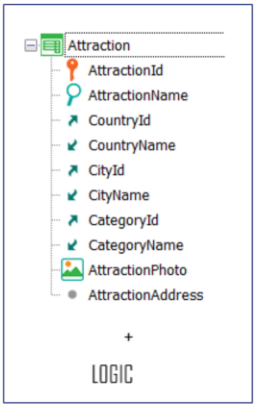
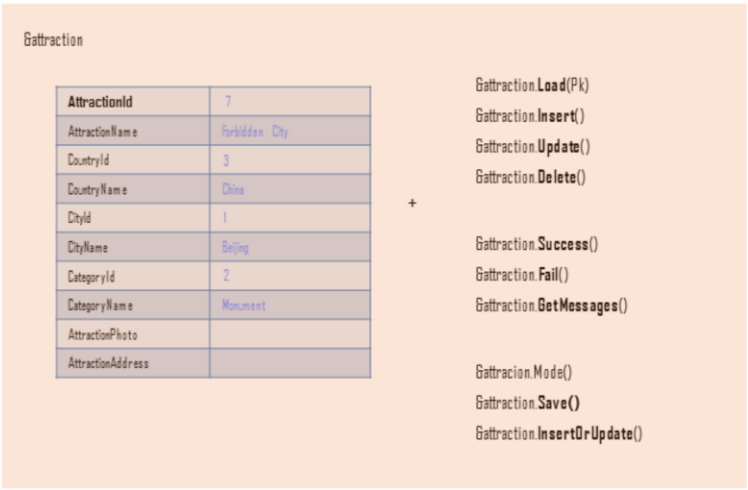
Database Update

Using Business Components - Example

GeneXus™

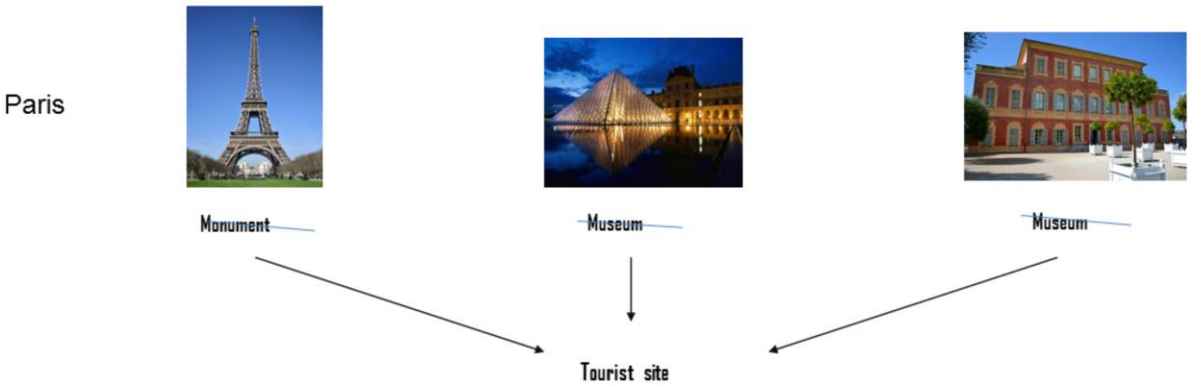
Business Component





In the previous video we saw how to insert, modify and delete a record from a business component corresponding to a single-level transaction.

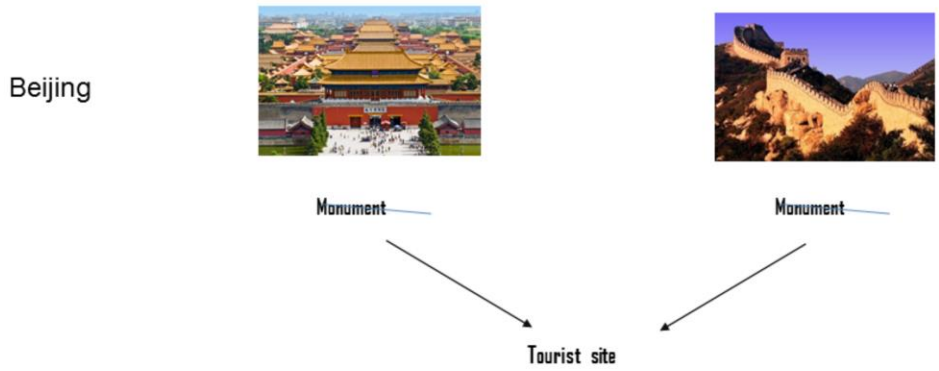
New back-office requirement



Now we will find it easy to implement the following requirement for the travel agency's back office:

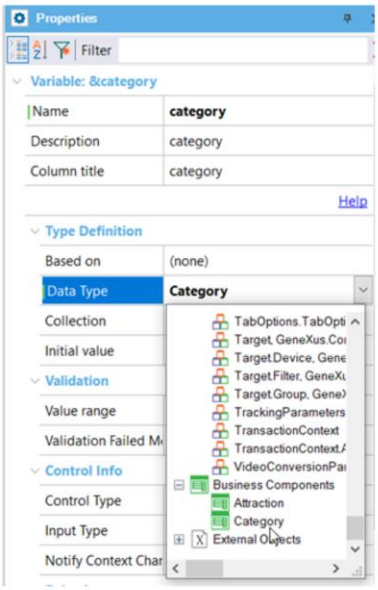
Suppose that at some point the need arises to create a special category, "Tourist site," which groups together very popular tourist attractions in a city, regardless of whether they were previously categorized as monuments, museums, etc.

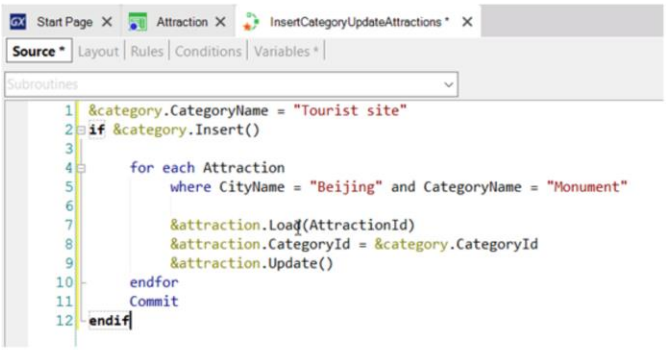
New back-office requirement



Let's start with something a bit simpler: suppose that the agency needs to change all the attractions in Beijing that used to be of Monument type, so now they are of this new category. We are asked to do it by code, so as not to do it interactively, one by one through the Work With screen (here we only see two attractions in Beijing, but there could be many more and the user could miss some of them).

Implementation



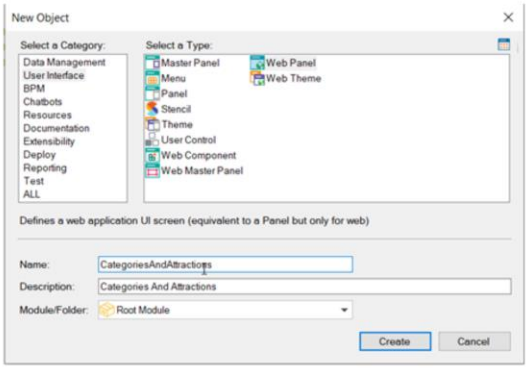


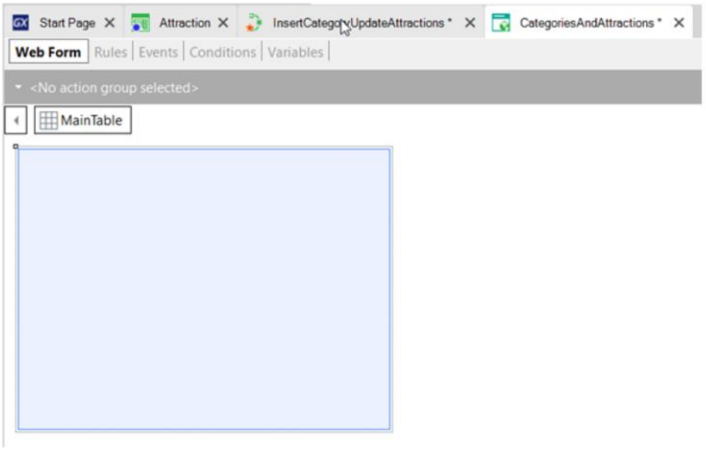
We'll use the procedure we saw. We delete everything we've coded to start over.

