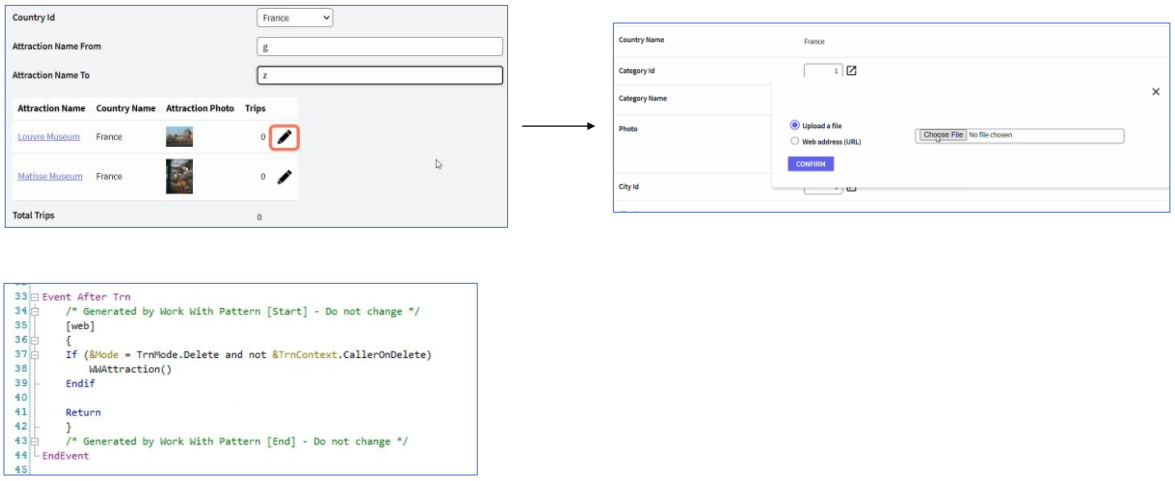
Interactive Screens

How to save context information

GeneXus™

GeneXus

Web panel WWAttractionFromScratch




```
Event After Trn
    /* Generated by Work With Pattern [Start] - Do not change */
    [web]
    {
    If (&Mode = TrnMode.Delete and not &TrnContext.CallerOnDelete)
        WWAttraction()
    Endif

    Return
    }
    /* Generated by Work With Pattern [End] - Do not change */
EndEvent
```

In this example, you will learn how to keep data in memory to avoid losing it after calling another object, and then return to the first one.
To illustrate this situation, the previous example will continue to be used.
Here is a short summary.
In the Web Panel that we see on screen, when entering values in the filters, the grid is refreshed and shows only the data we are interested in, conditioned by those values.

Selecting the update action in one of the rows, as you've seen, will take you to the Attraction transaction in Update mode. There, you can edit the values corresponding to the selected attraction.
You can choose one of the attractions to modify, for example, the Louvre Museum.

Change the image corresponding to the AttractionPhoto attribute and select Confirm.

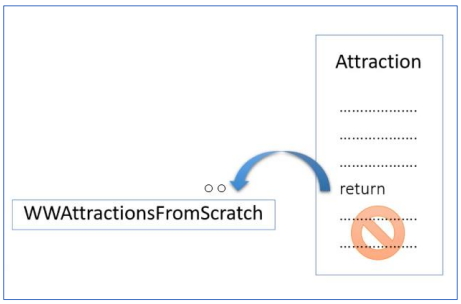
Note that this action brings you back to the Web Panel.

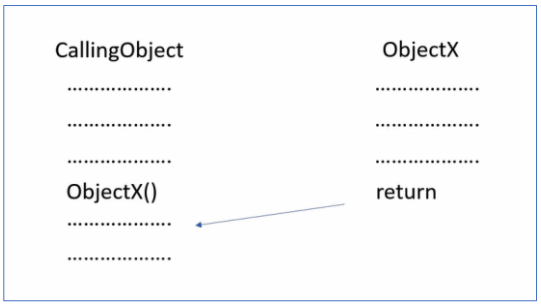
The reason is that the Work With pattern applied to the Attraction transaction has automatically added the Return command, in order to

return to the caller object. This can be seen in the Events section of the Attraction transaction, programmed within the “After Trn” event.
The “After Trn” event is triggered when the transaction has completed a cycle, immediately after the commit operation; that is, after a header has been recorded with the corresponding lines.

GeneXus

Web panel WWAttractionFromScratch






```
Event After Trn
    /* Generated by Work With Pattern [Start] - Do not change */
    [web]
    {
    If (&Mode = TrnMode.Delete and not &TrnContext.CallerOnDelete)
        WWAttraction()
    Endif

    Return
    }
    /* Generated by Work With Pattern [End] - Do not change */
EndEvent
```

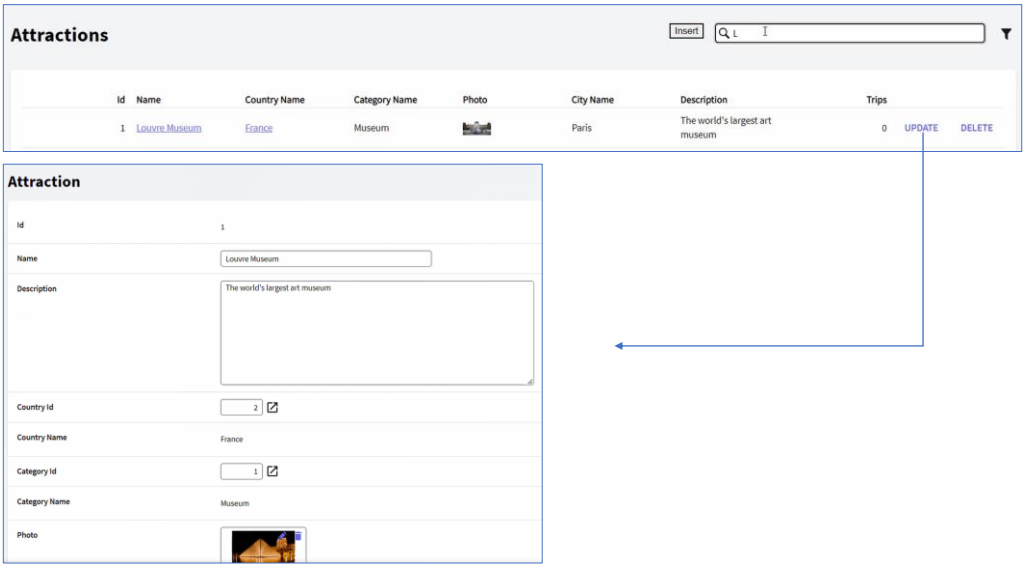
≡ WWAttractionsFromScratch()

The return command's function is to end the current execution on this object, and return to the caller object.
In this example, both the WWAttractionsFromScratch Web Panel and the Attraction transaction, i.e. both the calling and the called object, are objects with a graphical interface. So in this case, the Return command is the equivalent of invoking the Web Panel for the first time.
It would be different if one of these two objects didn't have a graphical interface, as is the case with a procedure. There the Return command of the object that was called will take you back to the next line right after the invocation.
In this case, when returning, the events associated with the Web Panel loading will be executed. First the Start event, followed by the Refresh and then the Load events; the latter is executed as many times as there are records in the grid that meet the conditions stated in the conditions property. Since here the conditions don't apply if the variables are empty, at runtime you'll see that all the attractions are reloaded.

Why does this happen? It will be explained very soon.

GeneXus

Pattern Work With Attraction

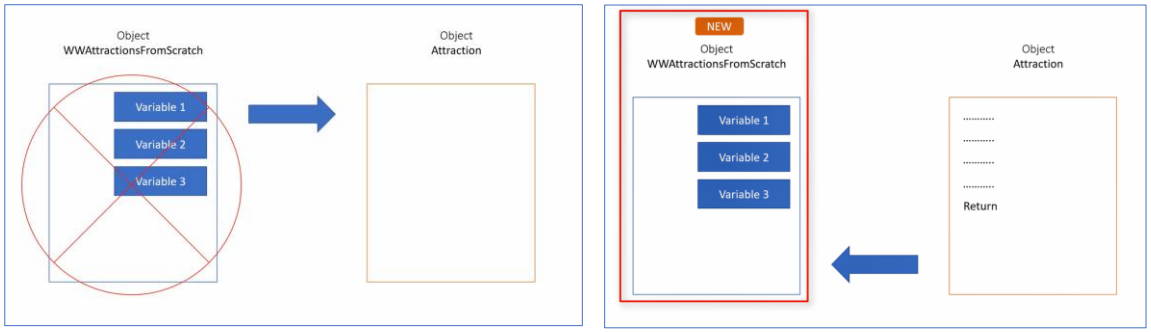


First, look at the behavior of the Web Panel created automatically from the patterns section of the Attraction transaction.
Filter by attraction name by entering the letter “L.” All the attractions that begin with this letter will be displayed; in this case, the only one that has been entered with this characteristic is the Louvre Museum.
Now, select the action to update this attraction.

Note that it will take you directly to the Attraction transaction and show you the selected attraction allowing you to edit it, exactly as in the case of the manually implemented web panel.
After changing the photo and confirming the update, you will return to the calling Web Panel.
Note that the filters are kept. This is exactly the feature you need to achieve.

GeneXus

Pattern Work With Attraction



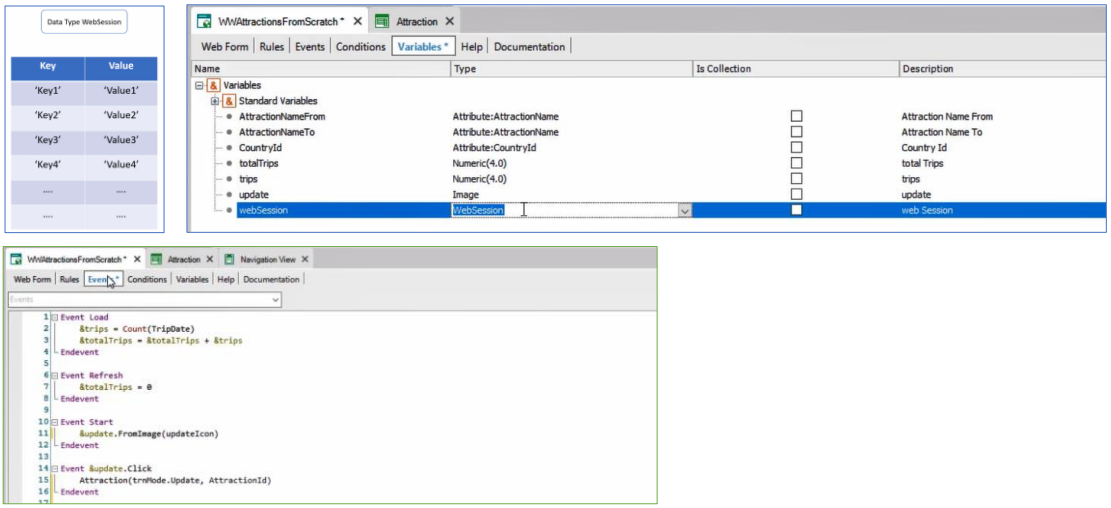
Next, a solution will be programmed in the Web Panel, and an explanation will be provided about the way the Pattern does it.
Remember that the variables stated in each object can only be used within them as long as the objects are active.

For example, in this case, the moment you invoke the Attraction transaction from the Web Panel, your WWAttractionFromScratch object is destroyed and you get its variables. Therefore, any values that were saved will be lost. On the other hand, the Attraction object will take an active status.
Then, when returning from the transaction to the Web Panel with the Return command, the latter object and its variables will be recreated.
This is why filter values are no longer displayed: it is actually a new object.
Also, the variables used as filters have been created again; the ones existing before invoking the Attraction object were destroyed.

This answers the question asked a moment ago, about the reason why the values entered in the filters were not kept after a record was updated.
You need to save the information of each one of the filters in a sort of global variables, so that it is not lost between executions, that is, when passing from one object to another. In this case, between the Web Panel and the transaction, and from the transaction back to the Web Panel.

GeneXus

WebSession Data Type



So how do you keep the value of a variable in memory?
Using GeneXus, you can program this feature using variables of Web Session type.
These variables allow you to handle a group of global variables, in which you can store data and access it from any object, while the session is active.
This is exactly the purpose of the example.
A major advantage of these variables is that they allow you to store a set of data of the key-value type. So you only need to create a Web Session variable, and use it to save all the filters you have, each of them with a unique key.

In this way, there is a key and a value for the CountryId filter, another key and a value for AttractionNameFrom and finally for AttractionNameTo, the three filters that you need to keep between runs of your panel.

This meets your need to temporarily save the information entered in the filters, to retrieve it later. See how to program this.
First of all, create a variable and call it “WebSession.”

When you give it this name, GeneXus assigns the WebSession data type, understanding that this is probably what you are interested in, which in

this case is correct. If this is not the case, you can always change the data type automatically assigned for the one you want.
Then, you will have to consider when you want this variable to keep the desired value(s).
Remember the main events that are currently scheduled:

- The Start event, which will be executed only once when the page is first loaded.
- The Refresh event, which will be triggered after the Start event and every time a filter within the grid conditions is changed or the page is refreshed from the browser.
And the Load event, which will be executed after the Refresh event, as many times as data is loaded in the grid.

This event will be executed N times because it has a base table associated with it.
Which of these three options do you consider to be the best to save this data?
From these options, clearly the most convenient one would be within the Refresh event, since every time you change any of the filter values, this event will necessarily be triggered, and that's when they would be saved in the session variable, so that you can retrieve them when returning from the transaction.
But there is also another event, which is the Click event of the Update variable.
The event will be triggered immediately after clicking on the Update action. This would also be useful for saving filter values here.
It is a good option, since this is the event where you will invoke the Attraction transaction, and as you've seen, this is the exact moment when the calling object is destroyed and you get its variables. That is why you need to save the values of the filter variables before doing this.

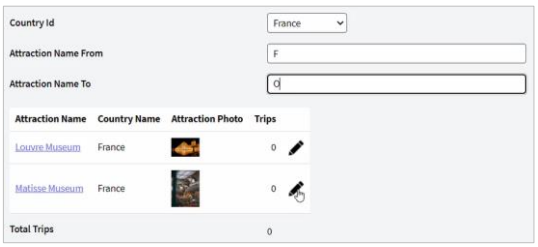
If you do it in the Refresh event, you will save the filter values every time you change data in any of them. Since it is in the grid conditions, every change triggers this event. But this is not necessary, as you may not need to update the filtered attractions at all in the current run. And in that case, why would you save those filters in the session, if the variables are not going to be destroyed?
However, if you do it in the event related to the Update action, you will save these values only once, when it is essential to do it, immediately before the object and its variables are destroyed.
This will be done in the Click event of the variable.

GeneXus

WebSession Data Type

```
Event Start
    &update.FromImage(updateIcon)
    &CountryId = &webSession.Get('CountryId').ToNumeric()
    &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
    &AttractionNameTo = &webSession.Get('AttractionNameTo')
Endevent
```

```
Event &update.Click
    &webSession.Set('CountryId', &CountryId.ToString())
    &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
    &webSession.Set('AttractionNameTo', &AttractionNameTo)
    Attraction(trnMode.Update, AttractionId)
Endevent
```



Type the webSession variable and apply the Set method to store values in it; as the first parameter you need to enter a key, which must be a value of character type, so you must enter it in quotes.
In this case, it is called CountryId. This unique key will later be useful to recover the value you save.
Then you must assign a value to it, which will be the data you want to store in memory. In this case you will want to save the value of the CountryId variable, which will be the value of the first filter you have on screen.

Keep in mind that the entered value must also be of Character type, so, as the CountryId variable is of numeric type, it must be converted to the Character type. This is achieved by applying the ToString method to the variable.
Do the same for the other two filters, AttractionNameFrom and AttractionNameTo.
Next, the application will be run again, and values will be entered in the filters.
Remember that every time the value of one of the filters is changed, the Refresh event is triggered, followed by the Load event for every record loaded in the grid.

Select the Update action of an attraction.
At that moment, the event &Update.Click will be triggered, which has been programmed to save the filter values in the &webSession variable. This is done in order to invoke the Attraction transaction later on. As you have seen, it will be the moment when your Web Panel object and its variables are destroyed.

After changing data in this attraction and confirming it, the Return command will be applied. Since in this case it is the equivalent of calling the Web Panel object for the first time, the Start, Refresh and Load events will be run again for each value to be loaded in the grid.
Now, you'll need to be able to retrieve these values saved in the webSession variable.

What do you think would be the best time to retrieve these values?
Go over the events you have and assess them:
In the Start event?
In the Refresh event?
In the Load event?
In the &Update.Click event?
Clearly, the only one that will work for you here will be the Start event. As you've seen, this event is executed only once when the Web Panel is loaded for the first time.

When returning from the transaction, it is the equivalent of calling the Web Panel for the first time, and it is precisely the moment when you need to retrieve those values.
In order to assign a key and value to the webSession variable, use the set method. Now you need to know how to retrieve the value stored there. You do this using the Get method, in which you must indicate the key of the value to be retrieved.

Then, to each variable used as a filter, apply the Get method, sending the key corresponding to each one by parameter.
This will be done first for the CountryId variable, step by step.
Enter the CountryId variable, and assign it the value of the Get method of the webSession variable, passing the key as a parameter, i.e. 'CountryId.'
Remember what has been said before: Web Session variables only store data of Character type. As the CountryId variable is of Numeric type, to assign it the value of the WebSession variable, which will be of Character type, you will have to convert this information into numeric type; this is achieved with the ToNumeric() method.

Next, do the same for the AttractionNameFrom and AttractionNameTo variables.
Run the application again and test its behavior.
Filter all the attractions in China between A and M.
In this case, only one attraction will be displayed.
Select the action to Update it, and remember that at that moment, before invoking the transaction, the data of your filter variables will be saved in memory.
Change the photo and confirm the action.
When returning to the Web Panel, as it has been mentioned, the first event that will be executed will be the Start event, where you program the retrieval of the information saved in the session variable, assigning those values to your filter variables.
Note that now, the values entered in the filters are maintained as desired.
In addition, after the Start event, the Refresh event will have been triggered, which in turn will have triggered the grid loading according to the filters. That's why the grid is filtered again, with the modified photo.

But once this has been programmed, what will happen when the user first opens this web panel in a browser? In that case, there will be nothing saved and therefore nothing to retrieve.
Close any active session on your browser, and run the application from GeneXus again for the entire cycle.
Open the Web Panel you've created.
And since this session is the first time the Web Panel is run, the first thing that is triggered is the Start event.

The session variable will look through the entered keys to see if there is any information to retrieve, but there won't be any, as no value has been stored in the &webSession variable yet. So, at this time the variables used for filters will remain empty.
All the attractions are displayed because in the grid conditions it has been indicated not to apply any filter if these variables are empty.

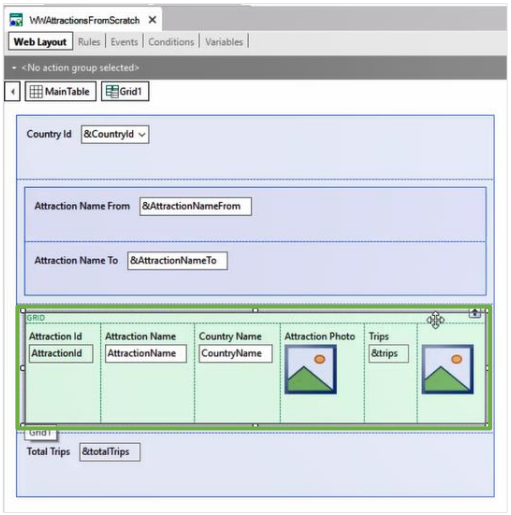
For example, select to filter by the country France, and you will see that the grid already applies the filter, showing only the attractions that have France as a country.
As Attraction Name From filter, enter the letter F, and at that moment the Refresh event is triggered again, followed by the Load event. Finally, in the Attraction Name To filter enter the letter O.
Select the Update action on the Matisse museum attraction.

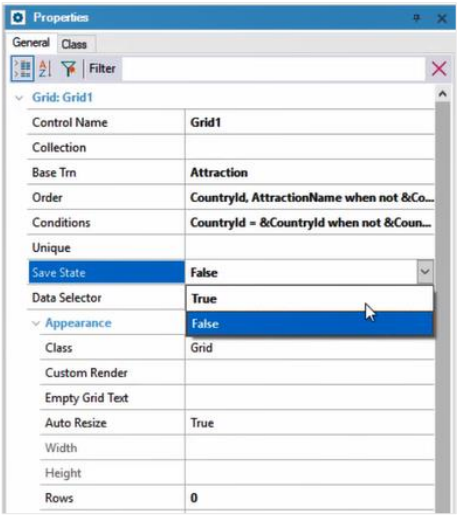
At that moment the &Update.Click event is triggered, in which you programmed that before invoking the Attraction transaction, the data of the filter variables is saved in the &webSession session variable.
To this end, the Set method of the session variable is used, passing the key and the value that you want to save in a parameter. In this case, there are three values to keep in memory. They are the variables &CountryId, &AttractionNameFrom and &AttractionNameTo.
At this point, the Attraction object is invoked and becomes active. And the Web Panel object is destroyed along with its variables.
Change some data of the attraction; for example, its photo, and confirm the action.

It will then take you back to the calling object, that is, your Web Panel. It does this because, as seen before, the Work With pattern applied to the Attraction transaction added the Return command within the After Trn event.
At this moment, the Start, Refresh and Load Events are executed again for each record to be loaded in the grid. This happens because the Return command in this case is the equivalent of invoking the Web Panel for the first time.

The values saved in the session variable will be loaded in the Start event. This is done using the Get method, passing by parameter the key used to save each value in the event &Update.Click using the Set method.
This data will be assigned to each corresponding variable, in this case to the three variables used for the filters: &CountryId, &AttractionNameFrom and &AttractionNameTo.

Save State propery





In this way, we learned to use the session variables provided by GeneXus, so that at any time we can save information in it and then retrieve it when necessary.

As GeneXus users, we often need to save the state of a grid to recover it later; for example, as we just saw, when we call another target and return to our web panel with a grid, and we want the filters we entered to remain. For this reason, a property called "Save State Property" has been added to the grid control in the latest versions of GeneXus. Basically, what this property does is to allow us to save in memory –in a "Web session" variable– all the information about the state of the grid (the grid page we were in, the filters applied, etc.). When this page is reloaded, the last known configuration is automatically restored, which is what we did manually during this video with the session variable, but it will be done by configuring a property, in an automatic and totally transparent way, since we will not see added code in the generated objects. Let's see it at runtime.

First of all, we delete what we did in the events section of the web panel, leaving it in the initial state of the video, so that it is not saved anywhere. From our web panel object we select the grid and look at this new property that we mentioned, which by default does not disappear when it

is false; we change it to true and run it again.
We enter values in the filters and select to modify the information of one of the attractions; as we saw, this calls from one object to another and then when we confirm the action in the called object, it returns to the calling object. We make changes, confirm, and when we return to the web panel, we see that the filters are kept, with the state it was in before calling the Attraction transaction.

Even if this grid had paging, we would still keep the position we were in before updating the record.
We have two other ways of saving and retrieving context information, which we will not go into in more detail.
One of them is by using the following methods of the grid control: SaveSessionState, which allows saving the state of the grid. And LoadSessionState, which will allow retrieving the state of the previously saved grid.

Using these methods is equivalent to using the Save State property of the grid that we just saw.
The other is by using the ClientStorage external object.
This object is an API, which is used to store context information in applications for mobile devices. Its use and operation is similar to that of WebSession variables.
It allows storing information with key-value pairs, locally, that is, in the mobile device we are in, and it will be possible to access it later, even without connectivity.

For more details on each of these implementations, please visit our Wiki.

GeneXus™

training.genexus.com
wiki.genexus.com