello contamos con la propiedad **Base Trn**, como la del for each, para especificar el nivel de la transacción cuya tabla asociada queremos recorrer. Llamamos a esa tabla, **tabla base del grid**. Si no la especificamos, como también sucede con un for each, GeneXus la infiere en base a los atributos que se utilicen. Aunque este caso no lo veremos.

Todos los atributos del grid deberán pertenecer, como en el caso de un for each, a la tabla extendida de esa tabla base.

Así como en un for each ordenamos la información con la cláusula Order y filtramos los datos a ser devueltos por la consulta con una o varias cláusulas where, para hacer lo mismo con el grid tenemos las propiedades Order y Conditions, respectivamente.

Carga del grid

< No action group selected >


MainTable Grid1

Country Id &CountryId ▾

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

Id	Attraction Name	Country	Photo
AttractionId	AttractionName	CountryName	

En procedimiento PDF

For each  
Main\_Code { Print Attractions  
endfor

Attractions			
Id	AttractionName	CountryName	AttractionPhoto
AttractionId	AttractionName	CountryName	AttractionPhoto

Ahora bien, en un for each programamos lo que queremos que se haga con cada registro que cumpla las condiciones, dentro de su cuerpo.

Por ejemplo, en el listado de las atracciones turísticas, el comando print Attraction enviaba a imprimir en la salida lo que en nuestro caso sería la línea del grid. En el caso del grid no hay que especificarlo. Esto se hace automáticamente.

Carga del grid: evento Load

< No action group selected >

MainTable Grid1

Country Id &CountryId

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

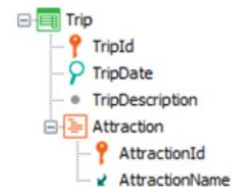
Id	Attraction Name	Country	Photo
AttractionId	AttractionName	CountryName	

En procedimiento PDF

For each  
*Main\_Code*  
endfor

&trips = Count( TripDate)  
Print Attractions

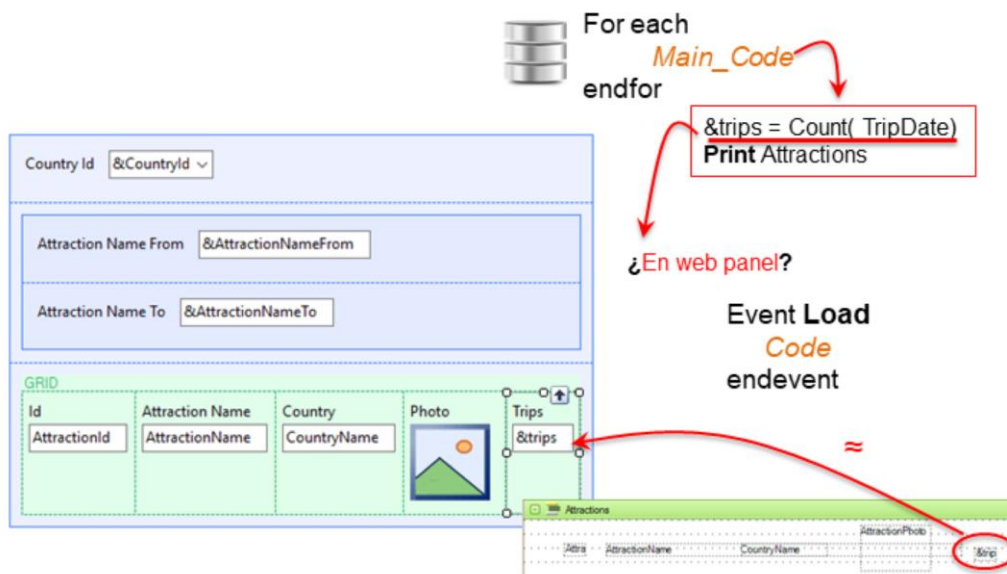
Attractions			
AttractionPhoto	AttractionName	CountryName	trips



