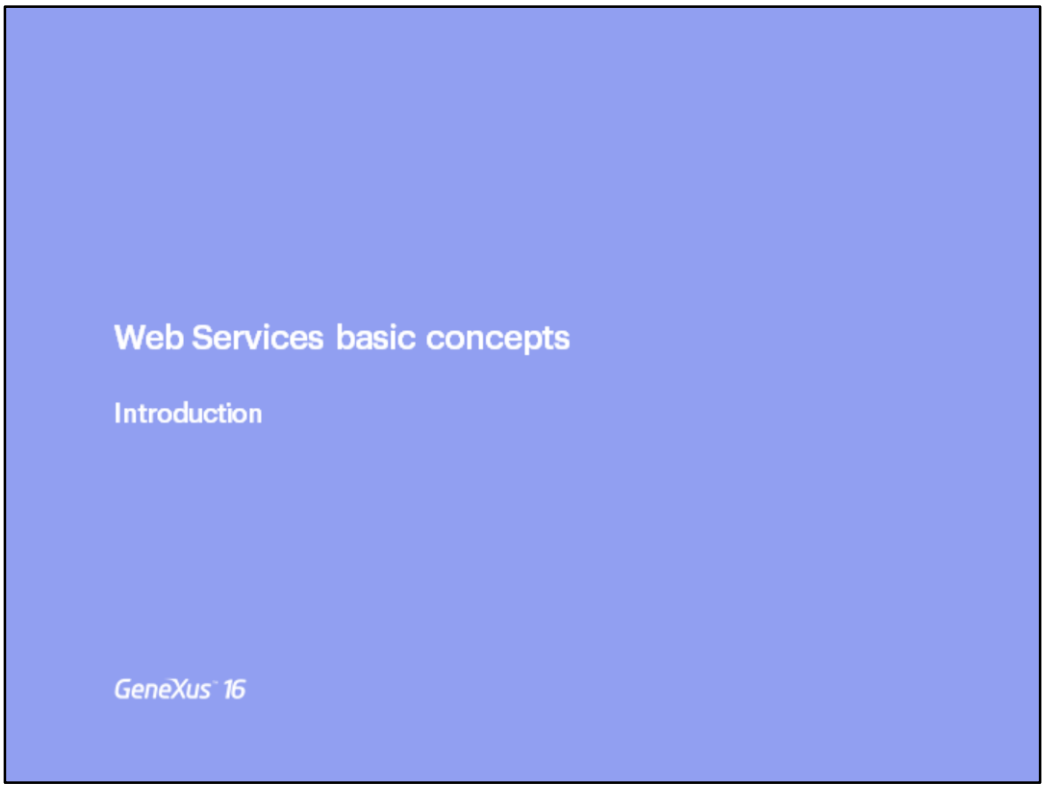
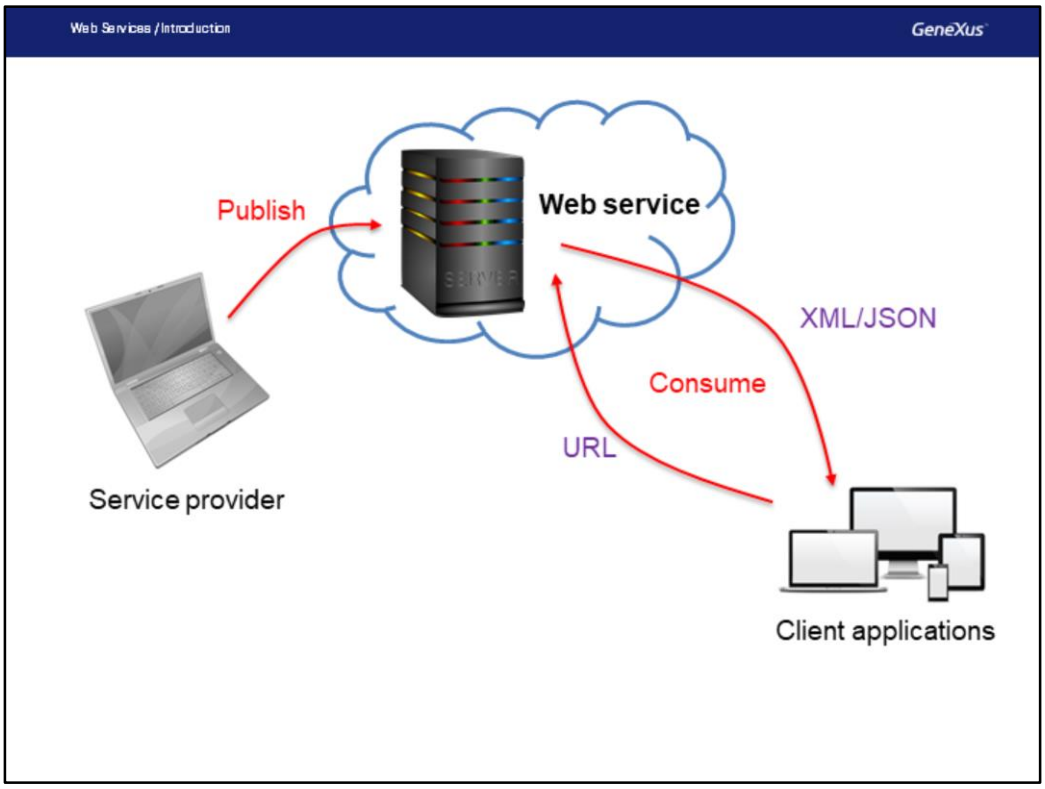


