

## Web Panel object. Loading data and events

*GeneXus*<sup>™</sup>

## Web Panel with base table and grid

**Application Name**

Country Id (None) ▾

Attraction Name From

Attraction Name To

Attraction Name	Country Name	Attraction Photo	Trips
<a href="#">Christ the Redeemer</a>	Brazil		2
<a href="#">Eiffel Tower</a>	France		2
<a href="#">Forbidden City</a>	China		0
<a href="#">Great Wall</a>	China		0
<a href="#">Louvre Museum</a>	France		1
<a href="#">Matisse Museum</a>	France		2
<a href="#">Smithsonian Institute</a>	United States		1

Total Trips 8

Start Page × WWAttractionsFromScratch ×

Web Form | Rules | **Events** | Conditions | Variables

Load ▾

```

1 | Event Load
2 |   &trips = count(TripDate)
3 |   &totalTrips = &totalTrips + &trips
4 | - Endevent
5 |
6 |
7 | Event Refresh
8 |   &totalTrips = 0
9 | - Endevent
10 |
11 |
12 | Event Start
13 |   &update.FromImage(UpdateIcon)
14 | - Endevent
15 |
16 |
17 | Event &update.Click
18 |   Attraction(TrnMode.Update, AttractionId)
19 | - Endevent
20 |
```

We were in the process of building our web panel – WWAttractionsFromScratch–, where we saw how to condition and sort the data shown on the grid, which had base table.

For that case, we also saw how to load a grid variable for each line with the Load event. We used the Refresh and the Start events, and we programmed an event at the line level to call the Attraction transaction in update mode.

In addition, we had seen that since we had a grid we could use the Load event of the grid instead of the generic one, to anticipate any future need to insert another grid. Also, that we had a grid Refresh. For now, to avoid confusion, **we'll** stick to the generic ones.

## Modify a grid line

The screenshot shows a web form design in GeneXus. At the top, there is a dropdown menu for 'Country Id' with the value '&CountryId'. Below it are two text input fields: 'Attraction Name From' with value '&AttractionNameFrom' and 'Attraction Name To' with value '&AttractionNameTo'. The main part of the form is a 'GRID' with the following columns: 'Id' (containing '&AttractionId'), 'Attraction Name' (containing '&AttractionName'), 'Country' (containing '&CountryName'), 'Photo' (with a landscape image icon), 'Trips' (containing '&trips'), and a new column '&newTrip' (with a landscape image icon). At the bottom, there is a 'Total Trips' label with the value '&totalTrips'.

The screenshot shows the 'Events' window in GeneXus for the '&newTrip.Click' event. The code is as follows:

```

1 | Event Load
2 |   &trips = count( TripDate )
3 |   &totalTrips = &totalTrips + &trips
4 | -Endevent
5 |
6 | Event Refresh
7 |   &totalTrips = 0
8 | -Endevent
9 |
10 | Event Start
11 |   &update.FromImage(updateIcon)
12 |   &newTrip = "New Trip"
13 | -Endevent
14 |
15 | Event &update.Click
16 |   Attraction( TrnMode.Update, AttractionId )
17 | -Endevent
18 |
19 | Event &newTrip.Click
20 |   |
21 | -Endevent

```

**Let's** now add another action at the line level. But this action will not call another object with interface as it happened in the case invoking the Attraction transaction.

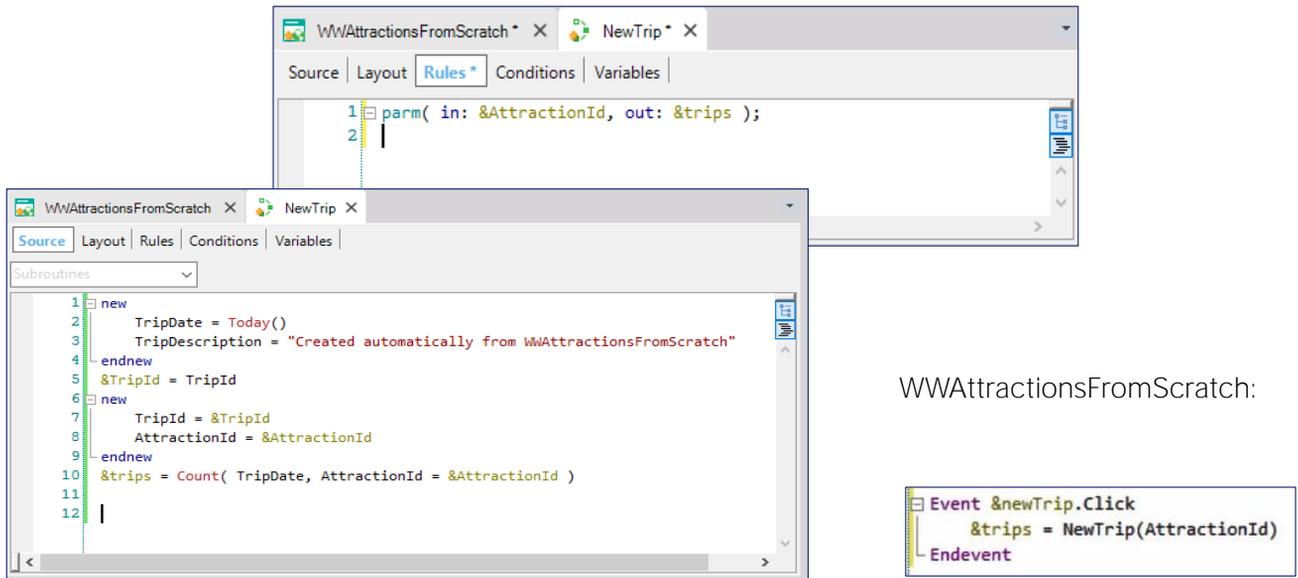
We can imagine that, for instance, we will enable the possibility to create, from a line (an attraction), a new trip in the database, with that attraction.