Pero, por ejemplo, imaginemos que tenemos una transacción Trip que registra las excursiones que ofrece la agencia de viajes. En forma muy simplificada, imaginemos que de una excursión sólo se registra la fecha en la que se realizará y la descripción y luego se registran las atracciones turísticas que serán visitadas por esa excursión. Bien, entonces ahora supongamos que en el listado de atracciones turísticas queremos ver también la cantidad de trips que tiene asociados cada atracción. Para ello alcanzaba con definir una variable &trips, numérica, y asignarle dentro del cuerpo del for each (es decir, cuando el for each está posicionado en el registro de su tabla base que está por procesar) el resultado de contar los trips de esa atracción. Y colocar esa variable en el print block.

Carga del grid: evento Load

En procedimiento PDF



Para hacer lo mismo en el web panel hacemos botón derecho sobre el grid e Insert Attribute/Variable. Creamos la variable &trips, Numeric(4.0), y la movemos para que ocupe la posición que nos interese dentro del grid. Esto corresponde a haber insertado la variable en el printblock. Pero ¿dónde le decimos cómo se calcula? En el for each es dentro de su cuerpo. ¿Aquí?

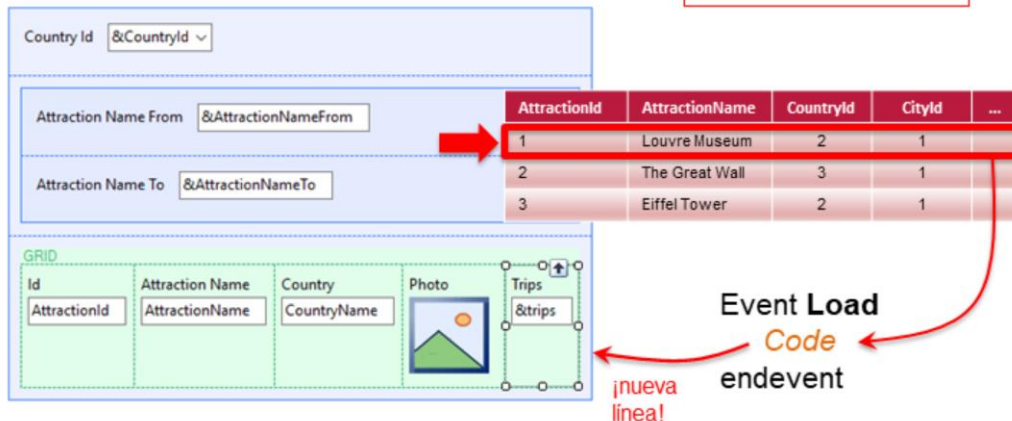
Contamos con el evento **Load**, del sistema. Allí dentro programaremos lo que queremos que se ejecute cuando se está posicionado en un registro de la tabla base del grid, inmediatamente antes de que la línea correspondiente sea cargada en el grid.

