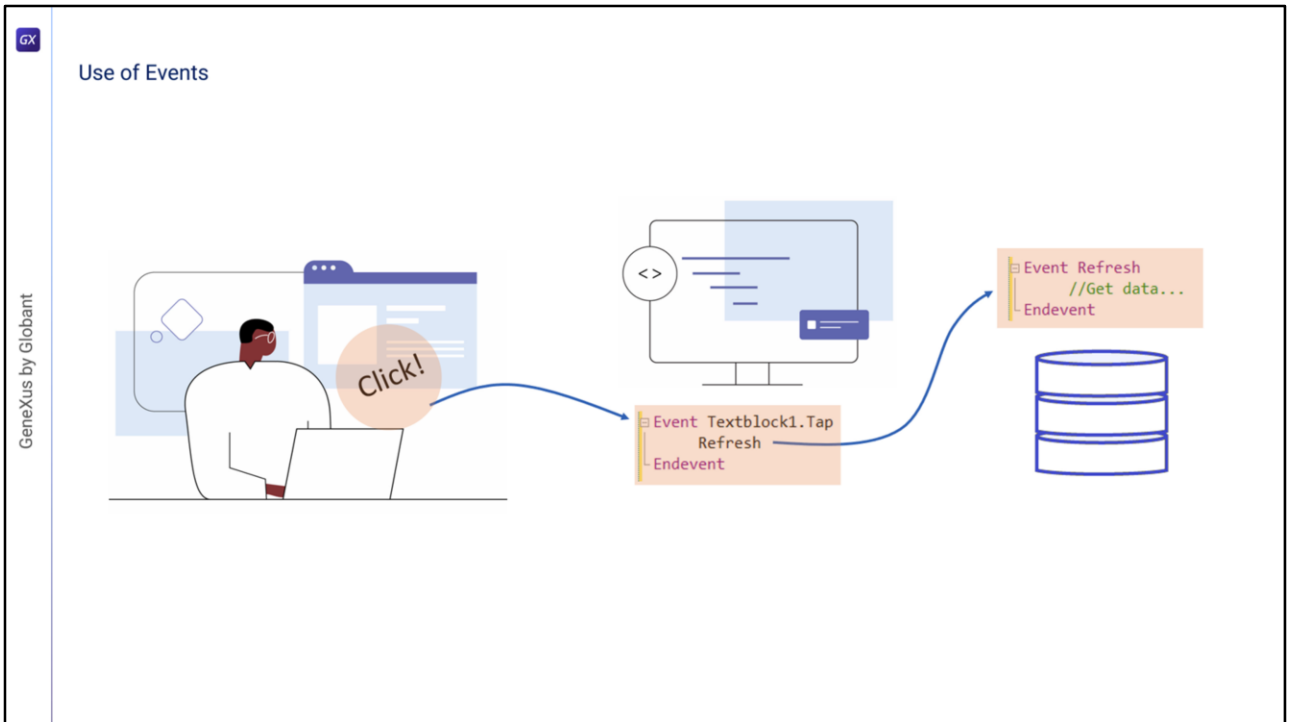


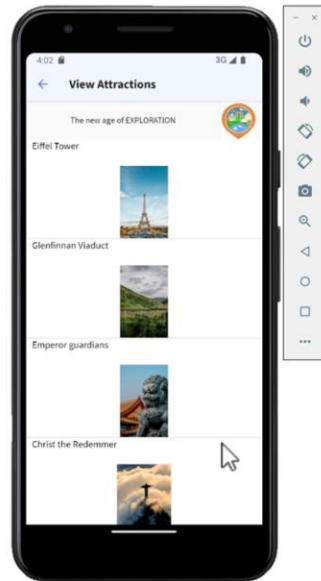
# Events in a Panel



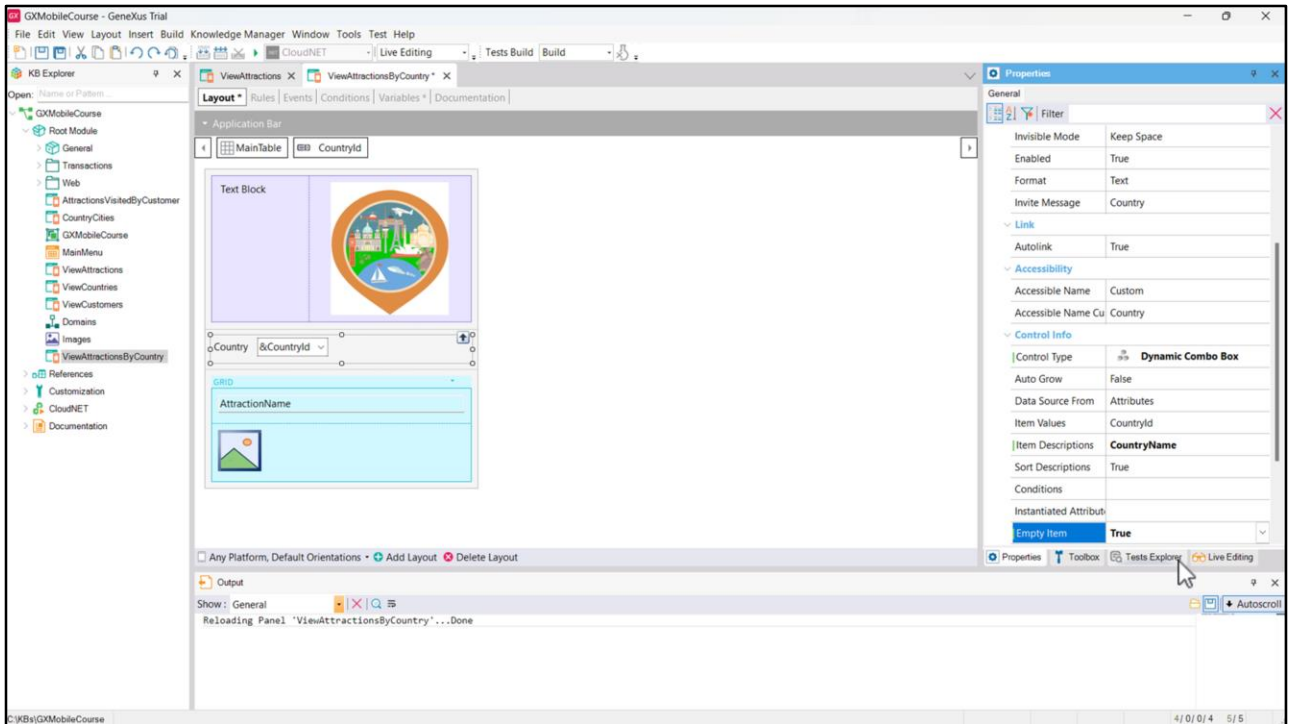
Vanesa Fernández



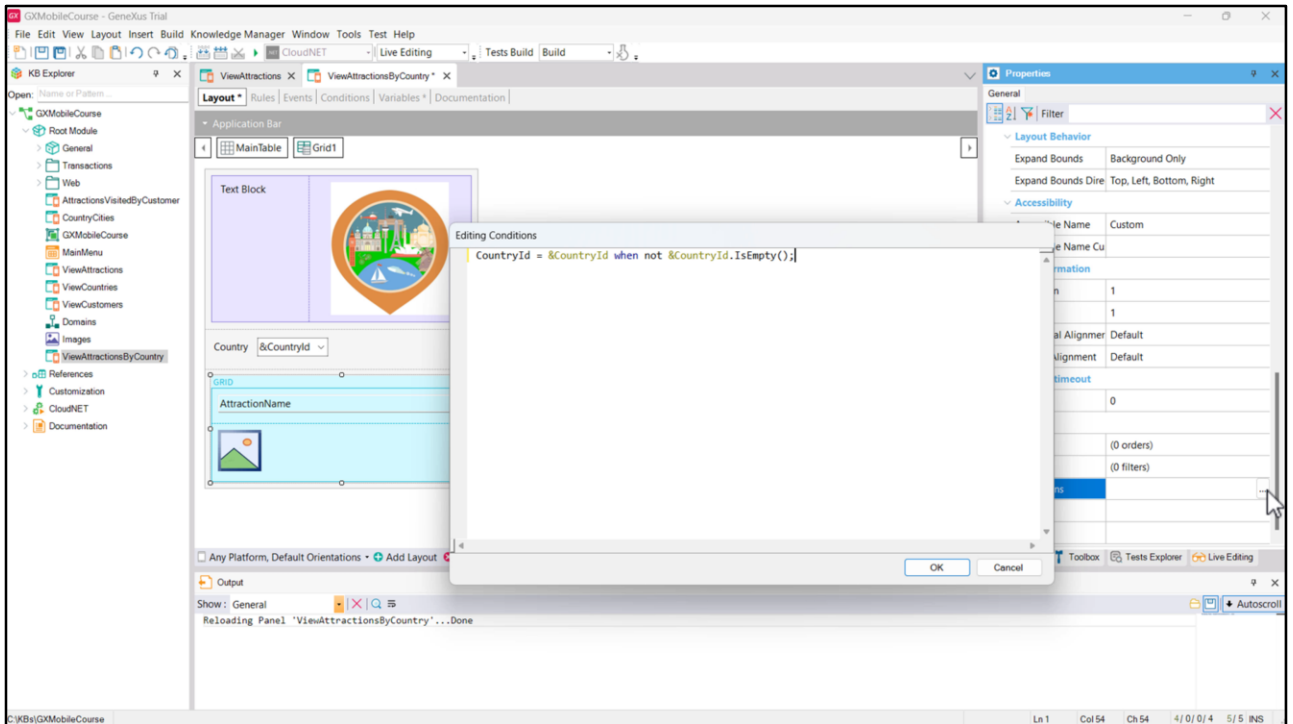
As we know, events are messages that the software or the system give at certain moments when the application is running, and this allows us to schedule actions to be executed at those moments. For example, if the user clicks on a button, or types something in a text control and exits, or if the server is required to send updated information, we can schedule a response after the event is triggered.



