

Components and Master Panel object

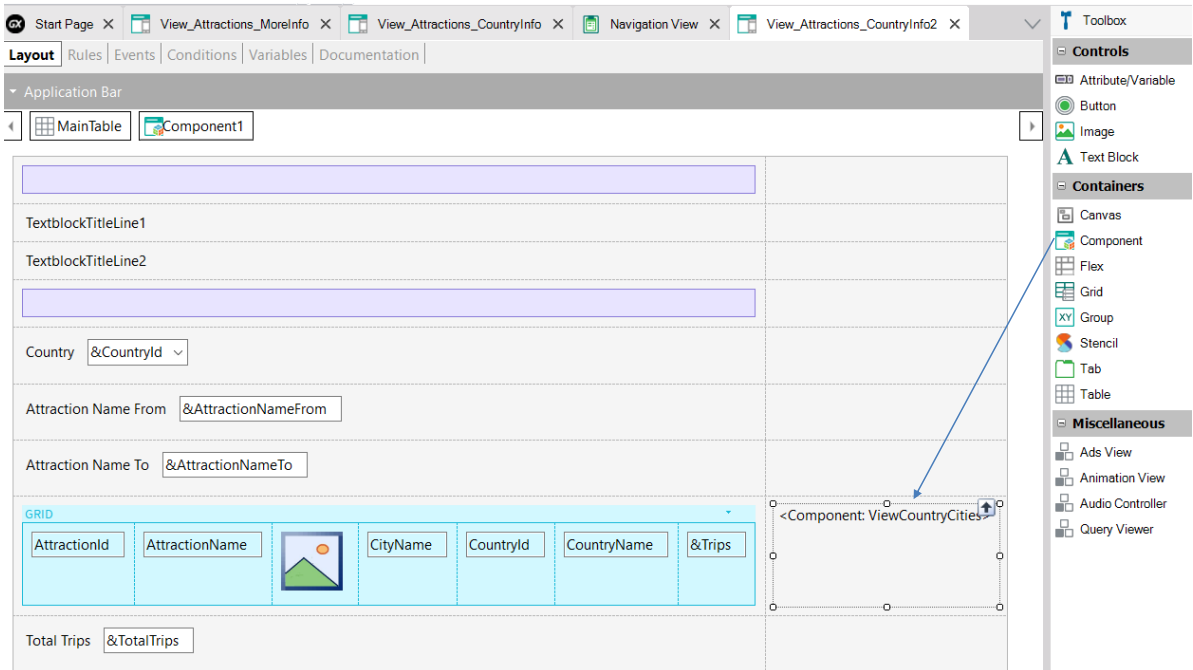


In this video, we will use components. They are containers that can be included in a panel and associated with another panel, in order to run a panel object inside another one. We will also see the Master Panel object, which will allow us to define a container screen inside which all the screens of the travel agency application will be executed, so that they all have the same appearance and context; for example, the agency's logo, among other things, will always be shown.

Components

The panel object allows encapsulating functionality through the use of components, in a similar way to web panels with web components. Let's see an example of use.

Using a panel inside another panel



We are going to build a panel object that shows the attractions and data of a selected country, but this time the information will be displayed using a component inside the panel being created.

To create the panel that will contain the component, we make a Save As of View_Attractions_MoreInfo with the name View_Attractions_CountryInfo2.

To the right of the attractions grid, from the toolbar we insert a Component control.

Similarly to what we did before, when clicking on the name of a country in the grid, the information of that country will be displayed.

Creating a panel to be used as a component

The screenshot illustrates the development of a panel component in Genexus. The main workspace shows the 'Rules' tab with a parameter rule: `Parm(in:CountryId);`. Below it, the 'MainTable' component is visible in the application bar. Two detailed views of the component's data provider are shown:

- Data Provider ViewCountryCities_Level_Detail:** This view shows the component's data provider configuration. It includes a 'Name' field set to 'ViewCountryCities_Level_Detail', a 'Description' of 'ViewCountryCities_Level_Detail', and 'Output' and 'Devices' set to 'None'. The 'LEVELS' section shows a 'For First Country (Line: 1)' level with an 'Order' of 'CountryId', an 'Index' of 'ICOUNTRY', and 'Navigation filters' for 'Start from: CountryId = @CountryId' and 'Loop while: CountryId = @CountryId'. The 'Optimizations' are set to 'First 1 record(s)'. The output is a 'Country (CountryId)' table.
- Data Provider ViewCountryCities_Level_Detail_Grid1:** This view shows the configuration for the grid component. It includes a 'Name' field set to 'ViewCountryCities_Level_Detail_Grid1', a 'Description' of 'ViewCountryCities_Level_Detail_Grid1', and 'Output' and 'Devices' set to 'None'. The 'LEVELS' section shows a 'For Each City (Line: 2)' level with an 'Order' of 'CountryId', an 'Index' of 'ICITY', and 'Navigation filters' for 'Start from: CountryId = @CountryId', 'Loop while: CountryId = @CountryId', and 'Server Paging'. The 'Optimizations' are set to 'Server Paging'. The output is a 'City (CountryId, CityId)' table.

