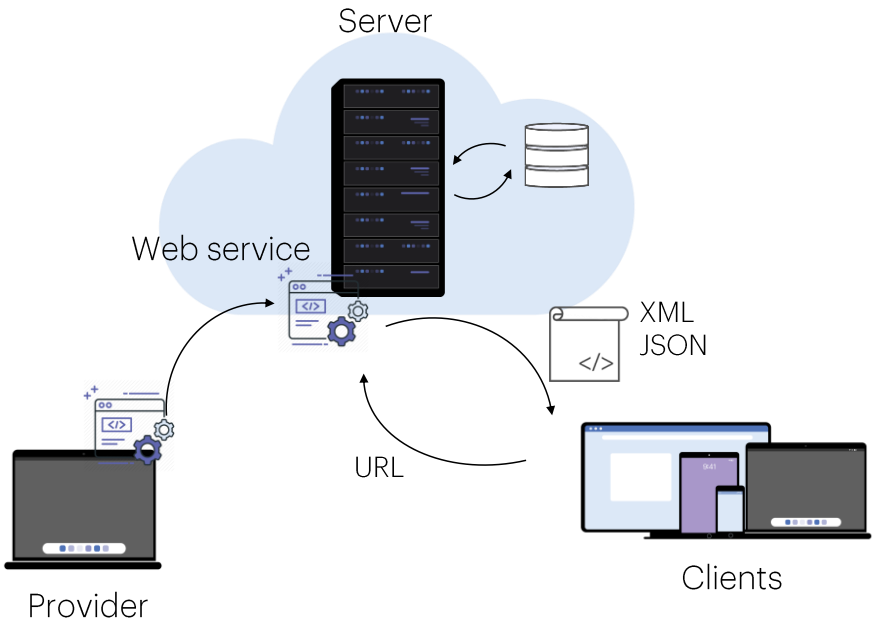
Web Services en GeneXus

Introducción

GeneXus™

A continuación veremos qué son los web services y cómo podemos usar estos servicios en una aplicación GeneXus.

Definición



Los Web Services son programas que brindan funcionalidades útiles a otros programas y son ubicados en servidores web para que puedan ser localizados e invocados a través de una red, generalmente Internet.

Cuando los servicios del backend de nuestra aplicación los publicamos en el servidor web de forma que puedan ser accedidos por otros sistemas, pasan a ser web services. Para facilitar el acceso a estos servicios se utilizan estándares que definen el mecanismo de interacción con los mismos y el formato de la información recibida.

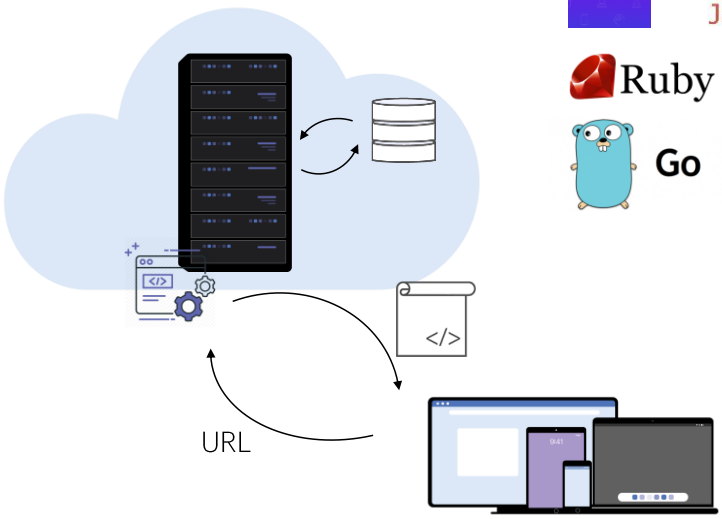
El proveedor de servicios “publica” un Web Service en un servidor y las aplicaciones cliente “consumen” el Web Service publicado.

Para acceder al servicio, la aplicación cliente utiliza su localización (URL) para invocarlo y eventualmente le envía los parámetros requeridos.

A cambio recibe la información devuelta, generalmente como una estructura en formato XML o JSON.

Protocolos de los web services

Protocol / Standard	
SOAP	WSDL
REST	OpenAPI



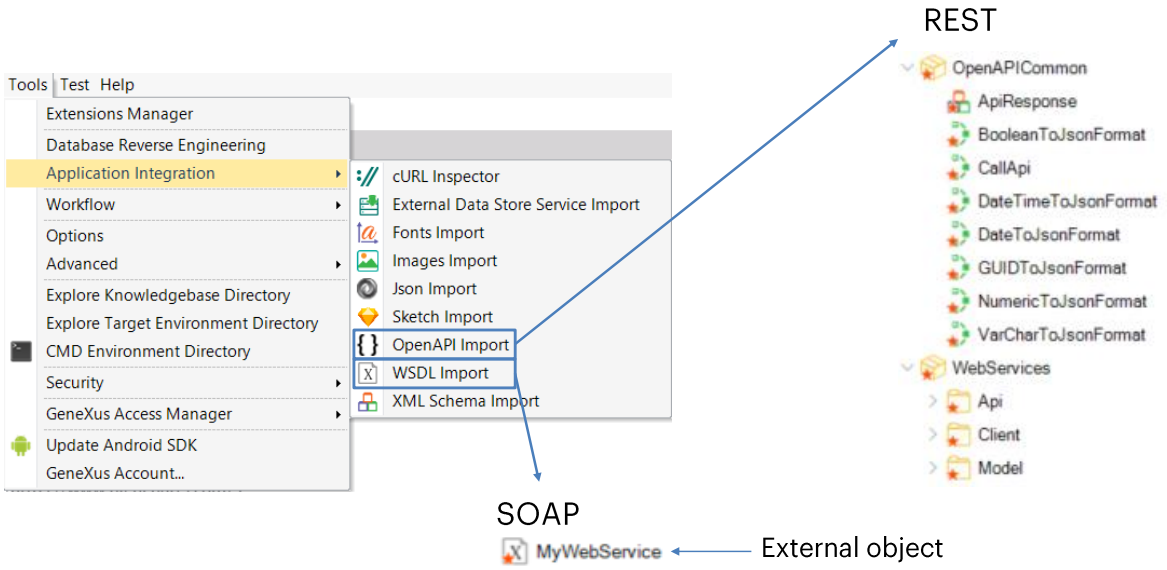
Los Web Services pueden ser desarrollados siguiendo distintos estándares, los más comunes en la industria son SOAP y REST. Cada estándar define cómo debe publicarse la información de las funciones disponibles en el web service.

