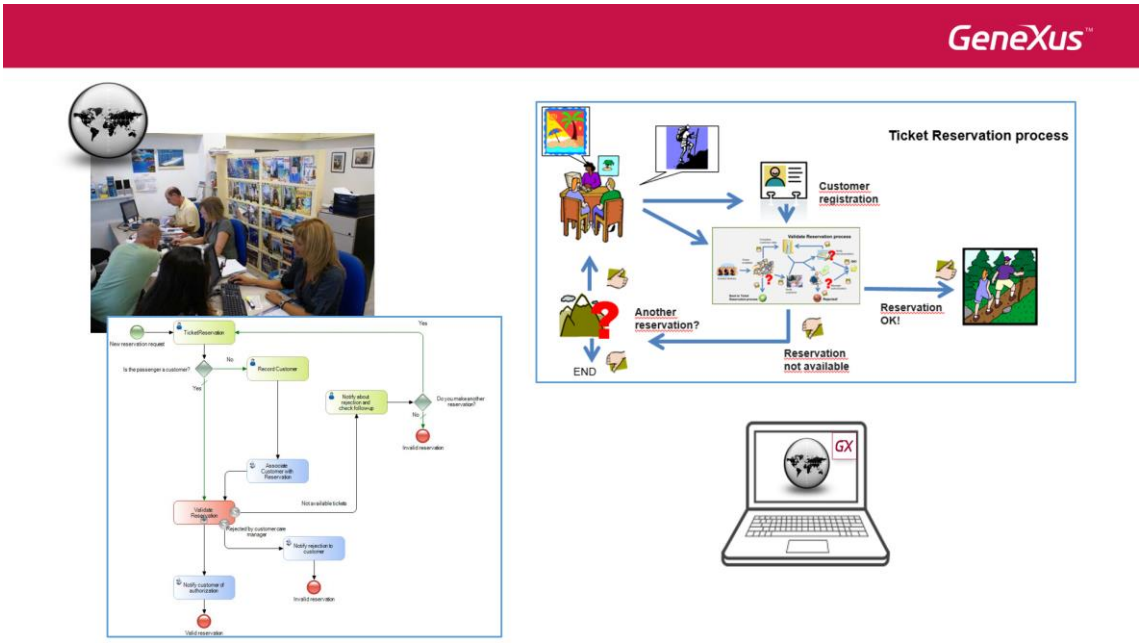
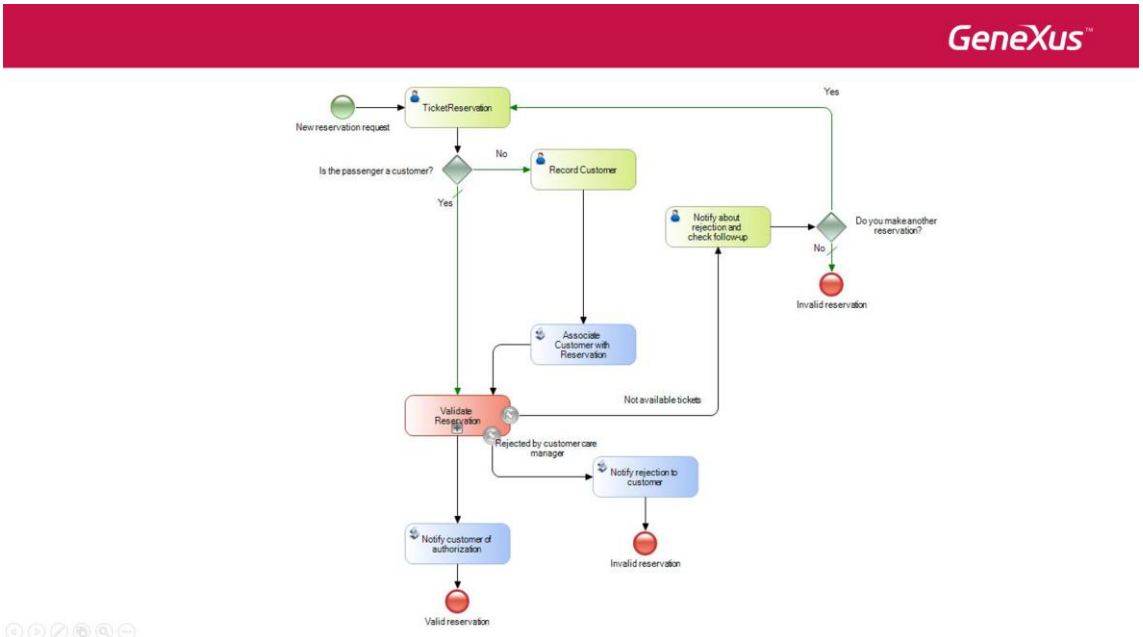
Turning the business process diagram into a functional application

