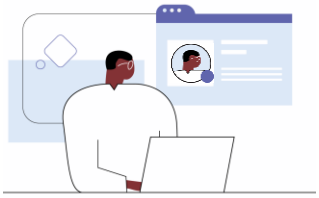
Web Screens with Customer-facing Focus

Event execution scheme, grammar of client-side events, and base table determination

GeneXus™

In this video, we will start to look at the events that can be defined on the screens in customer-facing applications, in particular on web screens. Also, we will describe some characteristics of the panel object.

Client and server events



WEB CLIENT with
a focus on UX

- ClientStart
- Back
- User Events
- Control Events

SERVER

- Start
- Refresh
- Load

This type of application includes two types of events: events that are triggered on the client side and events that are triggered on the server.

The server events are the same ones we already saw when we studied web panels: the Start, Refresh and Load events, which allow us to interact with the database.
The client-side events are ClientStart, Back, user-defined events and events associated with the screen controls.

When we use the panel objects to generate an application for mobile devices, we will also have other events associated with the mobile platform; for example, related to the type of device and its orientation when the application is executed.

Server-side events

SERVER

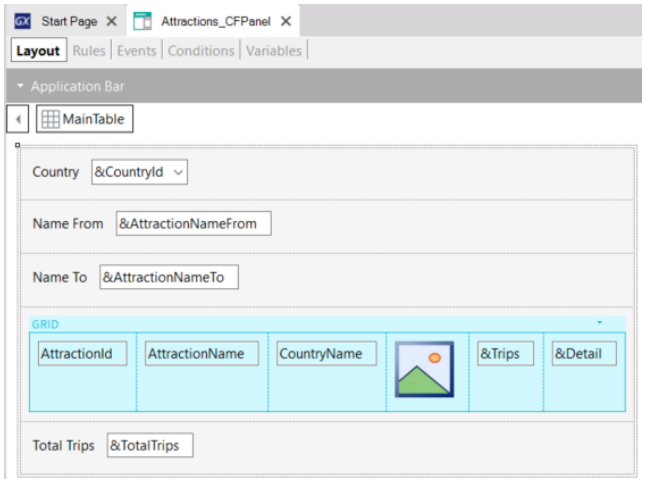
- Start
- Refresh
- Load

- Start Event is executed only once.
- 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.

Now we will see server-side events:

- The first one executed is the Start event. It is executed only once when the panel is opened, and will not be executed again unless we leave the object and reopen it.
- The Refresh event is executed after the Start Event, usually only once, but it can be invoked again using the Refresh command. In that case it will be executed more than once and it will be the first event, since Start will not be executed again.
- 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 of the system events to be executed:
  - 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.
  - And if the grid is based on an SDT, the load event will not be executed.

Example of server-side events




```
Event Load
    &Trips = Count(TripDate)
Endevent

Event Refresh
    &TotalTrips = 0
    For each Trip.Attraction
        &TotalTrips += 1
    Endfor
Endevent

Event &CountryId.ControlValueChanged
    Grid1.Refresh()
Endevent

Event &AttractionNameFrom.ControlValueChanged
    Grid1.Refresh()
Endevent


Event &AttractionNameTo.ControlValueChanged
    Grid1.Refresh()
Endevent

Event Start
    &Details = "Details"
Endevent

Event &Details.Tap
    AttractionDetail_CFPanel.Call(AttractionId)
Endevent
```

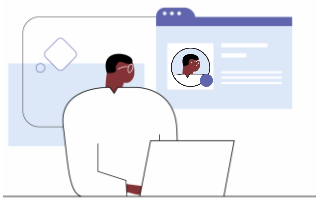
In the example of the previous video, when we implemented the Attractions_CFPanel panel object we programmed the events Start, Refresh, and Load.

In the Load event, we calculated the total trips of an attraction by means of a formula that accessed the TripAttraction table.

In the Refresh event, we programmed a For Each command to access the TripAttraction table to count the overall total of trips of the attractions.