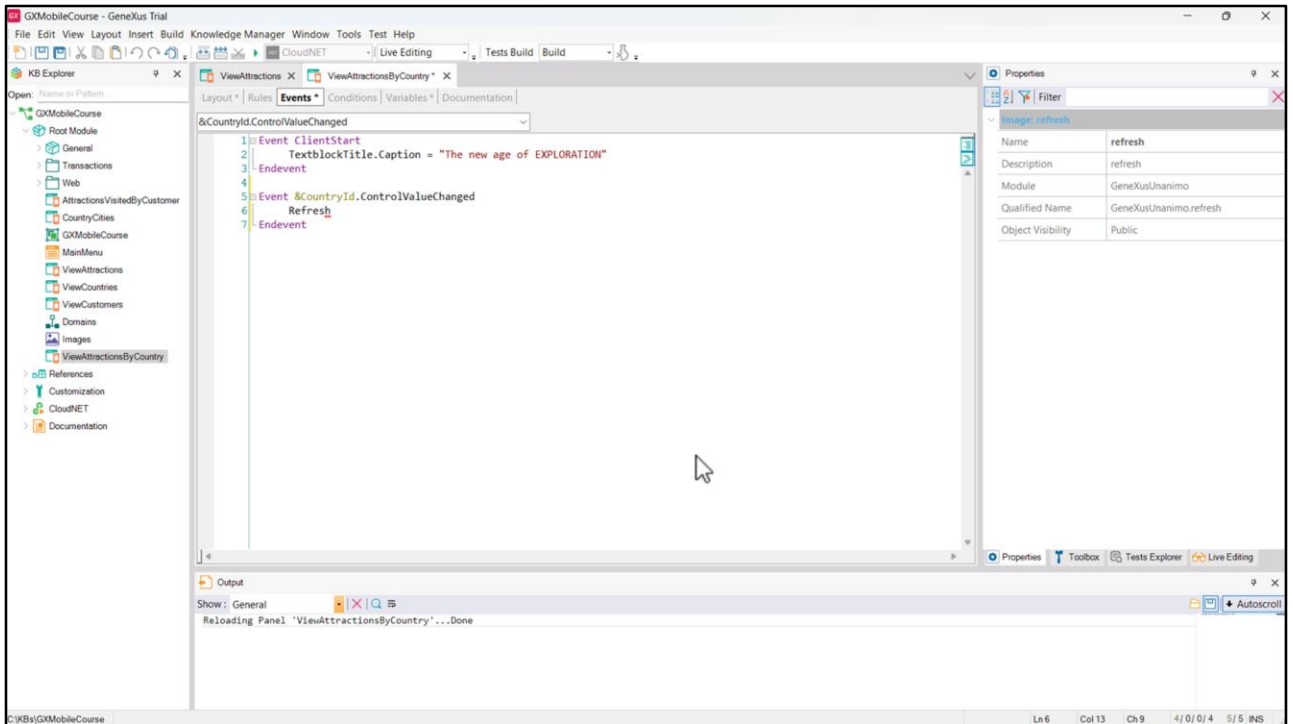
Suppose that when we view the attraction data, we want to have filters so that the displayed data meets the chosen constraint(s).



To implement this, we open the ViewAttractions Panel and save it with the name ViewAttractionsByCountry. Now we go to the variables section and add a variable named &CountryId, which is automatically based on the CountryId attribute. We drag it to the form after the title textblock and before the grid, and in its properties we change the CountryId label to Country, change the ControlType property to Dynamic Combo and set Item Descriptions to CountryName. We also set the EmptyItem property to True, so that the combo will appear at the start without any default value.

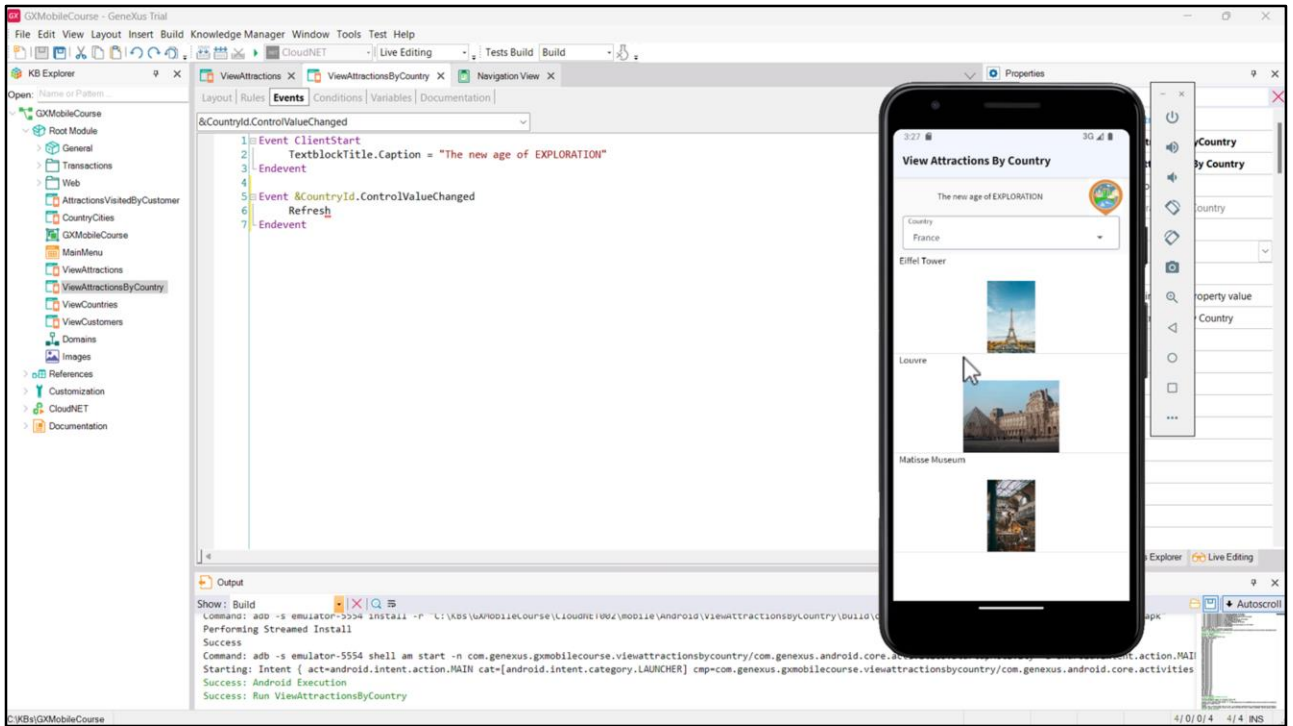


For the filter to have an effect on the attractions shown in the grid, we edit its properties and in the Conditions property we write: `CountryId = &CountryId when not &CountryId.IsEmpty();` and close with a semicolon. Once we choose a country, the Panel must be refreshed so that the grid loads only those attractions from that country.