In previous videos we have modeled the air ticket reservation process for a Travel Agency by means of a business process diagram, and we also saw a brief introduction to the BPMN standard, as well as the ways in which GeneXus facilitates the building of diagrams compliant with such standard.



Here, we will turn the model into a functional application using GeneXus objects associated with the symbols in our diagram.

Let's start by the process diagram we defined for ticket reservations.

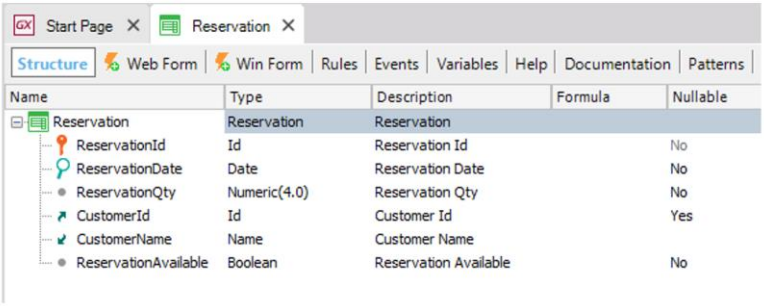


The first TicketReservation activity is interactive and its function is to record the ticket reservation.

To accomplish this we could use a transaction object with the following structure:

GeneXus

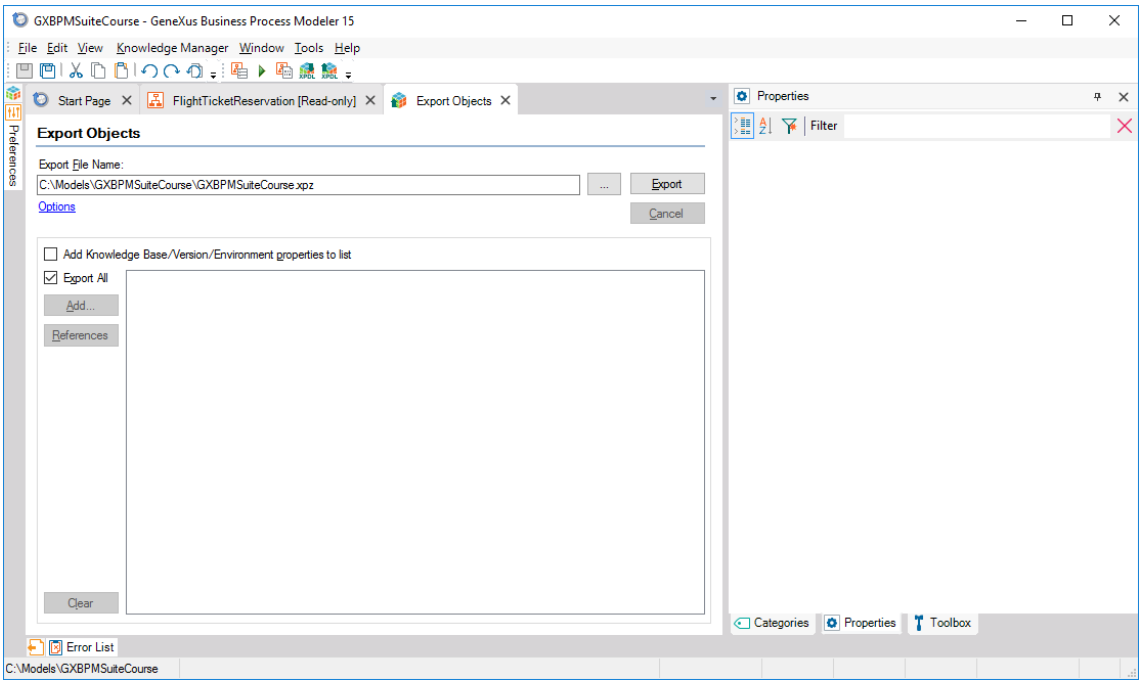
