

Web Services. Aspectos avançados.

Consumindo serviços REST com GeneXus

GeneXus™

Neste vídeo veremos como consumir com GeneXus serviços REST com protocolo OpenAPI feitos por terceiros ou publicados a partir do GeneXus.
E, em particular, como podemos invocar métodos HTTP de serviços REST, ou como consumir um serviço 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 'GetAttractionsInfo' service source with two methods: `GetAttractionsByCountry` and `GetAttractionsByCountryAndTrips`. A dialog box titled 'OpenAPI Import' is open, showing the steps to import the service specification. The 'File Path/Url' is set to 'E:\Models\TravelAgency_ExpertCourse\Local .NET Environment\web\GetAttractionsInfo.yaml'. The 'Module/Folder' is 'GetAttractionsInformation'. In Step 2, the operations 'GetAttractionsInfoGetAttractionsByCountry' and 'GetAttractionsInfoGetAttractionsByCountryAndTrips' are selected. The 'Procedure description' for 'GetAttractionsByCountry' is shown. Below the dialog, the 'Rules' tab of the 'GetAttractionsByCountry' procedure is visible, showing the parameter definition: `1 param(in:&ServerUrlTemplatingVar, in:&Id, out:&VarCharOUT, out:&HttpMessage, out:&IsSuccess);`

Vamos consumir o serviço `GetAttractionsByCountry`, que construímos publicando como serviço o procedimento `GetAttractionsByCountryWS`, usando o objeto API `GetAttractionsInfo` que vimos anteriormente.

Para importar a definição do serviço REST vamos em **Tools / Application Integration / OpenAPI import** e escrevemos a URL ou o file path do arquivo JSON com a especificação Swagger do serviço REST. Swagger é um conjunto de ferramentas de software de código aberto para projetar, construir, documentar e utilizar serviços REST que foi desenvolvido por SmartBear Software e inclui documentação automatizada, geração de código e geração de casos de teste. O arquivo que vamos importar com esta especificação pode ter uma extensão .JSON ou também .YAML, que é um superset de JSON.

Se quando publicamos nosso serviço REST definimos a propriedade `Generate Open API interface` como `True`, disponível no objeto API ou no procedimento exposto como REST, é gerado automaticamente o arquivo com especificação Swagger com extensão `yaml`, na pasta `Web` do `Environment`.

O arquivo Swagger que importamos pode ter uma especificação OpenAPI versão 2 ou 3. A partir da versão 17 upgrade 6, GeneXus suporta tanto a versão 2 como a versão 3 da especificação OpenAPI.

Continuando com nosso exemplo, na caixa de diálogo onde nos pede o path, procuramos o arquivo `GetAttractionsInfo.yaml` na pasta `Web` de nosso `environment` ativo. Em `Module/Folder` escrevemos o nome de um módulo que criamos anteriormente para conter tudo o que importamos. Esta é uma boa prática, caso importemos algum objeto que tenha o mesmo nome de algum

objeto já existente na KB.

Pressionamos o botão Import e vemos que o wizard encontra os dois serviços que expusemos com o objeto API.

Marcamos Select All e depois OK.

Agora se abrimos o módulo, veremos que existem 3 pastas, uma com o nome API onde encontramos dois objetos procedimentos com o nome dos serviços, que são os que vamos executar para invocar os serviços, uma pasta Client que tem um procedimento APIBaseURL que retorna a URL Base que será usada para invocar o serviço e que poderemos alterá-la se quisermos e uma pasta Model que no nosso caso está vazia, pois os métodos anteriores não retornam nenhum SDT.

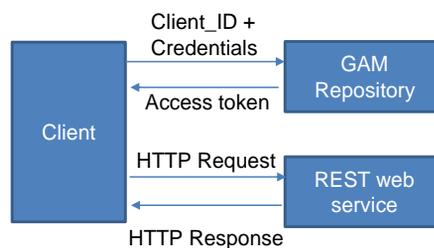
Se vemos as regras dos procedimentos, vemos os parâmetros de entrada, nos quais a variável &ServerUrlTemplatingVar está presente em todos os procedimentos que são consumidos e os demais são os que reconhecemos e como parâmetros de saída temos a variável &VarCharOUT que conterà a informação solicitada e as variáveis &HttpMessage e &IsSuccess que podemos usar para ter informação sobre a execução do serviço.