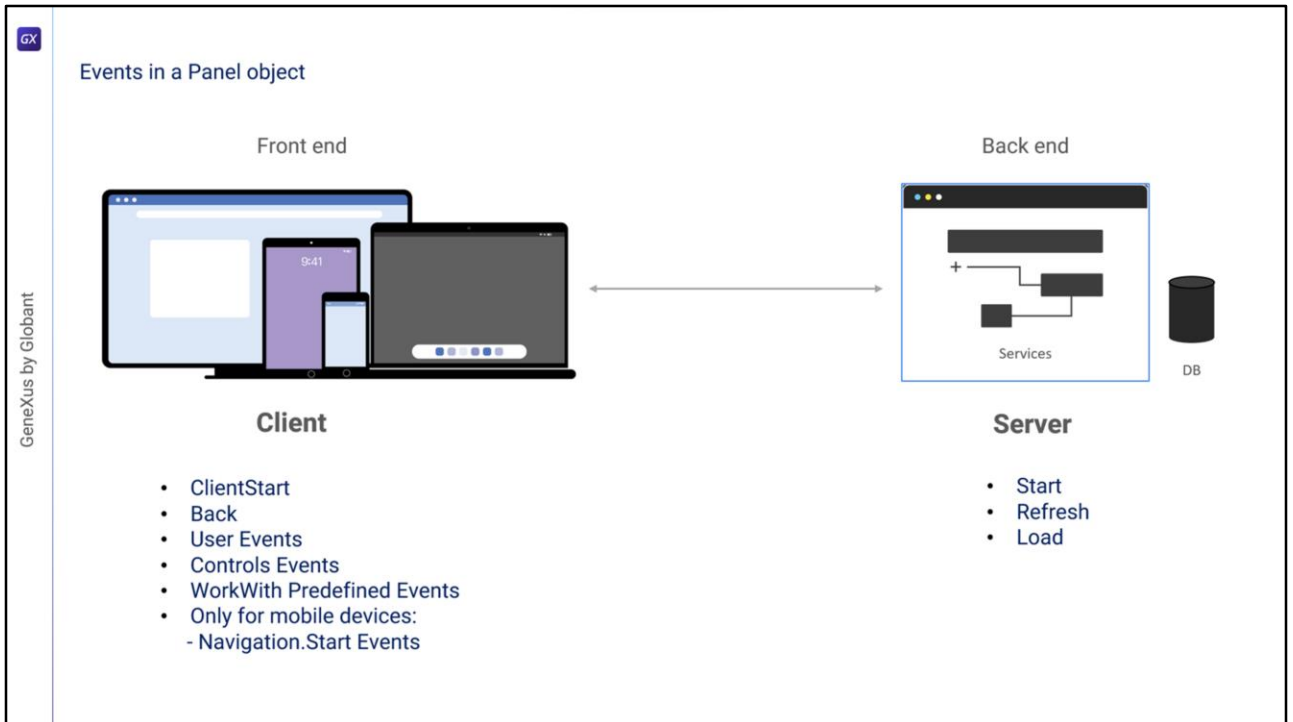
To do so, we use the ControlValueChanged event of the dynamic combo, so we go to the Events tab and in the menu we choose Insert event, click on the &CountryId variable, and select the event we mentioned.

We see that the event code opens so we can write what we want to be executed when this event is fired. We type Refresh. This command will cause the grid to get the data again, filtered this time by the chosen country.



We are going to set the Main program property of the Panel to True, to right-click and then Run. We see that, since we had set the Empty Item property to True, the combo appears without a country selected and all the attractions are shown. If we choose France, the grid reloads and now shows only the tourist attractions in France.

Let's take a closer look at events.



In a Panel object, we have two types of events: client-side events and server-side events.

Server-side events are the same as when working with WebPanels: the Start, Refresh, and Load events. These events will always be triggered on the server in the case of online native mobile applications. When implementing offline native mobile applications, they will be triggered internally on the device.

Client-side events are ClientStart, Back, user-defined events, and events associated with the screen controls, as well as the events predefined by the WorkWith pattern. We will also have the events associated with the data navigation styles, which depend, for example, on the type of device and its orientation when the application is started.