Next, we'll see what web services are and how they can be used in a GeneXus application.

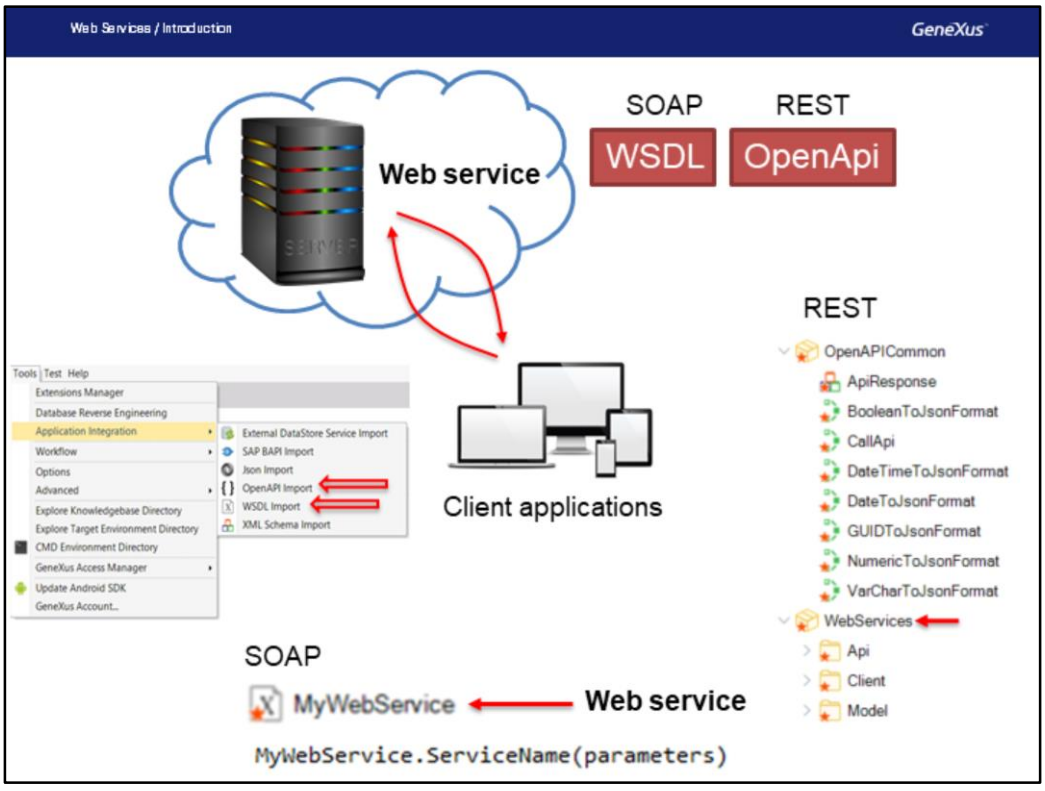


Web Services are programs that provide useful functionalities to other programs and are stored in web servers so that they can be located and invoked over a network, usually the Internet.

The service provider "publishes" a Web Service on a server and the client applications "consume" the Web Service published.

To access the service, the client application uses its location (URL) to invoke it and then sends it the required parameters.