Reservation transaction object



Here we see that the ReservationId attribute is the transaction's identifier, and also, we have the ReservationAvailable attribute to indicate whether the reservation is available or not.

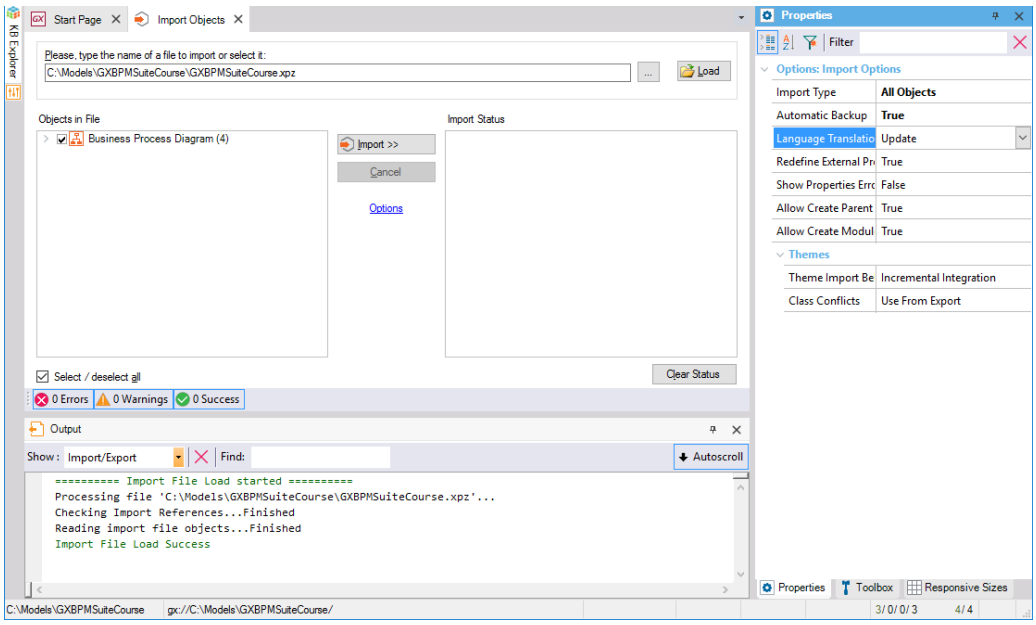
In order for the screen of the Reservation transaction to be opened when the TicketReservation task is executed, we will associate the task to the transaction object. We will be performing these assignments from a GeneXus KB, so we will need to import the diagrams we created with the Business Process Modeler.

We execute the Business Process Modeler, then open the GXBPMSuiteCourse KB, select Knowledge Manager and there we select Export. We check Export All and then Export.