GeneXus by Globant

Server-side events

Back end

SERVER

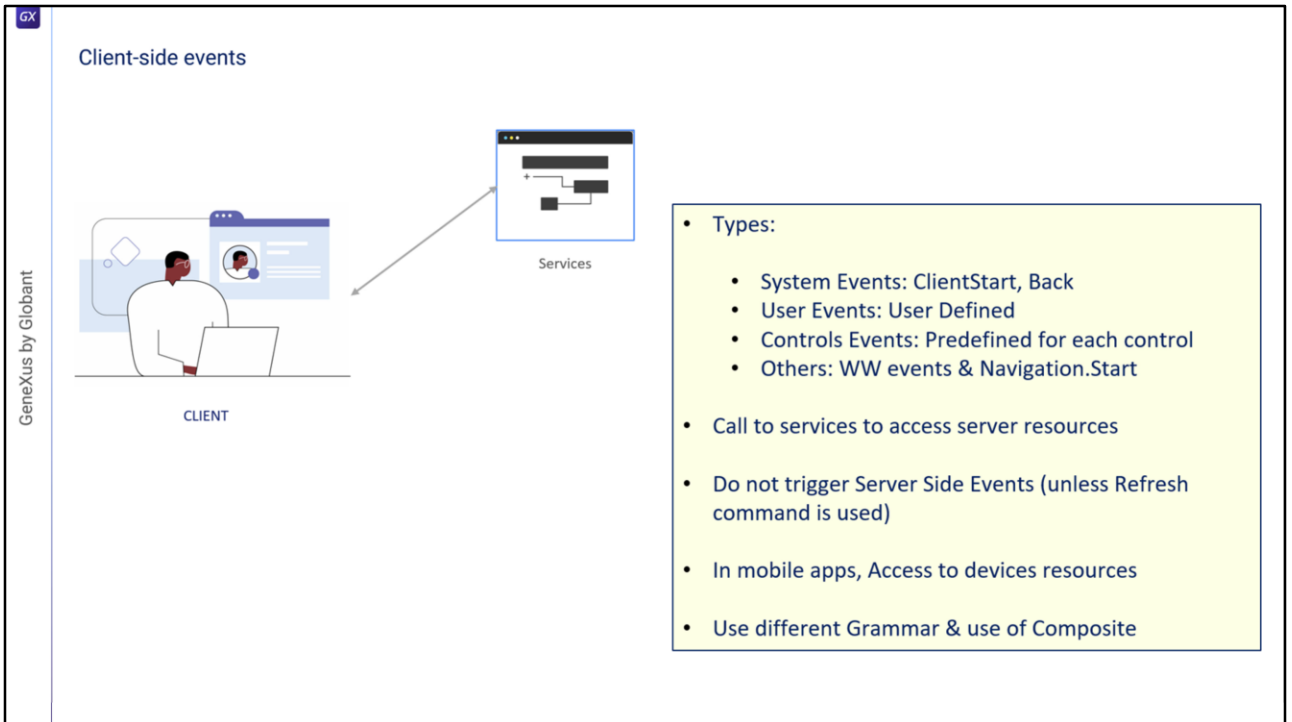
- Start
- Refresh
- Load

- Start Event is executed only once when it is necessary to go to the server.
- Refresh Event is executed after Start Event.
- Refresh Event triggers Load Event.
- Load Event is the last of system Events executed (only if a grid exists)
  - Grid with Base Table: Load is executed as many times as records in base Table.
  - Grids without Base Table: Load is executed only once.
  - SDT based Grids: Load is not triggered.

Let's take a closer look at server-side events.

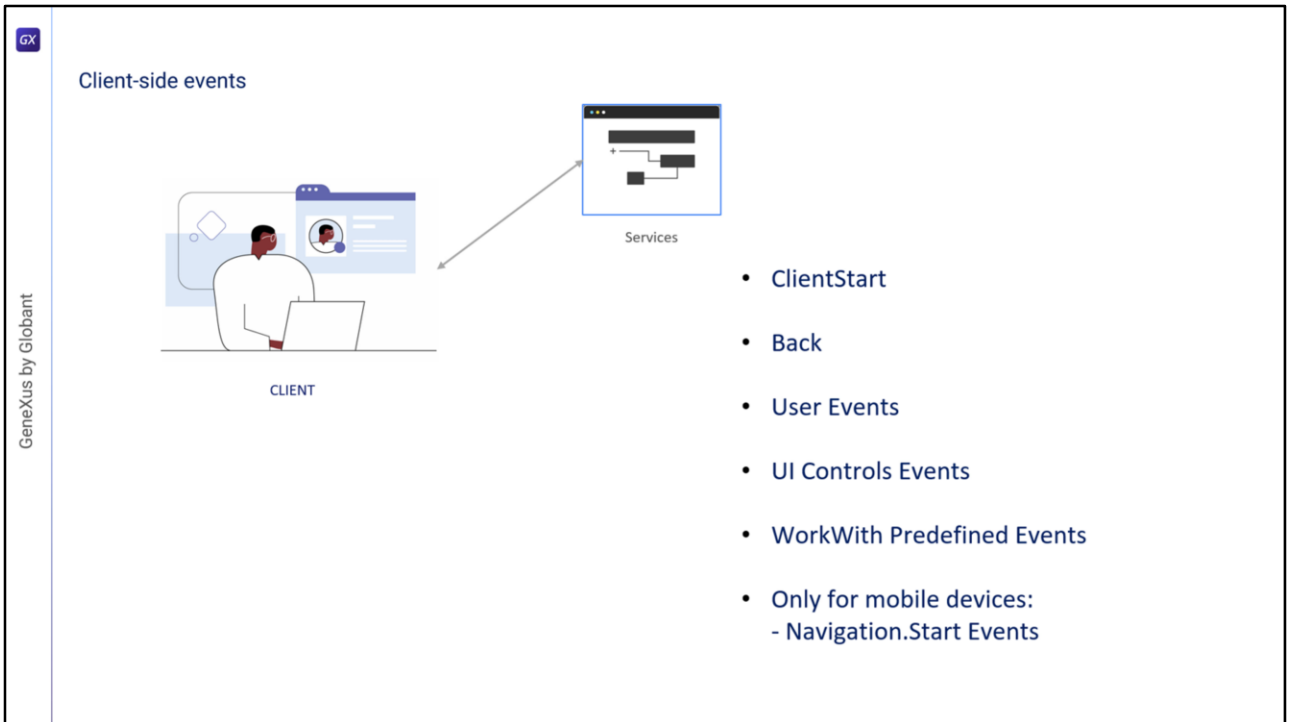
- The first one executed is the Start event. It is executed only once when the Panel is opened, as long as it is necessary to go to the server, and will not be executed again unless we exit the object and reopen it.
- The Refresh event is executed after the Start event, usually only once, but it can be invoked again with the Refresh command. In that case, it will be executed more than once and it will be the first event, since the Start event will not be executed again. The Refresh event will also be triggered if we refresh the browser.
- When the Refresh event is invoked, the Load event will be triggered at the end. This will only happen if there is at least one grid in the panel and the grid is not a collection SDT variable.
- The Load event is the last one of the system events to be executed, and...
- If it has a base table, it will be executed as many times as records exist in the base table;
- If the grid doesn't have a base table, it will be executed only once.
- If the grid is based on an SDT, the load event will not be executed.