In return, it receives the returned information, usually as a structure in XML or JSON format.



Web Services can be developed following different standards; the most common in the industry are SOAP and REST. Each standard defines how to publish the information about the functions available in the web service.

SOAP web services use a definition written in WSDL (Web Services Description Language), while REST services use the OpenAPI standard.

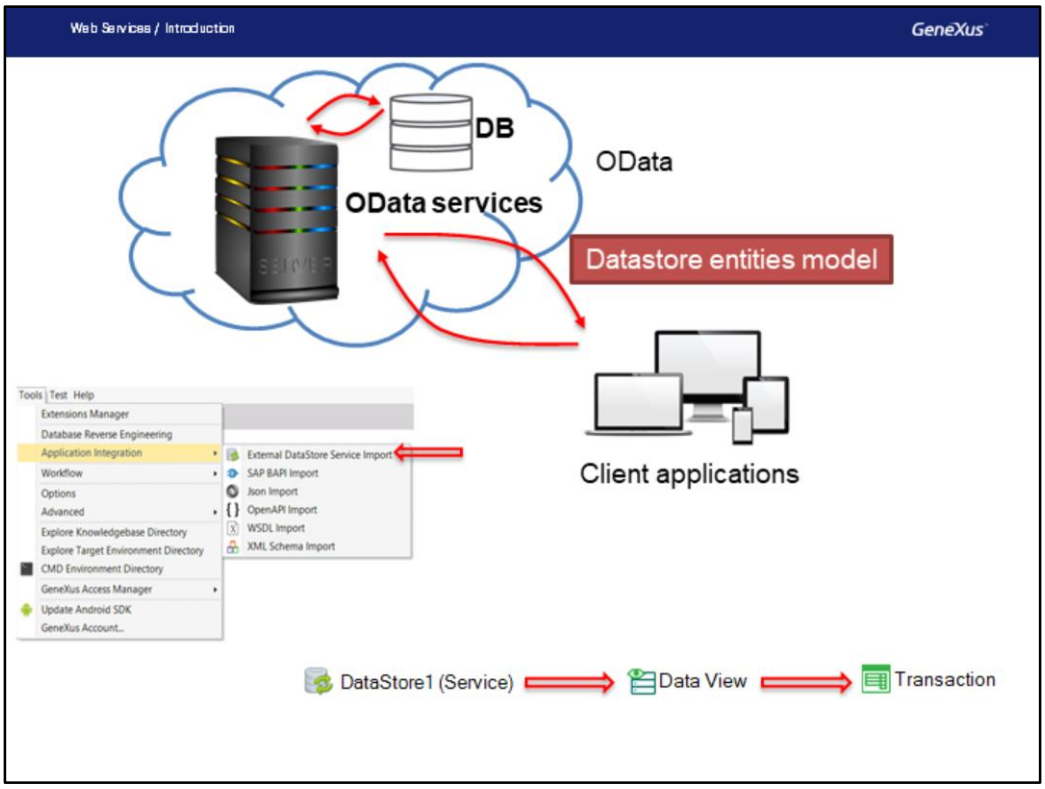
To access a published Web Service, we must know its location (URL) and import its definition in order to access the functions available in the web service.

