

Web Services. Aspectos avanzados.

Consumiendo servicios REST con GeneXus

GeneXus™

En este video veremos como consumir con GeneXus servicios REST con protocolo OpenAPI hechos por terceros o publicados desde GeneXus.
Y en particular, cómo podemos invocar métodos HTTP de servicios REST, o cómo consumir un servicio REST seguro.

Consuming a REST web service with GeneXus

The screenshot illustrates the process of consuming a REST web service in GeneXus. It shows the following components:

- Main Editor:** Contains the `GetAttractionsInfo` service definition with methods like `GetAttractionsByCountry` and `GetAttractionsByCountryAndTrips`.
- OpenAPI Import Dialog:** A dialog box with two steps:
 - Step 1 - Insert the yaml/json OpenAPI specification:** The file path is set to `E:\Models\TravelAgency_ExpertCourse\Local .NET Environment\web\GetAttractionsInfo.yaml`.
 - Step 2 - Select the operations you want to import:** Two operations are selected: `GetAttractionsInfoGetAttractionsByCountry` and `GetAttractionsInfoGetAttractionsByCountryAndTrips`.
- Variables Window:** Shows the mapping of API parameters to GeneXus variables:

Name	Type
Standard Variables	
PgmDesc	Character(256)
PgmName	Character(128)
RestCode	Numeric(3,0)
RestMethod	HttpMethod, GeneXus
Autodefined Variables	
CountryId	Attribute:CountryId
Attractions	LongVarChar(2M)
TripsQty	Numeric(4,0)
- Project Structure:** Shows the `GetAttractionsInformation` project with `Api` and `Client` folders. Arrows indicate the generated code for `GetAttractionsByCountry` and `GetAttractionsByCountryAndTrips` in the main editor.

Vamos a consumir el servicio `GetAttractionsByCountry`, que construimos publicando como servicio al procedimiento `GetAttractionsByCountryWS`, mediante el objeto API `GetAttractionsInfo` que vimos previamente.

Para importar la definición del servicio REST vamos a `Tools / Application Integration / OpenAPI import` y escribimos la URL o el file path del archivo JSON con la especificación Swagger del servicio REST. Swagger es un conjunto de herramientas de software de código abierto para diseñar, construir, documentar, y utilizar servicios REST que fue desarrollado por SmartBear Software e incluye documentación automatizada, generación de código, y generación de casos de prueba.

El archivo que vamos a importar con esta especificación, puede tener extensión `.JSON` o también `.YAML` que es un superset de JSON.

Si cuando publicamos nuestro servicio REST pusimos la propiedad `Generate Open API interface` en `True`, disponible en el objeto API o en el procedimiento expuesto como REST, se genera automáticamente el archivo con especificación Swagger con extensión `yaml`, en la carpeta `Web` del Environment.

El archivo Swagger que importemos puede tener una especificación OpenAPI versión 2 o 3. A partir de la versión 17 upgrade 6, GeneXus soporta tanto la versión 2 como la versión 3 de la especificación OpenAPI.

Continuando con nuestro ejemplo, en el cuadro de diálogo donde nos pide el path buscamos al archivo `GetAttractionsInfo.yaml` en la carpeta `Web` de nuestro environment activo. En `Module/Folder` escribimos el nombre de un módulo que creamos antes para contener todo lo que importemos. Esto es una buena práctica,

por si se da el caso de que importe algún objeto que tenga el mismo nombre que algún objeto ya existente en la KB.

Presionamos el botón Import y vemos que el wizard encuentra a los dos servicios que habíamos expuestos con el objeto API.

Marcamos Select All y luego OK.

Ahora si abrimos el módulo, vemos que hay 3 carpetas, una con el nombre API donde encontramos dos objetos procedimientos con el nombre de los servicios, que son los que vamos a ejecutar para invocar a los servicios, una carpeta Client que tiene un procedimiento APIBaseURL que retorna la URL Base que será usada para invocar al servicio y que la podremos cambiar si lo deseamos y una carpeta Model que en nuestro caso está vacía, porque los métodos anteriores no devuelven ningún SDT.

