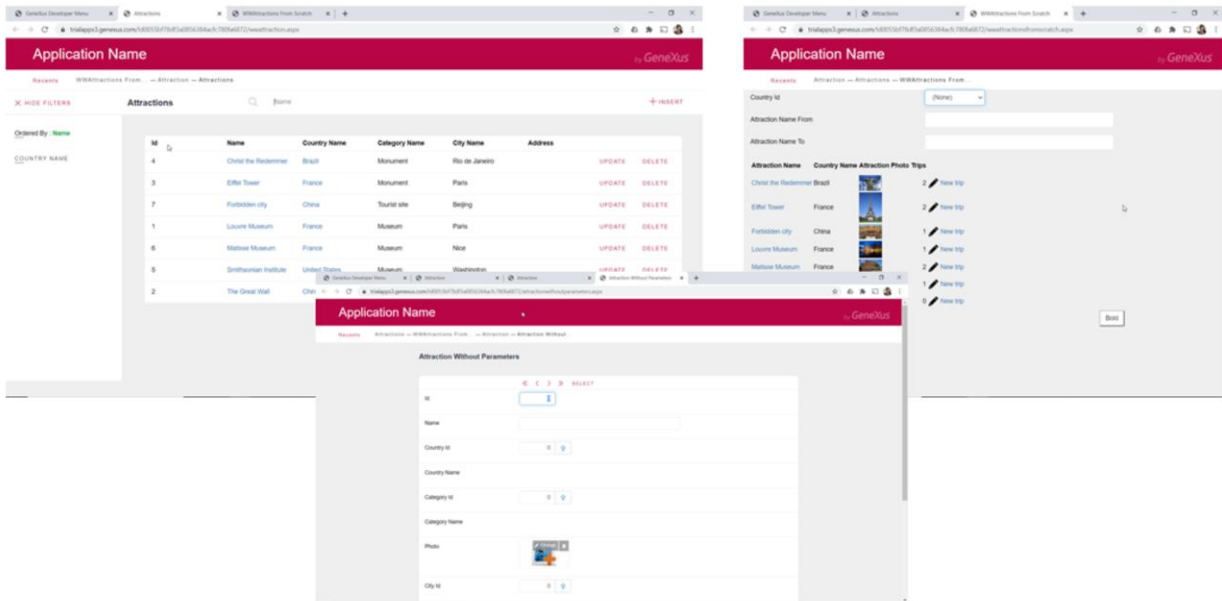


Web Screens with Back-office Focus

Types of Web Panels

GeneXus™

What all pages have in common



In previous videos, we have been building from scratch a solution that is very similar to the one automatically built by the Work With pattern for us.

Thus, if we compare both solutions at runtime, we see first that their design is different. So far we've focused on the logic and left out the User Interface, which we'll get into later on. But leaving aside those visual differences, in both web panels we can see a grid that shows information of the tourist attractions, with filters, and the possibility, for example, of updating the information.

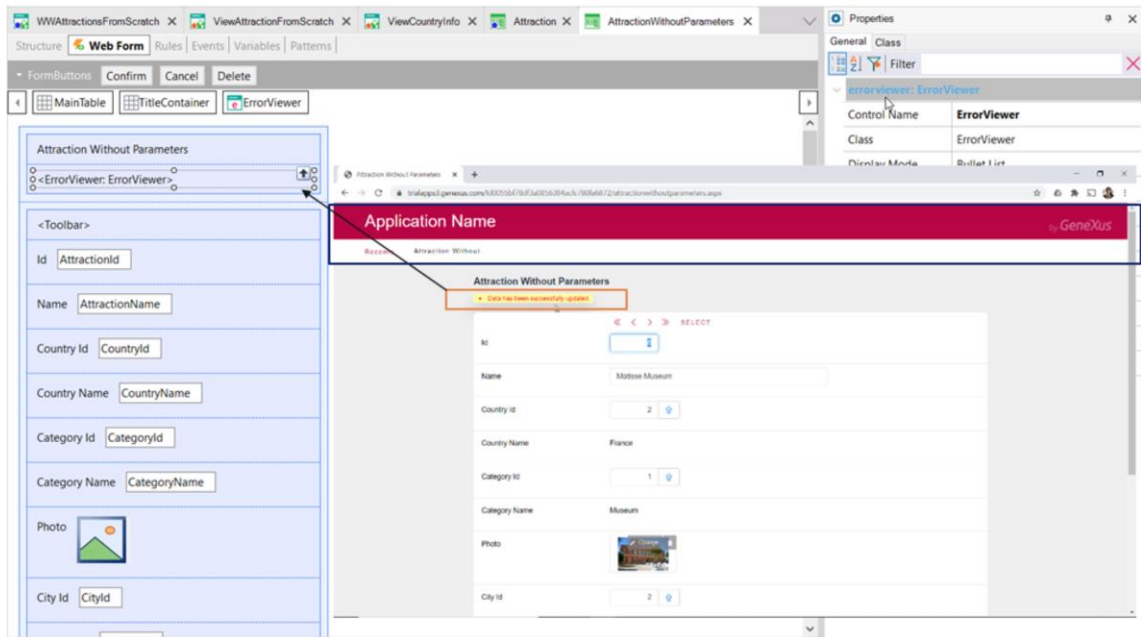
In both solutions, the transaction is invoked in Update mode.