GeneXus allows consuming Web Services that have been developed in any programming tool or platform, with SOAP or REST protocols.

To integrate a Web Service into a GeneXus application, go to Tools/Application Integration; select WSDL Import if the Web Service follows the SOAP protocol, and OpenAPI Import if the Web Service has a REST architecture.

This will trigger a wizard that will vary depending on the type of Web Service selected.

Finally, if the Web service is SOAP, GeneXus will automatically create an External Object associated with the Web Service and the types of structured data required for managing its data. Otherwise, if it was REST, several GeneXus objects will be automatically created (and we will usually include them in a module, for example, WebServices). Also, they will allow us to automatically run the service through these objects in our KB. The wizard will leave in an API folder the programs to invoke, and in a Model folder the SDTs to manage the data.

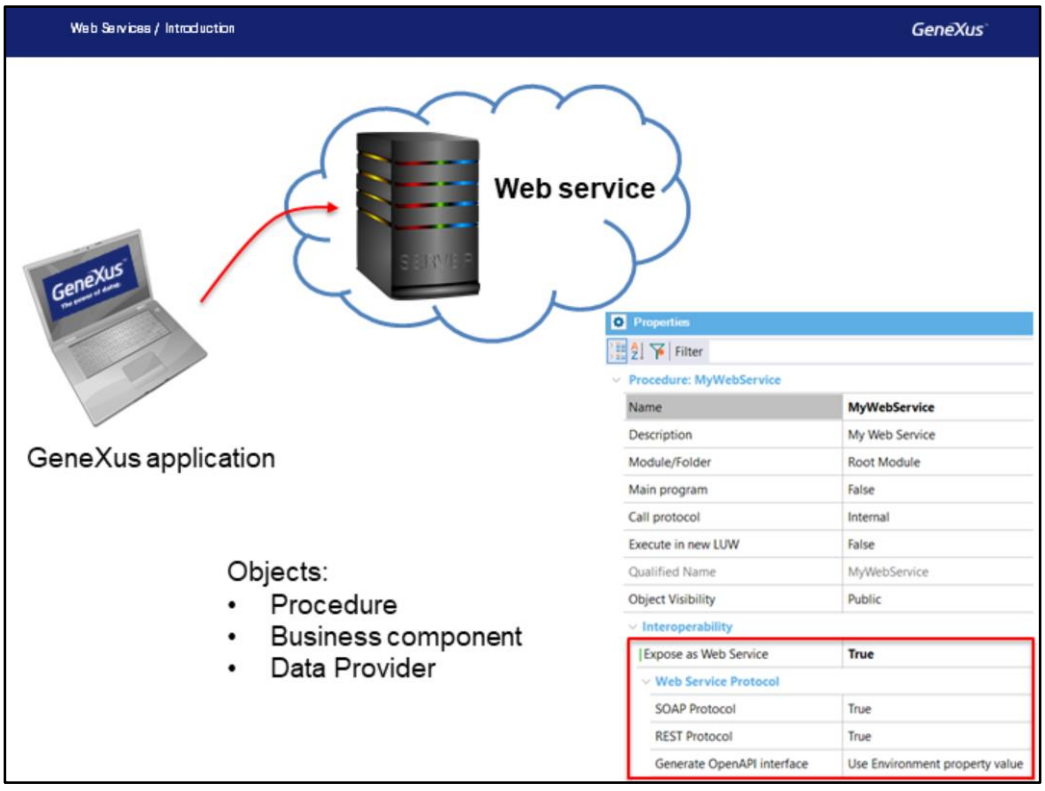


A special type of web services is called OData services. These services use the Open Data Protocol (Odata) designed to provide operations that insert, modify, or delete records in a database through a website.

Just as for a Soap or Rest Web Service its definition had to be imported first, in order to consume an Odata service we must first import the model with the database entities included in the service.

