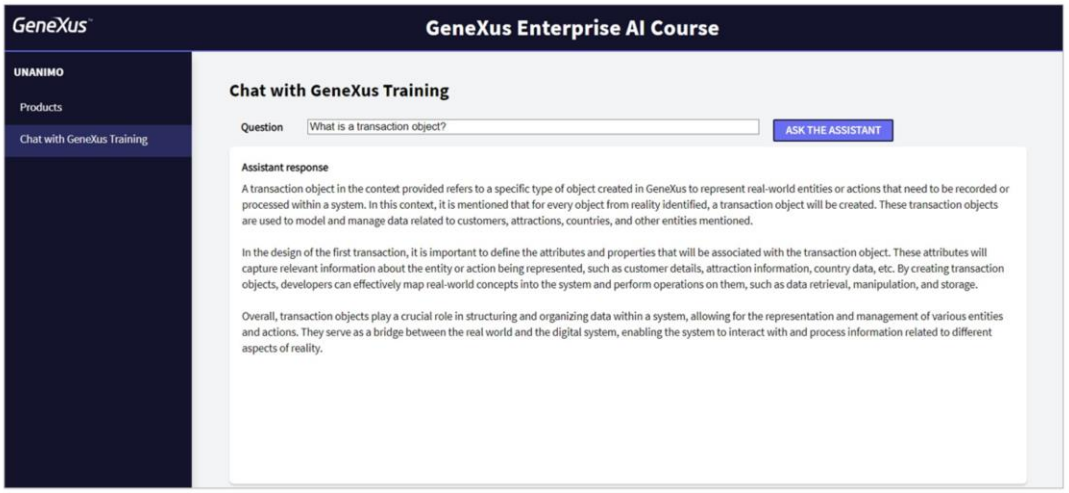
GeneXus by Globant

GeneXus application that interacts with a RAG assistant

Alejandra Caggiano

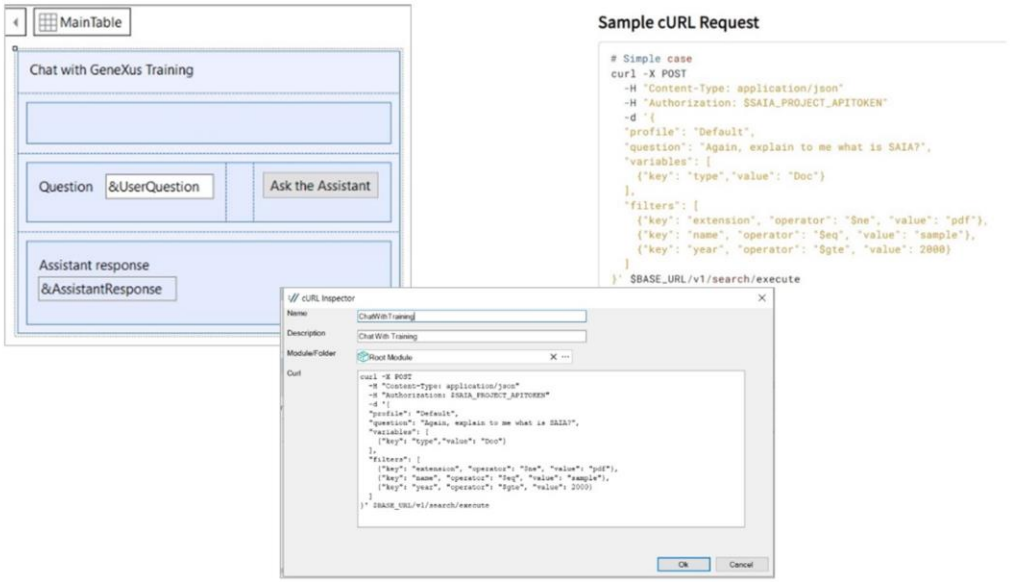
We have previously seen how to interact with a Chat assistant from a GeneXus knowledge base.

GeneXus application that interacts with a RAG assistant



We are now going to extend that same knowledge base so that it can also interact with our RAG Assistant named ChatWithGXTraining. This way we will enable the end user to ask questions about this content.

GeneXus application that interacts with a RAG assistant



If we go to GeneXus, we have already created the web panel named ChatGXTraining where the end user can enter a query and ask the assistant for the answer. We have two variables:

The &UserQuestion variable, on which the user will submit their query, and the &AssistantResponse variable that will be in charge of displaying the answer received.

Our objective then is to code the event associated with the button so that when it is pressed a procedure is called to send the query and receive the response.

To create this procedure that has the base of the request structure, we are going to load the cURL sample through the menu Tools / Application Integration / cURL Inspector.

We name it ChatWithTraining and paste the sample, available in the GeneXus Enterprise AI technical documentation.

https://wiki.genexus.com/enterprise-ai/wiki?8,Table+of+contents%3AEnterprise+AI,

In this case, we need to interact with the API that allows us to chat with documents, so the sample that we paste corresponds to the cURL Request.

GeneXus application that interacts with a RAG assistant

We click on OK, and we see the procedure created with the base structure we need. The first thing we indicate is the connection protocol. And for that, as we know, we declare the Secure method with the value 1 that corresponds to the HTTPS protocol.