Since we will have to enter a new category, we must create its Business Component. And now define a variable of that data type.
Since the ID is auto-numbered, we don't give it a value, but we do assign it to the CategoryName element and insert it.
If all went well, then the next step will be to re-categorize all of Beijing's tourist attractions that have so far been monuments.
Then we must run through those attractions with a For Each command, where the city is Beijing and the category is Monument.
In each iteration of the For Each command we will be positioned on one of those attractions, and what we do is to load it in the variable &attraction, change the category for the new one (note that here we are retrieving the value given by the database when the category was inserted; the ID is auto-numbered).
Finally, we change the record in the database, triggering all the transaction rules, as we saw, and write a Commit.

We will invoke this procedure from a screen, so we remove the outputfile rule and leave the Call protocol and Main program properties with their default values.

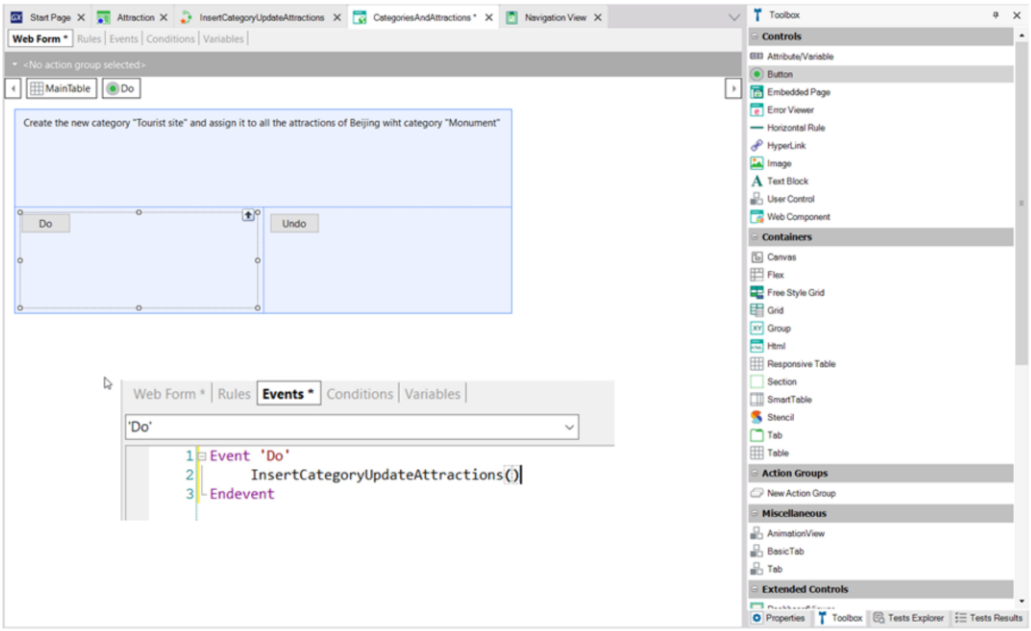
Web panel





We're going to create that screen, that panel. It will be an object, of which we will speak later, of the Web panel type.
We'll call it that...
Here we see its web form, which is empty. This is what users will see on their screen at runtime in their browser.