Carga del grid: Evento Load



```
For each  
Main_Code  
endfor
```

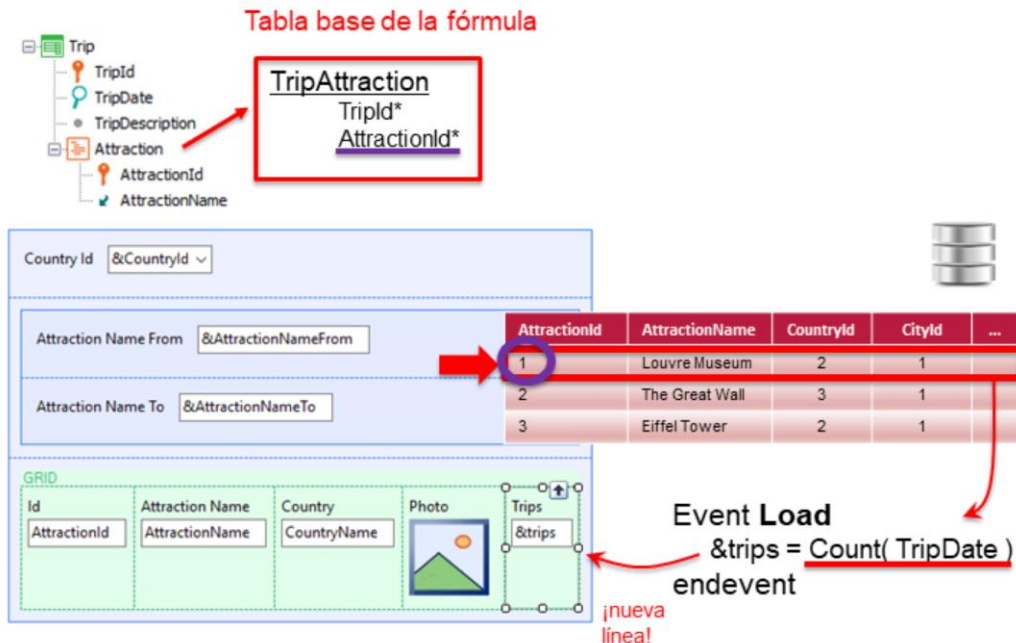
```
&trips = Count( TripDate)  
Print Attractions
```



Allí dentro programaremos lo que queremos que se ejecute cuando se está posicionado en un registro de la tabla base del grid, inmediatamente antes de que la línea correspondiente sea cargada en el grid.

En nuestro caso, allí es donde asignaríamos valor a la variable &Trips.

El evento Load se va a ejecutar automáticamente por cada registro de la tabla base del grid que cumpla con las condiciones de filtro, inmediatamente antes de que se agregue la línea al grid.



Es por esa razón que al ejecutarse su código se sabe que estamos trabajando con un registro de la tabla base y su extendida, y esta fórmula inline no contará todos los trips, sino solo aquellos de la tabla TripAttraction que correspondan a ese AttractionId, el del registro de Attraction que estamos a punto de cargar en el grid.

Obsérvese que pese a que el atributo TripDate es de la tabla Trip, GeneXus no escogerá la tabla Trip como tabla base de la fórmula, sino la tabla TripAttraction. No ahondaremos en esto aquí, pero TripDate está en la extendida de TripAttraction, tabla que tiene relación con la de Attraction. GeneXus buscó una tabla base que permitiera relacionar los datos.



DEMO



The screenshot shows a web application interface with a table of attractions. The table has columns: Id, Attraction Name, Country, Photo, and Trips. The data rows are as follows:

Id	Attraction Name	Country	Photo	Trips
25	Christ the Redemmer	Brazil		2
24	Eiffel Tower	France		2
28	Forbidden city	China		0
22	Louvre Museum	France		0
27	Matisse Museum	France		1
26	Smithsonian Institute	United States		1
23	The Great Wall	China		0

Below the table, it says "Total Trips: 6". A red box highlights the "Trips" column, and a red arrow points from the text "¿Variable fuera del grid para totalizar?" to the "Total Trips: 6" label.



Implementémoslo en GeneXus. Ya tenemos la transacción Trip creada. Vayamos a la sección de eventos del Web panel. En este combo, se nos ofrecen los eventos predefinidos, es decir, eventos del sistema que se producen en momento específicos, y a los que podremos programarles código.

Entre ellos, vemos el evento Load. Aquí dentro programaremos lo que queremos que se ejecute cada vez que se esté posicionado en un registro de la tabla Attraction, antes de cargar la línea en el grid.

Ejecutemos...

Ahora queremos agregar la suma de excursiones en las que las atracciones que se muestren en el grid en cada oportunidad estén incluidas. Es decir, la suma de los valores de esta columna.