Something that can start to call our attention is a common feature of all the screens we've been browsing: this bar up here and this one with the links we've recently browsed. We see that every page we open has those parts.

For example, if we directly invoke the parallel transaction `AttractionWithoutParameters`, here they are.

Or, if we now go to the View of an attraction, the one of the pattern or the one implemented from scratch by us, we see that beyond their differences, they still keep this in common.

Transaction controls



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

Or, if we open the Attraction transaction, or the parallel transaction that doesn't receive parameters, and we go to its Form, we don't see it either.

What we see here is a textblock control with the name of the transaction. Below is an ErrorViewer control that is where messages appear, for example, for success or failure. Then there is this "action group" control that shows the navigation and Select buttons. Next, there are attributes, and lastly another "action group" with the buttons.

Where is the upper sector?

Master Page

The image shows 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 **RwdMasterPage**. A dropdown menu is open, showing options: (none), AppMasterPage, PromptMasterPage, **RwdMasterPage** (selected), RwdPromptMasterPage, and RwdMasterPageCopy1.
- Web Panel Properties:** A table showing properties for a Web Panel named **WWAttractionsFromScratch**. The **Master Page** property is set to **RwdMasterPage**.
- Web Form Design:** A visual representation of a web form with a **<ContentPlaceHolder>** at the bottom, indicated by an arrow.
- Web Master Panel Properties:** A table showing properties for a **Web Master Panel: RwdMasterPage**. The **Type** property is set to **Master Page**.

If we look at the transaction properties we see a Web Transaction group with three very important properties: the first one, **Theme**, will control the design of the controls; the second one we'll see later, and now we'll see the third one, the **Master Page** property. There we indicate the master page inside which the page corresponding to this transaction object will be loaded. It offers some options, which correspond to all the master pages defined so far in our KB. When a KB is created, these pages are created so that we have some by default, but we will be able to create others.

The transaction was associated with this master page by default when it was created.

If we now open the other objects that we were seeing at runtime, we should not be surprised to find that they also have a master page, and it is this one.

By the way, note that the web panels also have the same three properties.