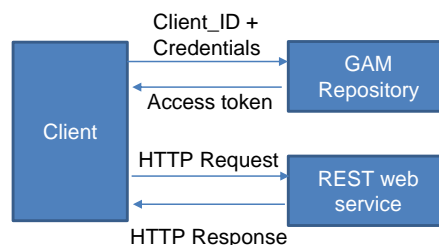
Si vemos las reglas de los procedimientos, vemos los parámetros de entrada, en los que la variable &ServerUrlTemplatingVar está presente en todos los procedimientos que se consumen y los otros son los que reconocemos y como parámetros de salida tenemos a la variable &VarCharOUT que contendrá la información solicitada y las variables &HttpMessage y &IsSuccess que podemos usar para tener información de la ejecución del servicio.

Consuming secure REST web services with GeneXus



- Client identification
- Users & Passwords
- Permissions

GeneXus™
Access Manager



```

//First get the access_token
&httpClient.Host= &server
&httpClient.Port = &port

&GAMOAuthAdditionalParameters.ClientId = "f719771ad52a42919a221bc796d0d6b0"
&GAMOAuthAdditionalParameters.ClientSecret = &ClientSecret
&GAMLoginAdditionalParameters.AuthenticationTypeName = !"local"
&AccessTokenSDT = GAMRepository.GetOAuthAccessToken(!"admin", !"admin123",&GAMLoginAdditionalParameters,&GAMOAuthAdditionalParameters,&GAMSession,&GAMErrors)

//call DPPProduct web service
&httpClient.BaseUrl = &urlbase + '/rest/'
&httpClient.AddHeader("Content-Type", "application/json")
&httpClient.AddHeader('Authorization','OAuth ' + &AccessTokenSDT.access_token)
&httpClient.AddHeader("GENEXUS-AGENT","SmartDevice Application")
&httpClient.Execute('GET','DPPProduct')
  
```

Así como remarcamos la importancia de la seguridad de los servicios SOAP, lo mismo debemos hacer con los servicios REST.

Los servicios REST seguros están basados en el esquema de seguridad OAuth y esto implica definir el Cliente, es decir la aplicación, los usuarios (UserId y UserPassword) y los permisos (Read, Write, FullControl, etc.).

En GeneXus esto está provisto a través del GAM, con una autenticación basada en OAuth versión 2.0.

Cuando exponemos un procedimiento, un data provider o un business component como servicio REST, si tenemos GAM aplicado, el servicio REST es identificado como aplicación dentro del repositorio del GAM. Para proveer acceso al servicio debemos configurar los roles, usuarios y permisos de la aplicación del servicio y luego proveer a quien consuma el servicio, el identificador de cliente (Client_Id) de la aplicación, usuario y password.

Antes de invocar al servicio, el cliente debe obtener un token de acceso. Para eso debe hacer un POST al repositorio del GAM con el Client_ID y las credenciales de acceso provistas anteriormente. La respuesta del GAM será un JSON con el token de acceso y el tipo de permiso (FullControl, etc.)

Esta invocación al repositorio del GAM se puede hacer mediante el método GetOAuthAccessToken() de la API del GAM.

Una vez obtenido el token, se debe utilizar una variable del tipo HTTPClient para consumir el servicio REST. Aquí vemos un ejemplo de consumo de un servicio REST

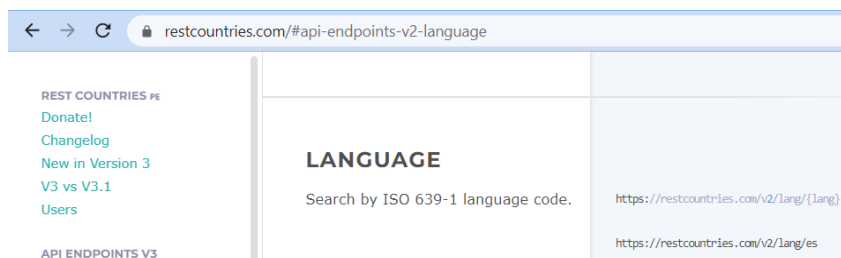
seguro.