Variable fuera del grid para totalizar

The screenshot shows a web panel with a grid of attractions. The grid has columns for Id, Attraction Name, Country, Photo, and trips. The first row is highlighted in red, showing the first attraction: Louvre Museum. A red arrow points from the first row of the grid to the first row of a data table. The data table has columns: AttractionId, AttractionName, CountryId, CityId, and ... The first row of the data table is highlighted in red, showing the first attraction: Louvre Museum. A red arrow points from the first row of the data table to the first row of the grid. The grid also has a 'Total Trips' variable labeled &totalTrips. To the right of the grid, there is a code block for the 'Event Load' event:

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

Below the code block, there are two variables: &trips and &totalTrips, both with a value of 2. The values are enclosed in red boxes.

Una forma eficiente de calcular el valor que deberá mostrar la variable es... cada vez que se vaya a cargar una línea en el grid, sumarle el valor de la variable &Trips de esa línea, al valor calculado hasta el momento en &totalTrips.

Es decir, en el evento Load, luego de calcular el valor de &trips, a &totalTrips asignarle el valor que contiene hasta el momento más el valor de la variable &trips.

Variable fuera del grid para totalizar

AttractionId	AttractionName	CountryId	CityId	...
1	Louvre Museum	2	1	
2	The Great Wall	3	1	
3	Eiffel Tower	2	1	

Country Id:

Attraction Name From:

Attraction Name To:

GRID

Id	Attraction Name	Country	Photo	trips
AttractionId	AttractionName	CountryName		&trips

Total Trips:

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

&trips: 1

&totalTrips: 3

Para la segunda línea a ser cargada, se calcula el valor de &trips y &totalTrips contendrá el valor anterior, al que se le sumará el de la variable &trips para esta segunda línea, y así sucesivamente.

Luego de que se cargue la última línea, la variable &totalTrips tendrá el valor deseado.

Pantallas interactivas
/ Web Panels
GeneXus

Refresh

¿Por qué muestra 9 en vez de 3?

Ejecutemos para probar.

Vemos dos cosas: primero, que está sumando correctamente; segundo: que por tratarse de una variable, es de entrada, por lo que el usuario puede modificarle el valor, lo que no tiene sentido. Así que primero que nada, configurémosla como Readonly.

Para ello nos posicionamos sobre la variable en el form y entre sus propiedades, modificamos la ReadOnly pasándola a True.

Puede llamarnos la atención, también, que &tips también es una variable, está mostrándose como readonly, y sin embargo no hicimos nada para ello. Las variables en los grids, cuando no se han programado eventos a nivel de las líneas, ni se los recorre por código, serán readonly. Volveremos a esto luego.

Pero además, veamos qué pasa si filtramos, por ejemplo, por Francia.

En vez de mostrarnos un total de 3, nos está mostrando 9, que es la suma del valor que nos mostraba antes, 6, más los 3 que ahora debería estar mostrando. ¿Por qué sucedió esto?

Al modificar una de las variables de filtro, el web panel volvió a cargar el grid. Es decir, volvió a consultar la tabla Attraction de la base de datos, y volvió a ejecutar el evento **Load** para cada registro que cumpliera los filtros. El problema es que la variable &totalTrips tendría que haberse reseteado, es decir, vuelto a cero, antes de lanzar la carga del grid.

¿Dónde hacemos esto? En el **evento Refresh**.

El orden en que los eventos quedan escritos no tiene la menor importancia. Aquí solamente se indica el código que se ejecutará al producirse cada uno de ellos.

## Evento Refresh

- Evento del sistema: se produce siempre que se va a refrescar la pantalla antes de ejecutarse el evento Load por c/linea a ser

```

Event Load
  &trips = count( TripDate )
  &totalTrips = &totalTrips + &trips
Endevent

Event Refresh
  &totalTrips = 0
Endevent

```

Application Name by GeneXus

Recents WWAttractions From...