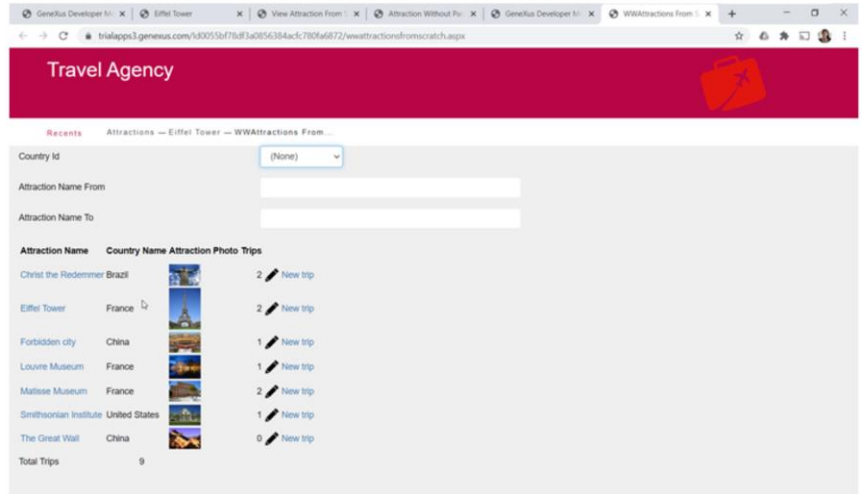
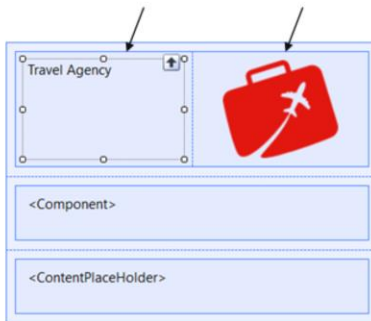
So let's look for this master page and see what it's all about. If we don't know where it is, we can look for it here and open it. And if we want to locate it in the KB Explorer, we can right-click on the tab to do so. It's inside a web folder where GeneXus places objects that have to do with the pattern, for example, and so on.

If we look at its properties, we also see one named **Theme** and one named **Type**, and the **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, **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, the view, and everything we saw.

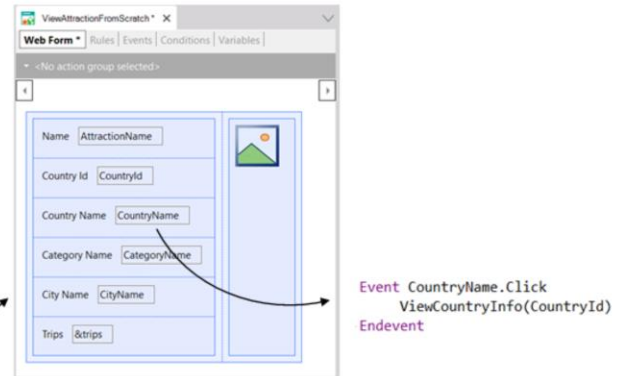
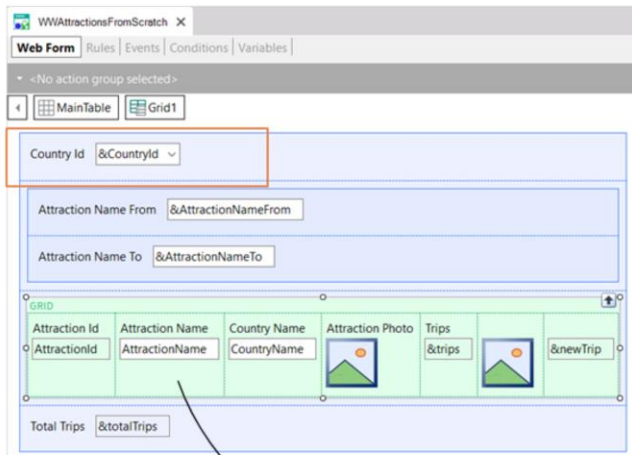
Master Page



Here is the text block control that we see at runtime called Application Name. For example, let's try to change it to Travel Agency. Here we have an image that we can also change for this one that we previously inserted in the KB.

Let's test the effect of these changes on our application. Here we have them...

Web Component



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 web component design interface for 'ViewCountryInfo'. The main form is divided into several sections:

- Country Name:** A text field labeled 'CountryName'.
- Attraction Name Range:** Two text fields: 'Attraction Name From' with value '&AttractionNameFrom' and 'Attraction Name To' with value '&AttractionNameTo'.
- Attraction Grid:** A table with columns: 'Attraction Id' (value: 'AttractionId'), 'Attraction Name' (value: 'AttractionName'), 'Attraction Photo' (with a photo icon), 'Trips' (value: '&trips'), and 'newTrip' (with a photo icon and value: '&newTrip'). Below the grid is a 'Total Trips' field with value '&totalTrips'.
- City Information:** A text field 'City Name' with value '&cityName' and a 'Total Attractions' field with value '&totalAttractions'.
- City Grid:** A smaller grid with columns: 'City Id' (value: 'CityId'), 'City Name' (value: 'CityName'), and 'Attractions' (value: '&attractions').