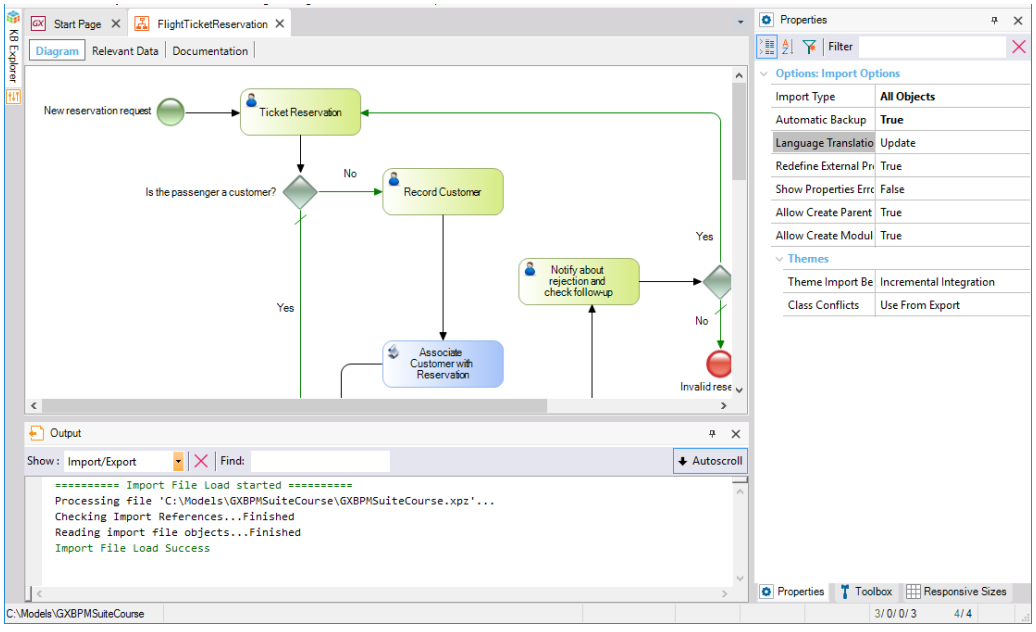
We now execute GeneXus and open the GXBMPSuiteCourse KB that we had created previously, where we had created several objects.

We select Knowledge Manager and then Import. We search for the file we had created from the Business Process Modeler and press Load.



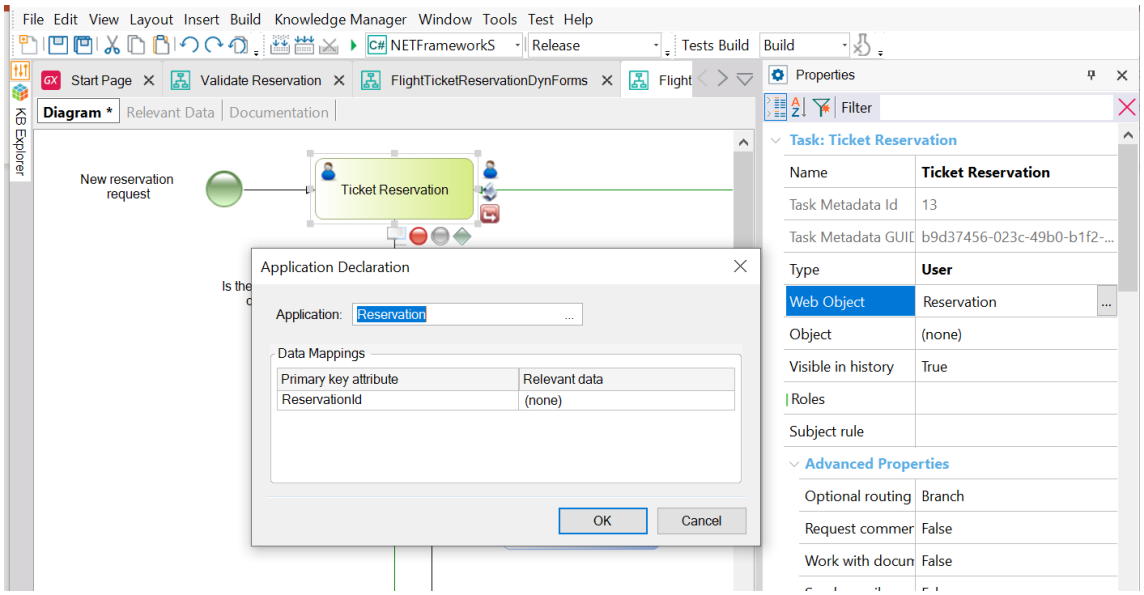
In this case we are using an export file with extension .xpz to take diagrams from the GXBPM to GeneXus. Another possibility could have been to send, from the GXBPM to the KB, to a GXServer and then do a Create KB from Server from GeneXus.

Now we open the FlightTicketReservation object with Ctrl-O.