We also programmed the Start event to initialize the &Details variable with the text we wanted to appear on the screen.

Client-side events



WEB CLIENT with customer-facing focus

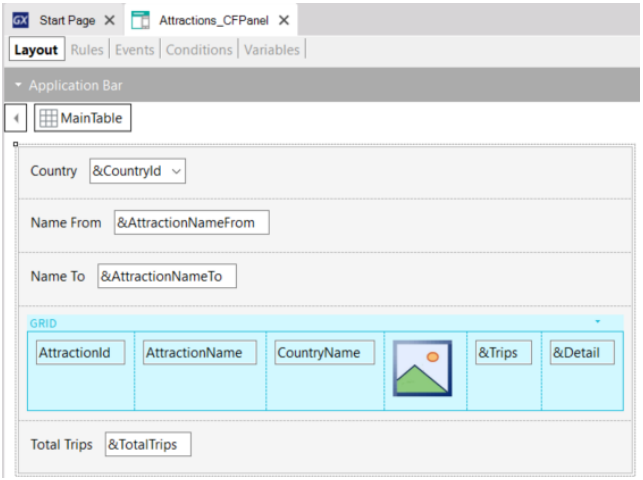
- ClientStart
- Back
- User Events
- Control Events

- Types:
  - System Events: ClientStart, Back
  - User Events: User-defined
  - Control Events: Predefined for each control
- Call to services to access server resources
- Do not trigger server-side Events (unless Refresh command is used)
- Use different grammar.

Client-side events are the application's response to user interaction.

- There are three types of client events: system, user and control events. The system events are ClientStart and Back, which we will not see in this course. User events are those created by the user, and control events are those associated with the screen controls, such as Tap, Double Tap, or ControlValueChanged events, among others.
- The code associated with these events is executed in the client, unless access to a server service is required; for example, if you want to access the database.
- During the execution of a client-side event, server-side events are not executed, unless explicitly required through the Refresh command.
- These client-side events will have a particular grammar that is different from the server-side events.

Examples of client-side events




```
1  Event Load
2      &Trips = Count(TripDate)
3  Endevent
4
5  Event Refresh
6      &TotalTrips = 0
7      For each Trip.Attraction
8          &TotalTrips += 1
9      Endfor
10 Endevent
11
12 Event &CountryId.ControlValueChanged
13     Grid1.Refresh()
14 Endevent
15
16 Event &AttractionNameFrom.ControlValueChanged
17     Grid1.Refresh()
18 Endevent
19
20
21 Event &AttractionNameTo.ControlValueChanged
22     Grid1.Refresh()
23 Endevent
24
25 Event Start
26     &Details = "Details"
27 Endevent
28
29 Event &Details.Tap
30     AttractionDetail_CFPanel.Call(AttractionId)
31 Endevent
```

In the same object we saw before, we program only events associated with controls, we do not program any user events.
To each variable of the on-screen filters, we program its ControlValueChanged event, to trigger the Grid Refresh method, which will cause the server Refresh and Load events to be triggered to update the contents of the grid with the filters entered.

We also programmed the Tap event of the &Details variable, to invoke the AttactionDetail_CFPanel, to see the details of the selected attraction, whose AttractionId identifier is sent in a parameter.

Grammar of client-side events

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 assignment: &var = ***<expr>***

SDT or BC elements assignment:
&SDT.A = <value>
&BC.A = <value>

Return
Refresh

Inside an **expression**:
- Variables
- Attributes
- Constants
- Methods
- Functions
- Control properties
- Operators (+, -,, /, ^)