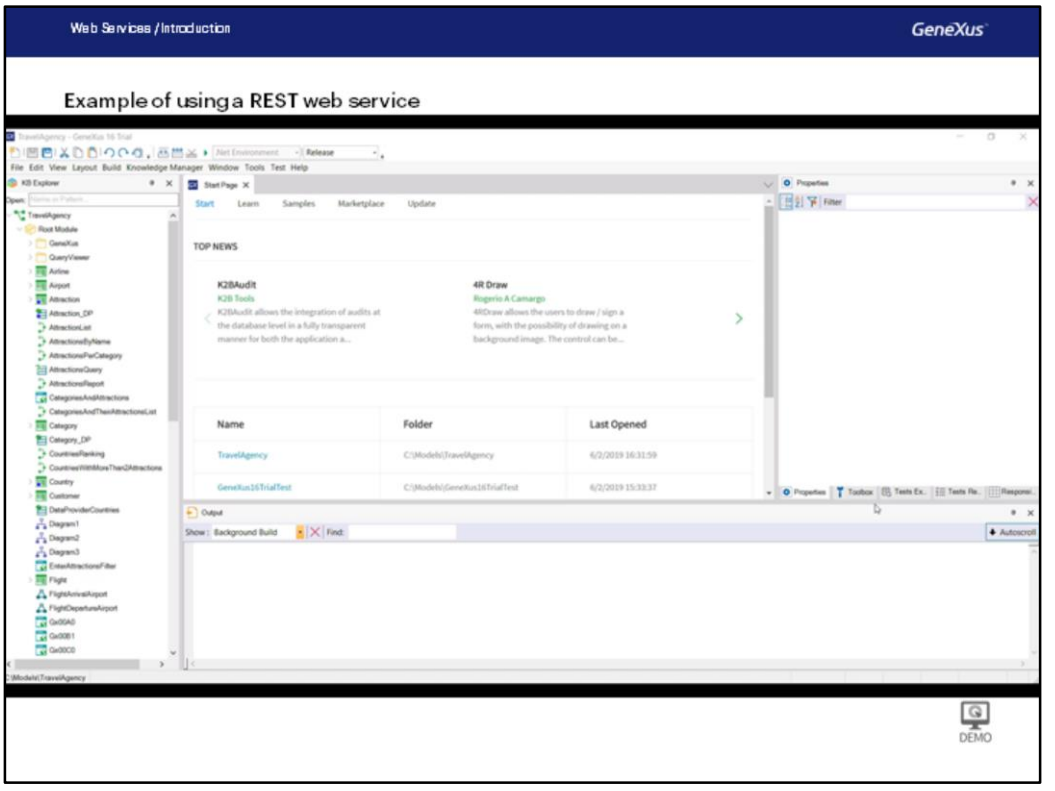
To do so, go to Tools/Application Integration and select External DataStore Service Import. Since the process implies creating a DataStore of Service type to associate it with the external datastore, we can only use this feature in GeneXus Full.

After executing the wizard, as many transaction objects will be created as entities were included in the model, and we will be able to work with them as with any other transaction in our application.



GeneXus also allows creating web services and publishing them on a web server.

The GeneXus objects that can be exposed as web services are procedures, business components and data providers.



[DEMO: https://youtu.be/bvmt0Gjcxpw]

As an example, we will create a REST Web Service that inserts a new airline in the database and we will invoke it from our application.

To do so, first we open the Airline transaction and set its Business Component property to True.
Then we are going to create a procedure object that receives by parameter the data of an airline and inserts a new record in the Airline table.

We call the procedure CreateNewAirlineWS and create a Parm rule with 2 input variables: &AirlineName and &AirlineDiscount. We create these two variables using Add Variable, and then in the Variables section we create another variable called Airline, of Airline business component type.

In the source, we write the code to insert an airline with the data received by parameter. We don't have to enter a value for AirlineId because it is autonumbered.

