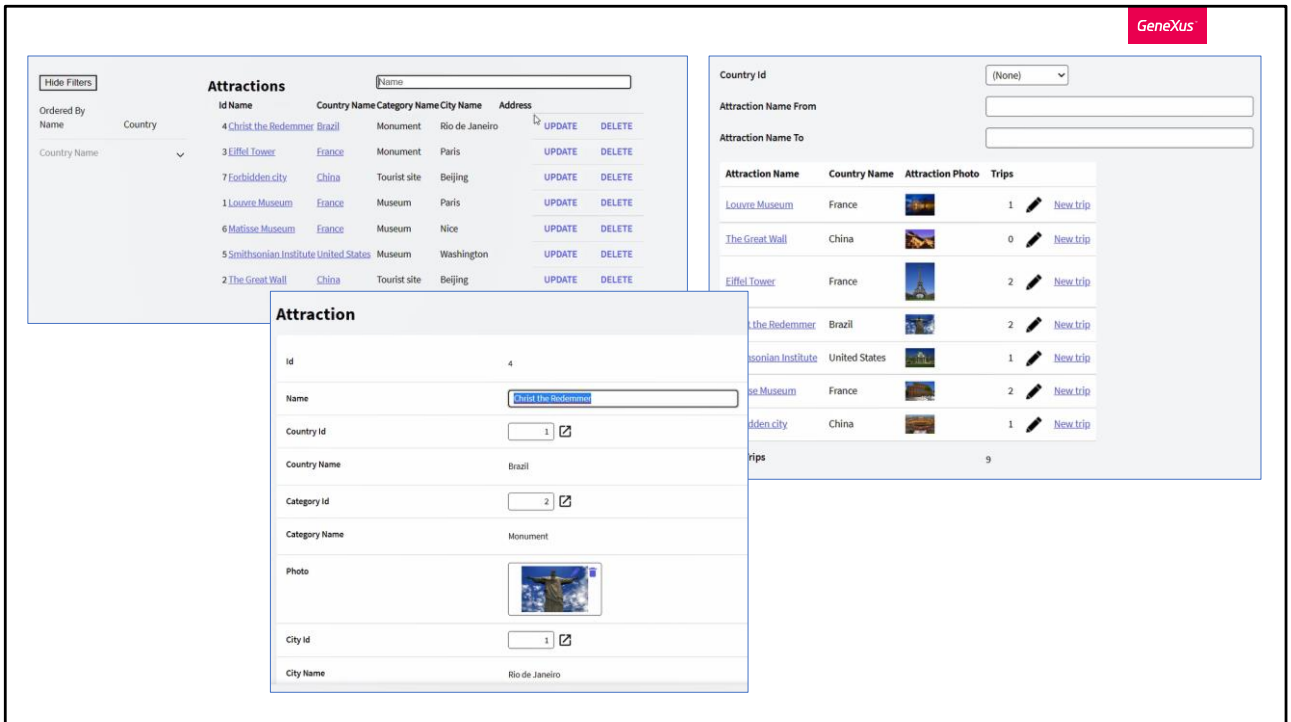


## Types of Web Panels

GeneXus™



In previous videos, we have built a solution from scratch, very similar to the one that the Work With pattern builds automatically.