The interface includes a 'MainTable' tab and a 'Web Form' menu with options for Rules, Events, Conditions, and Variables. The status bar indicates '<No action group selected>'.

```
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

parm(in: CountryId);

The image displays two GeneXus IDE screenshots side-by-side, illustrating the development of a web component. The left screenshot shows a 'Web Form' for 'ViewAttractionFromScratch' with a 'MainTable' and a 'Grid1'. The right screenshot shows a 'Web Form' for 'ViewCountryInfo' with a 'MainTable' and a 'Grid1'. Below the IDE screenshots, three rows of UI elements are shown, representing the output of the web panels. Each row contains a text label, a small image, a number, and a button. The first row shows 'Louvre Museum', a photo, '1', and a 'New trip' button. The second row shows 'Louvre Museum', a photo, '1 UPDATE', and a 'New trip' button. The third row shows 'Louvre Museum', a photo, '1', and a 'New trip' button. Arrows indicate the mapping from the 'New Trip' buttons in the IDE panels to the buttons in the UI rows.

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 );
```

The screenshot displays the GeneXus IDE interface for a web form named 'CountryAttractionsInfo'. The form contains several elements: a 'MainTable' with two input fields for 'Attraction Name From' and 'Attraction Name To', a 'GRID' with columns for 'Attraction Id', 'Attraction Name', 'Attraction Photo', and 'Trips', and a 'Total Trips' field. A 'Grid's Conditions' dialog box is open, showing the following conditions:

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

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

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

The Properties window on the right shows the following configuration for the 'Web Component: CountryAttractionsInfo':

Name	CountryAttractionsInfo
Description	Country Attractions Info
Module/Folder	Root Module
Theme	Carmine
Type	Component
URL access	No

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 displays the GeneXus IDE interface. On the left, the 'Web Form' design view shows a 'Country Id' dropdown menu and a component control labeled '<Component: CountryAttractionsInfo>'. The 'Properties' window on the right shows the configuration for 'Web Component: Component1':

Control Name	Component1
Object	CountryAttractionsInfo
Parameters	&CountryId
Cell information	
Cell Control Name	
Cell Class	
Horizontal Alignment	Default
Vertical Alignment	Default
Row information	
Row Height	
Row Class	

On the right, a preview window shows the rendered web form. A dropdown menu for 'Country Id' is highlighted with a red box and contains the text 'France'. Below it, a table lists attractions with columns for 'Attraction Name', 'Attraction Photo', and 'Trips'.

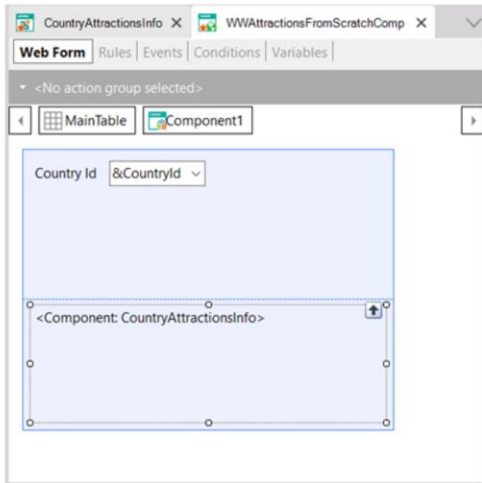
So, the next thing we did was a Save as of our original panel, and replaced all those controls that we now place in the component web panel, with a component type control. Of course we deleted (here we leave comments) all the events that will now be in the component.

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.