Now, we open the properties of the procedure object and set **Expose as Web Service** to True, **SOAP Protocol** to False; we **leave REST Protocol** set to True and set **Generate OpenAPI interface** to Yes. The latter will allow the definition of our Web Service to be generated in the OpenAPI standard.
We right-click on the procedure and select Build With This Only, so that the service will be published on our web server, in this case on our local machine. In the Output window we see a message confirming that the documentation of the Rest API of the Web Service was generated, with its definition.

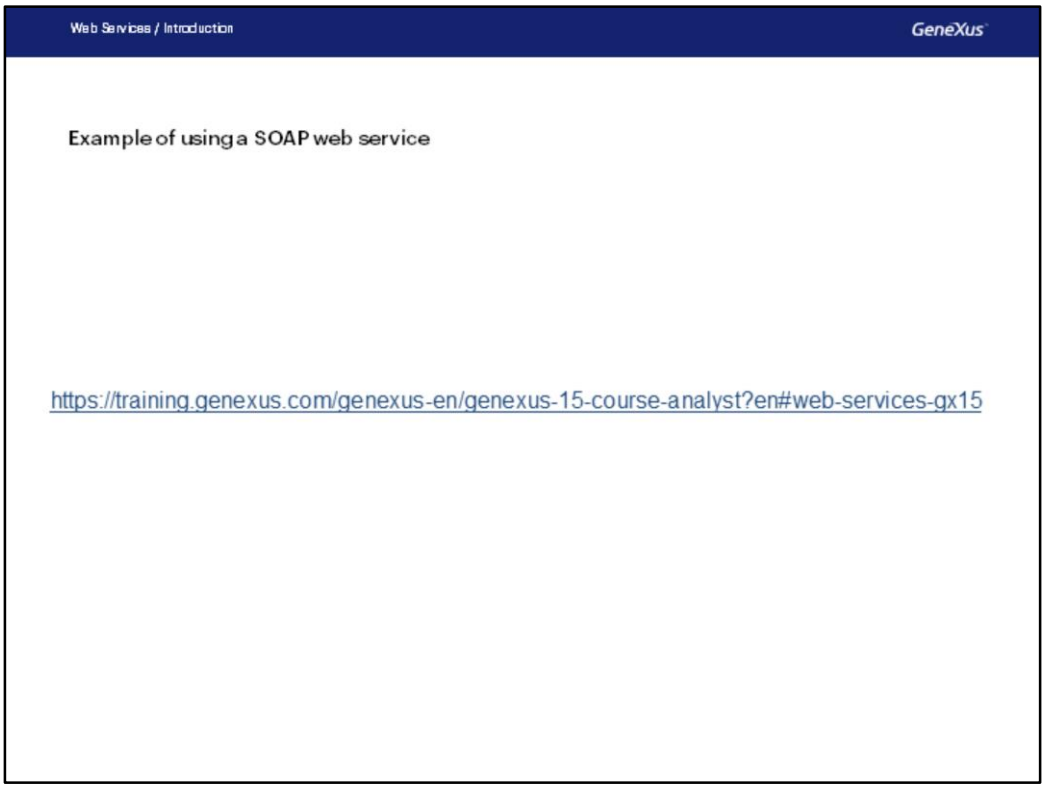
Before importing the Web Service, we'll create a new module called WebServices, so everything is stored in that module.
To import the Web Service, we go to Tools/Application Integration and select OpenAPI Import. In the File Path / URL we type: C:\Models\TravelAgency\CSharpModel\Web\default.yaml because it is where the

Rest API documentation was generated, and select the WebServices module as target. We confirm that everything has been imported correctly; if we open the WebServices module, the imported procedure is located in the API folder. Also, we see that in the Model folder there is an SDT called CreateNewAirlineWSInput, and if we open it we see that the parameters we need to pass to the service are available here.