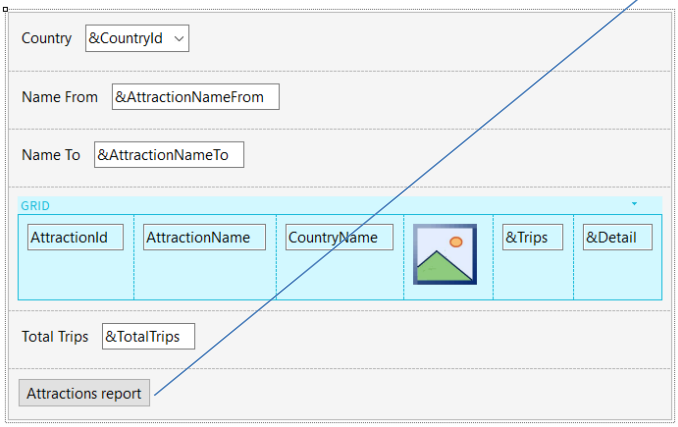
Not allowed: **procedures returning a value** or **external objects**

Let's now look at the grammar of client-side events.

For client-side events, there are restrictions regarding the commands that can be used, but not for server-side events.

As for commands, the accepted ones are shown here. We will not go into more detail in this course.

Composite Command




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

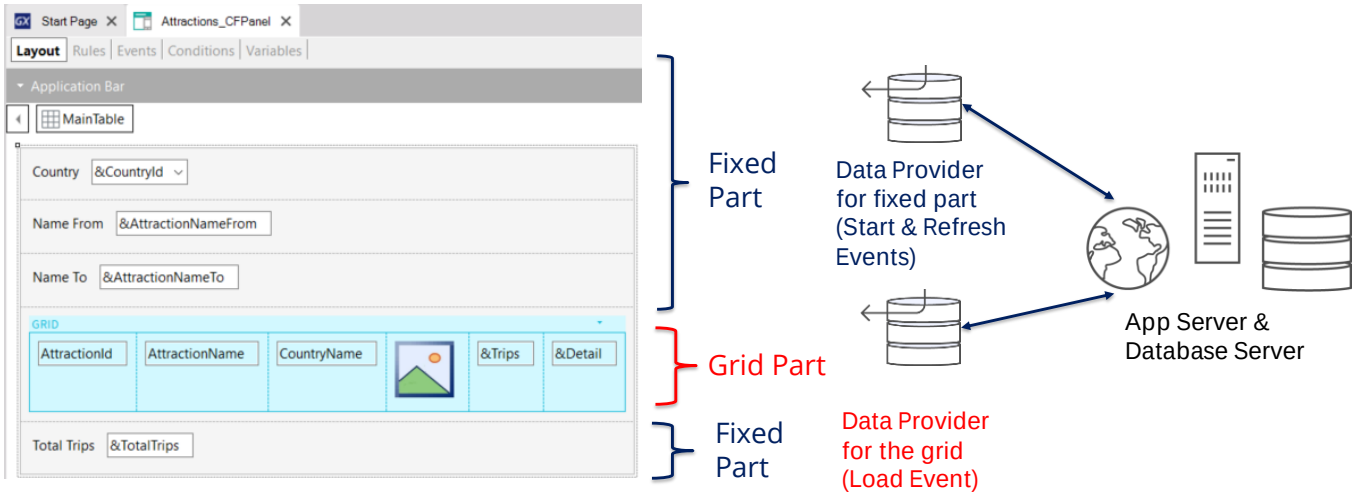
- Only for Client-side Events with more than 1 line of Code.
- Stop execution on Error.
- Automatic Error Handling.

The Composite command is used in client-side events in panel objects, when two or more lines have to be written in an event, and in this case it is mandatory to group the entire code of the event within this command.

This command 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 web panels, 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 be taken.

Parts of a panel object



Let's now look at some characteristics of the panel object.