First we add a new variable –called newTrip and 10 character- to the grid.

**Let's** change it to ReadOnly. We want it to contain the “**New Trip**” text, so we assign it in the Start event because there will be no variations by line:

What do we want to do when the user clicks on New Trip?

A new trip is added to an attraction



For example, call a procedure to which we pass the identifier of the **line's** attraction and create the trip with that attraction.

Note that we implemented it with the new command, to create a record directly in the Trip table and another one in the TripAttraction table. This solution shows how the commands are used. However, a more recommended solution would be to use the business component of Trip for inserting. In this course we did not see how to load a two-level business component, which is very simple to do.

We will see that, following the insertion of the header and line of Trip, we calculate the number of trips that include this attraction. Since the inline formula is triggered without positioning on any table (it is "loose" inside the code), we must indicate the explicit condition that we want to filter the trips of the attraction.

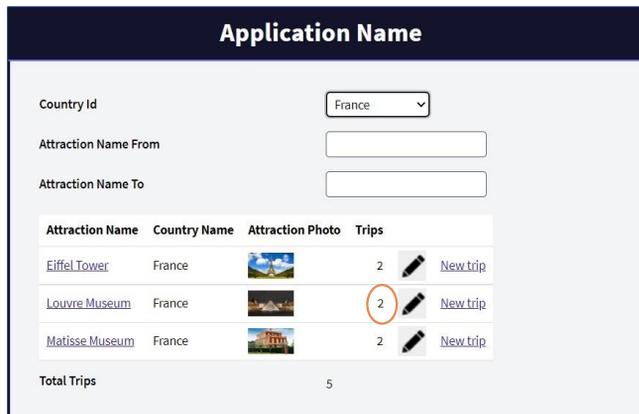
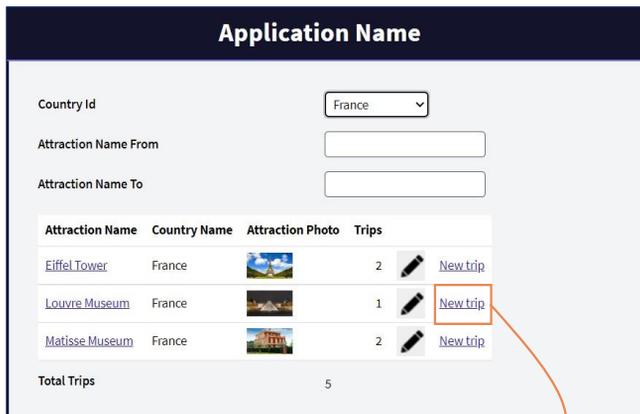
And that is the value returned when this procedure is called.

So, from the click event of the &newTrip control (variable), we invoke this procedure and pass the AttractionId of the line from where the click is done. Since the parameter returned is the one we need to show in the &Trips column, we assign it directly to the variable:

It's only logical that, upon the **user's** click on New Trip.

For that line, in the Trips column, the value shown should be the one it had prior to the click plus one. That is the line should be refreshed.

A new trip is added to an attraction



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

Only the line was refreshed.  
But the total was not updated!

Let's execute and try.

For example, let's filter by France, and for the Louvre museum attraction, which for the time being is not part of any trip. Let's click on New Trip:

We can see that the line has been automatically refreshed:

Now it shows number of Louvre trips: 1.

However, the total count was not refreshed. It should be 4 but it keeps the previous value.

Why?

Upon executing the click event associated with the &NewTrip variable, only its code was executed. And because inside it a variable of the grid was assigned a value, the line was refreshed –and only that line. No other event was executed, not even the Load event.