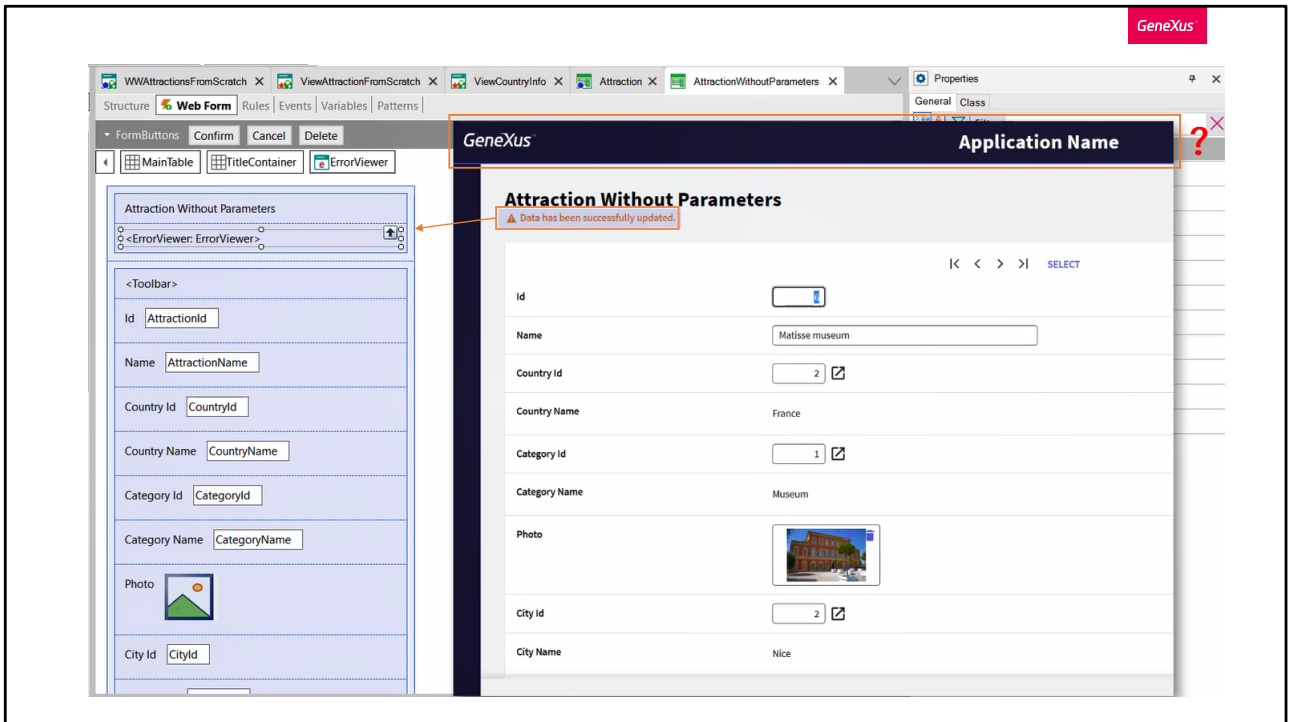
When comparing the two solutions in runtime, the first thing you'll see is their different design. That is because, so far, we were focused on logic leaving the User Interface aside, an aspect we'll get into later on. Apart from the visual differences, both web panels are showing a grid with information on tourist attractions, with filters, and the possibility to, for example, update information.

In both solutions, the transaction is invoked in Update mode.

Something to be noted from the beginning is a common feature on all the screens navigated. Regardless of the page you open, you will see this part.

For instance, if you directly invoke the parallel transaction `AttractionWithoutParameters`, it is right here. it is right here.

Or, if you now go to the View of an attraction -the pattern's View or the one you implemented from scratch-, note that, beyond those differences, they continue to have this aspect in common.



If you now go to GeneXus to see the form of this View you implemented, where is this common feature?

You will not find it either when opening the Attraction transaction, or the parallel transaction that doesn't receive parameters, and going to its Form.

What you see here is a textblock control with the name of the transaction. Below is an ErrorViewer control showing messages such as success or failure. Then there is this “action group” control that shows the navigation and Select buttons. And then come the attributes, and another “action group” with the buttons.

Where is the upper sector?

## Master Page

The screenshot displays the GeneXus IDE interface with three main components:

- Web Transaction Properties:** A table showing properties for a Web Transaction. The 'Master Page' property is set to 'MasterUnanimoSidebar'.
 

Web Transaction	
Style	TravelAgency
Form Layout	UnanimoTemplate
Type	Web Page
Master Page	MasterUnanimoSidebar
- Web Panel Properties:** A table showing properties for a Web Panel named 'WWAttractionsFromScratch'. The 'Master Page' property is set to 'MasterUnanimoSidebar'.
 

Web Panel: WWAttractionsFromScratch	
Name	WWAttractionsFromScratch
Description	WWAttractions From Scratch
Module/Folder	Root Module
Style	TravelAgency
Type	Web Page
Master Page	MasterUnanimoSidebar
- Web Master Panel Properties:** A table showing properties for a Web Master Panel named 'MasterUnanimoSidebar'. The 'Type' property is set to 'Master Page'.
 