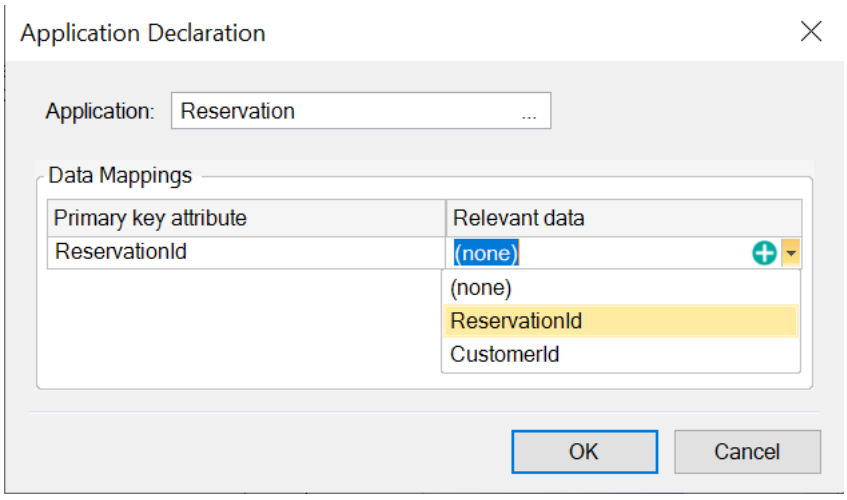
We select the task TicketReservation and go to its properties. We click on the Web Object property and press the button with three dots.

In the dialog box we click on the button reading "Web Object" and select the Reservation transaction prior to pressing Ok.



We can see that the Data Mapping sector shows that the transaction's primary key is the ReservationId attribute. We then run the mouse over the RelevantData column, where we will see a button and an arrow.

We press the button to the left of the arrow and we will see that ReservationId has been selected.

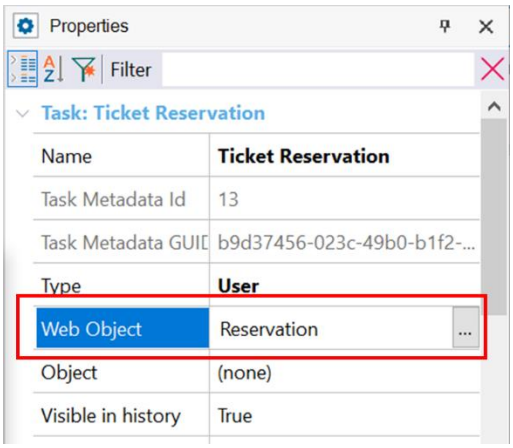


This means that ReservationId will be the name of relevant data in our process.

Relevant data is global data that is known throughout a process. We can compare it to a global variable for it is the mechanism used by Workflow for passing information between tasks.

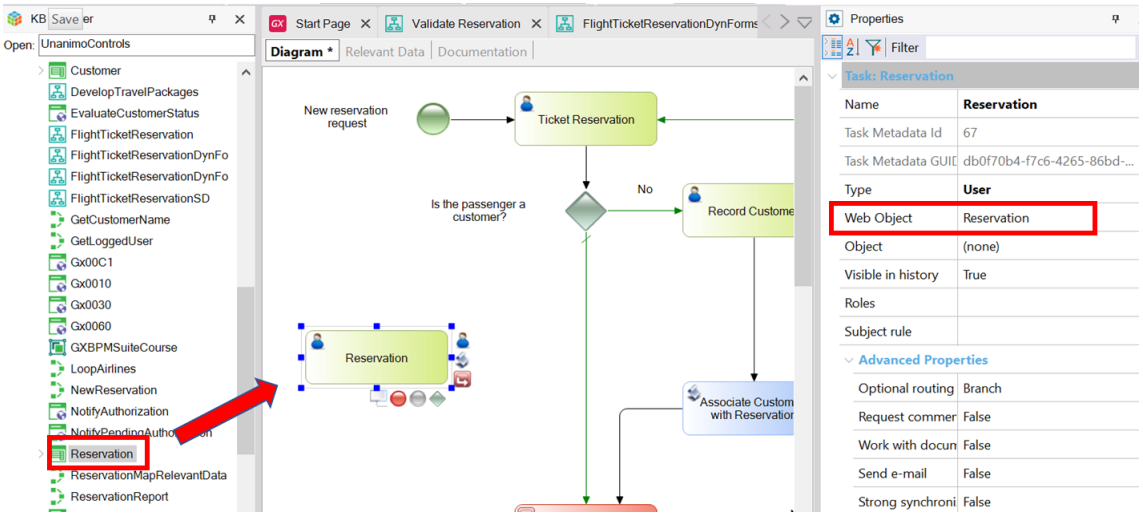
When we associate a transaction to an interactive task GeneXus automatically creates relevant data with the same name as the transaction's identifier attribute.