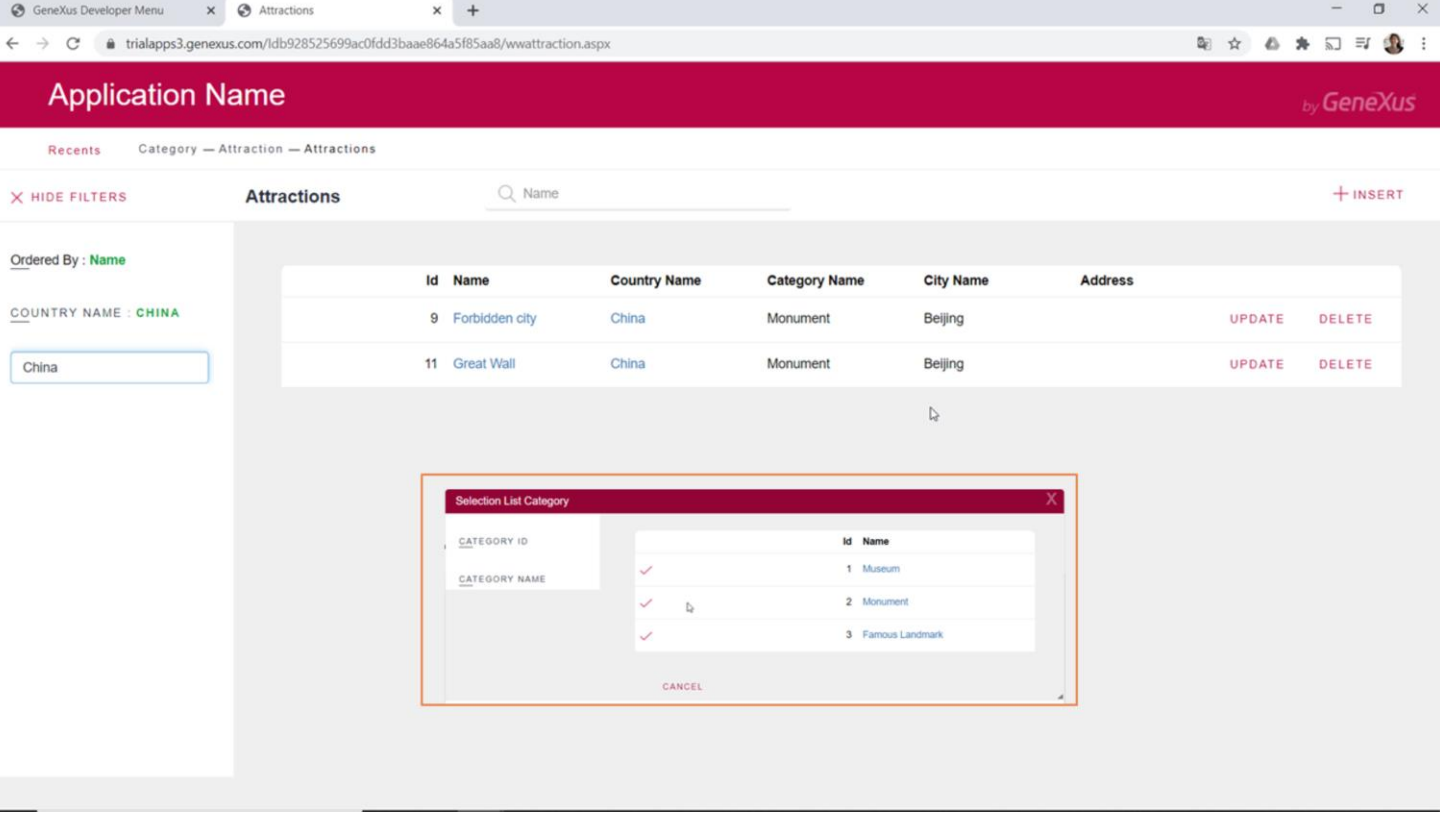
Web panel



From the Toolbox, we drag a text block type control, to show the user a message, a text, in the panel. We include this caption, indicating that what it is going to do is to insert the category "Tourist site" and change Beijing's attractions. And we drag a button, which will have an associated event of this name Do. And we're going to add another button to undo the action.