Web Master Panel: MasterUnanimoSidebar	
Name	MasterUnanimoSidebar
Description	Master Unanimo Sidebar
Module/Folder	UI
Style	TravelAgency
Type	Master Page
Show Master Page when P	False
On session timeout	Ignore
Focus control	Use Environment property value
Cache expiration lapse	
Automatic refresh	Yes
Auto compress http traffic	Use Environment property value

The central diagram shows a visual representation of the master page layout with placeholders for '<Image=list\_view\_white>', '<Image=genexus\_logo>', 'Application Name', '<Sidebar: SidebarMenu>', and '<ContentPlaceHolder>'.

When you look at the transaction properties you will see a Web Transaction group with four significant properties: the first one to assign the object with which the style will be applied, the second one to specify the template that will be used to generate the Form, the third one will be explained hereafter, and the fourth one is the Master page property. There, you will indicate the master page where the page corresponding to this transaction object should be loaded. Note that it offers options that correspond to all master pages defined so far in your KB. These pages are created upon the creation of a KB so you will have some by default, but you can still create others. Note that the transaction was created by default in association with this master page.

If you now open the other objects that you saw in runtime, it will be no surprise that they also have a master page, and it is the same one. And you will also note that the web panels also have the same four properties.

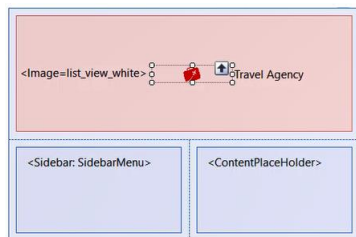
So let's look for this master page and see what it's all about. If you don't know where it is, you may look for it here and open it.

If you look at its properties, you will see one called Style and one called Type, but Master Page is no longer there. Why? Because it is a special type of web panel, the Master Page type. A web panel of this type will have a

special control called `ContentPlaceHolder`. This is where any page that has this one as its master page will be loaded.

Then the transaction will be loaded into this space, with the view, and everything else we saw.

## Master Page



The screenshot shows a web application titled "Travel Agency". It features a sidebar menu on the left and a main content area. The main content area includes a search form with fields for "Country Id" (a dropdown menu showing "(None)"), "Attraction Name From", and "Attraction Name To". Below the search form is a table listing attractions with columns for "Attraction Name", "Country Name", "Attraction Photo", and "Trips".

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

At the bottom of the table, it says "Total Trips 9".

Here is the text block control that you see in runtime called Application Name. For example, try to change it to Travel Agency. Here you have an image that may also be changed for this one that you previously inserted in the KB.

Note the effect of these changes on your application. Here they are...

Another control that you will see in this Master Panel is a User Control called Sidebar, that is typical of the Unanimo design system. This control comes by default showing a deployable side bar that provides access to various objects.

## Web Component

WWAttractionsFromScratch X

Web Form Rules Events Conditions Variables

<No action group selected>

MainTable Grid1

Country Id &CountryId

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

GRID

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

Total Trips &totalTrips

Grid1's Conditions

```

CountryId = &CountryId
when not &CountryId.IsEmpty();

AttractionName >= &AttractionNameFrom
when not &AttractionNameFrom.IsEmpty();

AttractionName <= &AttractionNameTo
when not &AttractionNameTo.IsEmpty();
  
```

OK Cancel

ViewAttractionFromScratch X

Web Form Rules Events Conditions Variables

<No action group selected>

Name AttractionName

Country Id CountryId

Country Name CountryName

Category Name CategoryName

City Name CityName

Trips &trips

Event CountryName.Click  
ViewCountryInfo(CountryId)  
Endevent

We have already seen two of the three types of web screens: the master page, and the common web page, which is what we have been working with so far every time we create a web panel or a transaction. We only have to study the web page of component type.

To do this, let's go back over our steps a little and remember what we had implemented in the previous videos. We had the limitation of the attractions Work With element, where the user could filter by country by choosing a value in this variable, which was being used in the grid conditions to filter by that country if the variable was not empty.