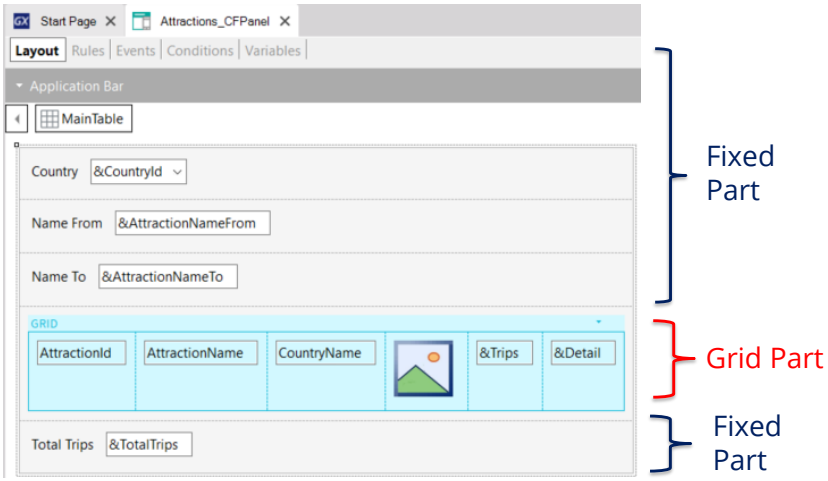
As we mentioned before, in the panel objects we can identify two different parts; the first part is called the fixed or flat part that contains everything in the form that is not included in any grid. In the example we see on the screen, the fixed part is made up of the filter variables &CountryId, &AttractionNameFrom, &AttractionNameTo and the total trips &TotalTrips.

The second part will be called grid or variable part, and in this case it will be made up of the grid that contains the attractions' data.

We will always have a fixed part, and we can have a variable part for each of the grids of the panel.

For each part (fixed and grid) GeneXus will automatically generate Data Providers that will be published as services on the server and will resolve data access. We will not see these data providers in the Knowledge Base because GeneXus will generate and maintain them, but we will be able to see the data accessed by each one, if we see their navigation list, as we did before.

Event execution order



- ClientStart
- *Call to Start & Refresh Data Provider*
- Start
- Refresh

- ***Fixed Part is Drawn***

- *Call to Grid Data Provider*
- Load

- ***Grid Part is Drawn***

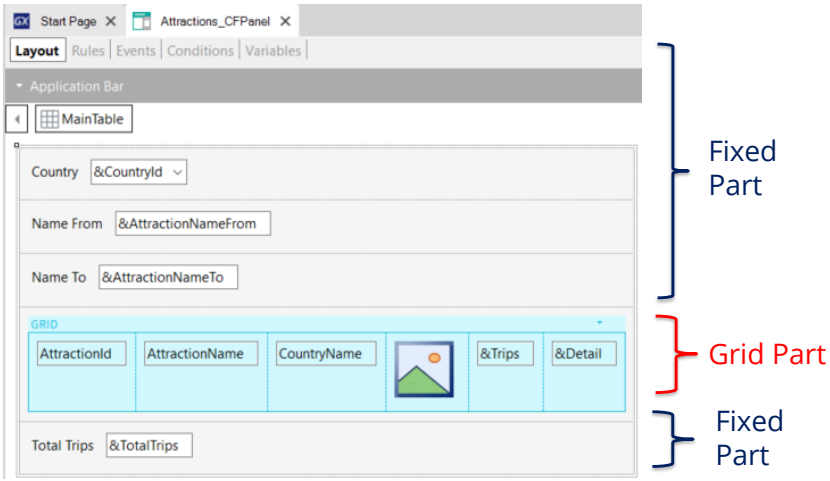
Now let's see what happens when we run a panel object.

First, the ClientStart event is executed only once and runs on the client as we saw before. Next, a data provider is executed that will return the necessary data to load the fixed part of the panel. This data provider Is part of the execution of the code of the Start and Refresh events that will be executed in the server. Also, it returns the information to load the fixed part in a single result. Afterwards, the fixed part of the panel is drawn.

A second Data Provider will be executed to retrieve the data required by the grid. Within the execution of this data provider, the Load event code is executed in the server. This Load event will be executed N times when the grid has a base table, once for each record and only once if the grid doesn't have a base table.
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, with which the grid will be loaded, and then, the grid will be drawn completely.