Puede obtener más información en el wiki, en el artículo “HowTo: Develop Secure REST Web Services in GeneXus”.

Customizing the consumption of a REST web service

Customizing the consumption of a REST web service

```
&HttpClient.Execute("GET", ...)
&HttpClient.Execute("POST", ...)
&HttpClient.Execute("PUT", ...)
&HttpClient.Execute("DELETE", ...)
```



```
//REST API: https://restcountries.com/v3.1/all?fields=name
&httpClient.Host = "restcountries.com"
&httpClient.Port = 443 // for https
&httpClient.Secure = 1
&httpClient.BaseUrl = "/v2/"
&httpClient.AddHeader("Content-Type", "application/json")
&httpClient.Execute("GET", "lang/es")

if &httpClient.StatusCode = 200
    &result = &httpClient.ToString()
else
    msg("Error: " + &httpClient.StatusCode.ToString())
endif
```

Así como personalizamos el consumo de servicios SOAP, veamos cómo podemos personalizar el consumo de un servicio REST.

Si bien lo recomendable es importar las definiciones de un servicio REST con el wizard Import OpenAPI que vimos, a veces el archivo de información del servicio (con extensión .yaml) no está disponible. En esos casos, es posible invocar los métodos HTTP: GET, PUT, POST y DELETE, utilizando una variable del tipo HTTPClient.

Veamos un ejemplo de la invocación de una API REST pública que retorna datos de países

Para invocar al web service, primero creamos la variable y luego le asignamos las propiedades: Host, Port, Secure y BaseUrl. Luego agregamos el header del tipo JSON e invocamos al método Execute, pasando el método que queremos usar y los parámetros requeridos por el servicio, en este caso estamos pasando el lenguaje porque queremos recuperar los países de habla hispana.

Luego de la invocación procesamos el status code devuelto. Si es 200 obtenemos el string JSON y de lo contrario damos un mensaje de error.

Para usar los otros métodos HTTP, sustituimos los parámetros del método Execute, por el método HTTP que deseamos y utilizamos los parámetros adecuados según el método, por ejemplo en un DELETE debemos pasar el identificador del registro que queremos eliminar.

Customizing the consumption of a REST web service

Web Layout | Rules | Events | Conditions | Variables | Help | Documentat...

Call Http GET for Countries Info

```

1 Event 'Call Http GET for Countries Info'
2 //REST API: https://restcountries.com/v3.1/all?fields=name
3 &httpClient.Host = "restcountries.com"
4 &httpClient.Port = 443 // for https
5 &httpClient.Secure = 1
6 &httpClient.BaseUrl = "/v2/"
7 &httpClient.AddHeader("Content-Type", "application/json")
8 &httpClient.Execute("GET", "lang/es")
9
10 if &httpClient.StatusCode = 200
11     &result = &httpClient.ToString()
12 else
13     msg("Error: " + &httpClient.StatusCode.ToString())
14 endif
15 -Endevent
  
```

localhost/TravelAgency_ExpertCourseNET.Local/getcountriesinfousinghttpget.aspx

Travel Agency - Backoffice

Recents Get Countries Info ...