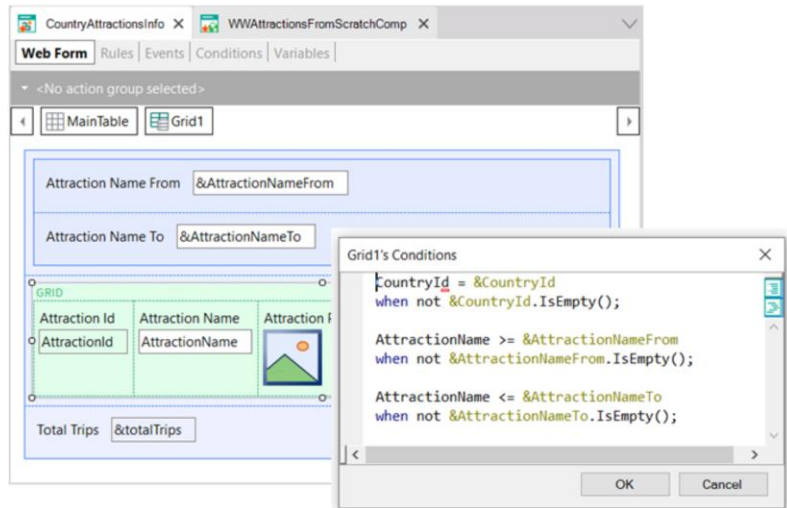
We see exactly the same. If we filter by name of attraction... it's working perfectly.

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

Component



```
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 a web form editor with a panel containing a grid and a total attractions field. The grid has three columns: City Id, City Name, and Attractions. The total attractions field is labeled &totalAttractions. The Properties window on the right shows the component's configuration:

Control Name	Component2
Object	CountryAttractionsInfo
Parameters	CountryId

Below the table, the Cell information section shows the following code:

```
Event Grid2.Load
    &attractions = Count(AttractionName)
    &totalAttractions = &totalAttractions + &attractions
endevent