Los web services SOAP utilizan una definición escrita en WSDL (Web Services Description Language), mientras que los servicios REST utilizan el estándar OpenAPI.

Para acceder a un Web Service publicado, debemos conocer su localización e importar su definición para poder tener acceso a las funciones disponibles en el servicio web.

GeneXus permite consumir Web Services que hayan sido desarrollados en cualquier herramienta de programación o plataforma, con protocolos SOAP o REST.

Cómo consumir un web service en GeneXus

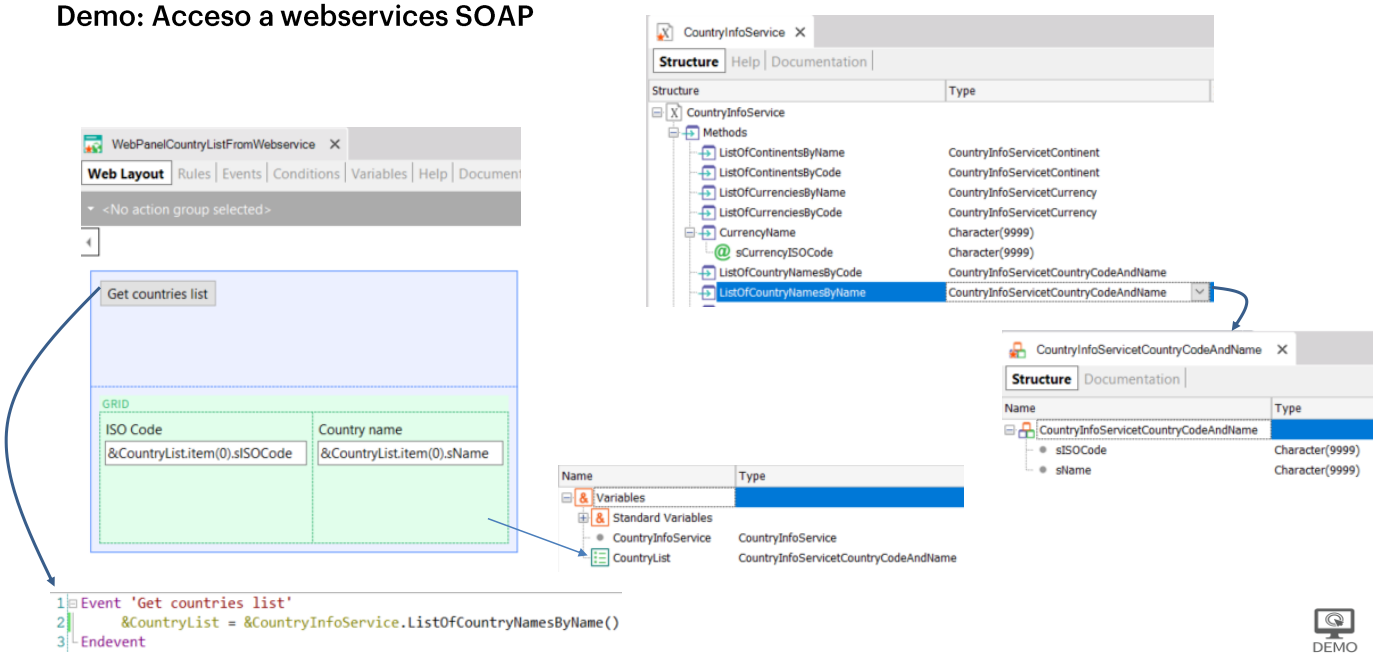


Para integrar un Web service a una aplicación GeneXus, vamos a Tools/Application Integration, elegimos WSDL Import si el web service sigue el protocolo SOAP y OpenAPI Import si el webservice tiene arquitectura REST.

Esto disparará un wizard que variará dependiendo del tipo de servicio web elegido. Al finalizar, si el webservice era SOAP, GeneXus creará automáticamente un Objeto Externo asociado al Web Service y los tipos de datos estructurados necesarios para manejar sus datos.

Si en cambio era REST, se crearán automáticamente una serie de objetos GeneXus, (que por lo general incluiremos en un módulo, por ejemplo WebServices) y que nos permitirán ejecutar el servicio directamente a través de esos objetos en nuestra KB. El wizard dejará en una carpeta API los programas a invocar y en una carpeta Model los SDTs para manejar los datos.