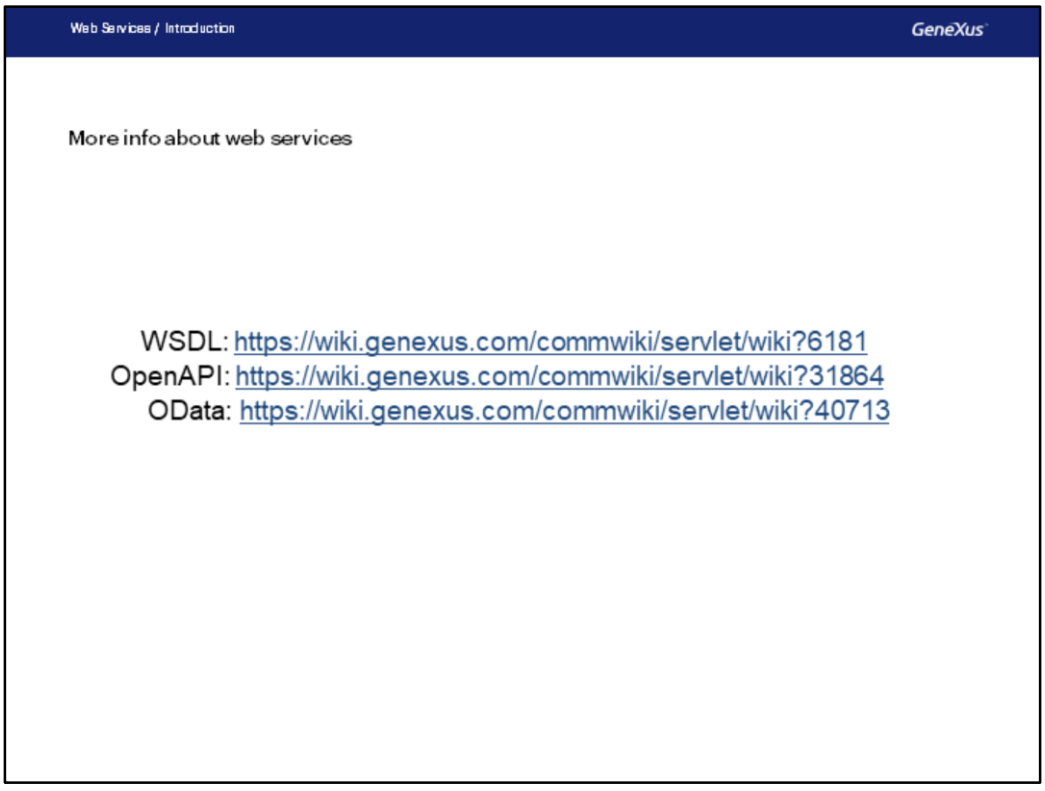
To test whether the Web Service works correctly, we created a web panel called CreateNewAirlineUsingWS. Next, we drag a button to the form and call the event Create airline. We open the variables and create a variable &CreateNewAirlineWSInput which is automatically created as of SDT type.
In order to receive feedback about the result of the Web Service execution, we also created a variable &IsSuccess and another one &HttpMessage. Note that the types are assigned by default.

We double-click on the button and in the event we load the members of the SDT; next, we invoke the Web Service passing the SDT as a parameter, and finally we write the following code to show messages on screen.

We press F5… and run the web panel we've just created.
We press the button and see that the service informs us that the airline was created correctly.
To check it we select the Airline transaction… And see that the airline we wanted has been actually added.



Here we saw an example of using a web service of REST type. To see how to use a SOAP service, you can watch the following video:
https://training.genexus.com/genexus/curso-genexus-15-analista?es#web-services-gx15



For more information about web services in GeneXus, read the following wiki links:

WSDL: https://wiki.genexus.com/commwiki/servlet/wiki?6181
OpenAPI: https://wiki.genexus.com/commwiki/servlet/wiki?31864
OData: https://wiki.genexus.com/commwiki/servlet/wiki?40713

GeneXus™

The power of doing.

Videos	training.genexus.com
Documentation	wiki.genexus.com
Certificactions	training.genexus.com/certifications