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')
  
```

Assim como enfatizamos a importância da segurança dos serviços SOAP, devemos fazer o mesmo com os serviços REST.

Os serviços REST seguros são baseados no esquema de segurança OAuth e isto implica definir o Cliente, ou seja, a aplicação, os usuários (UserId e UserPassword) e as permissões (Read, Write, FullControl, etc.).

No GeneXus isto é fornecido através do GAM, com uma autenticação baseada em OAuth versão 2.0.

Quando expomos um procedimento, um data provider ou um business component como serviço REST, se tivermos GAM aplicado, o serviço REST é identificado como aplicação dentro do repositório do GAM. Para fornecer acesso ao serviço devemos configurar as roles, usuários e permissões da aplicação do serviço e então fornecer a quem consome o serviço, o identificador de cliente (Client_Id) da aplicação, usuário e password.

Antes de invocar o serviço, o cliente deve obter um token de acesso. Para isso, deve fazer um POST no repositório do GAM com o Client_ID e as credenciais de acesso fornecidas anteriormente. A resposta do GAM será um JSON com o token de acesso e o tipo de permissão (FullControl, etc.)

Esta invocação ao repositório do GAM pode ser feita por meio do método GetOAuthAccessToken() da API do GAM.

Uma vez obtido o token, deve ser utilizada uma variável do tipo HttpClient para consumir o serviço REST.

Você pode obter mais informações na wiki, no artigo “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
```

Assim como personalizamos o consumo de serviços SOAP, vamos ver como podemos personalizar consumo de um serviço REST.

Embora seja recomendado importar as definições de um serviço REST com o wizard Import OpenAPI que vimos, às vezes o arquivo de informação do serviço (com extensão .yaml) não está disponível. Nesses casos, é possível invocar os métodos HTTP: GET, PUT, POST e DELETE, utilizando uma variável do tipo HttpClient.

Vejamos um exemplo da invocação de uma API REST pública que retorna dados de países

Para invocar o web service, primeiro criamos a variável e depois atribuímos as propriedades: Host, Port, Secure e BaseUrl. Em seguida, adicionamos o header do tipo JSON e invocamos o método Execute, passando o método que queremos usar e os parâmetros requeridos pelo serviço, neste caso estamos passando o idioma porque queremos recuperar os países de língua espanhola.

Após a invocação, processamos o status code retornado. Se for 200, obtemos a string JSON e, caso contrário, damos uma mensagem de erro.

Para usar os outros métodos HTTP, substituímos os parâmetros do método Execute pelo método HTTP que queremos e utilizamos os parâmetros adequados conforme o método, por exemplo em um DELETE devemos passar o identificador do registro que queremos excluir.

Customizing the consumption of a REST web service

The screenshot shows the GeneXus IDE interface. On the left, the design view displays a web layout with a button labeled 'Call Http GET for Countries Info' and a data table labeled '(Countries with Spanish language)' containing a column '&result'. The right pane shows the code for the 'Call Http GET for Countries Info' event:

```

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
  
```

The screenshot shows a web browser window at localhost/TravelAgency_ExpertCourseNET.Local/getcountriesinfousinghttpget.aspx. The page title is 'Travel Agency - Backoffice'. Below the title, there are links for 'Recents' and 'Get Countries Info...'. The main content area displays 'REST API: restcountries.com' and '(Countries with Spanish language)'. A scrollable area shows the JSON response for Argentina:

```

[{"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","e
s":"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":
  
```

At the bottom of the page, there is a button labeled 'Call Http GET for Countries Info'.

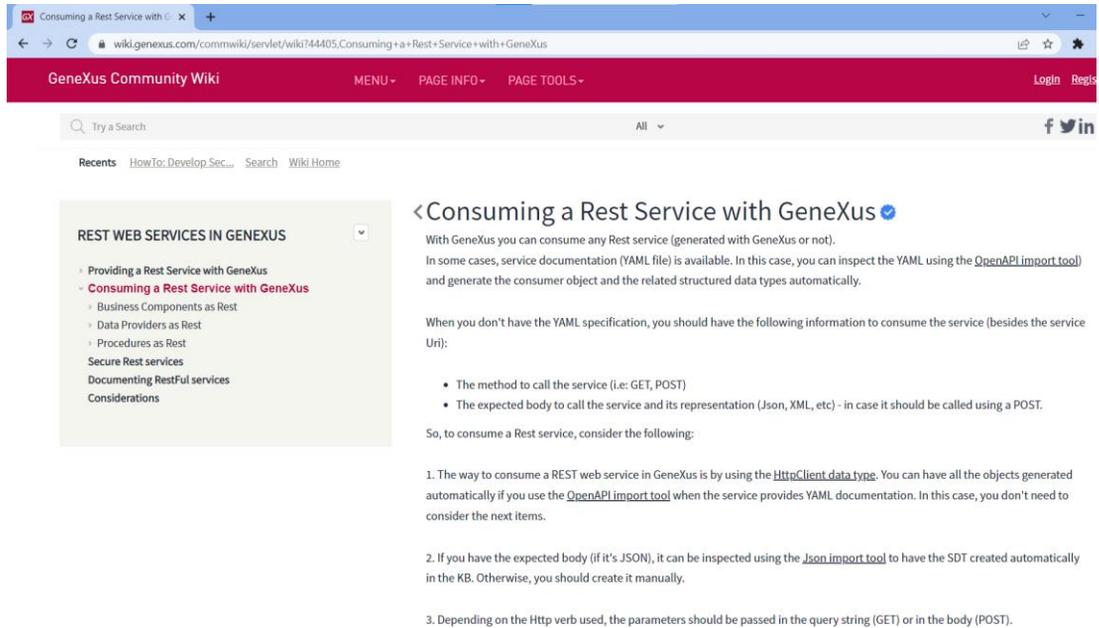
Vamos executar este exemplo no GeneXus.