Demo: Acceso a webservices SOAP

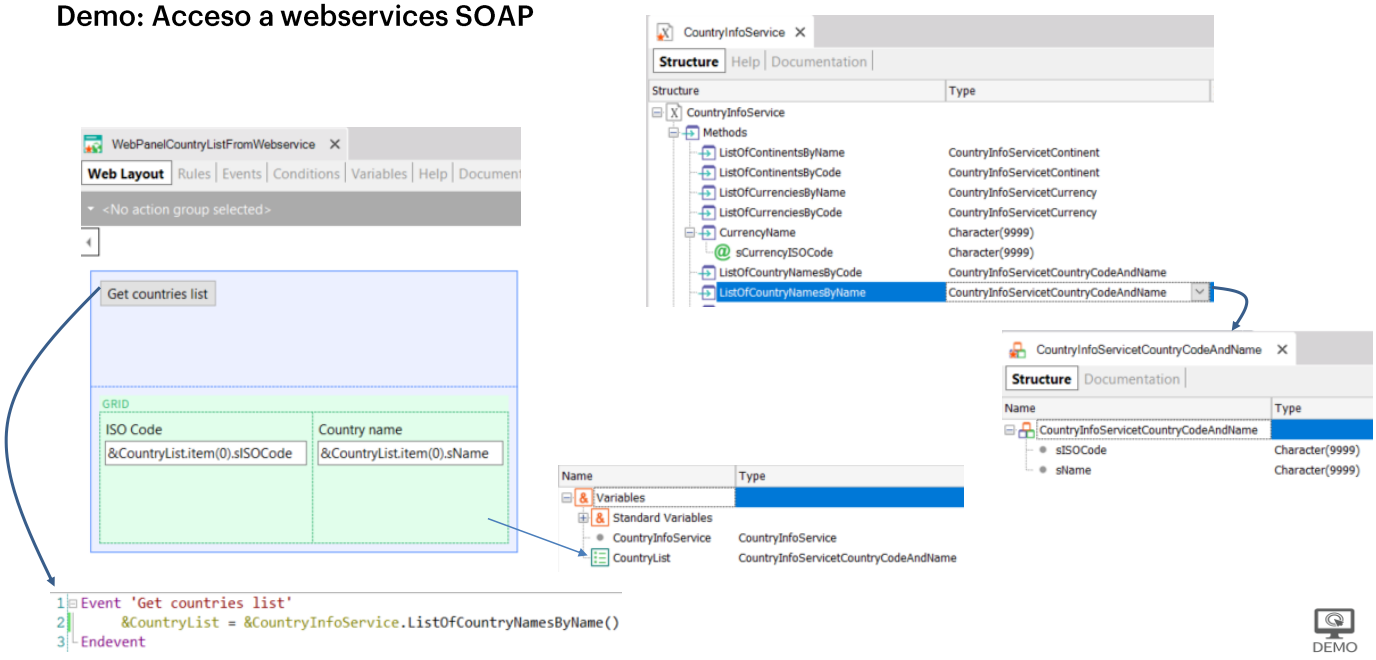


Comencemos importando a nuestra aplicación una lista de países desde un webservice SOAP.
Para eso accedemos al menú Tools/Application Integration/WSDL import y escribimos la url que vemos en pantalla (http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL) , y presionamos Next.
Vemos que se encontró un servicio llamado CountryInfoService. Para ordenar lo que importemos vamos a guardar los objetos en el folder SOAPWebService, dejamos el prefijo sugerido y presionamos Next.
Hacemos clic en el signo de "+" y luego hacemos lo mismo en el nodo Service Description. Observamos que se abre una lista de las funciones que ofrece el webservice, como por ejemplo: ListOfCountryNamesByName para obtener la lista de nombres de países, CountryCurrency para obtener la moneda de un país, CountryFlag para obtener la imagen de su bandera, etc.

Presionamos Import para que GeneXus importe la definición del webservice y vemos que en la ventana de output se nos informa que se están importando una serie de tipos de datos estructurados y otros componentes. Al finalizar, comprobamos que aparece en el KBNavigator la carpeta SOAPWebService y si vemos su contenido, encontramos al objeto externo CountryInfoService y una serie de SDTs que nos permitirán almacenar la información recibida desde cada método del servicio.

Si damos doble clic sobre CountryInfoService, podemos ver que todos los métodos ofrecidos por el webservice CountryInfoService fueron incorporados al objeto externo, detallándose los parámetros necesarios de cada método y qué tipo de datos devuelve. En particular nos interesa el método LisfOfCountryNamesByName, que nos devolverá una lista de países ordenada por nombre.