But also, from this panel the user could see the information of an attraction, clicking on its name, and at the same time, from there, a link was displayed above the country name to show all the relevant information of the country.

## Web Component

The screenshot shows a GeneXus Web Form design interface. The top bar includes tabs for 'ViewAttractionFromScratch', 'ViewCountryInfo', and 'Web Form'. Below the tabs is a menu with 'Rules', 'Events', 'Conditions', and 'Variables'. A dropdown menu shows '<No action group selected>'. The main design area contains a 'MainTable' component. The layout includes:

- A 'Country Name' label and a text input field labeled 'CountryName'.
- A section with 'Attraction Name From' and 'Attraction Name To' labels, each followed by a text input field.
- A 'GRID' component with columns: 'Attraction Id' (AttractionId), 'Attraction Name' (AttractionName), 'Attraction Photo' (with a photo icon), 'Trips' (Trips), and a 'newTrip' button.
- A 'Total Trips' label and a text input field labeled '&totalTrips'.
- A 'City Name' label and a text input field labeled '&cityName'.
- A 'GRID' component with columns: 'City Id' (CityId), 'City Name' (CityName), and 'Attractions' (&attractions).
- A 'Total Attractions' label and a text input field labeled '&totalAttractions'.

```
parm( in: CountryId );
```

That's why we called this web panel that received the country identifier in the attribute, and showed its name, and then a grid identical to the first one, with the same filters by attraction name, the same columns and actions and general info, and also information about the cities.



## Web Component

WWAttractionsFromScratch X

Web Form Rules Events Conditions Variables

--No action group selected--

MainTable Grid1



Country Id &CountryId

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo

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

Total Trips &TotalTrips

Louvre Museum  1  New trip

ViewAttractionFromScratch X ViewCountryInfo X

Web Form Rules Events Conditions Variables

--No action group selected--

MainTable

Country Name &CountryName

Attraction Name From &AttractionNameFrom

Attraction Name To &AttractionNameTo



City Name &CityName


City Id	City Name	Attractions
&CityId	&CityName	&Attractions

Total Attractions &TotalAttractions

Attraction Id Attraction Name Attraction Photo Trips &newTrip

Total Trips &TotalTrips

Louvre Museum  1  New trip

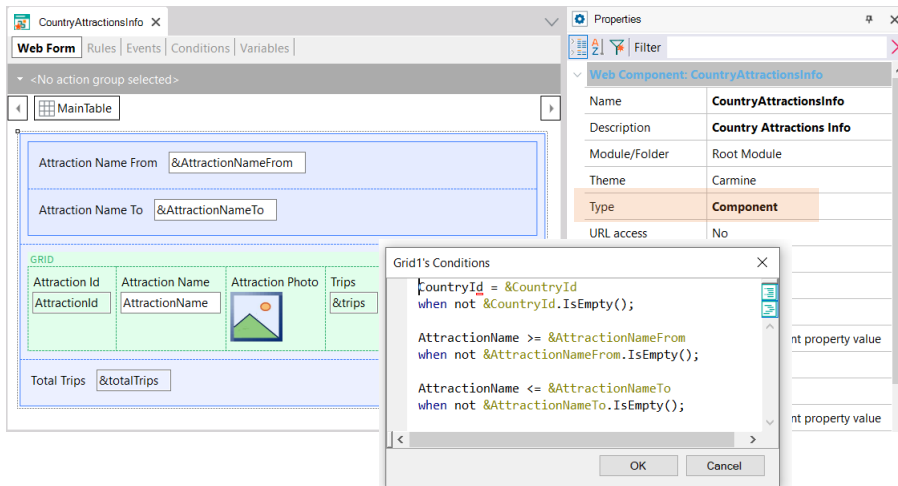
Louvre Museum  1 UPDATE New trip

We can clearly see that part of this panel is almost identical to the other one. Let's suppose that we no longer want the Update option to be offered with this image, but we want it to be as it is in the pattern, with the word UPDATE.

We will have to replace this image with a text block on both panels. To avoid these duplications and to program the behavior and design of a part of the panel only once, we have web panels of the component type.

## Web Component

```
parm( in: &CountryId );
```



What is repeated in this case? This whole section of the screen and its behavior.