## Refresh command

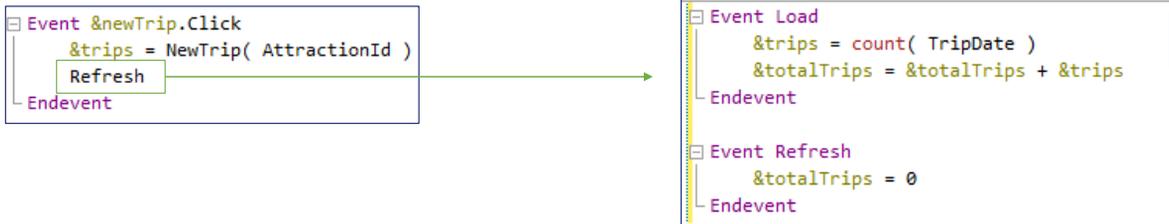
### Alternative 1

```

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

```

### Alternative 2



Therefore, we have two options to keep the total of trips loaded on the grid updated when this event occurs.

One possibility is to add one to &totalTrips, because the procedure only added one trip to that attraction.

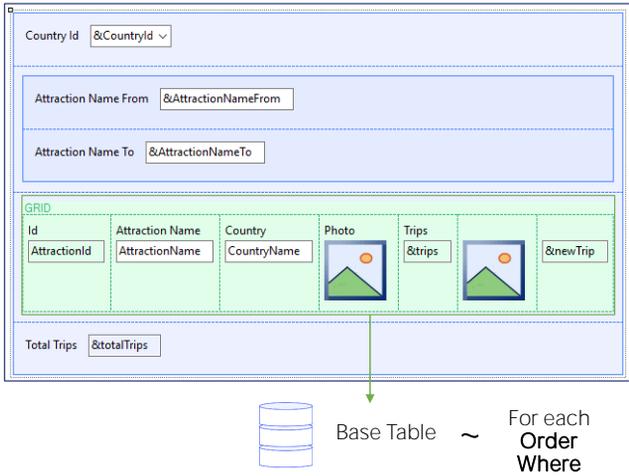
But if the procedure is later changed with the addition of more trips, then we will have to remember to do the change in a way consistent with this event.

A better solution is to request the web panel to execute the Refresh and Load events again. To do this we use the Refresh command.

When we do this the grid will be loaded in the data base again.

## Summing up: Order of events execution

Web Panel with a grid (with attributes)



1st Time:

Start

Refresh

Load (N times)

Nth Time:

User / Control Event

Let's try executing this:

We add a trip for the Matisse museum.

We can see that now it has updated it all. It executed the Refresh and Load events again. The latter was executed seven times, once for every line to be loaded.

Let's review what we have seen so far.

For the case of a web panel with a grid with attributes, GeneXus will understand that the grid has an associated base table, that is: a table to be navigated in order to load the grid lines. It will load a line for each record in that base table. To filter, we have the Conditions property of the grid, and to sort we have the Order property. A grid with a base table is analogous to a For each.

For the system events produced in the web panels we can program code to be executed at the time when they are triggered. We saw three of such events that are always triggered when a web panel is opened, that is, in the first execution:

The Start is triggered only once. There we can, for example, initialize variables.

The Refresh is triggered prior to the loading of the screen data. Following this event, the access to the database will take place in order to bring the data of the base table and its extended table.

A Load event is produced for each record on the base table that is about to be loaded on the grid. Therefore, this will be the point where all the actions we want executed before the line is actually loaded on the grid will have to be programmed. The data loaded on the grid is exclusively the data from the visible or invisible columns included in it.

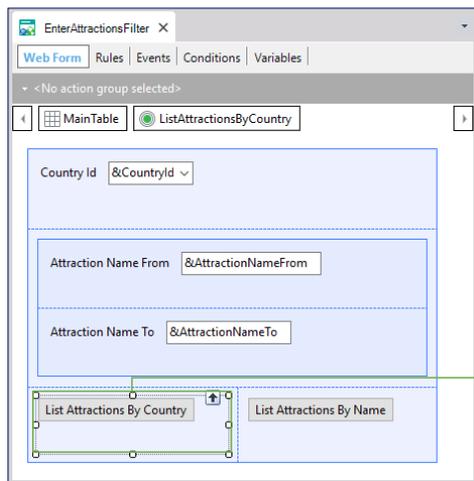
After this the web panel will be loaded with the information obtained from the database and it is then disconnected from it.

Now the user starts to interact with the screen. For example, he or she changes the value of one of the variables used to filter the **Grid's** data. A Refresh is automatically made, by which the server is invoked again to execute the Refresh and the Load event. In this case, if CountryId corresponds to that of France, the Refresh will be executed and then the Load for each attraction in France. Note that the biggest difference with the first run is that here the Start is not run again.

However, web panels allow you to define other events, which will also be triggered after the first run; that is, once the web panel has been loaded and always as a result of the **user's** action.

For example: the Click event that we programmed on this image, or the Click on New Trip.

