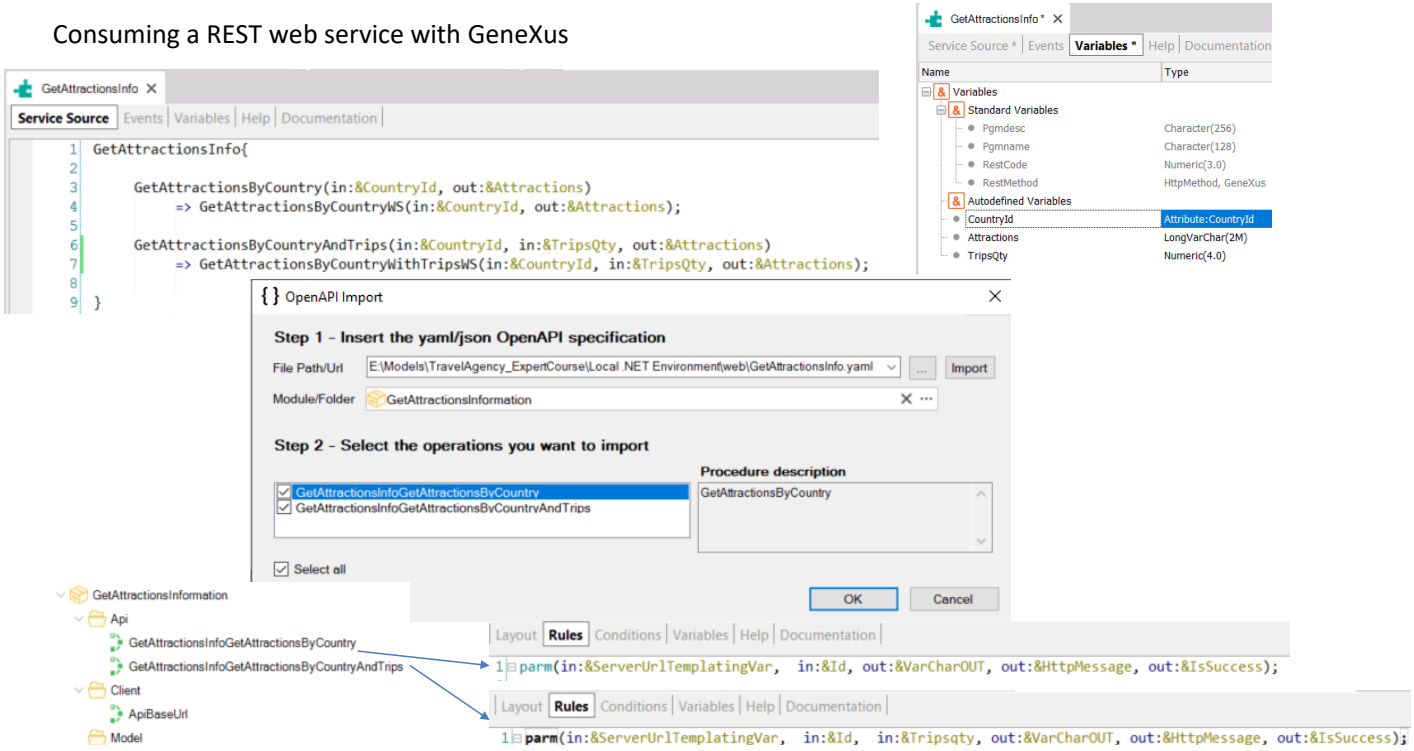
# Web Services. Aspectos avançados.

## 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




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.

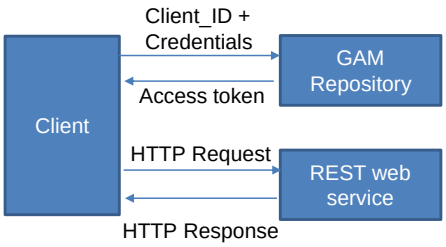
Agora se abrirmos o módulo, veremos que existem 3 pastas, uma com o nome API onde encontramos dois objetos procedimentos com o nome dos serviços, que são os que vamos executar para invocar os serviços, uma pasta Client que tem um procedimento APIBaseURL que retorna a URL Base que será usada para invocar 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 conterá 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,&GAMOauthAdd
itionalParameters,&GAMSession,&GAMErrors)

//call DPProduct 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','DPProduct')
```

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 consuma 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.