Demo: Acceso a webservices SOAP



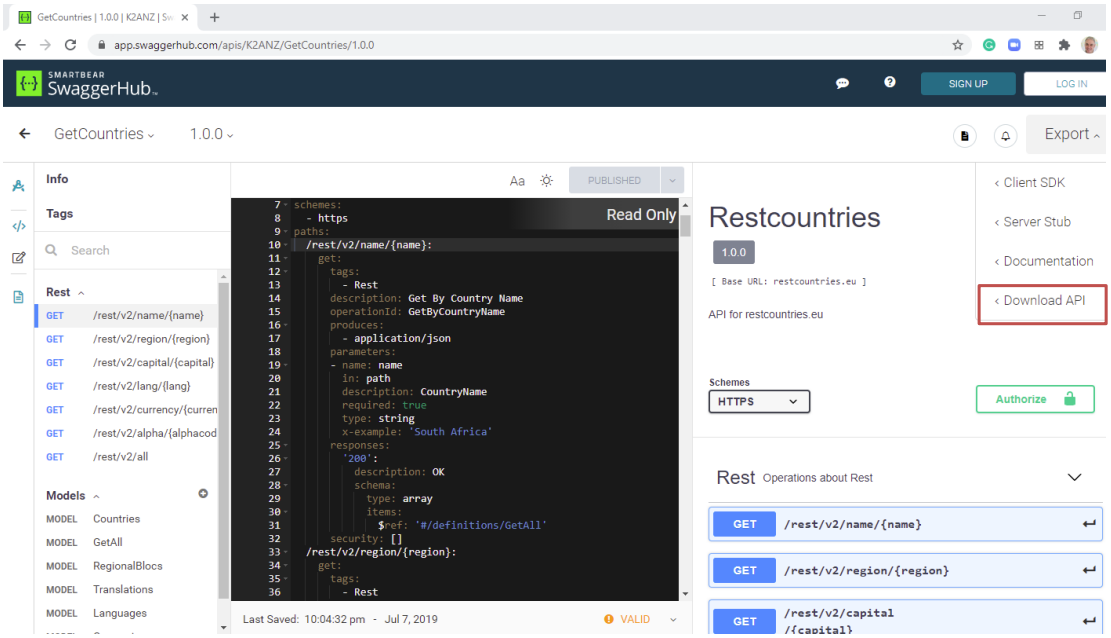
Ahora creamos un webpanel de nombre WebPanelCountryListFromWebService. En sus variables, creamos una variable del tipo CountryInfoService que automáticamente queda del tipo de datos del objeto externo.
Si volvemos a la definición del objeto externo, vemos que el tipo de datos devuelto por el método LisfOfCountryNamesByName es un SDT llamado CountryInfoServicetCountryCodeAndName. Si lo abrimos vemos que almacena el Código ISO del país y el nombre.
Volvamos al webpanel, creemos una variable CountryList, del tipo de datos del SDT CountryInfoServicetCountryCodeAndName y la marcamos como colección.
Ahora en el form, arrastramos un botón con el nombre del evento: Get countries list, damos doble clic sobre el mismo y en el evento insertamos la variable &CountryList y le asignamos la variable &CountryInfoService. Si escribimos punto, vemos que tenemos acceso a todos los métodos del webservice, así que elegimos ListOfCountryNamesByName.
Volvemos al form y arrastramos la variable CountryList basada en el sdt y presionamos OK.
Ejecutamos....

Si abrimos el webpanel WebPanelCountryListFromWebService y presionamos el botón Get countries list, vemos que obtenemos la lista de países ordenada alfabéticamente con el Código ISO de cada uno, tal como esperábamos.

Estos datos los podemos usar luego en nuestra aplicación de la agencia de viajes como lista de selección para filtrar por un país, o en diferentes usos.

Demo: Acceso a webservices REST con protocolo OpenAPI

GeneXus actualmente puede importar OpenAPI versión 2.0



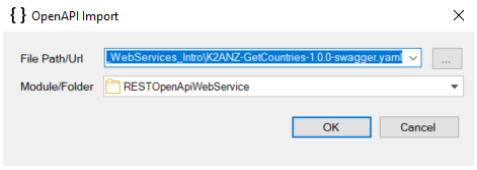
DEMO

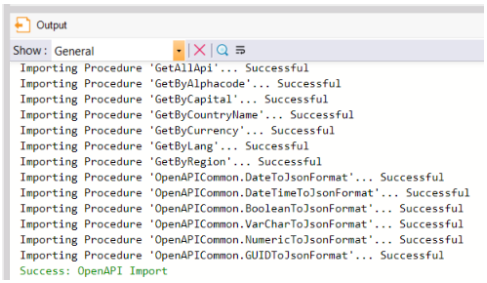
Veamos el caso de importar un webservice REST que cumple con la especificación OpenAPI, también conocida como especificación Swagger.

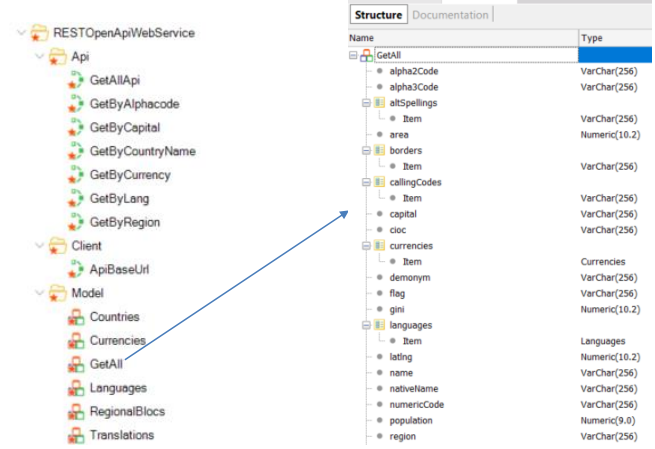
El ejemplo lo obtenemos de la página: https://app.swaggerhub.com, una api llamada GetCountries.

Elegimos la opción Download API y descargamos el archivo .yaml.