Remember that when we work on a mobile device application, the code of the Start, Refresh, and Load events does not have access to device resources such as the camera, GPS, etc.



Now, let's see Client-side events. These events are the application's response to user interaction.

- There are several types of client events: system events such as ClientStart and Back, user events, on-screen control events and others that we will see next.
- The code associated with these events is run on the client, unless access to a server resource is requested; for example, if you want to access the database. In this case, the client will have to invoke a server service.
- During the execution of a client-side event, server-side events are not executed unless explicitly requested through the Refresh command. This causes the server's Refresh event to be triggered, followed by the Load event (if there is at least one grid, as we saw before).
- In native applications for mobile devices, client-side events have access to all hardware and software resources of the device, such as the camera, GPS, microphone, calendar, contacts, etc.
- These client-side events will have a specific grammar that is different from that of server-side events.

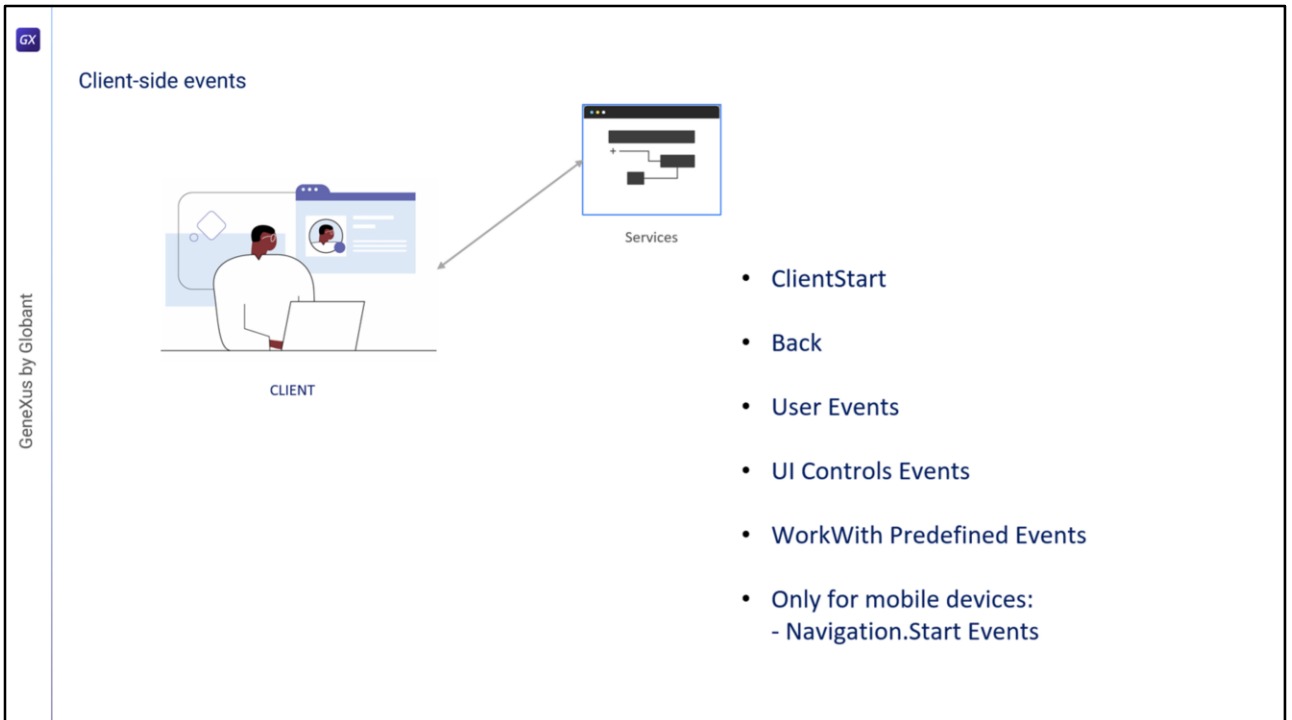


Let's take a brief look at each client-side event.

The ClientStart event is the first event to be triggered, even before the Start event that is triggered on the server and with no need for any user interaction with the application. It is used to initialize the home screen and UI-related aspects.

The Back event is used only on mobile platforms and allows you to capture that the user has pressed the back button on Android devices, or that the back gesture was made on iOS devices, and program an appropriate action.

User events have a name given by the user and allow the developer to associate a certain code that is run when a certain on-screen control is activated by clicking (or tapping) on it.



On-screen controls have their own events, depending on the control, such as: click (or tap), double-click (or double-tap), drag, swipe, `ControlValueChanged`, etc. These events are triggered when some of the above actions take place and the developer can program a response of the application to the user's input.

The `WorkWith` pattern has predefined events that are triggered depending on the action performed on the data of the entity to which the pattern is applied. These include the `Insert`, `Update`, `Delete`, `Save`, and `Cancel` events, among others. The events available will depend on the part of the `WorkWith` object being executed (list, detail, etc.).

In mobile devices, depending on the type of device and orientation, when we start the application a `Start` event will be triggered depending on the navigation. For example, if we are using a Tablet, by default the application starts in `Split` mode, meaning that the screen is divided into two sections, with a list on the left and the details of the selected item on the right. In this case, when the application starts, the `Split.Start` event is triggered. In the case of phones, for example, the screen will show only the list and a separate screen will be opened to view the details. This mode is called `Flip` and is the default behavior in these devices. Therefore, when starting the application, the `Flip.Start` event will be triggered. Although there is a default navigation for each device, the main objects of mobile applications have a property that allows changing the way the application starts. These `Start` events associated with the type of navigation are triggered at the start of the mobile application and immediately after the `ClientStart` event.