Then we did a Save as of this web panel, to which we only removed the CountryId variable from the form and placed it, instead, as a parameter. It will no longer be entered directly by the user in the form of this screen, but will be received from the caller.

We see that the conditions remain identical, as do the events and everything else. The other change is that we modified the Type property, switching it to Component. From now on, this web panel can be a component of another one.

## Component

The screenshot illustrates the process of creating a reusable component in GeneXus. On the left, the 'Web Form' tab shows a design view with a 'MainTable' containing a 'Country Id' dropdown and a component placeholder '<Component: CountryAttractionsInfo>'. The 'Properties' tab on the right shows the configuration for 'Web Component: Component1', including its 'Control Name', 'Object' ('CountryAttractionsInfo'), and 'Parameters' ('&CountryId'). On the right, a preview of the 'Travel Agency' application shows the dropdown set to 'France', displaying a table of attractions.

Attraction Name	Country Name	Attraction Photo	Trips
<a href="#">Louvre Museum</a>	France		2 <a href="#">New trip</a>
<a href="#">The Great Wall</a>	China		0 <a href="#">New trip</a>
<a href="#">Eiffel Tower</a>	France		2 <a href="#">New trip</a>
<a href="#">Christ the Redeemer</a>	Brazil		2 <a href="#">New trip</a>
<a href="#">Smithsonian Institute</a>	United States		1 <a href="#">New trip</a>
<a href="#">Museum of Modern Art</a>	France		2 <a href="#">New trip</a>
<a href="#">Forbidden city</a>	China		1 <a href="#">New trip</a>
Total Trips			9

So, the next thing we did was a Save as of our original panel, and replaced all those controls that we now place in the component web panel, with a component type control. Of course we deleted (here we leave comments) all the events that will now be in the component.

By placing a control of this type, we are saying that whatever we specify must be loaded and executed in there. We have to tell it that an instance of the web component we have just shown, CountryAttractionsInfo, has to be created in that component. We can do it in a static way, indicating it in the properties; here we indicate which will be the component type object and here the parameter sent. Let's try it.

We see exactly the same. If we filter by name of attraction... it's working perfectly.

Now, what happens if we want to filter by country? Nothing happens. Why?

## Component

The screenshot displays two web forms in the GeneXus IDE. The left form, titled 'CountryAttractionsInfo', features a dropdown menu for 'Country Id' with the variable '&CountryId'. The right form, titled 'WWAttractionsFromScratchComp', includes input fields for 'Attraction Name From' (&AttractionNameFrom) and 'Attraction Name To' (&AttractionNameTo), a grid named 'Grid1', and a 'Total Trips' label with the variable '&totalTrips'. The grid 'Grid1' contains columns for 'Attraction Id', 'Attraction Name', and 'Attraction Image'. A 'Grid1's Conditions' dialog box is open, showing the following logic:

```

CountryId = &CountryId
when not &CountryId.IsEmpty();

AttractionName >= &AttractionNameFrom
when not &AttractionNameFrom.IsEmpty();

AttractionName <= &AttractionNameTo
when not &AttractionNameTo.IsEmpty();

```

Below the web forms, the event logic for the 'Country Id' dropdown is shown:

```

Event &CountryId.ControlValueChanged
  Component1.Object = CountryAttractionsInfo.Create(&CountryId)
EndEvent

```

The CountryId variable is on a different panel than the attraction grid through which it is filtered. Here we have to use the ControlValueChanged event, which captures the moment the value of the &CountryId variable control is changed, and have a new instance of the component created, passing it now the new value of the variable.

Let's try it.

We refresh, and try to filter by France. Perfect. And there by attraction name. Perfect also.

## Component

```
parm( in: CountryId );
```

The screenshot shows the GeneXus IDE interface. On the left, a web form is being designed. It contains a 'Country Name' text box, a '<Component: CountryAttractionsInfo>' placeholder, and a 'City Name' text box with a '&cityName' parameter. Below the city name is a grid with three columns: 'City Id', 'City Name', and 'Attractions'. The grid cells contain '&cityId', '&cityName', and '&attractions' respectively. Below the grid is a 'Total Attractions' text box with a '&totalAttractions' parameter. On the right, the 'Properties' window is open, showing the 'Web Component: Component2' properties. The 'Control Name' is 'Component2', the 'Object' is 'CountryAttractionsInfo', and the 'Parameters' are 'CountryId'. The 'Cell information' section shows 'Cell Control Name'. The 'Event' section shows two events: 'Grid2.Load' and 'Grid2.Refresh', both with associated code snippets.