REST API: restcountries.com

(Countries with Spanish language)

```

[{"name":"Argentina","topLevelDomain":
[".ar"],"alpha2Code":"AR","alpha3Code":"ARG","callingCodes":["54"],"capital":"Buenos
Aires","altSpellings":["AR","Argentine Republic","República Argentina"],"subregion":"South
America","region":"Americas","population":45376763,"latlng":
[-34.0,-64.0],"demonym":"Argentinean","area":2780400.0,"gini":42.9,"timezones":["UTC-
03:00"],"borders":
["BOL","BRA","CHL","PRY","URY"],"nativeName":"Argentina","numericCode":"032","flags":
{"svg":"https://flagcdn.com/ar.svg","png":"https://flagcdn.com/w320/ar.png"},"currencies":
[{"code":"ARS","name":"Argentine peso","symbol":"$"}],"languages":
[{"iso639_1":"es","iso639_2":"spa","name":"Spanish","nativeName":"Español"},
{"iso639_1":"gn","iso639_2":"grn","name":"Guarani","nativeName":"Avañe'ê"}],"translations":
{"br":"Archantina","pt":"Argentina","nl":"Argentinië","hr":"Argentina","fa":"آرژانتین","de":"Argentinien","es":
"Argentina","fr":"Argentine","ja":"アルゼンチ
>...
["it":"Argentina","hu":"Argentina"],"flag":"https://flagcdn.com/ar.svg"},"regionalBlocs":
[{"acronym":"UNASUR","name":"Union of South American Nations","otherAcronyms":
["UNASUR","UNASUL","UZAN"],"otherNames":["Unión de Naciones Suramericanas","União de
Nações Sul-Americanas","Unie van Zuid-Amerikaanse Naties","South American
Union"]},"cioc":"ARG","independent":true},{"name":"Belize","topLevelDomain":
[".bz"],"alpha2Code":"BZ","alpha3Code":"BLZ","callingCodes":
["501"],"capital":"Belmopan","altSpellings":["BZ"],"subregion":"Central
America","region":"Americas","population":397621,"latlng":
[17.25,-88.75],"demonym":"Belizean","area":22966.0,"gini":53.3,"timezones":["UTC-
06:00"],"borders":["GTM","MEX"],"nativeName":"Belize","numericCode":"084","flags":
  
```

Call Http GET for Countries Info

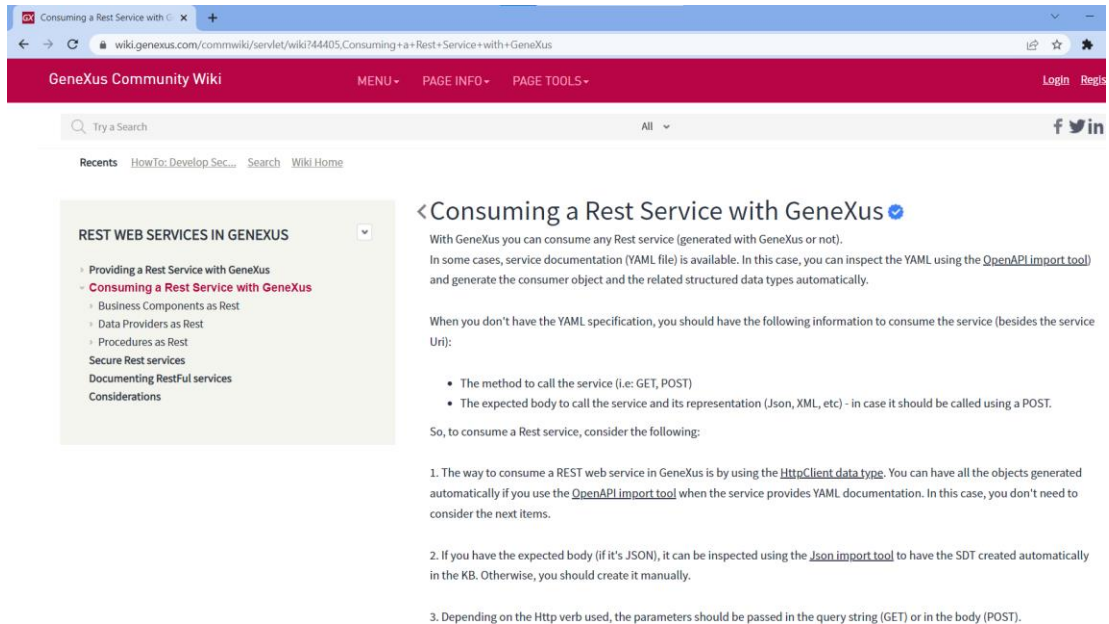
Vamos a ejecutar este ejemplo en GeneXus.

Tenemos creado al web panel GetcountriesInfoUsingHTTPGET e incluimos en el web layout un botón para invocar al servicio y una variable &result para mostrar el JSON que obtendremos.

En el evento del botón, vemos el código que ya explicamos.