We press OK, and we will have established a relation between the task and the GeneXus object, in addition to a mapping between the ReservationId attribute and the &ReservationId relevant data.



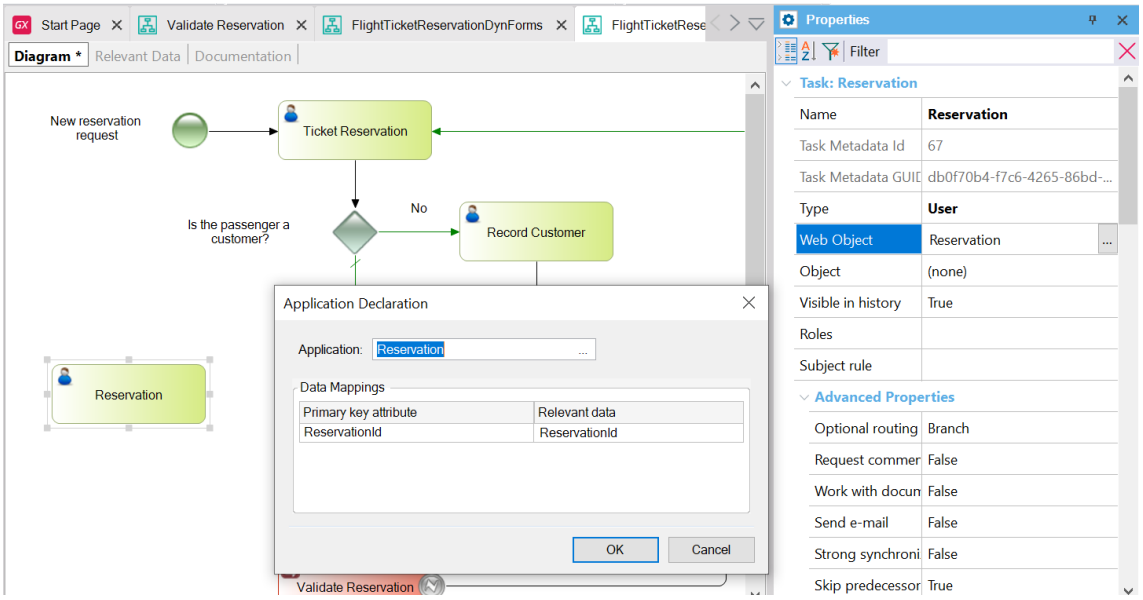
This means that, every time that the TicketReservation task is instanced, the Reservation transaction will be executed because it is the application associated with the task.

In this modeling process we first created the interactive task and then associated it to the application. Instead of creating the task, we could have dragged the diagram from the View Folder directly to the Reservation transaction.



When we do this we can see that an interactive task is automatically created with the same name as the transaction, and its Web Object property reads: Reservation, because the GeneXus object was automatically associated with the task.