Web Component: Component2	
Control Name	Component2
Object	CountryAttractionsInfo
Parameters	CountryId
Cell information	
Cell Control Name	
Event Grid2.Load	
<pre>&amp;attractions = Count(AttractionName) &amp;totalAttractions = &amp;totalAttractions + &amp;attractions endevent</pre>	
Event Grid2.Refresh	
<pre>&amp;totalAttractions = 0 Endevent</pre>	


And, of course, we did something similar with the panel that showed the country information.

We did a Save as of this panel, replacing this whole section with a component that will be loaded with this panel.

This way, in the new panel the CountryId is not a variable, but it is received in a parameter. So, we can create the component instance in a static way, only once inside the execution of this web panel, passing it here the parameter that in this case comes in the attribute received in the parm rule.

Note that from the original panel we removed the controls and programming of events associated with the attractions, and only left those of the cities. Let's try it at runtime.

To do this, we invoke this panel and not the previous one.









## Travel Agency

Country Name

France

Attraction Name From

Attraction Name To

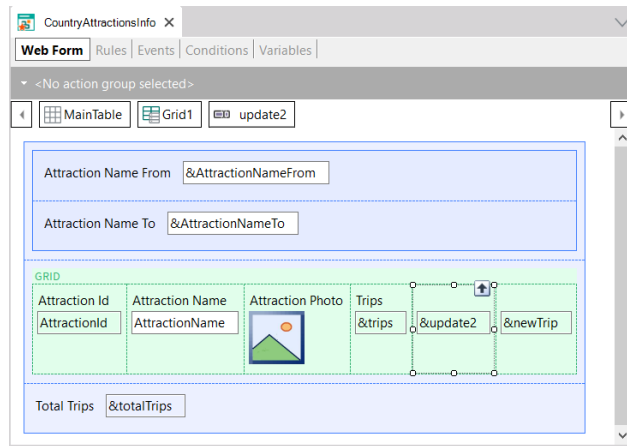
Attraction Name	Attraction Photo	Trips	update
<a href="#">Eiffel Tower</a>		2 	<a href="#">New trip</a>
<a href="#">Louvre Museum</a>		1 	<a href="#">New trip</a>
<a href="#">Matisse Museum</a>		2 	<a href="#">New trip</a>
Total Trips		5	

City Name

City Id	City Name	Attractions
1	Paris	2
2	Nice	1
Total Attractions		3

Now let's change the image to perform the UPDATE, and place the text UPDATE instead.