Ejecutamos el web panel que es main... presionamos el botón y vemos que recibimos la información de los países de habla hispana, tal como queríamos.

Customizing the consumption of a REST web service



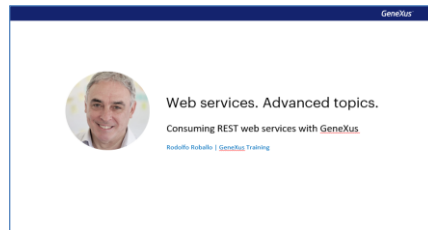
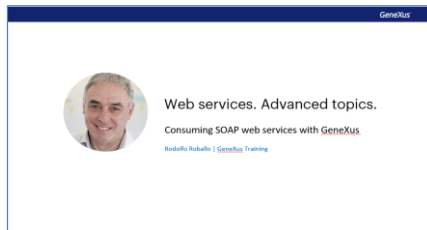
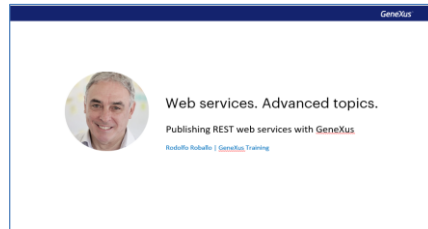
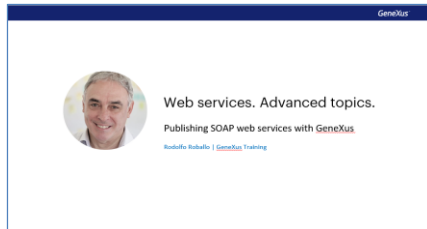
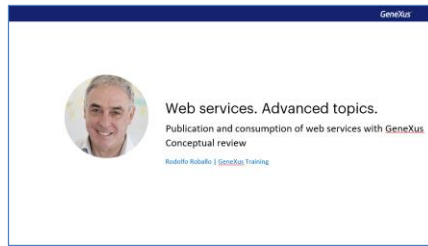
The screenshot shows a web browser displaying the GeneXus Community Wiki page. The page title is "Consuming a Rest Service with GeneXus". The left sidebar contains a navigation menu under "REST WEB SERVICES IN GENEXUS" with the following items: "Providing a Rest Service with GeneXus", "Consuming a Rest Service with GeneXus" (highlighted), "Business Components as Rest", "Data Providers as Rest", "Procedures as Rest", "Secure Rest services", "Documenting RestFul services", and "Considerations". The main content area has the heading "< Consuming a Rest Service with GeneXus" and the following text: "With GeneXus you can consume any Rest service (generated with GeneXus or not). In some cases, service documentation (YAML file) is available. In this case, you can inspect the [YAML](#) using the [OpenAPI import tool](#) and generate the consumer object and the related structured data types automatically. When you don't have the YAML specification, you should have the following information to consume the service (besides the service Uri):

- The method to call the service (i.e: GET, POST)
- The expected body to call the service and its representation (Json, XML, etc) - in case it should be called using a POST.

So, to consume a Rest service, consider the following:

1. The way to consume a REST web service in GeneXus is by using the [HttpClient data type](#). You can have all the objects generated automatically if you use the [OpenAPI import tool](#) when the service provides YAML documentation. In this case, you don't need to consider the next items.
2. If you have the expected body (if it's JSON), it can be inspected using the [Json import tool](#) to have the SDT created automatically in the KB. Otherwise, you should create it manually.
3. Depending on the Http verb used, the parameters should be passed in the query string (GET) or in the body (POST).

Por más información, consulte el artículo del wiki: "Consuming a Rest Service with GeneXus".



En estos videos de web services con GeneXus, tratamos de cubrir los casos de uso más comunes de publicación y consumo de servicios tanto SOAP como REST, en los ejemplos más simples y en situaciones donde se requiere de personalización.

Lo invitamos a profundizar sobre este y otros temas relacionados en nuestro Wiki.

GeneXus™

training.genexus.com

wiki.genexus.com

training.genexus.com/certifications