Event Grid2.Refresh
    &totalAttractions = 0
Endevent
```

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.



Recents WWAttractions From... — View Attraction Fr... — View Country Info ...




Country Name: France

Attraction Name From:

Attraction Name To:

City Name:

Attraction Name Attraction Photo Trips

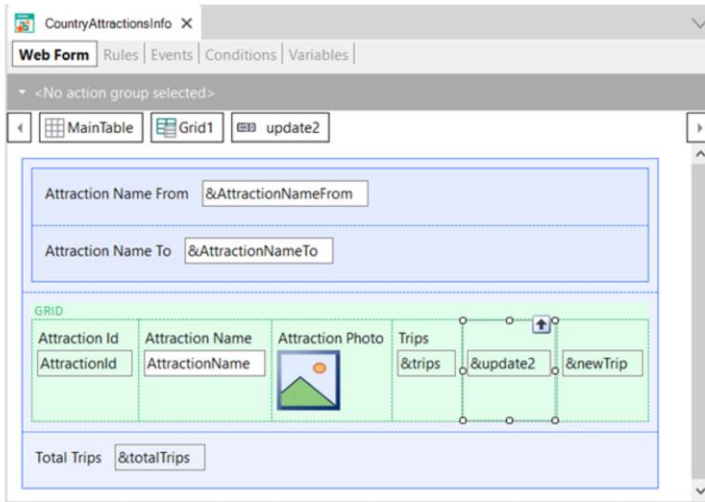
Attraction Name	Attraction Photo	Trips
Eiffel Tower		2 New trip
Louvre Museum		1 New trip
Matisse Museum		2 New trip
Total Trips		5

City Id City Name Attractions

1 Paris	2
Total Attractions	2

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

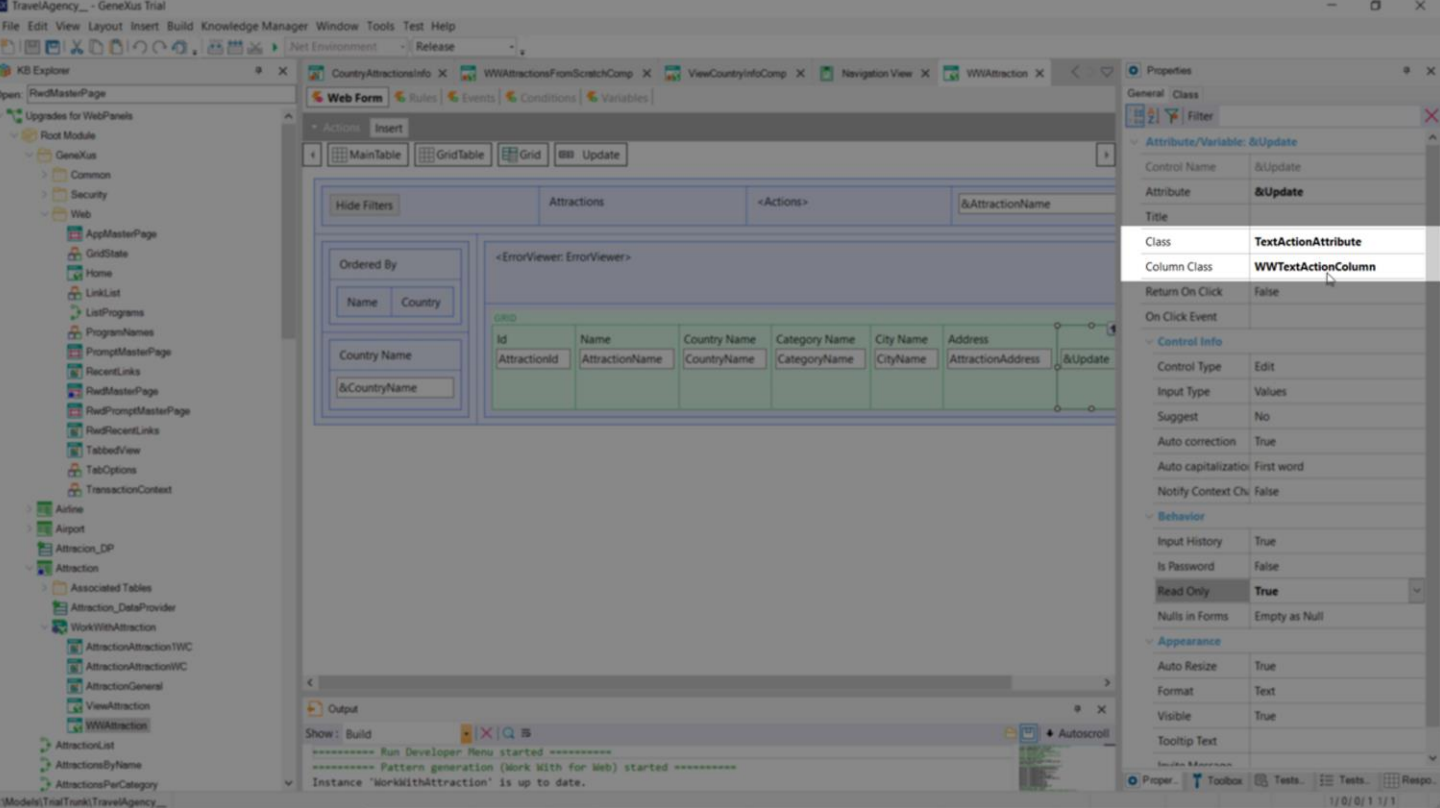
```

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.