Now we will create the panel to associate with the component. We call it ViewCountryCities, and from the toolbar we drag the CountryName attribute and a grid with the CityName attribute. We assign the City base transaction to the grid.

Now we add a Parm rule with the CountryId attribute.

In the navigation list of the panel, in the Level_Detail node we can see that the Country table is run through, filtered by the CountryId attribute included in the Parm rule. So, it will retrieve a single record corresponding to the country received by parameter.

In the Level_Detail_Grid1 node we see that the grid runs through the City table, also filtered by the CountryId received by parameter.

Component properties

The screenshot shows the GeneXus IDE interface. The main workspace displays a form with the following elements:

- TextblockTitleLine1
- TextblockTitleLine2
- Country: &CountryId
- Attraction Name From: &AttractionNameFrom
- Attraction Name To: &AttractionNameTo
- GRID table with columns: AttractionId, AttractionName, CityName, CountryId, CountryName, &Trips
- Total Trips: &TotalTrips

The Properties panel on the right is configured for 'Component: Component1'.

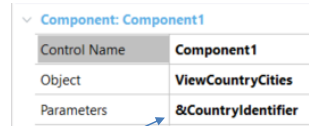
Component: Component1	
Control Name	Component1
Object	ViewCountryCities
Parameters	&CountryIdentifier
Appearance	
Auto Grow	True
Class	Table
Visible	True
Invisible Mode	Keep Space
Cell information	
Row Span	1
Col Span	1
Horizontal Alignment	Default
Vertical Alignment	Default

In the properties of Component1, in Object we choose the ViewCountryCities panel and in Parameters we add the &CountryIdentifier variable. This variable will store the value of the CountryId attribute of the selected grid row.

One difference we can see compared to web panels is that here it is not necessary to change the type of the ViewCountryCities panel to make it a component: when inserting a Component control, I can choose to insert any of the panels we created before.

Dynamic creation of the panel used as a component

```
34 | Event CountryName.Tap|
35 |     &CountryIdentifier = CountryId
36 |     Component1.Object = ViewCountryCities.Create(&CountryIdentifier)
37 | Endevent
```



Component: Component1	
Control Name	Component1
Object	ViewCountryCities
Parameters	&CountryIdentifier

Now we have to invoke the component panel when clicking on the country name in the attractions grid, so we program the Tap event of the CountryName attribute with the invocation.

Since this panel was created from a Save As, we already had code in that event, so we delete it.

First, we save the CountryId corresponding to the selected CountryName in the &CountryIdentifier variable that we defined before. Then we dynamically create the **ViewCountryCities** object associated with Component1, using the Create command and sending the chosen country by parameter.

Since the panel we are building is already set to main, we are going to run it with Run.

At runtime...

The screenshot shows a web browser window with the URL `localhost:49467/View_Attractions_CountryInfo2-Level_Detail`. The page title is "View Country Cities". Below the title, there is a heading "The new age of EXPLORATION".

The main content area features a table of attractions and a dropdown menu for selecting a country. The table has columns for attraction name, location, and country. The dropdown menu is currently set to "(None)" and shows a list of countries: China, Beijing, Shanghai, and Hong Kong.

Attraction Name	Location	Country	Trips
1 Eiffel Tower	Paris	France	6
2 Glenfinnan ...	Glenfinnan	Scotland	0
3 Meet the ...	Beijing	China	0
4 Christ the ...	Rio de ...	Brazil	6
5 Rifugio ...	Belluno	Italy	0
6 London ...	London	England	0
7 Louvre	Paris	France	0
8 Forbidden ...	Beijing	China	0