Country Id (None)

Attraction Name From

Attraction Name To

Start (only the first time)

Refresh (once)

Load (7 times)

Id	Attraction Name	Country	Photo	Trips
25	Christ the Redemmer	Brazil		2
24	Eiffel Tower	France		2
28	Forbidden city	China		0
22	Louvre Museum	France		0
27	Matisse Museum	France		1
26	Smithsonian Institute	United States		1
23	The Great Wall	China		0
Total Trips				6

Presionemos F5.

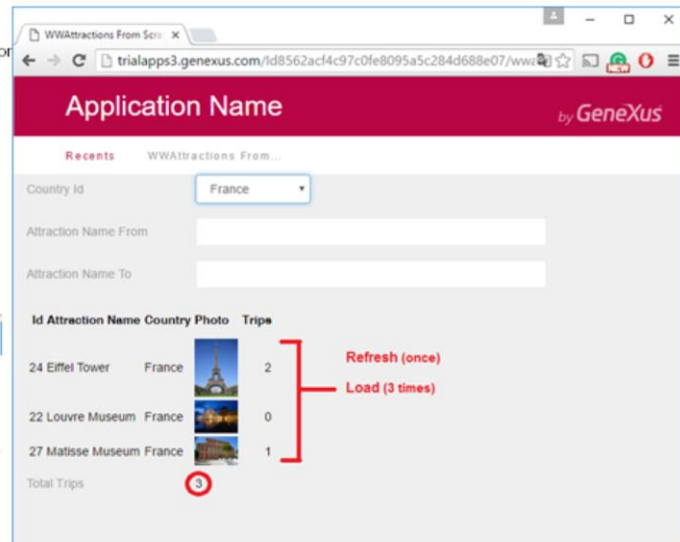
Podemos ver que ahora el total de tips aparece Readonly. Al ejecutar este web panel por primera vez, se dispararon tres eventos en forma consecutiva: el **evento Start**, que se ejecuta sólo al abrirse el web panel, la primera vez, el **evento Refresh**, que puso la variable en cero, y el **evento Load**, tantas veces como líneas se fueran a cargar en el grid. En este caso, fueron 7.

## Evento Refresh

- Al cambiar valor de variable implicada en las Condiciones de Carga

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

Event Refresh
  &totalTrips = 0
Endevent
```



Ahora, si elegimos filtrar por un país, por ejemplo, Francia, vemos que se está calculando bien el número de trips.

Al cambiar el valor de una variable que tiene efectos sobre las condiciones que deben cumplir los registros para cargarse en el grid, se vuelve a disparar el evento **Refresh** (y por tanto la variable `&totalTrips` se vuelve a poner en cero) y la ida a la base de datos para volver a filtrar y cargar los registros en el grid. Por tanto el **Load** se vuelve a disparar por cada atracción de Francia a ser cargada.

Pantallas interactivas
/ Web Panels
GeneXus

### Work With Attractions del Pattern

Acciones sobre los datos: Insertar, Actualizar, Eliminar una atracción

Attraction