GeneXus by Globant

### Client-side events

```

1 Event ClientStart
2   TextblockTitle.Caption = "The new age of EXPLORATION"
3 Endevent
4
5 Event &CountryId.ControlValueChanged
6   Refresh
7 Endevent

```

In the object we saw before, we program only an event associated with a control; we don't program any user events. We program the `ControlValueChanged` event to the on-screen filter variable in order to trigger the Grid's `Refresh` method, which in turn will trigger the server's `Refresh` and `Load` events to update the contents of the grid with the entered filter.

## What can be done in a client-side event?

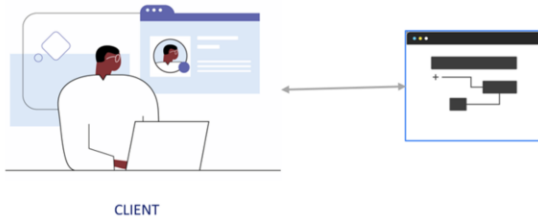


- Call other panels
- Call Rest Services
- Use Business Component
- Call WorkWith objects. In mobile call to Menu
- Call External Objects of GeneXus module
- Call Subroutines
- In addition:
  - Control properties assignments
  - Simple or SDT variable assignment
  - For each Line and Selected Line in grids
  - Use If-Else, Do-Case and Do-While code blocks

In short, let's see what can be done in a client-side event:

- Invoke another Panel object.
- Invoke Rest Services; when we invoke Procedures or Data Providers, they will be automatically exposed as Rest Services on the server. If those procedures or DPs were invoked only from a server-side event (within Start, Refresh or Load events), they would not be exposed as REST services because it would not be necessary.
- Use Business Components to retrieve or update information. In this case, these Business Components will also be exposed as Rest services automatically.
- Directly invoke any node of the Work With pattern. In mobile applications, we can invoke a Menu object.

## What can be done in a client-side event?



- Call other panels
- Call Rest Services
- Use Business Component
- Call WorkWith objects. In mobile call to Menu
- Call External Objects of GeneXus module
- Call Subroutines
- In addition:
  - Control properties assignments
  - Simple or SDT variable assignment
  - For each Line and Selected Line in grids
  - Use If-Else, Do-Case and Do-While code blocks

- Invoke the external objects defined in the GeneXus module. Some of these APIs only make sense for a particular platform, such as mobile devices. Meanwhile, some of them can only be used on web applications, and others can be used on both platforms.
- From a client-side event we can call subroutines.
- In addition, we can:
  - Assign properties to controls.
  - Assign simple or SDT variables.
  - Run For Each Line and For each Selected Line on grids.
  - Use the blocks IF-Else, Do-Case, and Do-While.

If we try to use commands not allowed in a client-side event, we will see an error in the output screen for those lines that don't comply with the grammar constraints. For example, if we try to use a For Each command, a New or any attempt to access or modify information that is only accessed from the server.

## Client-side event grammar

## COMMANDS

Composite

`<Control>.<Property> = <value>`

If `<Bool_expr>`

Do case... endcase

Do while `<Bool_expr>`

Do-sub (except Menu for Smart Devices)

**For each selected line**

Simple variable assignation: `&var = <expr>`

SDT or BC elements assignation:

`&SDT.A = <value>`

`&BC.A = <value>`

Return

Refresh

Inside an **expression**:

Variables

Attributes

Constants

Methods

Functions

Control properties

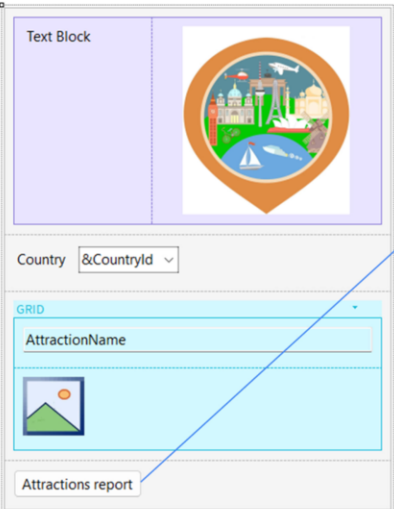
Operators (+, -, /, ^)

Here is a summary of the commands we can use in the client-side event code. These restrictions are only for client-side events, not for server-side events (Start, Refresh, and Load) where we can use all the commands and functions available in GeneXus. The commands accepted at the moment are shown here.