Total Trips: 18

Country Name: China
Beijing
Shanghai
Hong Kong

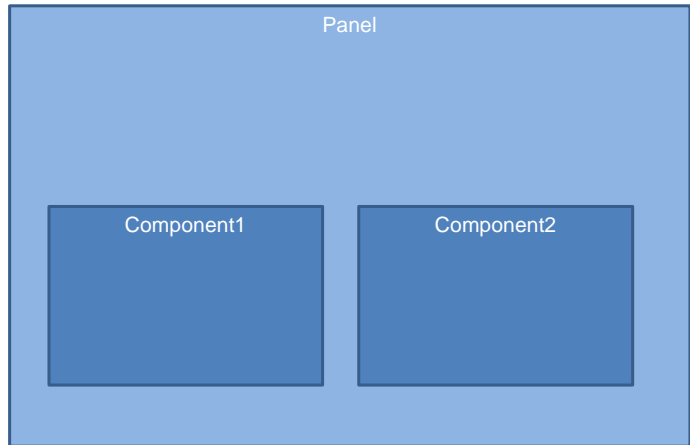
We click on the country China and see that China's Cities are displayed: Beijing, Shanghai, and Hong Kong.

Event firing order

ClientStart (Panel)
 Start (Panel)
 Refresh (Panel)
 Load (Panel)

ClientStart (Component1)
 Start (Component1)
 Refresh (Component1)
 Load (Component1)

ClientStart (Component2)
 Start (Component2)
 Refresh (Component2)
 Load (Component2)



Now let's see the firing order of the events when we have panels inside another panel, when using components.

The panel objects that we insert as components inside another panel have the same events that they had as a panel; that is, client-side events (ClientStart, Back, User events, etc.) and server-side events: Start, Refresh, and Load.

In other words, we must consider the order in which the events of the component(s) we have in the panel will be fired relative to the events of the host panel.

First, the events of the panel containing the components will be fired in the usual order we saw before; that is, ClientStart, Start, Refresh, and Load.

Then, from top to bottom and from left to right, the events of each panel included as a component, also in the usual order.

Master Panel Object

Now let's see what the Master Panel object is. This object can only be used in web applications generated in Angular.

Master Panel Object

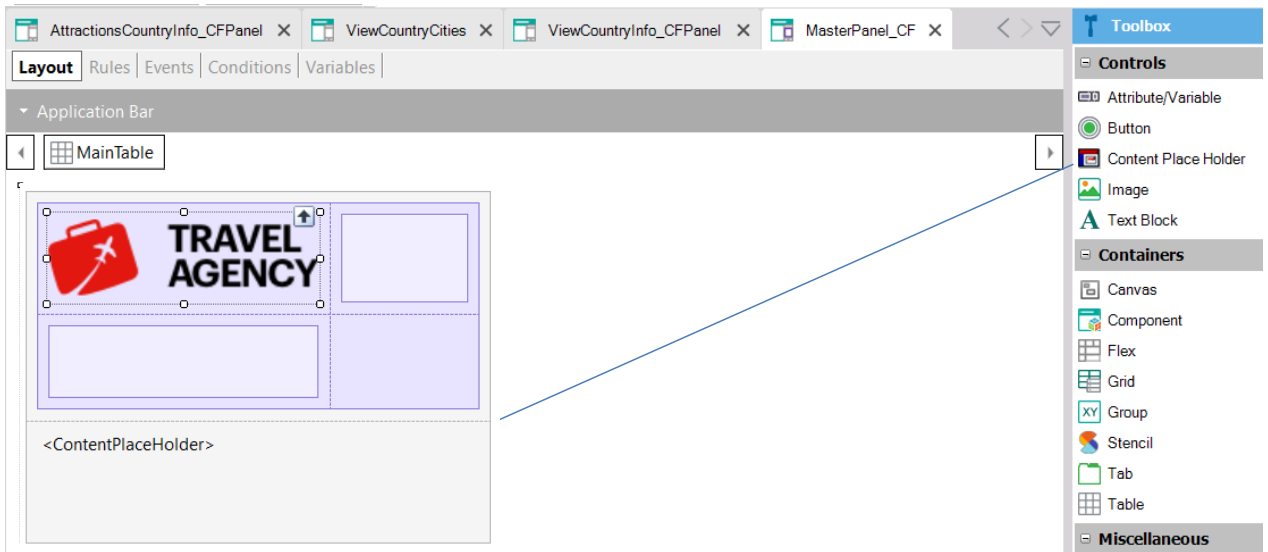
Properties	
Filter	
Panel: Attractions_CFPanel	
Name	Attractions_CFPanel
Description	Attractions_CFPanel
Module/Folder	FrontendAngular
Qualified Name	Attractions_CFPanel
Object Visibility	Public
Main program	False
Master Panel	(none)
Caption	Attractions_CFPanel

In the properties of the **View_Attraction_MoreInfo** panel object, we can see that one of them is the Master Panel property, which in our panel appears empty. There you indicate which master page will contain the panel object and where it will be loaded.