Demo: Acceso a webservices REST con protocolo OpenAPI








```
GetAllApi(&GetAllOUT, &HttpMessage, &IsSuccess)
```

DEMO

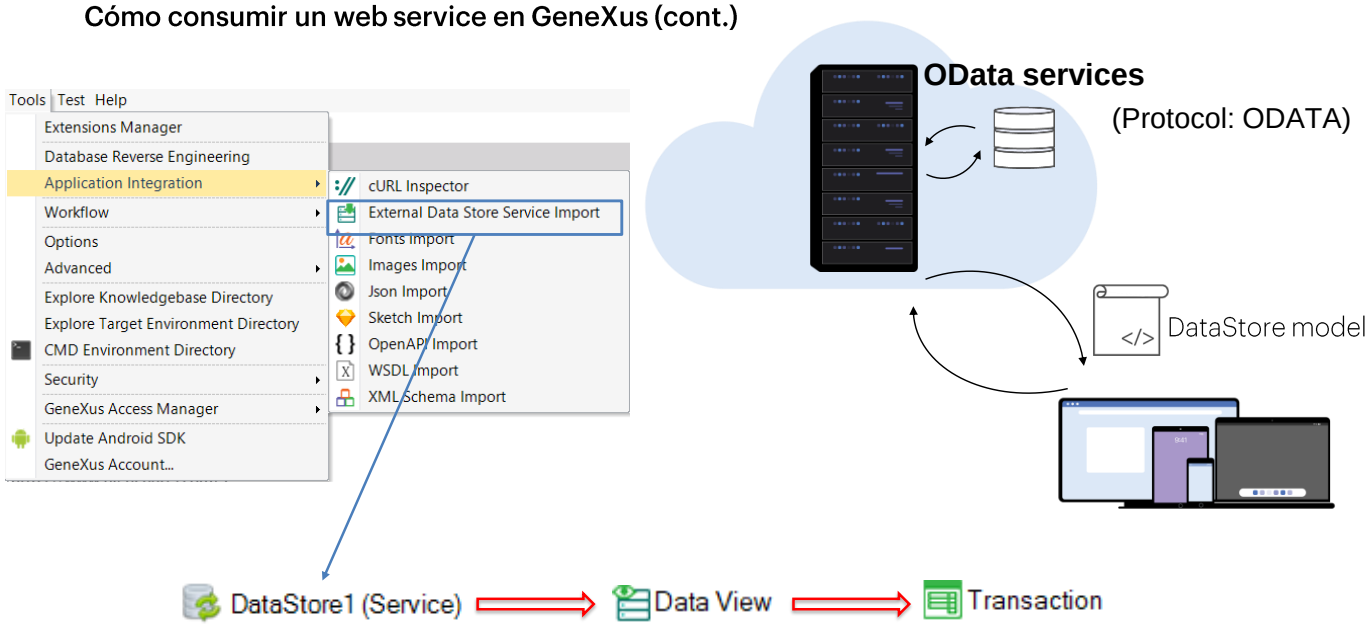
Luego Vamos a Tools/Application Integration/OpenApi import y elegimos el archivo .yaml que habíamos descargado. Como folder escribimos RESTOpenApiWebService y presionamos OK.

Vemos en la ventana de Output que se importan varios elementos y al finalizar abrimos la carpeta de destino. Encontramos que se crearon las carpetas Api que contiene los métodos que podemos invocar, la carpeta Client con el método ApiBaseURL y la carpeta Model que contiene varios SDTs que son los tipos de datos devueltos por los métodos anteriores.

Si abrimos el SDT de nombre GetAll podemos ver todos los datos que podemos recuperar de los países. Y si queremos obtener la lista de los países, podemos invocar al método GetAllApi que nos devolverá los datos en las variables &GetAllOut (colección de elementos GetAll), &HttpMessage y la variable booleana &IsSuccess.

Luego podremos recorrer la colección &GetAllOut para obtener los datos de los países.

Cómo consumir un web service en GeneXus (cont.)



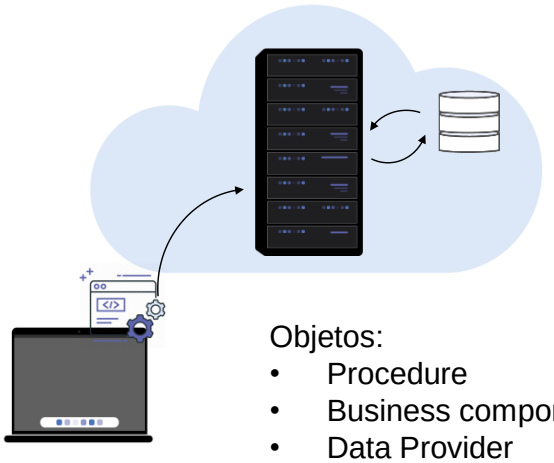
Un tipo especial de servicios web son los llamados OData services. Estos servicios utilizan el protocolo Open Data Protocol (Odata) diseñado para proveer operaciones de insertar, modificar o eliminar registros de una base de datos, a través de un servicio web.

Así como en un webservice Soap o Rest había que importar primero su definición, para consumir un servicio Odata primero debemos importar el modelo con las entidades de la base de datos incluidas en el servicio.

Para hacerlo, vamos a Tools/Application Integration y elegimos External DataStore Service Import. Como el proceso implica crear un DataStore del tipo Servicio para asociarlo al datastore externo, sólo podemos usar esta funcionalidad en GeneXus Full.