The fact that the screen is drawn in two different moments has its consequences, as we will see next.

Determining the base tables of the fixed part and the grid



Attributes involved in determining the **Fixed Part base table**:

- Attribs. in fixed part of panel (form)
- Attribs. outside For Each command in events: Refresh, Buttons, or controls in fixed part and Application Bar
- Attribs. in Conditions Tab

Attributes involved in determining the **Grid Part base table:**

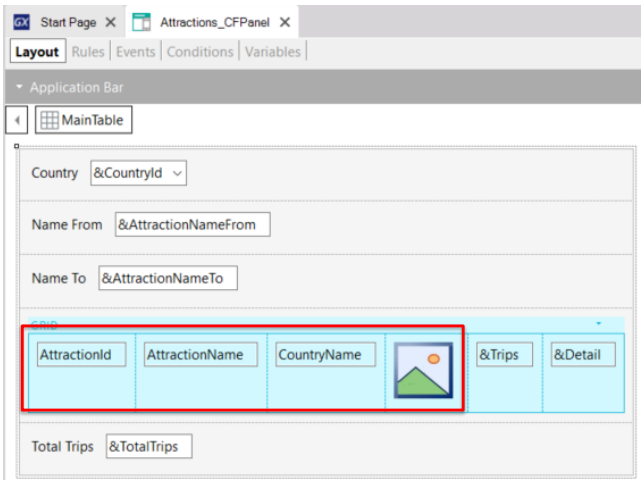
- Attribs. in grid columns
- Attribs. in Order, Search, Advanced Search, and Conditions
- Attribs. outside For Each command in events: Load, Buttons, or controls inside the grid

In a panel object, the fixed part and the grid determine independent navigations and each part will have its base table, as if there were two parallel For Each commands. This creates an important difference with web panels.

To determine the base table of the fixed part, the attributes that belong to the fixed part of the form, the attributes that belong to the events associated with the fixed part as long as they are outside a For Each command (in the Refresh event and events associated with buttons or controls of the fixed part, including those of the Application Bar), and attributes of the Tab Conditions of the Panel object will be taken into account.

To determine the grid's base table, the attributes included in the grid columns –both visible and hidden–, the attributes referenced in the grid Order, Search, Advanced Search and Conditions, and the attributes outside the For Each clauses included in the Load event and in the button or control events within the grid will be taken into account.

Determining the base tables of the fixed part and the grid



Fixed Part: No base table

Fixed Part

Grid Part

Fixed Part

Grid Part base table: Attraction

```
Event Load
    &Trips = Count(TripDate)
Endevent

Event Refresh
    &TotalTrips = 0
    For each Trip.Attraction
        &TotalTrips += 1
    Endfor
Endevent

Event &CountryId.ControlValueChanged
    Grid1.Refresh()
Endevent

Event &AttractionNameFrom.ControlValueChanged
    Grid1.Refresh()
Endevent


Event &AttractionNameTo.ControlValueChanged
    Grid1.Refresh()
Endevent

Event Start
    &Details = "Details"
Endevent

Event &Details.Tap
    AttractionDetail_CFPanel.Call(AttractionId)
Endevent
```

Therefore, in the example we saw, as in the form there are no attributes in the fixed part (only variables), there are no buttons either, as well as no attributes in the Conditions Tab of the panel, and no attributes in the events associated with the variables. In the Refresh event there are no attributes outside the For Each, and the fixed part of the panel has no base table.

In the grid we have the attributes AttractionId, AttractionName, CountryName and AttractionPhoto, so the base table will be Attraction, because in the Load event the only attribute included is in the Count formula.

More information...

https://wiki.genexus.com/commwiki/servlet/wiki?24829

If you want to learn more about the topics addressed in this video, you can read the documentation about programming for native mobile devices (Smart Devices) since, as we said before, the panel object is also used in those applications.

You can find more information in the wiki page displayed on the screen.

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications