Events in Transactions

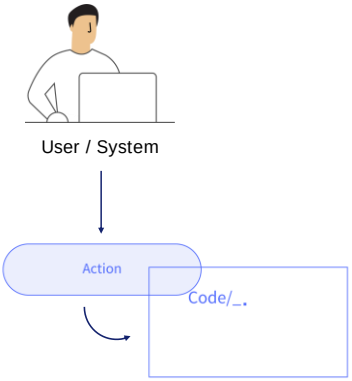
GeneXus™

Let's continue to advance our knowledge of the Transaction object.

We have already seen that GeneXus uses the structure defined in a transaction to automatically create its form, and that its behavior can be defined by declaring rules.

Let's talk now about the events in transactions.

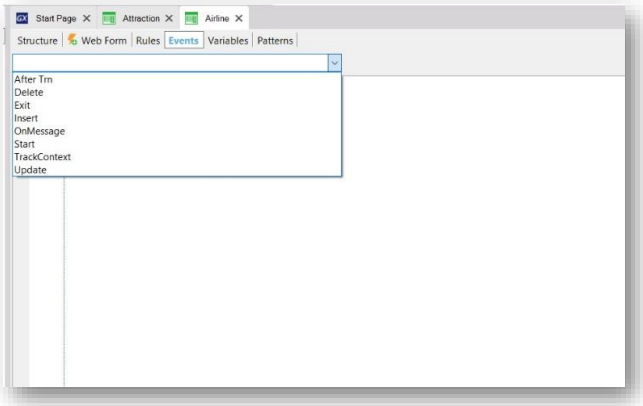
Event: Concept



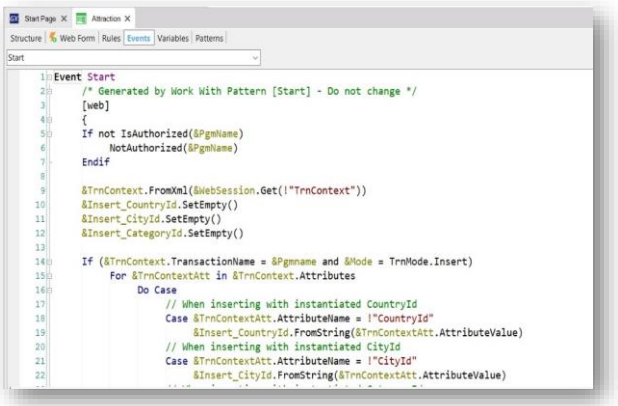
An event is an action performed by the user or by the system, which activates a certain code in response to that action.

Events in Transactions

In a common transaction.



In a transaction with pattern Work With for Web.



First take a look at the Airline transaction. If you go to its events sector, you will see that it is blank, and that you can edit and schedule the events that you need.

Note that there is declared code here, and that even though we are not going to analyze it, it is important to keep in mind that it was automatically generated when the Work With for Web pattern was applied.

Available events

Start: Defines an action to be executed when you open and start working with the transaction.

Track Context: Allows you to schedule what action to take when a change is made in the context of the transaction execution.

OnMessage: This event is related to web notifications that allow performing actions in real time.

Insert, Update y Delete: Associated with the concept of updating the Dynamic Transactions that will not covered in this course.

Exit: Allows you to indicate what actions to take when the execution of the transaction is completed; that is, when the transaction is already closed.

After Trn: Allows you to indicate what action to take after each execution cycle of the transaction; that is, immediately after the Commit has been made.

Begin with the **Start** Event: This event It is a system event, and is generally used to assign values to variables that will later be used during the execution of the transaction.

Continue with the **TrackContext** event: This event allows you to schedule what action to take when a change is made in the context of the transaction execution. For example, this event can track the position of the cursor on a certain control, and then execute the code programmed here.

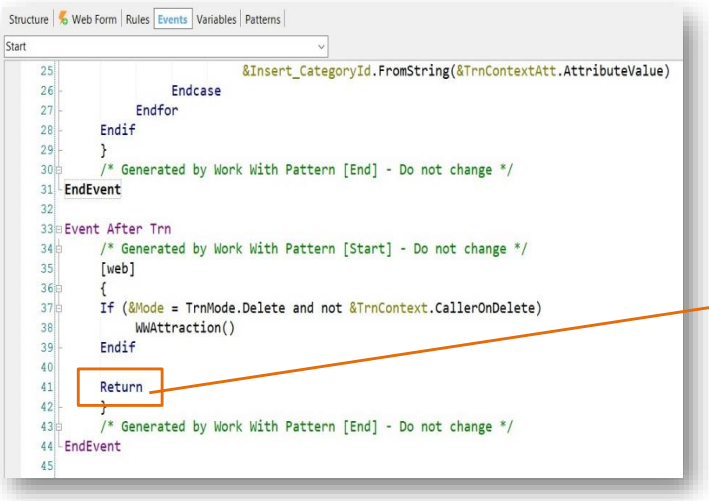
The **OnMessage** event is related to web notifications that allow performing actions in real time.

The events **Insert, Update, and Delete** are associated with the concept of updating the Dynamic Transactions that will not covered in this course. For this reason, I will only say that these events apply to a special case of use of transactions.

The **Exit** event allows you to indicate what actions to take when the execution of the transaction is completed; that is, when the transaction is already closed.

Finally, focus on the **After Trn** event. **This event allows you to indicate what action to take after each execution cycle of the transaction; that is, immediately after the Commit has been made.** Looking back at what has already been discussed about rule triggering moments, we can see that this AfterTrn event is executed just like the triggering moment on AfterComplete.

AfterTrn event



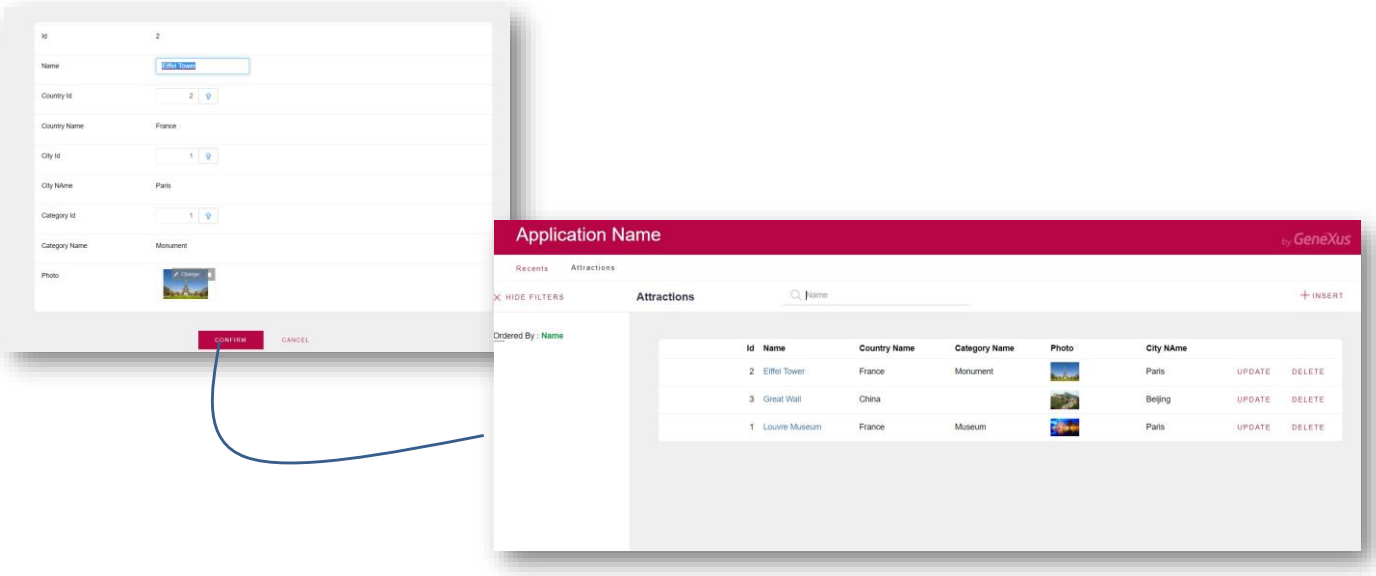
When inserting, deleting, or modifying a tourist attraction at runtime, it automatically returns to the Work With Attractions main screen.

Go back to the Attraction transaction and look at the code declared in its AfterTrn event.

As you can see, when the Work With for Web pattern was applied, code was added in this AfterTrn event. In particular, it added the Return command.

As a result, when inserting, deleting, or modifying a tourist attraction at runtime, it automatically returns to the Work With Attractions main screen. Remember that the Commit is made for each work cycle with the transaction form; therefore, the AfterTrn event will be triggered and this Return command will be executed.

AfterTrn event at runtime



Suppose that you change its name. When you press the Confirm button, a work cycle with the transaction form ends, and the Commit is triggered followed by the AfterTrn event.
Therefore, the Return command declared in this event will be executed and you will return to the initial Work With Attractions screen.

You may wonder, why don't you indicate the Return by conditioning it to the on AfterComplete moment? Because it is a command and not a rule. So, it must be declared in an event, and the event that is triggered after Commit is the AfterTrn event.

Finally, remember that if you need to implement some functionality that requires adding code to an event, and you see that the event already has code that has been automatically generated by GeneXus, then you must declare your code outside the marks of the code that is automatically maintained by GeneXus.

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