## Web Component



```

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

Event Grid1.Refresh
    &totalTrips = 0
Endevent

Event Start
    &update2 = "UPDATE"
    &newTrip = "New trip"
Endevent

Event &update2.Click
    Attraction(trnMode.Update, AttractionId)
Endevent

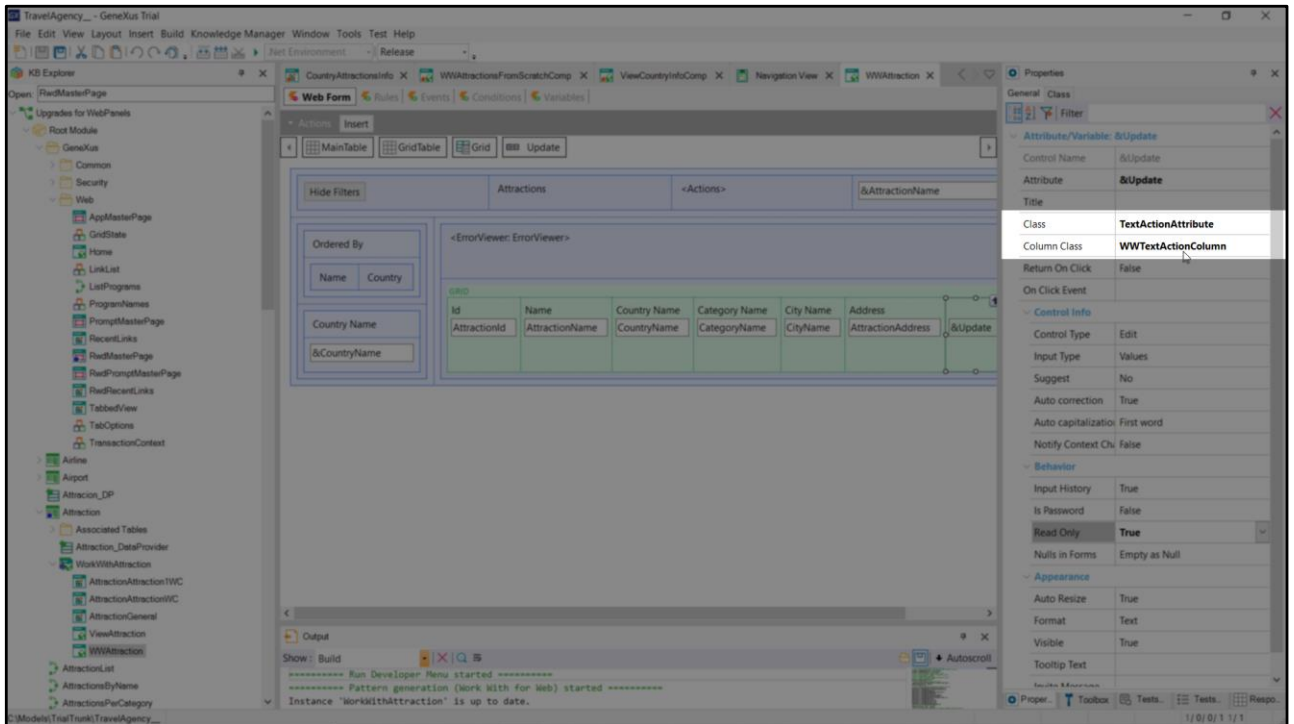
```

To do so, in the component web panel we create a variable of Character (20) type. We insert it in the grid. In the Start event we assign it the UPDATE value, which is what we want the user to see. We removed the assignment of an image to the variable we had, because we are going to remove it from the grid.

And now let's take the title off the new variable and make it Read only.

Let's try to run it. It says that the click is not a valid event. We forgot to change it so that it is the click of the new variable and not the old one.

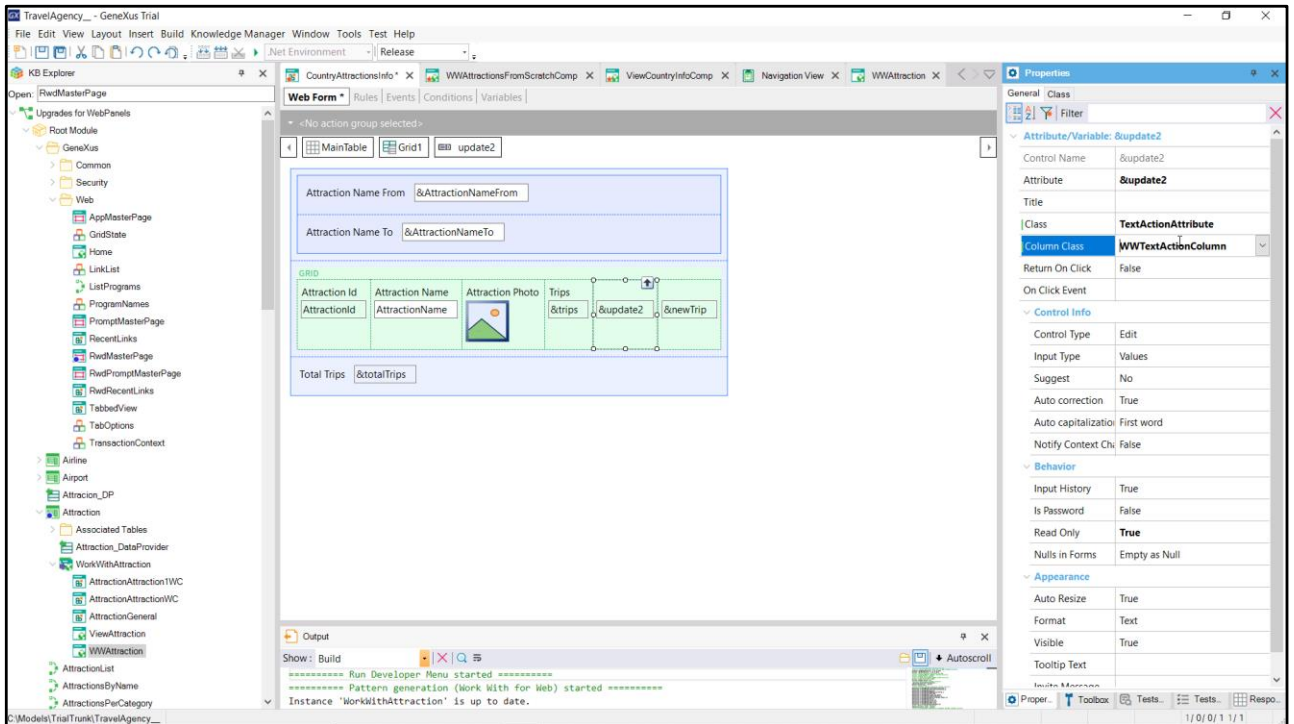
Now we run it...