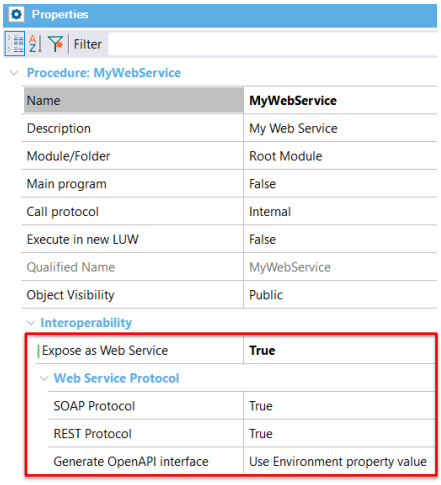
Luego de ejecutado el wizard, se crearán tantos objetos transacción como entidades estaban en el modelo y podremos trabajar con ellas como con cualquier otra transacción de nuestra aplicación.

Cómo publicar un web service con GeneXus



Objetos:

- Procedure
- Business component
- Data Provider



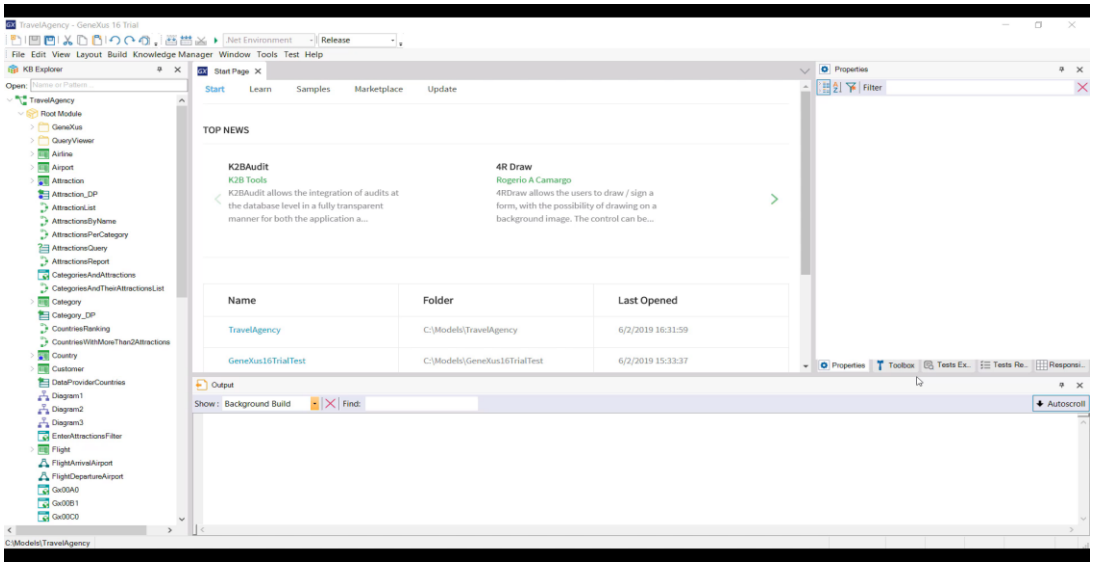
GeneXus también permite crear web services y publicarlos en un servidor web.

Los objetos GeneXus que pueden ser expuestos como un web service son los procedimientos, los business components y los data providers.

Para exponer uno de estos objetos como web service asignamos la propiedad Expose as Web Service > True, y luego podremos especificar si queremos usar el protocolo SOAP, REST o ambos.

Si queremos que nuestro servicio provea operaciones de CRUD sobre una base de datos, también podemos generar la información necesaria siguiendo el protocolo Odata

Ejemplo de uso de un web service REST



DEMO

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

Como ejemplo de lo visto, crearemos un web service REST que inserte una aerolínea nueva en la base de datos y lo invocaremos desde nuestra aplicación.

Para hacerlo, antes que nada, abrimos la transacción Airline y ponemos su propiedad Business Component en True.
Luego vamos a crear un objeto procedimiento que reciba por parámetro los datos de una aerolínea e inserte un registro nuevo en la tabla Airline.

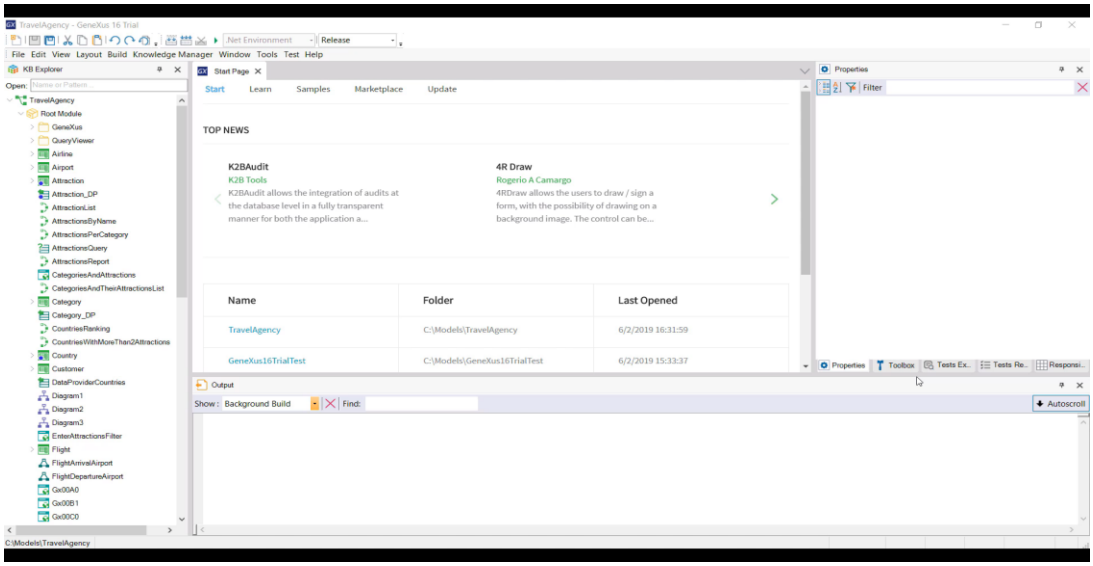
Llamamos al procedimiento CreateNewAirlineWS y definimos una regla Parm con 2 variables de entrada: &AirlineName y &AirlineDiscountPercentage. Definimos estas dos variables usando Add Variable y luego en la sección Variables definimos una variable más, Airline del tipo business component Airline.