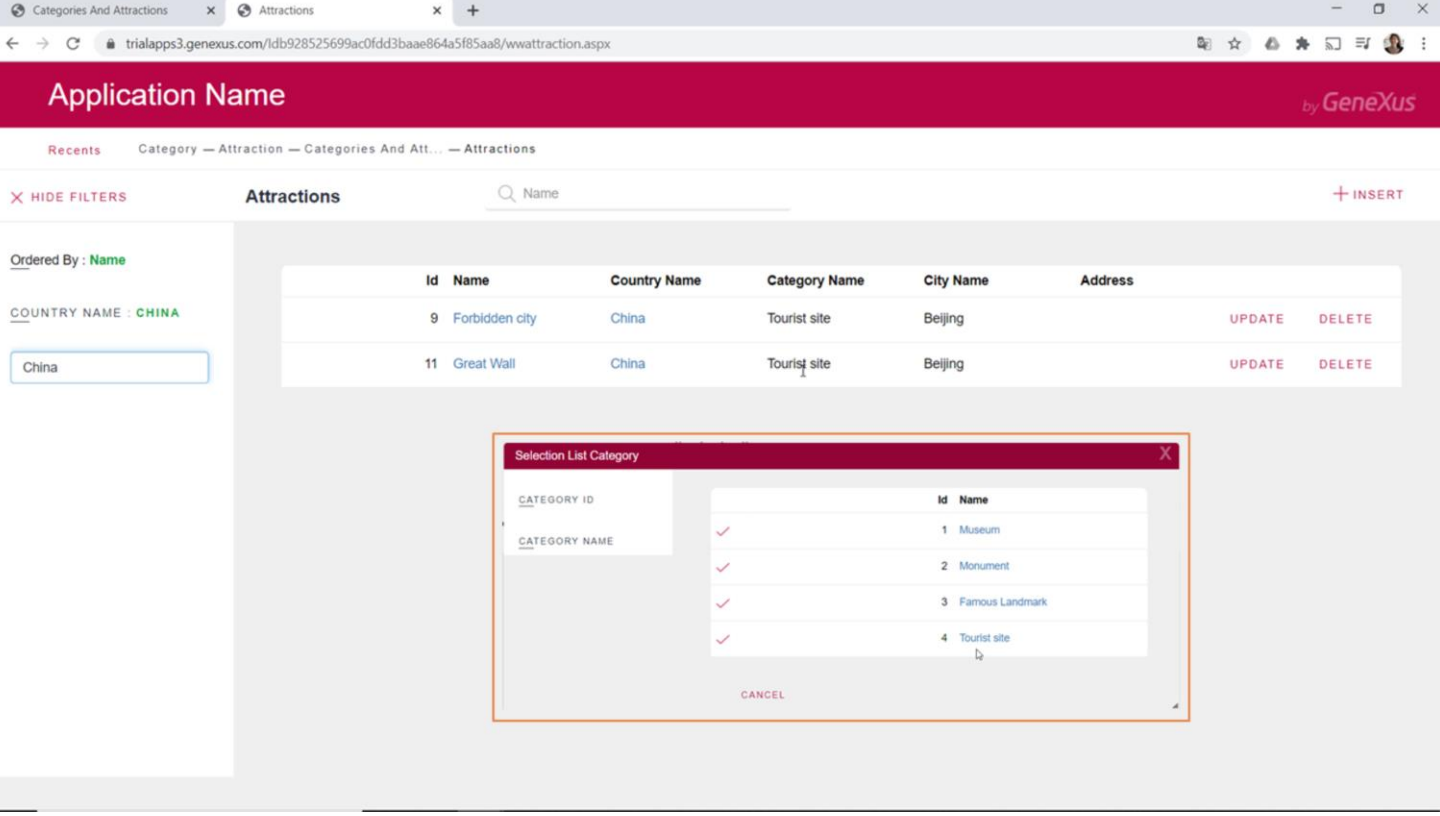
We want to program the code that will be executed when the user clicks on Do. Then we see that the web panel has an events section which is where we're going to specify that code. Here we will invoke the procedure that we had created. We can type it directly, or better yet, to avoid errors, drag it from the KB Explorer. And add the parentheses.

Let's give it a try. F5.