## Summing up: Order of events execution



1st Time:

Start

Refresh

Load (N times)

Nth Time:

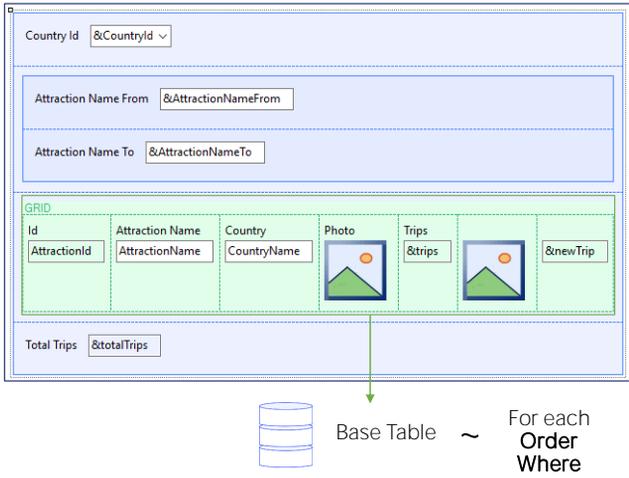
```

Event 'List Attractions By Country'
  AttractionsList(&CountryId)
  //AttractionsReport(&CountryId)
-Endevent
  
```

Or even in the web panel we defined several classes ago, the user event we associated with the button that we called.

## Summing up: Order of events execution

Web Panel with a grid (with attributes)



1st time

Start

Refresh

Load

Nth time:

User / Control Event

Refresh command

These events are known as user events or control events (as the Click event we saw).

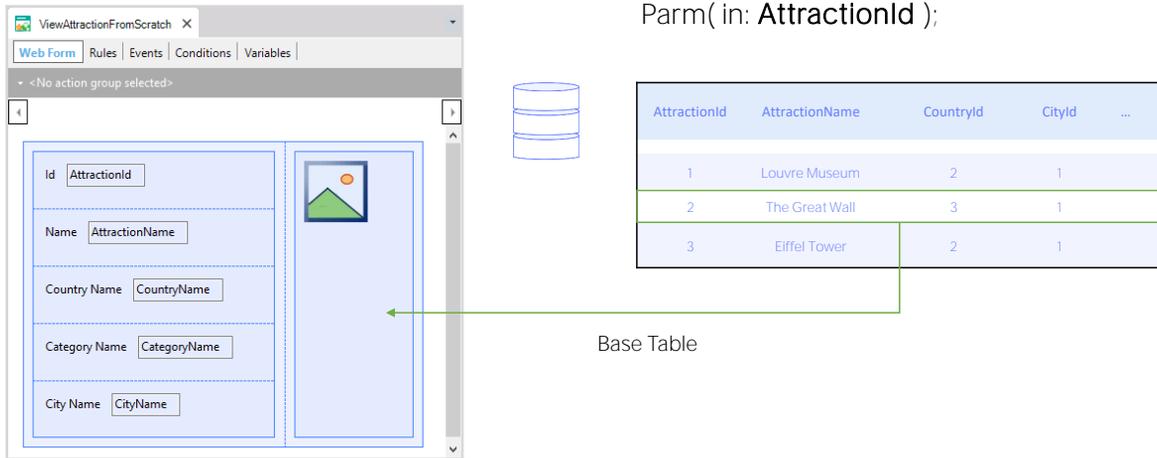
When the user causes one of these events only its code is executed, without screen refreshing. The only exception occurs when the event takes place at the level of a line in the grid, as in the case we saw of &newTrip, where a variable from the same grid –in our case &Trips– is assigned inside its code. In this case, the value of the variable is refreshed on screen, for that line, so it is shown updated.

When we need the Refresh to be executed again and the lines loaded again on the grid from the database (for example to update the total of lines), we may write the Refresh command in the event.

The Refresh command causes the execution of the Refresh and Load events.

## Summing up and more: Web Panel with base table and without grid

### Web Panel without a grid but with attributes in the form



Parm( in: **AttractionId** );

We studied the case of a web panel with a grid with attributes. But, what if we have a web panel without grid and with attributes in the form?

**Let's** suppose that we want to click on the name of the attraction and call a web panel to show all data on that attraction.

To do that we have already implemented this web panel where we have inserted, in its form, the attributes of an attraction that we want to show. We also wrote a parm rule to receive a parameter. We can see that, instead of receiving in a variable we decided to receive in the AttractionId attribute.

What we want is that when this web panel is called from the Click event of AttractionName in our other web panel to pass the Id of the attraction of the grid line, GeneXus automatically go to the attractions table to find the attraction with that Id, and show, for that attraction, the information on the attributes located in the form.

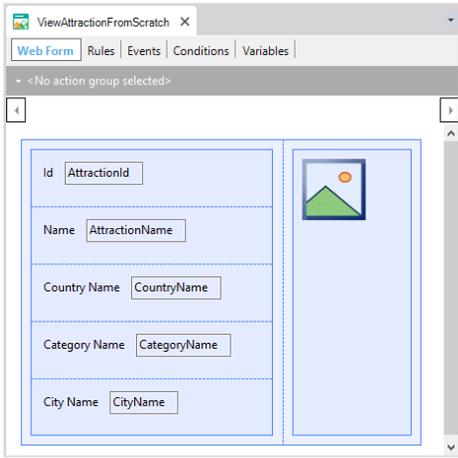
In sum, a web panel that has no grid but containing attributes on the form will also have base table. How does GeneXus determine it if we **don't** have base transaction to indicate? That is something we will not see in this course, but **it's** something similar to the case of a For each where no Base Transaction is indicated.