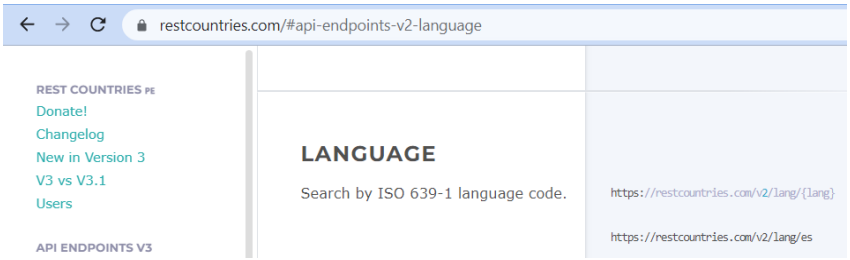
Uma vez obtido o token, deve ser utilizada uma variável do tipo HTTPClient para consumir o serviço REST.

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.

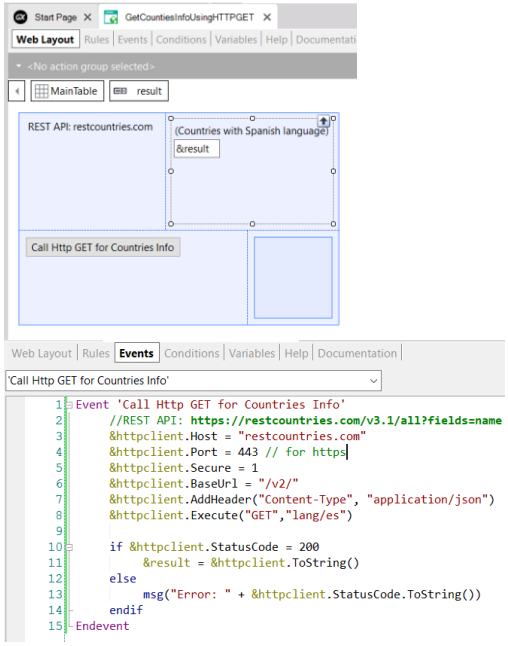
Vejamos um exemplo da invocação de uma API REST pública que retorna dados de países

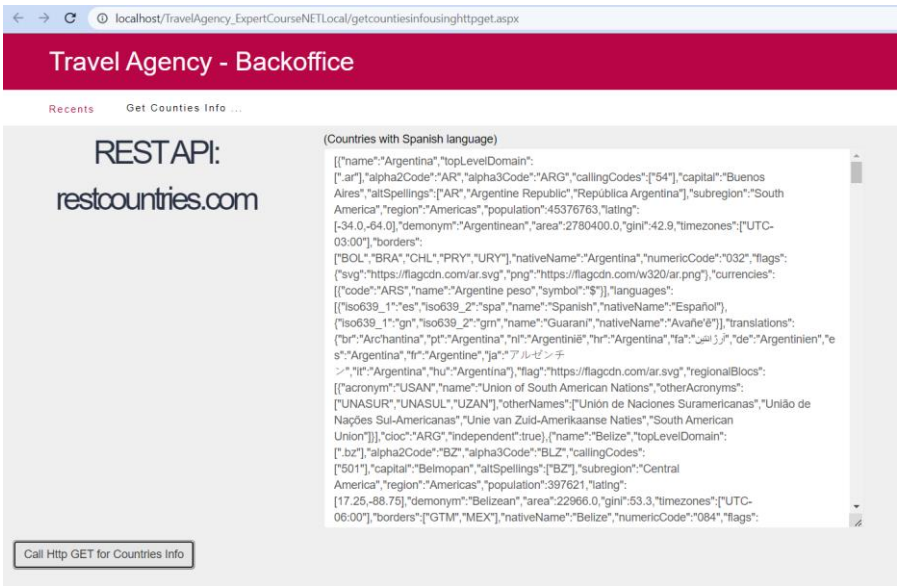
Para invocar o web service, primeiro criamos a variável e depois atribuímos as propriedades: Host, Port, Secure e BaseUrl.