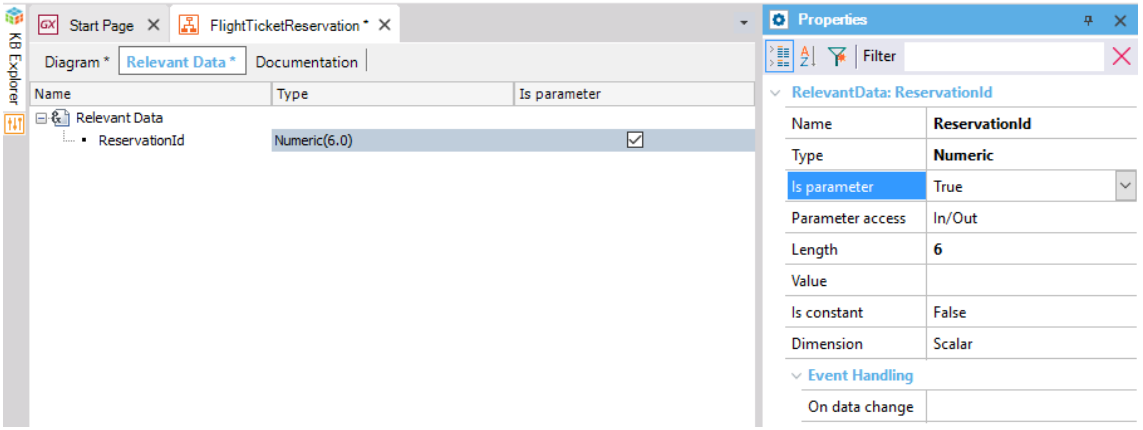
If we press the button with the three dots we will also see that the relevant data was created and that the primary key attribute in the transaction was automatically associated with it.



This relevant data will have the same name and type as the transaction's primary key.

If the primary key includes more than one attribute then relevant data will be created for each attribute that is part of the primary key.

In the Relevant Data tab of the FlightTicketReservation process we can see the relevant data defined.

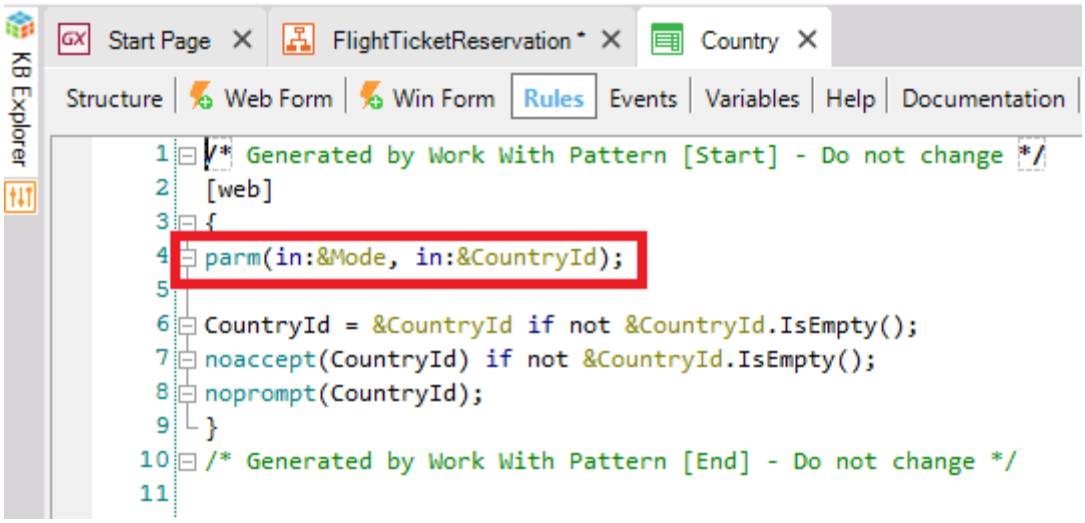


This data appears under the ampersand symbol to indicate that it consists of variables.

When we refer to the relevant data ReservationId we always do it by using the ampersand symbol as prefix in order to set it apart from the ReservationId attribute, as well as in the case of variables in GeneXus objects.

One more thing to bear in mind when we drag a transaction object to the diagram is whether is has a Parm rule defined where the value of the primary key is received in a variable.

Here we have the Country transaction to which the Work With pattern was applied and consequently a Parm rule was added to receive the &CountryId variable as parameter.



The dragging of the transaction to the diagram creates the relevant data, but if the names of variables in the Parm rule match those of the attributes in the primary key then there will be a mapping between the relevant data and the variables, so that **when the transaction is executed, the variables take the values stored in the relevant data at the time.**

In the next video, Automation Part 2, we will continue to associate GeneXus objects with the tasks in our ticket reservation diagram.