But in this case, since we **don't** have a grid, only ONE record from that base table and its extended table will have to be loaded. Where do we indicate the filter to enable the return of that record?

In our case, it received in the AttractionId attribute. There we specify the automatic filter to indicate which record from the base table must be taken.

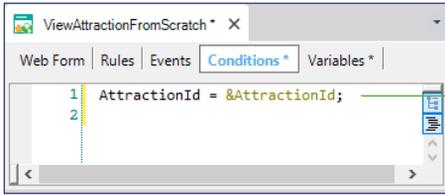
## Summing up and more: Web Panel with base table and without grid

Web Panel without a grid but with attributes in the form



Base Table

Parm( in: **&AttractionId** );



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

If we need to establish another type of condition, or if we received in variable instead of in attribute, we have the Conditions tab to establish the filter.

### Load event in Web Panel with base table and without grid

### Application Name

Attraction Name	Eiffel Tower	
Country Name	France	
Category Name	Monument	
City Name	Paris	

#### Event Load

Order: AttractionId  
Index: IATRACTION  
Navigation filters: Start from: AttractionId = @AttractionId  
Loop while: AttractionId = @AttractionId  
Join location: Server

- Attraction (AttractionId)
- Country (CountryId)
- CountryCity (CountryId, CityId)
- Category (CategoryId)
- count\_TripDate.navigation
- TripAttraction (AttractionId)
- Trip (TripId)

#### Web Form

Actions: Update Delete

MainTable AttributesTable trips

Name	AttractionName
Country Name	CountryName
Category Name	CategoryName
City Name	CityName
Trips	&trips

```
1 | Event Load  
2 |   &trips = Count(TripDate)  
3 | Endevent
```

If here we would also like to show the number of trips in which the attraction is included, we would need the variable &trips, but where do we load it this time? In the Load event too! **Let's** try it.

**It's** the same as if there was a grid. **Let's** see what the navigation list shows. The Load event is associated with the Attraction table, which is the base table. But instead of running through it all, it just keeps this record.

## Load event in Web Panel with base table and without grid

```
1 param(in:&AttractionId);
```

Event Load

Order: AttractionId  
Index: IATTRACTION

Navigation filters: Start from: FirstRecord  
Loop while: NotEndOfTable

Join location: Server

```

-Attraction ( AttractionId )
  -Country ( CountryId )
    -CountryCity ( CountryId, CityId )
      -Category ( CategoryId )
        -count( TripDate ) navigation
          -TripAttraction ( AttractionId )
            -Trip ( TripId )

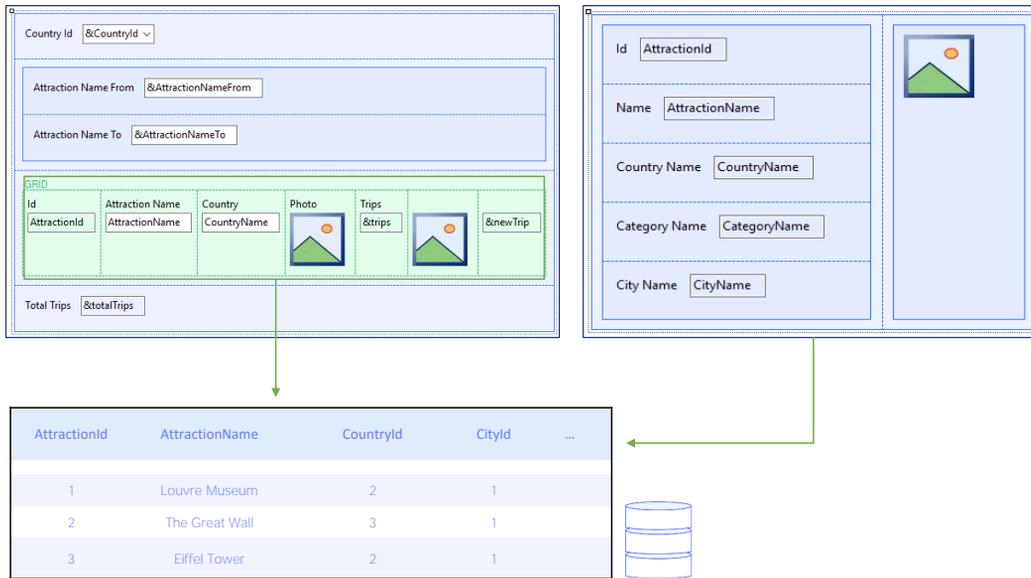
```

Name	AttractionName
Country Id	CountryId
Country Name	CountryName
Category Name	CategoryName
City Name	CityName
Trips	&trips

For example, what would have happened if we had not received the data in the attribute, but in a variable, but we had failed to specify the filter in the conditions? Which attraction values will be loaded on the screen? **Let's** look at the navigation list before running. It will run through the entire base table and make a load for each record. However, since there is no grid, for each record in which it is positioned, it will overwrite the attributes displayed on screen. What we will finally see is the last record of the run which, since it is done by primary key, corresponds to the last attraction entered. **Let's** see. Forbidden city was the last attraction entered.

If we now add the filter condition and look at the navigation list, we see that it will filter again, keeping only one attraction. We run it.

## Web panels with base table



So far we have seen two web panels with base table

- the first one with a grid. There, the base table of the grid is the base table of the web panel, that is: the table that the web panel automatically decides to navigate to retrieve the information to be loaded on screen.
- the second one without grid, but with attributes. There, the base table of the web panel is found from those attributes.

There is also a mixed case that happens when we have a web panel with a grid with attributes, but there are also loose attributes in the so-called fixed part of the panel; that is, outside the grid.

## Web panels with base table

Country Name

Attraction Name From

Attraction Name To

Attraction Id	Attraction Name	Country Name	Attraction Photo	Trips	&newTrip
AttractionId	AttractionName	CountryName		&trips	

Total Trips

Event Load

Order: CountryId, AttractionName  
No index

Navigation filters: Start from: CountryId = @CountryId  
Loop while: CountryId = @CountryId

Constraints: AttractionName >= &AttractionNameFrom WHEN not &AttractionNameFrom.isempty()  
AttractionName <= &AttractionNameTo WHEN not &AttractionNameTo.isempty()

Join location: Server

- Attraction (AttractionId)
- Country (CountryId)
- count(TripDate).navigation
- TripAttraction (AttractionId)
- Trip (TripId)

WWAttractionsFromScratch x ViewAttractionFromScratch x

Web Form | Rules | **Events** | Conditions | Variables

CountryName.Click

- Event Load
- &trips = Count(TripDate)
- Endevent
- 
- 
- Event CountryName.Click
- ViewCountryInfo(CountryId)
- Endevent
- 
- 
- Event Start
- CountryId.Visible = False
- Endevent
- 

ViewCountryInfo x

Web Form | **Rules** | Events | Conditions | Variables

- parm( in: CountryId );
- 

For example, **let's** suppose that by clicking on the name of the tourist attraction country we want to see some general information about the country, and all its tourist attractions. **It's** similar to the panel we created, but here we are not allowing to filter by country, but only by name of attraction.

Looking back at how we implemented it, from the View panel of the attraction we programmed the Click event over the country name (note that we had to set the CountryId

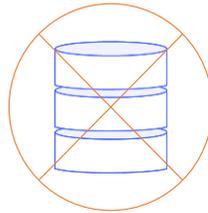
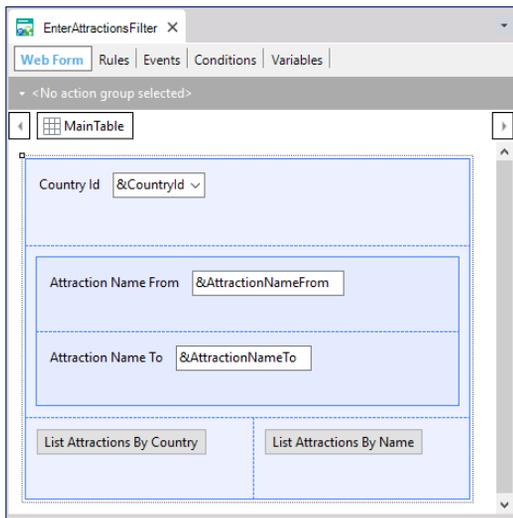
attribute as invisible to be able to make this invocation. Why?) and there we called this new Web Panel, which we created by saving the one we had under this other name. As we can see, this new web panel is very similar, except that it receives a parameter: the country identifier.

Instead of having the variable CountryId, it has the CountryName attribute. Here the order of the grid –which in the first one was conditional– is fixed by CountryId, AttractionName because the country will be instantiated and sent by parameter. And the conditions for loading the grid, which previously included one for filtering by country, no longer do so. When it is received in the attribute, it will act as an automatic filter.

If we look at the navigation list, we see that it still says that the Load event used to load the data to be displayed on screen still runs through the same Attraction table.

We were able to remove the attribute from the grid and place it in the fixed part, because the country will always be the same, for all the attractions, since we were receiving it by parameter in the attribute. We could remove it from the grid, where it **won't** make sense anymore.

## Web panels without base table



We will now see the case of web panels without base table, that is: web panels that do not have a query to the database programmed automatically.

The case most evident is when no query is made at all to the database, as in the case of our initial web panel, which only requested data from the user and called other objects.