En el source escribimos el código para insertar una aerolínea con los datos recibidos por parámetro. No necesitamos darle valor al AirlineId ya que es autonumerado.

Ahora vamos a las propiedades del objeto procedimiento y asignamos **Expose as Web Service** en True, **SOAP Protocol** en False, **REST Protocol lo dejamos** en True y **Generate OpenAPI interface** en Yes. Esto último permitirá que se genere la definición de nuestro webservice en el estándar OpenAPI.
Damos botón derecho sobre el procedimiento y elegimos Build With This Only, con lo que el servicio se publicará en nuestro servidor web, en este caso en nuestra máquina local. En la ventana de Ouput vemos un mensaje que nos confirma que se generó la documentación de la API Rest del webservice, con la definición del mismo.

Antes de importar el webservice, vamos a crear un módulo nuevo, al que llamamos WebServices, así queda todo guardado en ese módulo.

Ejemplo de uso de un web service REST



DEMO

[DEMO: https://youtu.be/bvmt0GjcxpW]

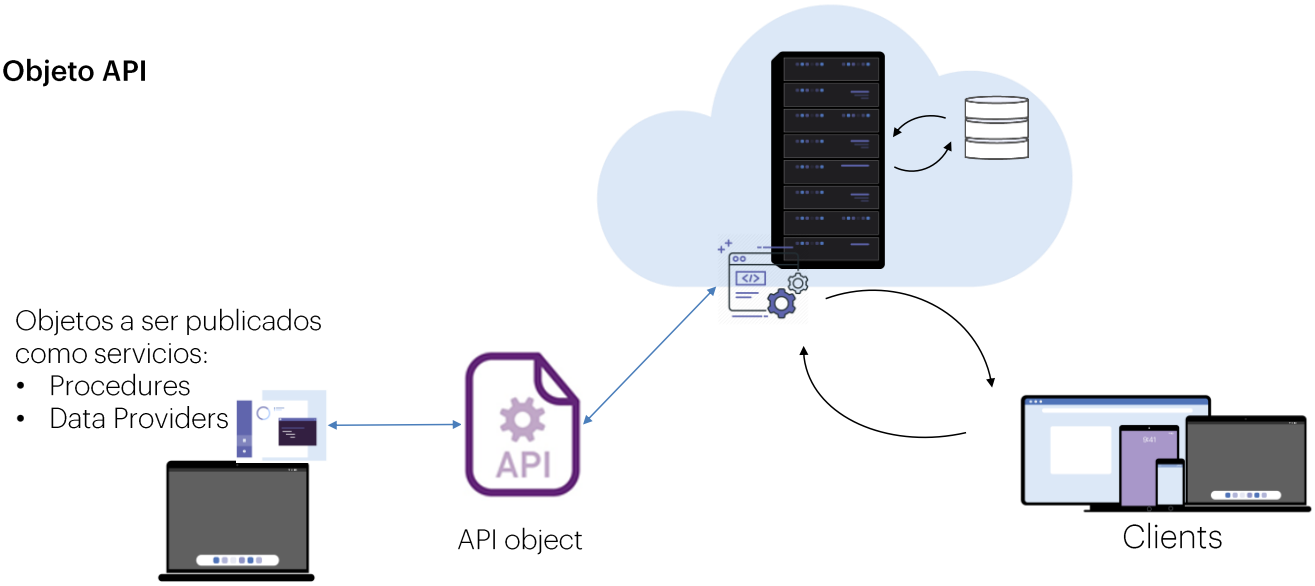
Para importar el webservice, vamos a Tools/Application Integration y elegimos OpenAPI Import. En el File Path / URL escribimos: C:\Models\TravelAgency\CSharpModel\Web\default.yaml que es donde quedó generada la documentación de la API Rest y elegimos como destino el módulo WebServices. Vemos que se importó todo correctamente y si abrimos el módulo WebServices, en el folder Api está el procedimiento importado. También vemos que en el folder Model hay un SDT llamado CreateNewAirlineWSInput, que si lo abrimos vemos que los parámetros que precisamos pasarle al servicio están disponibles aquí.

Para probar que el web service funcione correctamente, creamos un web panel llamado CreateNewAirlineUsingWS. Luego arrastramos un botón al form y al evento lo llamamos Create airline. Ahora vamos a las variables y definimos una variable &CreateNewAirlineWSInput que automáticamente queda definida del tipo del SDT.
Para poder tener un feedback del resultado de la ejecución del webservice, también creamos una variable &IsSuccess y otra &HttpMessage, vemos que los tipos se asignan por defecto.

Damos doble clic sobre el botón y en el evento cargamos los miembros del SDT, luego invocamos al webservice pasando el SDT como parámetro y por último escribimos el siguiente código para dar mensajes por pantalla

Presionamos F5... Vemos las arolíneas que tenemos ingresadas... Y ejecutamos el webpanel que creamos recién.
Presionamos el botón y vemos que el servicio nos informa que la aerolínea se creó correctamente.
Para verificar vamos a la transacción Airline... Y vemos que efectivamente se agregó la aerolínea que queríamos.