This concept is analogous to that of the Web Master Panel seen for web panels.

However, unlike them, a panel object doesn't have a property to change the type; instead, the Master Panel object is a separate object from the panel object.

Master Panel Object (continued)



We will create an object of Master Panel type and call it **MasterPanel_TravelAgency**.

In the toolbar there is a Content Place Holder container control available, where the panel objects will be loaded.

We will insert the Content Place Holder in the web panel. Above it we insert a table and inside it an image with the agency's logo; below and to the right we insert tables so that the logo is displayed aligned to the left.

Master Panel Object (continued)

The screenshot displays the GeneXus IDE interface for a Master Panel Object. On the left, the design view shows a table with a logo and the text "TRAVEL AGENCY". The table is divided into four quadrants. The top-left quadrant contains a red suitcase icon and the text "TRAVEL AGENCY". The top-right quadrant is empty. The bottom-left quadrant is empty. The bottom-right quadrant is empty. Below the table is a content placeholder labeled "<ContentPlaceholder>".

The properties view on the right shows the configuration for two tables:

Table: MainTable	
Control Name	MainTable
Appearance	
Columns Style	100%
Rows Style	100dip;pd
Width	100%
Height	100%
Auto Grow	True

Table: Table1	
Control Name	Table1
Appearance	
Columns Style	320dip;100%
Rows Style	90dip
Width	100%
Height	90dip

On the right, a class diagram shows the inheritance structure:

```

classDiagram
    class Image
    class ImageHeader_TravelAgency
    ImageHeader_TravelAgency --|> Image
    class ImageHeader_TravelAgency {
        Scale Type: Fit
    }
  
```

We adjust the properties of the containing table so that the image looks right. And we do the same with the Main Table.

Now we must change the behavior of the image to fit the size we set. For now, we are going to create an ImageHeader_TravelAgency class to which we assign the Scale Type property with Fit value. Later on, when discussing the Design topic, we will see how to do this with a Design System Object.

We assign the created class to the logo image and save.

Master Panel Object (continued)

Panel: Attractions_CFPanel

Name	Attractions_CFPanel
Description	Attractions_CFPanel
Module/Folder	FrontendAngular
Qualified Name	Attractions_CFPanel
Object Visibility	Public
Main program	True
Master Panel	MasterPanel_CF
Caption	Attractions_CFPanel

Panel: AttractionDetail_CFPanel

Name	AttractionDetail_CFPanel
Description	Attraction Detail_CFPanel
Module/Folder	FrontendAngular
Qualified Name	AttractionDetail_CFPanel
Object Visibility	Public
Main program	False
Master Panel	MasterPanel_CF
Caption	Attraction Detail_CFPanel

Now we assign the Master Panel property to the **View_Attractions_MoreInfo** and **View_Attractions_CountryInfo** panels with the **MasterPanel_TravelAgency** object we've just created.

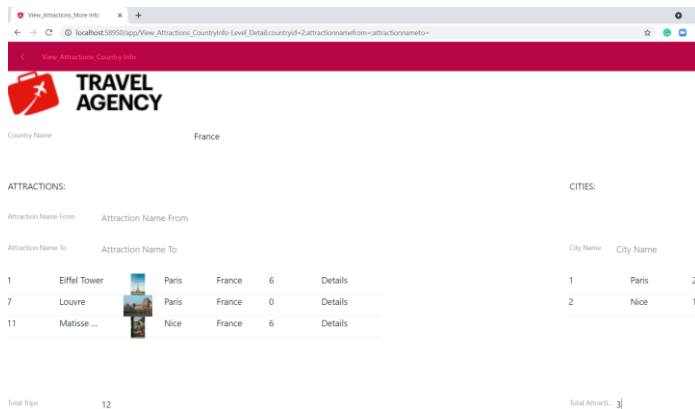
We run it.

Running the application with an assigned master panel



The new age of
EXPLORATION

Country	(None)	Attraction Name From	Attraction Name To
1	Eiffel Tower	Paris	2
2	Glenfinnan Viaduct	Glenfinnan	5
3	Meet the Emperor	Beijing	3
Total Trips			18



We see that the agency logo is shown and the panel `View_Attractions_MoreInfo` is loaded underneath. If we click on the country France, we see that the information panel of the country also has the same logo.

Here we will not go into the details of the Design System of the Panel object, but we can say that the use of the Design System Object is similar to that of the web panels.

We will get more into design in a future video.

GeneXus™

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