## Web panels without base table

The image displays a web panel design and its associated code. The design features a search interface with filters for Country Id, Attraction Name From, and Attraction Name To. Below these filters is a grid with columns for Attraction Id, Attraction Name, Country, Photo, Trips, and a new Trip button. A 'Total Trips' label is also present. The code on the right shows events for Refresh, Start, Update Click, New Trip Click, and Attraction Name Click, all using variables like &totalTrips, &newTrip, and &AttractionId.

Only variables!

```

Event Refresh
  &totalTrips = 0
Endevent

Event Start
  &Update.FromImage(updateIcon)
  &newTrip = "New Trip"
Endevent

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

Event &newTrip.Click
  &trips = NewTrip( &AttractionId )
  Refresh
Endevent

Event &AttractionName.Click
  ViewAttractionFromScratch( &AttractionId )
Endevent

```

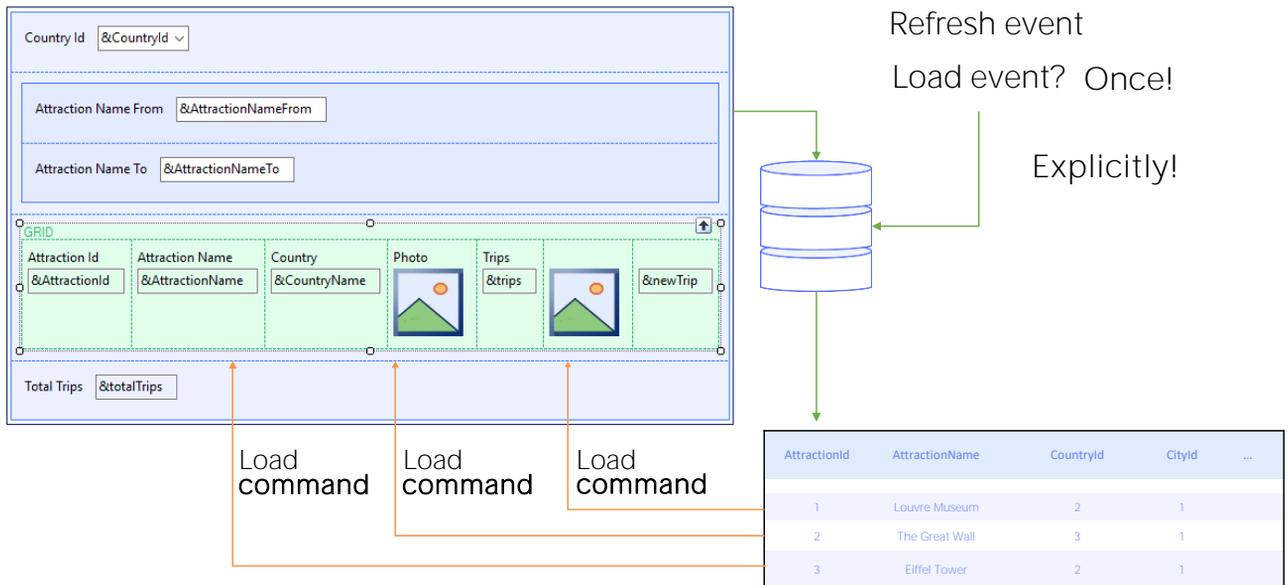
But we can also have a web panel that does query the database, with that query and load on screen fully in hands of the developer.

The web panels we implemented with base table could also have been implemented in this manner.

**Let's** see the case of the web panel that shows the attractions in the grid, but this time, instead of having attributes in the grid we will have variables.

So, in the events we have to change the invocations we had, where we passed the AttracionId attribute in the &AttractionId variable.

## Web panels without base table



If we now have only variables, how does GeneXus know that it has to navigate the Attraction table and its extended table to load a line in the grid per record?

In fact, it **doesn't** know that. The load of this grid will not be automatic as in the case where we used attributes.

This means that the Load event will not be executed when it has gone to navigate a table for each record where we are positioned at a given moment. This is because we are not positioned anywhere!

However, the Load event will indeed be triggered, except that it will be triggered only once after the Refresh and not being in the database at all.

In that triggering, in that execution, we will have to program the grid load manually.

We will have to explicitly request access to the database (in our case by going to the Attraction table) and indicate that every time that a line is loaded.

In order to instruct it to insert a new line in the grid with the values that the variables corresponding to the columns have at the time we have the Load command –not the event but the command. Every time that a Load command is inside the Load event a line is inserted in the grid.