We check how it looks in the pattern... Why does it have this much nicer design here?

Let's find the control properties in the web panel generated by the pattern. Let's look at these two: Class and Column Class.





And let's see the values of these properties for our control, the one we inserted manually in the web panel. They are different. Let's assign them the same ones as those of the pattern. And try it.






## Travel Agency

Country Name

France

Attraction Name From

Attraction Name To

Attraction Name	Attraction Photo	Trips	
<a href="#">Eiffel Tower</a>		2	<a href="#">UPDATE</a> <a href="#">New trip</a>
<a href="#">Louvre Museum</a>		1	<a href="#">UPDATE</a> <a href="#">New trip</a>
<a href="#">Matisse Museum</a>		2	<a href="#">UPDATE</a> <a href="#">New trip</a>
Total Trips		5	

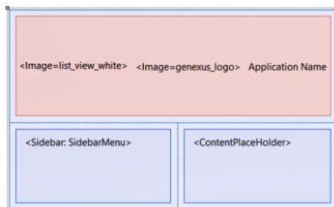
City Name

City Id	City Name	Attractions
1	Paris	2
2	Nice	1
Total Attractions		3

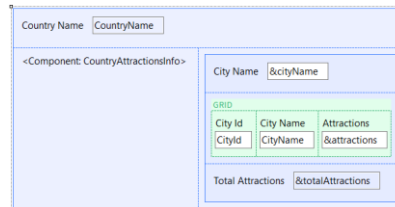
We can see it here. Here we begin to understand how to manipulate the design of the screens.

## Types of Web Panels

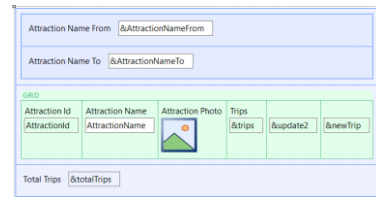
Master Page



Web Page



Component



In summary: We have seen three types of web panels that are related to each other.

The Master Page type, which offers a common layout for all the pages of the application or a part of it, so as not to have to repeat the same thing every time, for each page. For example, the menus usually go there. This object in particular contains in its form a special control, ContentPlaceHolder, where the web pages will be loaded.

That of Web Page type, which is the one we have been studying, can have a specific Master Page associated with it and only one, since it will be loaded in its ContentPlaceHolder.

And that of Component type, which is useful precisely to reuse the same design and programming in different objects. In order to have an object defined as a component type web panel loaded into another web object, the component type control is used, which can be loaded either statically or dynamically.

The more components we identify, and use, the better the app will be.

Of course, there is much more to study on this topic (for example, what

happens with the execution of events in a panel with components, how a component is refreshed, etc.). But this is more than enough for now.

*GeneXus*<sup>™</sup>

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