Em seguida, adicionamos o header do tipo JSON e invocamos o método Execute, passando o método que queremos usar e os parâmetros requeridos pelo serviço, neste caso estamos passando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



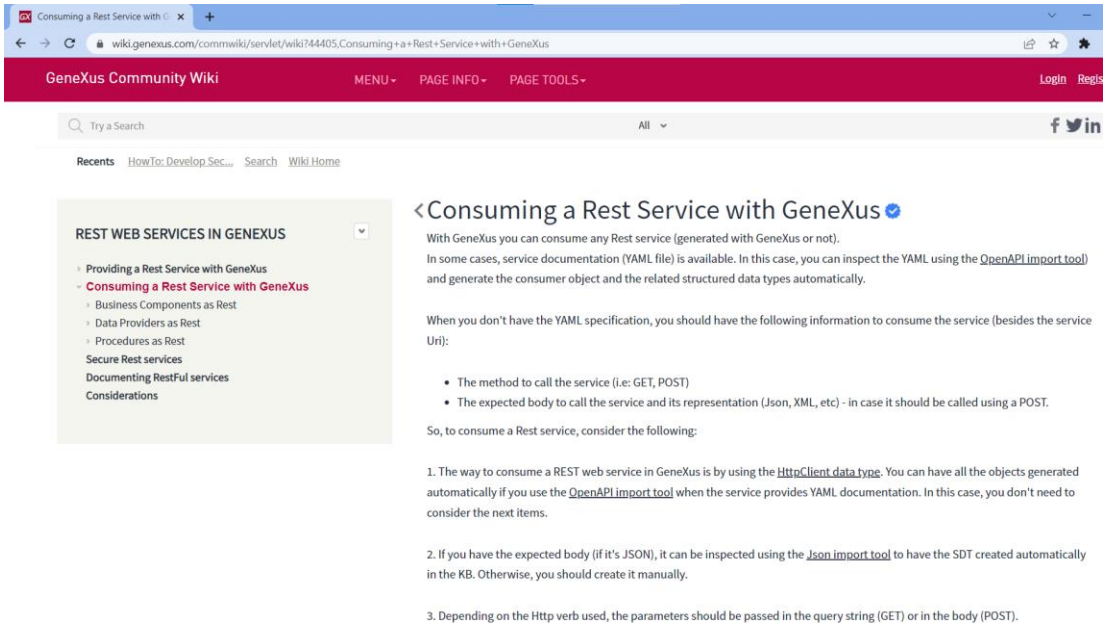

Vamos executar este exemplo no GeneXus.

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.

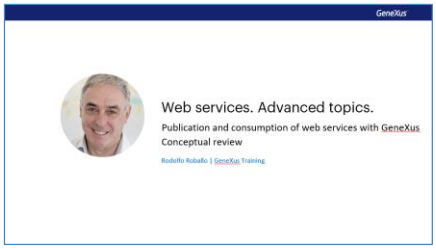
No evento do botão, vemos o código que já explicamos.

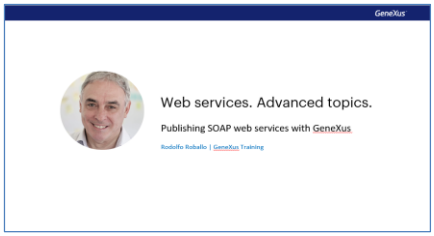
Executamos o web panel que é main… pressionamos o botão e vemos que recebemos a informação dos países de língua espanhola, exatamente como queríamos.

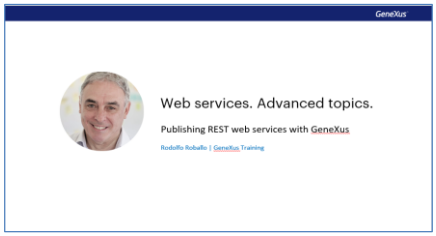
Customizing the consumption of a REST web service

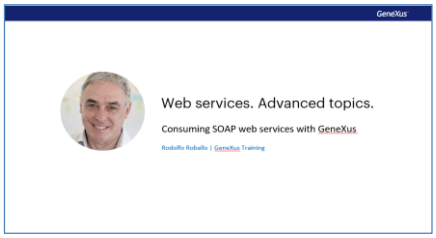


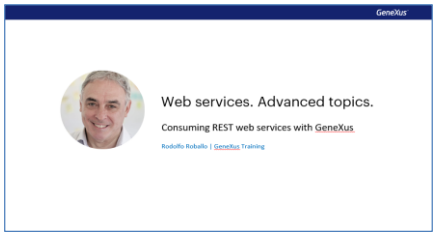

Para mais informações, consulte o artigo da wiki: “Consuming a Rest Service with GeneXus”.




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