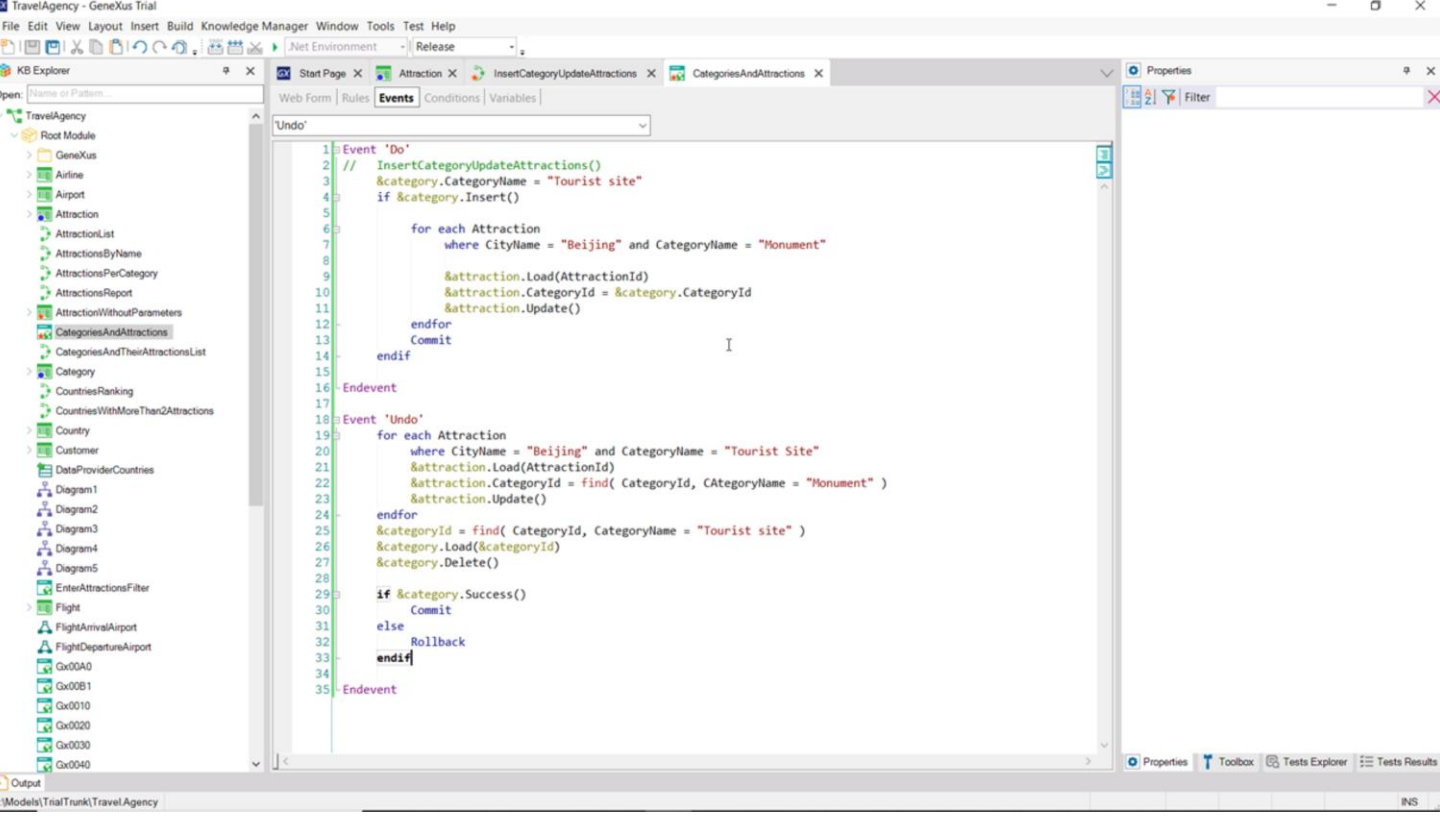


Let's look at the categories we have entered so far in the database. They are these three.
Now let's look at the tourist attractions. We are going to re-insert the Chinese wall, which we had deleted in the previous video.

If we filter by country China, we see that we have here the two attractions of Beijing, which are of Monument type. Therefore, they must be replaced with the new Tourist Site category when we execute the event code of the Web panel.



We refresh the screen, and see that the category has been effectively changed. It was previously created in the table.



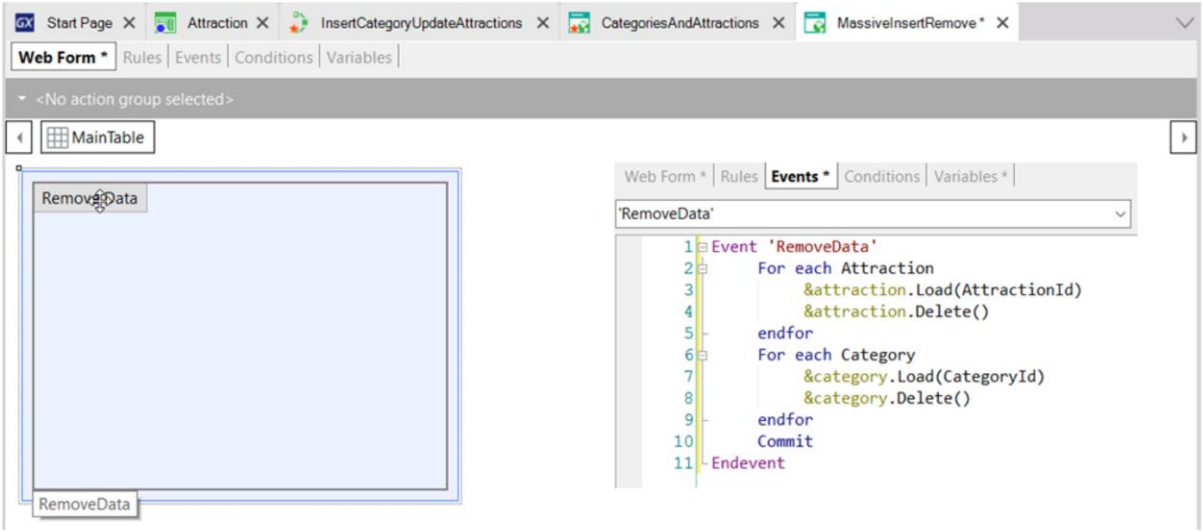
As we mentioned in the previous video, updates by means of business components can be made in almost any object that accepts code. Thus, we could have written directly here the code we had in the procedure, without the need to invoke any other object. We define the variables, simply category and attraction, business components, and that's it.