```
parm(in:&Mode, in:&AttractionId);
AttractionId = &AttractionId if not &AttractionId.IsEmpty();
```

Dominio TrnMode: Enum Values    Insert, Insert, INS, Update, Update, UPD, Delete, Delete, DLT, Display, Display, DSP

Ahora bien, si observamos el web panel que tenemos implementado hasta el momento, podremos apreciar que se va pareciendo al objeto que había creado el Pattern Work with aplicado a la transacción Attraction.

Este objeto era, evidentemente, un web panel.

Lo más interesante es que además de permitir filtrar los datos del grid, el Work With ofrece ejecutar acciones sobre los datos. Por ejemplo, poder actualizar los datos de una atracción, o eliminar la atracción, así como insertar una nueva.

Para ello el pattern insertó dos controles a nivel de las líneas del grid, y uno fuera. En cualquiera de los tres casos, la acción asociada a cada control consiste en llamar a la transacción Attraction, enviándole como parámetro el **modo** en que debe abrirse la transacción, esto es, si se la llama para actualizar, eliminar o insertar. En los primeros dos, Update o Delete, como corresponderán a eventos de una línea, se contará con el id de la atracción de la línea, que se le enviará como segundo parámetro a la transacción, de modo de actualizar los datos de **esa** atracción, o eliminar **esa** atracción. En el caso del Insert, control fuera del grid, se le enviará 0 como segundo parámetro, ya que las atracciones en Insert se autonumeran.

Es por ello que el pattern modificó la transacción Attraction, agregándole, entre otras cosas, la regla Parm que como vemos, recibe dos variables: la variable &Mode es una variable estándar en transacciones, Character de 3, que acepta uno de cuatro valores, especificados en el dominio enumerado TrnMode: Insert (INS), Update (UPD), Delete (DLT) y Display (DSP).

Recibiendo uno de estos cuatro valores, la transacción sabrá en qué modo se le pide que se abra.

Y por otro lado, recibirá como segundo parámetro el id de atracción, en la variable &AttractionId, para cuando se quiera hacer update, delete o display.

## Acción de Update a nivel de las líneas del grid

Country Id

Attraction Name From

Attraction Name To

GRID

Id	Attraction Name	Country	Photo	trips	Update
AttractionId	AttractionName	CountryName		&trips	

Total Trips

Event Start  
 &Update.FromImage(updateIcon)  
 Endevent

Event &Update.Click  
 Attraction( TrnMode.Update, AttractionId )  
 Endevent

updateIcon X

Images

New Image

Image

updateIcon24.png  
 (Default)  
 Size: 24x24 px

Ingresamos imagen en la KB

&Update tipo Image

En nuestro web panel implementaremos una de estas acciones sobre las atracciones. Por ejemplo, la de Update. La intención es mostrar un ejemplo de acciones sobre los datos.

Tendremos que insertar un control en el grid. En el caso del pattern se inserta una variable character a la que se ha llamado update, y a la que se le asigna el texto "UPDATE" que vemos en ejecución. Pero nosotros elegiremos insertar una imagen, que antes que nada debemos insertar en la KB.

La llamamos updateIcon.

Ahora vamos al web panel y arrastramos de la toolbox el control Attribute/Variable a la última columna del grid, definimos la nueva variable como &Update, pero no de tipo character, sino Image.

Le quitamos el título para que no figure como título de columna, y nos resta cargar esa variable con la imagen que acabamos de insertar en la KB. ¿Dónde lo hacemos?

Si la imagen variara por línea del grid, lo haríamos en el evento Load, pero la imagen será la misma para cada línea, y no variará nunca, así que una buena opción es hacerlo en el **evento Start**, que se ejecutará una única vez, cuando se abra el web panel, y ya no más.

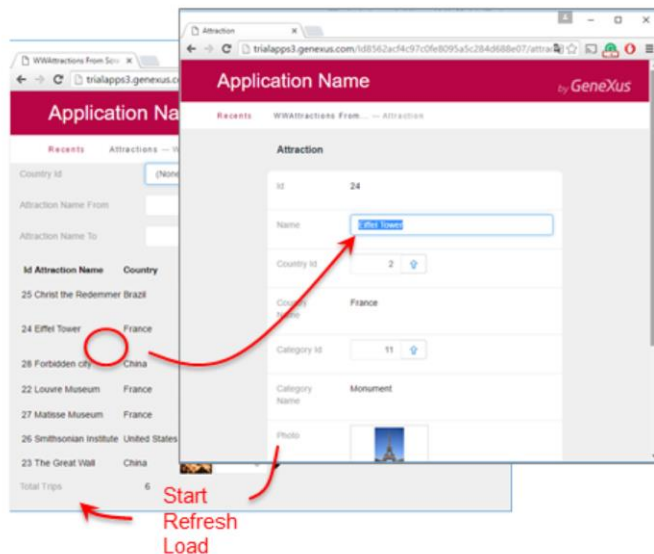