Criamos o web panel GetcountriesInfoUsingHTTPGET e incluímos no web layout um botão para invocar o serviço e uma variável &result para mostrar o JSON que obteremos.

No evento do botão, vemos o código que já explicamos.

Executamos o web panel que é main... pressionamos o botão e vemos que recebemos a informação dos países de língua espanhola, exatamente como queríamos.

Customizing the consumption of a REST web service



The screenshot shows a web browser window with the URL wiki.genexus.com/commwiki/servlet/wiki?44405,Consuming+a+Rest+Service+with+GeneXus. The page is titled "Consuming a Rest Service with GeneXus" and is part of the GeneXus Community Wiki. The left sidebar contains a navigation menu with the following items:

- REST WEB SERVICES IN GENEXUS
 - Providing a Rest Service with GeneXus
 - Consuming a Rest Service with GeneXus**
 - Business Components as Rest
 - Data Providers as Rest
 - Procedures as Rest
- Secure Rest services
- Documenting RestFul services
- Considerations

The main content area of the page is titled "Consuming a Rest Service with GeneXus" and contains 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.

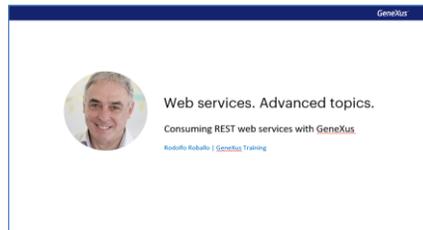
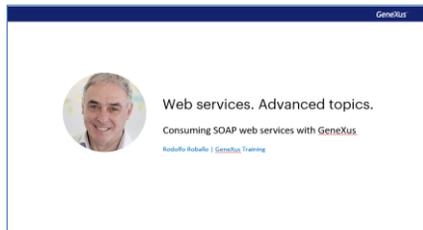
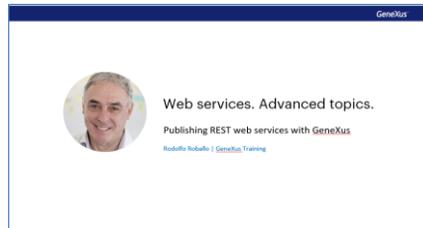
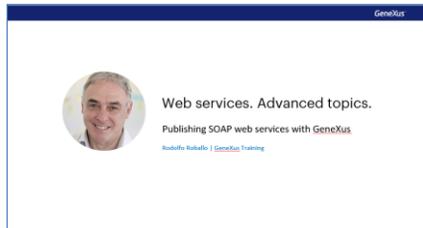
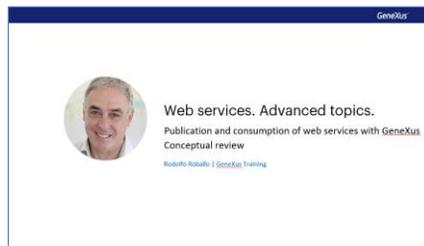
When you don't have the YAML specification, you should have the following information to consume the service (besides the service Uri):

- 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:

- 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.
- 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.
- Depending on the Http verb used, the parameters should be passed in the query string (GET) or in the body (POST).

Para mais informações, consulte o artigo da wiki: "Consuming a Rest Service with GeneXus".



Nestes vídeos de web services com GeneXus, tentamos cobrir os casos de uso mais comuns de publicação e consumo de serviços, tanto SOAP quanto REST, nos exemplos mais simples e em situações em que é necessária personalização.

Convidamos você a se aprofundar neste e em outros temas relacionados em nossa Wiki.

GeneXus™

training.genexus.com

wiki.genexus.com

training.genexus.com/certifications