GeneXus by Globant

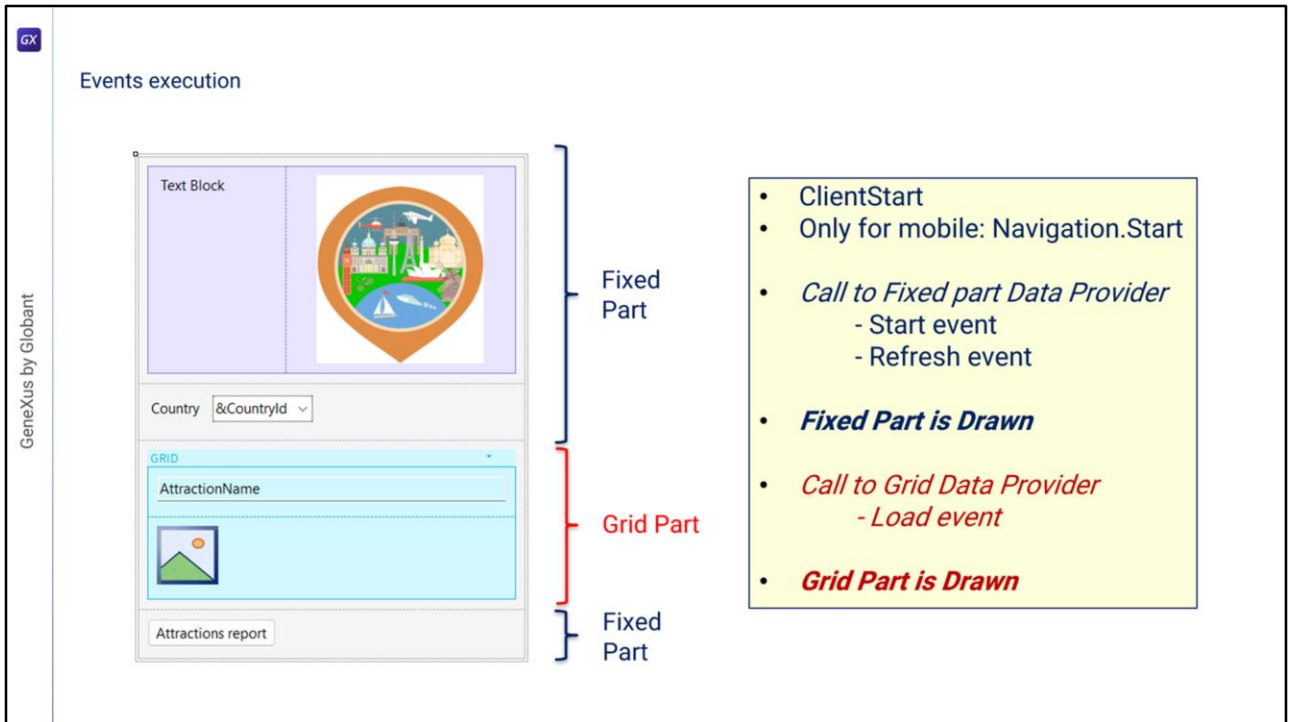
### Composite command



```
Event 'Attractions report'  
  Composite  
    AttractionsByName("F", "M")  
  Return  
EndComposite  
Endevent
```

- Stop execution on Error.
- Automatic Error Handling.

Now, let's look at the Composite command mentioned before. This command is used in client-side events on Panel objects. It is important because if an error occurs in the call sequence, it stops running and the errors are automatically handled and displayed on the screen with no need to implement any programming. This is a great difference with WebPanels, because when a called object in an event causes an error, the execution is not interrupted—it continues in the following statement and the developer is responsible for handling the errors and programming the actions to take. This Composite command is optional: if not used, the operation will be identical to that of WebPanels.



Now let's see what happens with events when we run a Panel object.

The ClientStart event is executed first, and it is executed only once on the client. In mobile applications, the Start event corresponding to the type of navigation set in the main object is then triggered using the Navigation Style property. Next, a Data Provider is executed that will return the data needed to load the fixed part of the Panel. This Data Provider is involved in the execution of the code of the Start and Refresh events to be executed on the server and returns, in a single result, the information to load the fixed part. The fixed part of the Panel is drawn later. Next, a second Data Provider is run that will retrieve the data required by the grid. Within the execution of this Data Provider, the Load event code is run on the server. This Load event will be run N times when the grid has a base table, once for each record, only once if the grid doesn't have a base table, and never if the grid was from a collection SDT variable. At the end of the execution of the Data Provider, it will return the information generated by all these executions of the Load event in a single result, which will be used to load the grid. Afterwards, the grid will be completely drawn.

GeneXus by Globant

Refresh command

```

1 Event ClientStart
2   TextblockTitle.Caption = "The new age of EXPLORATION"
3 -Endevent
4
5 Event &CountryId.ControlValueChanged
6   Refresh
7 -Endevent
8
9 Event 'Attractions report'
10  Composite
11    AttractionsByName("F", "M")
12  Return
13  EndComposite
14 -Endevent

```

Conditions: CountryId = &CountryId when not &CountryId.IsEmpty(): ...

We've mentioned that the fixed part of the Panel is loaded independently from the loading of the grid. Different Data Providers are invoked that are published on the server as services and access the database to retrieve the information of each part. When we change the value of a filter, the grid's information needs to be refreshed. For this, we need to add the Refresh method, which will trigger the server's Refresh and Load events. In turn, it will cause the grid to be reloaded applying the programmed conditions and displaying the filtered results as expected. Since the Refresh command must be invoked after we change the value of the filter variable, we use the ControlValueChanged event of the variable to invoke the method. In this way, after changing the filter value, when leaving the field the corresponding event will be triggered, which is expected to refresh the content of the grid.

GeneXus by Globant

### Refresh command

```

1 Event ClientStart
2   TextblockTitle.Caption = "The new age of EXPLORATION"
3 -Endevent
4
5 Event &CountryId.ControlValueChanged
6   Refresh
7 -Endevent
8
9 Event 'Attractions report'
10  Composite
11    AttractionsByName("F", "M")
12    Return
13  EndComposite
14 -Endevent

```

Conditions: CountryId = &CountryId when not &CountryId.IsEmpty(): ...

But what happens when the Refresh command is invoked within a client-side event? In this architecture, since the objective is to have the page loaded as few times as possible, the data cache is prioritized; that is to say, we always try to retrieve the information previously saved. That is, depending on the caching configuration, it is decided whether or not to go to retrieve data from the server. When it is decided to go to the server, if there are no changes in the server data, nothing is returned to the client. Otherwise, the Refresh and Load events are executed (if there is a grid not based on an SDT, as in our case) and the fixed and variable parts of the Panel are updated, as expected.



In the following videos, we will discuss other aspects of the design and implementation of Panel objects.