Ahora nos resta asociar un evento a esa imagen, de manera tal de que cuando el usuario haga clic sobre ella, se produzca ese evento y se ejecute su código, en el que invocaremos a la transacción Attraction.

Hay varias alternativas para realizar esto. Una de ellas es utilizar el evento click del control variable &Update.

De este modo, cuando el usuario haga clic sobre la imagen para una línea, se ejecutará el código que escribamos dentro de este evento. Lo que queremos hacer, en ese caso, es invocar a la transacción Attraction, pasándole el modo Update, es decir, el valor Update del dominio enumeado TrnMode que vimos antes, y el valor de AttractionId correspondiente a la línea del grid donde se hizo clic.



## DEMO



Ejecutemos para probar.

Primero observamos que la columna de la variable &Trips aparece ahora como editable. No la habíamos definido como Readonly explícitamente porque al ejecutar vimos que ya lo estaba. Como dijimos antes, las variables del grid en principio se colocan como readonly, salvo que se defina algún evento a nivel de las líneas, como ha sido nuestro caso, o salvo otras excepciones que ahora no veremos. Configurémosla como Readonly para la próxima ejecución.

Seleccionemos por ejemplo el país Francia. Y ahora hagamos clic sobre la imagen de update para la Torre Eiffel. Vemos cómo se llama a la transacción en update. Modifiquemos algo... por ejemplo pasemos de mayúscula a minúscula la T de Tower.

Confirmamos... y como el pattern work with, aunque no lo vimos, ha agregado a la transacción un comando Return, retorno, para volver a quien la llamó, es que se vuelve al web panel. Este comando Return es como hacer una invocación al web panel por primera vez, por lo que en éste se ejecutará el evento Start, seguido del Refresh y del Load tantas veces como registros se vayan a cargar.

¿Cuándo no es posible no incluir columna en el grid?

The screenshot shows the GeneXus IDE interface. At the top, there are tabs for 'Start Page', 'WebForm', 'Rules', 'Events', 'Conditions', and 'Variables'. Below these, there's a 'MainTable' and a 'Grid1' tab. The main workspace displays a web form with fields for 'Country Id', 'Attraction Name From', 'Attraction Name To', and a grid. The grid has columns for 'Id', 'Attraction Name', 'Country', 'Photo', and 'Trips'. A red arrow points from the 'Id' column in the grid to the 'Visible' property in the 'Properties' window on the right. The 'Properties' window shows the 'Visible' property set to 'False'. A text box with the code `Event &Update.Click`  
`Attraction( TrnMode.Update, AttractionId )`  
`Endevent` is overlaid on the grid. A blue arrow points from the 'AttractionId' parameter in the code to the 'AttractionId' column in the grid. A text box on the right says: '¡Tiene que estar en el grid! Si no la queremos ver, podemos ocultarla'.

¿Qué pasaría si no hubiéramos colocado el atributo `AttractionId` en el grid? Al hacer clic sobre la imagen para actualizar, ¿qué `AttractionId` se habría enviado como parámetro a la transacción? No tendría ese valor para enviar.

Como lo vamos a usar en un evento a nivel de las líneas, que se va a disparar después de que las líneas se hayan cargado, no podemos quitar `AttractionId` del grid. Es que aquí ya no se está más en la base de datos. El grid almacenó, al cargarse con el `Load`, todos los valores de sus columnas y nada más. Un evento posterior trabajará únicamente sobre los datos cargados en el grid. Entonces, lo que podemos hacer si no queremos ver esa columna en el grid, es ocultarla. Seguirá estando presente, pero invisible. Para ello utilizamos la **propiedad Visible**.



Videos

[training.genexus.com](http://training.genexus.com)

Documentation

[wiki.genexus.com](http://wiki.genexus.com)

Certifications

[training.genexus.com/certifications](http://training.genexus.com/certifications)