We are going to program the code associated with the Undo event, which is going to undo the previous operations. That is to say, delete the Tourist Site category, and at the same time restore the Beijing attractions that were now Tourist Sites to the Monument category.

The code will look similar to this. We are running through Beijing's attractions in the Tourist Site category with a For Each command. We are loading the Attraction variable, business component, with each attraction found, and changing its category ID to Monument. We update.
And delete the category. We define a variable, CategoryId, to retrieve the identifier of the Tourist site category, in order to load that category into the business component and remove it.
If everything goes well, then we make a Commit, and otherwise a Rollback.

Why do we define this order, that is to say, to run through the attractions first and then eliminate the category, and not the other way around? What if we wanted to remove the category first and then change that same category of tourist attractions? Obviously, it won't be allowed. Referential integrity will fail, because we will be trying to remove a category that has associated records. So the order can't be reversed.

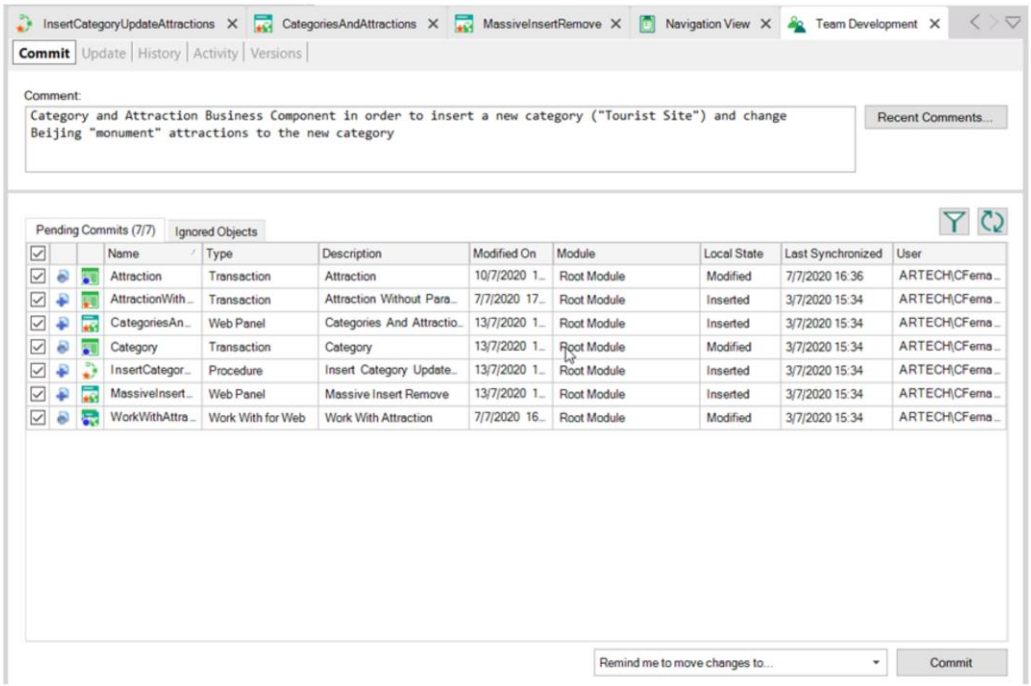
Let's try it. We click on Undo. Note that the category has been effectively deleted. And if we go to the attractions and filter through China, we see the Monument category has been restored for both attractions.



Now let's create another panel to remove all the information from the category table and the entire table of tourist attractions.

Again, the web form is empty. We drag a button, associate the RemoveData event to it and code it this way. And once again the order has to be like this. We couldn't just run through the categories and eliminate them, and then go on to eliminate the attractions. Referential integrity would fail.

Let's try it. We click on Remove Data. And now if we open the information in the Category transaction we see that it is empty. The same happens if we go to see the attractions.



Finally, we send the changes made to the server.

GeneXus™

training.genexus.com

wiki.genexus.com

training.genexus.com/certifications