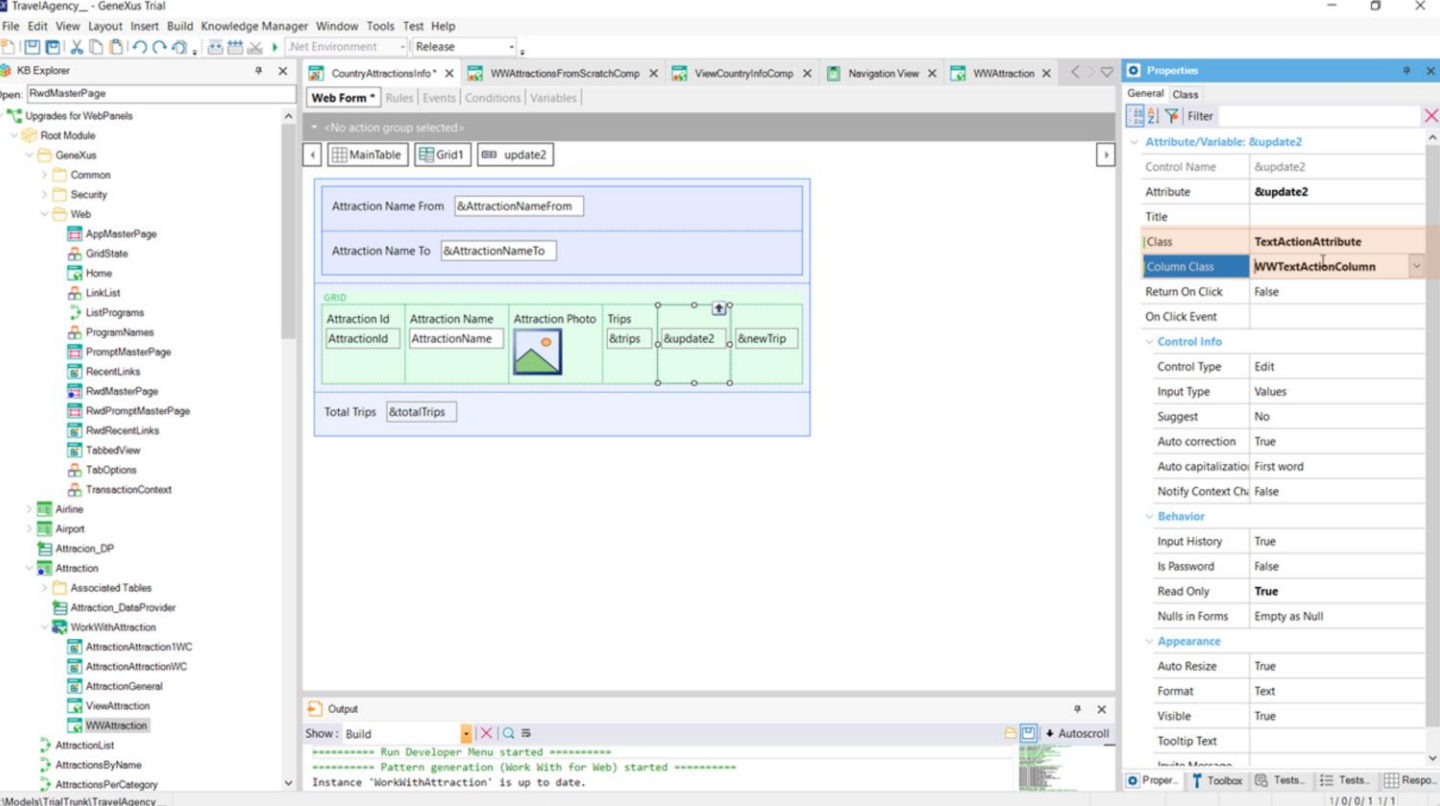
Let's try to run it. It says that the click is not a valid event. We forgot to change it so that it is the click of the new variable and not the old one.

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.






Recents View Country Info ...

Country Name France

Attraction Name From

Attraction Name To

City Name

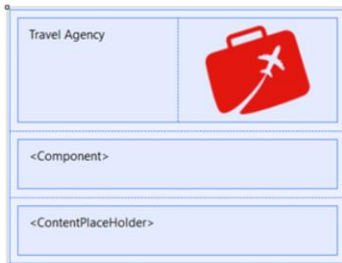
Attraction Name	Attraction Photo	Trips
Eiffel Tower		2 UPDATE New trip
Louvre Museum		1 UPDATE New trip
Matisse Museum		2 UPDATE New trip
Total Trips		5

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

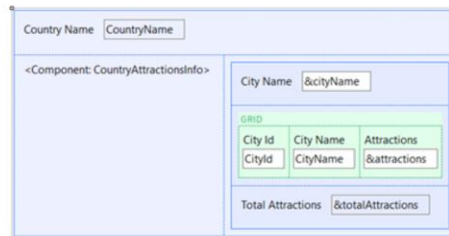
We can see it here. Here we begin to understand how to manipulate the design of the screens.

Types of Web Panels

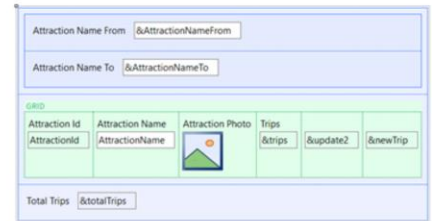
Master Page



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™

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