Then we indicate the Host corresponding to the content of the &BASE_URL variable, which in turn depends on the GeneXus Enterprise AI installation.

Good. Next, we need to indicate the Project Api Token, which as we already know, we copy it from the platform and paste it.

Let's go to the body of the query.

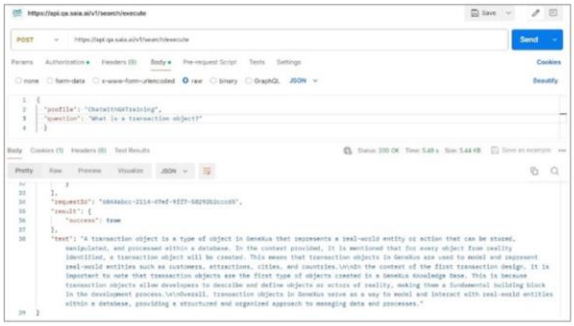
First, we indicate the “profile”, that is, the name of the assistant, which in this case is ChatWithGXTraining.

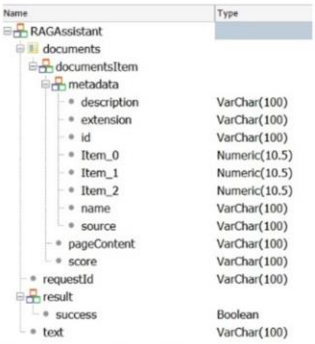
As we discussed in the previous example, this procedure should receive the query from the end user to send it to the assistant and get the answer.

So we declare the corresponding Parm rule, where the input parameter is the user's question, and the output parameter, or return value, is the assistant's answer.

We may also set filters for the query, but we won't do it in this case. Next, we start running the POST, filling in the path.

GeneXus application that interacts with a RAG assistant






```
//curl -X POST  -H "Content-Type: application/json"  -H "Authorization: $SAIA_PROJECT_APITOKEN"  -d '{  "profile": "Defaul

&HttpClient.Secure = 1
&HttpClient.Host = "api.qa.saia.ai"

&HttpClient.AddHeader(!"Authorization", !"Bearer default_OuK5BwzqSLjNEHwUf-GV0QCLI2YHYxBBGAshAg1CiLSa9lFgeZKcQr5S0xsehbjbg
&HttpClient.AddHeader(!"Content-Type", !"application/json")

&HttpClient.AddString(!'{ "profile": "ChatWithGXTraining", "question": "' + &UserQuestion.Trim() + '" }')

&HttpClient.Execute(!"POST", "/v1/search/execute")

&RAGAssistant.FromJson(&HttpClient.ToString())

&AssistantResponse = &RAGAssistant.text
```

This is how we send the query. Now we must receive the assistant's response.

When we remember the output we saw from Postman when we tested this assistant via API, we can see that this output corresponds to a structure, and the response of the assistant is in the “text” item.

So we save it as a JSON file, and then import it into the KB and automatically generate the structured data type we need to receive the response.

We go back to the KB and import it. Next, we go to Tools / Application Integration / Json import, enter the name RAGAssistant and load the file.

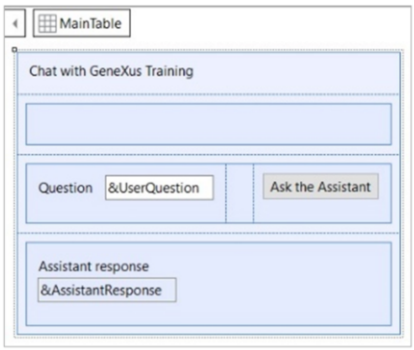
Note that the structured data type has been created. We open it, and we see the “text” item, where we will receive the assistant's response.

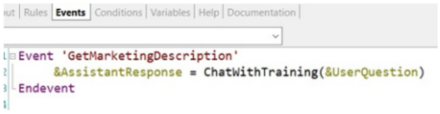
What should we do then? We must create a variable based on this data type and load the assistant's response there.

In the procedure, we then define the variable &RAGAssistant and load the response from JSON. We already have the complete answer, but what we need is the content of the “text" item. So we load our return variable, &AssistantResponse, with that content.

Good. We have completed the procedure.

GeneXus application that interacts with a RAG assistant





Now we must complete its call from the web panel. For that, we go to the event associated with the button and call the procedure, sending as parameter the variable &UserQuestion with the user's question, and receiving its answer in the variable &AssistantResponse, both present in the form.

To test it, we press F5, and run the web panel.

GeneXus application that interacts with a RAG assistant

Remember that our RAG assistant was fed with only 18 documents, which is not a lot of information, and for that reason we will ask simple questions whose answers are in the scope of those documents. This will allow us to test the application.

First, let's ask: What is a GeneXus knowledge base? Let's see the answer.

Good. Let's ask now: What is a transaction object?

Very good. And if we ask: what is a formula attribute in GeneXus?

Perfect. The answers received were thorough and correct. We consider it valid.

Now let's look at the menu. We have access to the chat assistant and the assistant to chat with documents.

In this way, we have achieved a GeneXus application that interacts with two artificial intelligence assistants created through GeneXus Enterprise AI.

GeneXus by Globant

GeneXus™
by Globant

training.genexus.com