Objeto API



Veremos ahora otro mecanismo para publicar objetos como servicios, utilizando el objeto API.

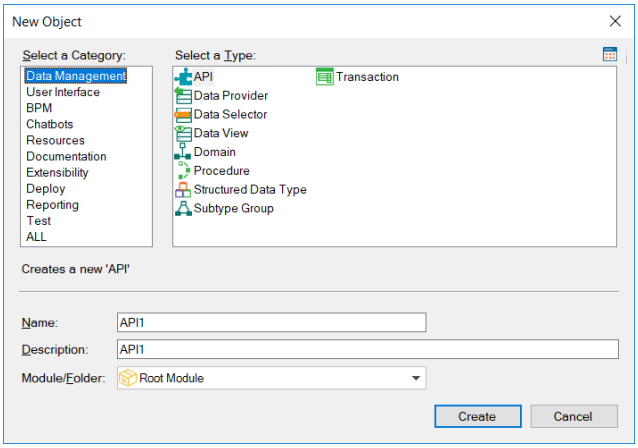
El objeto API (su nombre viene de Application Programming Interface) es un objeto GeneXus que nos permite definir una interfase de acceso por programación a algunos objetos de nuestra aplicación, como procedimientos o data providers.
Esto significa que una aplicación externa a la nuestra podrá acceder a estos objetos publicados como web services.

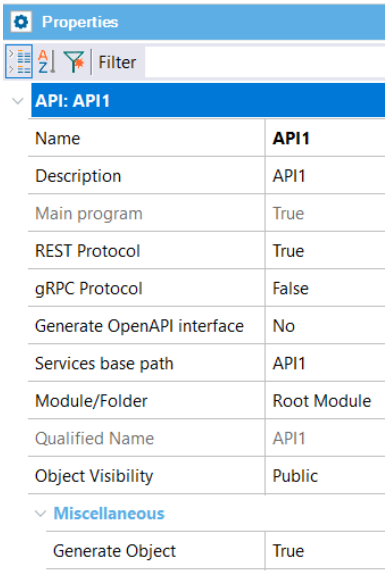
El objeto API agrega una capa intermedia que separa la interfase, de los detalles de implementación, lo que permite que futuros cambios en la programación en los objetos no afecten la forma en que son invocados por las aplicaciones externas.

Por ejemplo, se hace un mapeo entre el nombre interno del objeto en la KB y el nombre con el que el mismo es expuesto como servicio, así como también se puede cambiar el tipo y nombre de los parámetros o cambiar el path de acceso, sin que eso afecte a la forma en que el servicio es invocado.

Esta abstracción provee una flexibilidad importante porque podemos evolucionar nuestros servicios sin obligar a que las aplicaciones que los usan deban cambiar su código para adaptarse a esos cambios.

Objeto API (cont.)

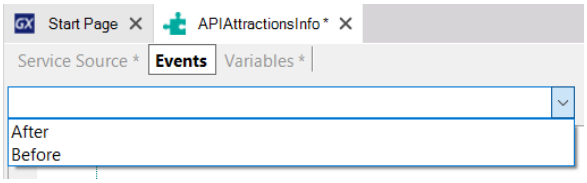


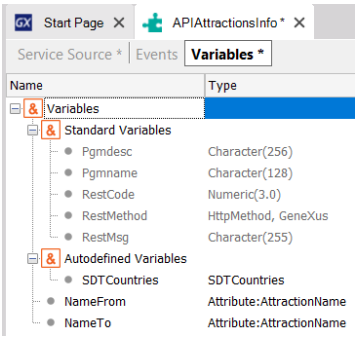


Para crear un objeto API, vamos a la categoría Data Management y seleccionamos el tipo API. En sus propiedades podemos definir en tiempo de diseño su nombre, dirección, protocolo a usarse, entre otras cosas.

El protocolo gRPC es un moderno framework open source desarrollado por Google, que permite hacer llamadas a procedimientos remotos con una alta performance y en forma bidireccional.

Objeto API (cont.)





El objeto API tiene tres secciones: Service Source, Events y Variables.

En Service Source se define, mediante una sintaxis declarativa, el nombre de cada servicio a ser publicado, con los parámetros que correspondan y el nombre del objeto GeneXus en nuestra KB que estamos exponiendo como servicio, con sus respectivos parámetros.

En el ejemplo vemos que el data provider que creamos anteriormente de nombre RankingCountriesWithAttractionsQty lo estamos publicando como servicio con el nombre RankingCountriesAttractions y los parámetros que exponemos, son los mismos que los del objeto.
Lo mismo para el listado AttractionsByName que lo publicamos con el nombre ListAttractionsByName.

Esta definición nos permite en el futuro cambiar el nombre o los parámetros del objeto GeneXus, sin que cambie la definición de cómo está publicado este objeto.

En los eventos disponemos del evento Before y After, con el que podemos realizar acciones que se ejecuten antes o después de la invocación al objeto como servicio.

En las variables hay algunas del tipo estándar, que nos permiten definir o cambiar características del objeto API en tiempo de ejecución.

Más información sobre Web Services

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
API Object:https://wiki.genexus.com/commwiki/servlet/wiki?46151

Por más información sobre webservices en GeneXus, siga los siguientes links del Wiki.

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