## Web panels without base table

```

Event Load
For each Attraction
  order CountryId, AttractionName when not &CountryId.IsEmpty()
  order AttractionName
  where CountryId = &CountryId when not &CountryId.IsEmpty()
  where AttractionName >= &AttractionNameFrom when not &AttractionNameFrom.IsEmpty()
  where AttractionName <= &AttractionNameTo when not &AttractionNameTo.IsEmpty()
  &AttractionId = AttractionId
  &AttractionName = AttractionName
  &CountryName = CountryName
  &AttractionPhoto = AttractionPhoto
  &trips = count( TripDate )
  Load
  &totalTrips = &totalTrips + &trips
endfor
Endevent

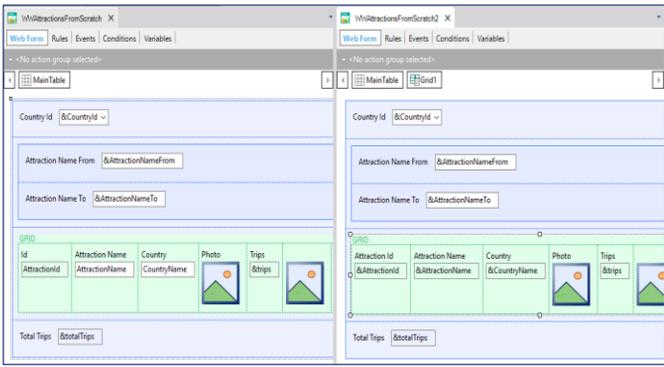
```

Attraction Id	Attraction Name	Country	Photo	Trips	
&AttractionId	&AttractionName	&CountryName		&trips	 &newTrip

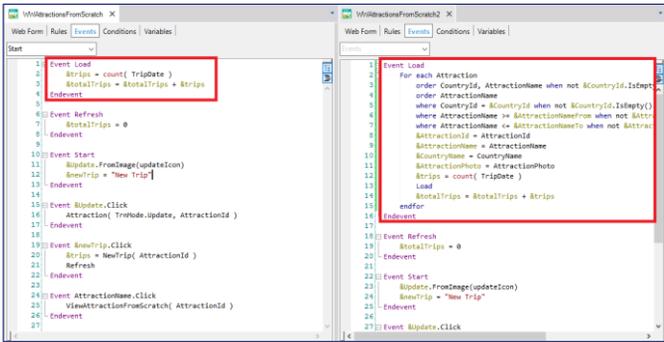
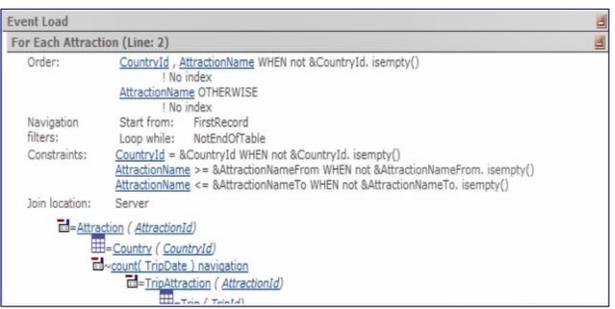
AttractionId	AttractionName	CountryId	CityId	...
1	Louvre Museum	2	1	
2	The Great Wall	3	1	
3	Eiffel Tower	2	1	

What we will have to do then is to program, inside the Load event, the access to the Attraction table with a For each, where we will specify the order clauses, and, in the Where clauses, the conditions that the data must fulfill. And for each record compliant with those filters we will load the grid variables with the values of that attraction.

When all the variables are loaded we then write the Load command to add a line in the grid with those values.



### Navigation Report:



Control Name	Grid1
Collection	
Base Trm	
Order	
Conditions	

Grid without base table

Here we have this web panel that is already implemented. We did a Save as of the one we had with attributes and change it with variables.

And we also change the events, as we just saw.

Let's now filter by France for example.

If we note the navigation listing of the web panel without base table we have:

We can see the For each in the Load, indicating the navigation.

One remark: to be able to say that the grid and the web panel **don't** have a base table, it is not enough that there are only variables on the screen and no attributes. There should be no attributes in the Order of the grid, nor in the conditions; of course there should be no base Transaction indicated, nor attributes outside For Each commands in the events, like here.

# Load event: More than one grid

The screenshot shows a GeneXus web form editor for a form titled "ViewCountryInfo". The form contains several input fields: "Country Name" (CountryName), "Attraction Name From" (&AttractionNameFrom), and "Attraction Name To" (&AttractionNameTo). Below these is a main grid labeled "GRID" with columns: "Attraction Id" (AttractionId), "Attraction Name" (AttractionName), "Attraction Photo" (Attraction Photo), "Trips" (Trips), and a new trip button (&newTrip). At the bottom of the form is a "Total Trips" field (&totalTrips). Two arrows point to the grid: one to the main grid labeled "Grid1" and another to the "Total Trips" field labeled "+ Grid2". To the right, a code block shows the event logic for Grid1.Load:

```
Event Grid1.Load  
  &trips = Count(TripDate)  
  &totalTrips = &totalTrips + &trips  
Endevent
```

We've said a couple of times that maybe it would have been better to use the Load event of the grid and not the generic one, which only works in the case of a web panel without a grid or with a single grid. Using the Load event of the grid we anticipate a future need to enter another grid.

*GeneXus*<sup>TM</sup>

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