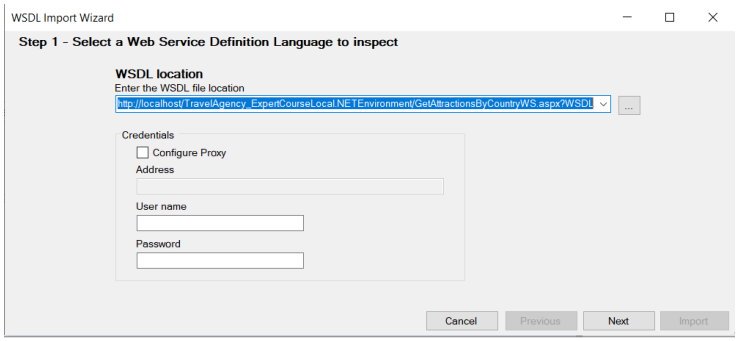
Web Services. Aspectos avançados.

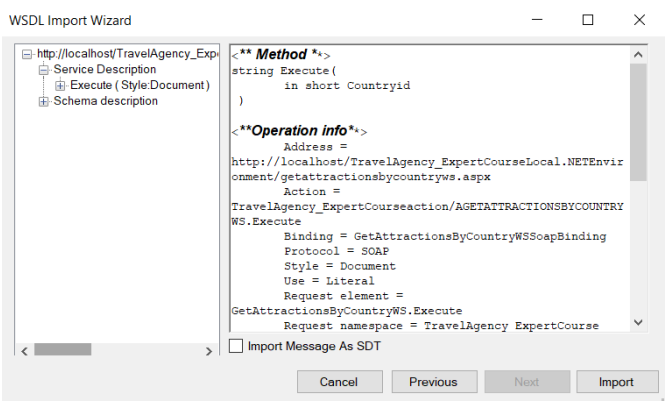
Consumindo serviços SOAP com GeneXus

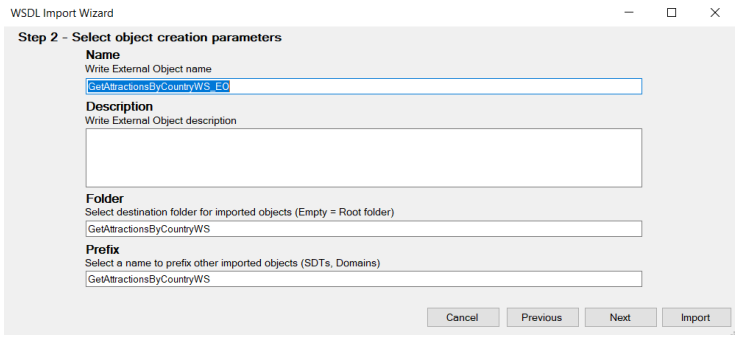
GeneXus™

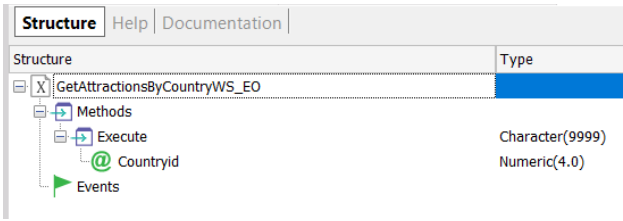
Vejamos agora como podemos testar um serviço SOAP publicado com GeneXus, a partir do próprio GeneXus, importando-o como objeto externo.
Em seguida, vamos nos concentrar em alguns conceitos mais avançados que flexibilizam o consumo deste tipo de serviços.

Testing the procedure published as SOAP web service with GeneXus









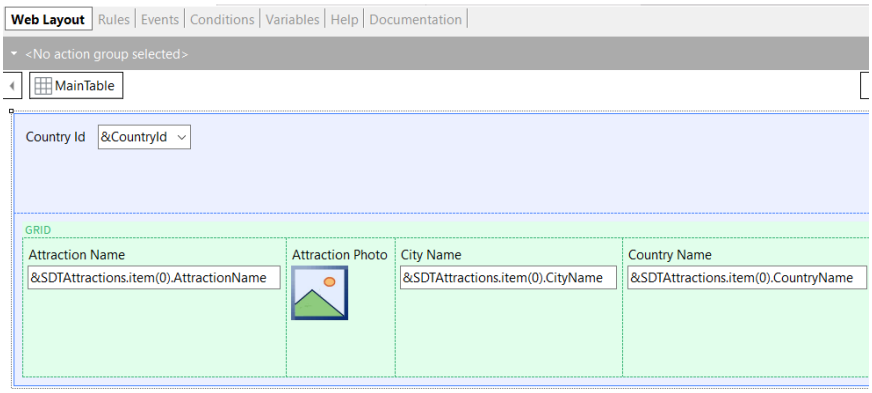
Para importar o serviço SOAP que publicamos, vamos para Tools /Application integration / WSDL Import e escrevemos o mesmo WSDL que testamos antes. [http://localhost/TravelAgency_ExpertCourseNETLocal/GetAttractionsByCountryWS.aspx?WSDL]

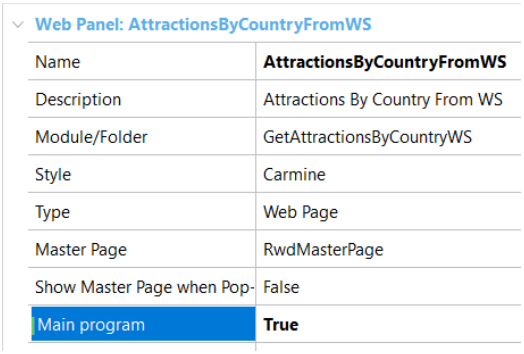
Na etapa 2 do wizard, escrevemos o nome de um folder de destino e pressionamos Next.

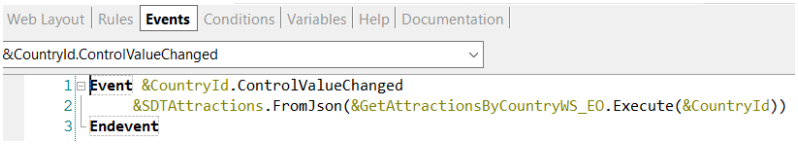
Na etapa 3, vemos a estrutura do serviço com dois nós. Se abrirmos o nó Service Description vemos que existe um único método chamado Execute, e que se clicarmos sobre ele, vemos na tela da direita que recebe por parâmetro o identificador do país CountryId.

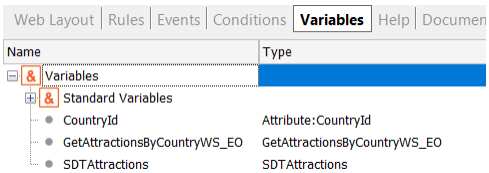
Se formos ao KB Explorer, vemos que foi criado o folder que definimos e, se o abrimos, vemos que aparece o objeto externo. Se abrirmos o objeto, podemos ver que tem definido um único método chamado Execute e que recebe por parâmetro o CountryId, assim como o programamos no procedimento.

Testing the procedure published as SOAP web service with GeneXus







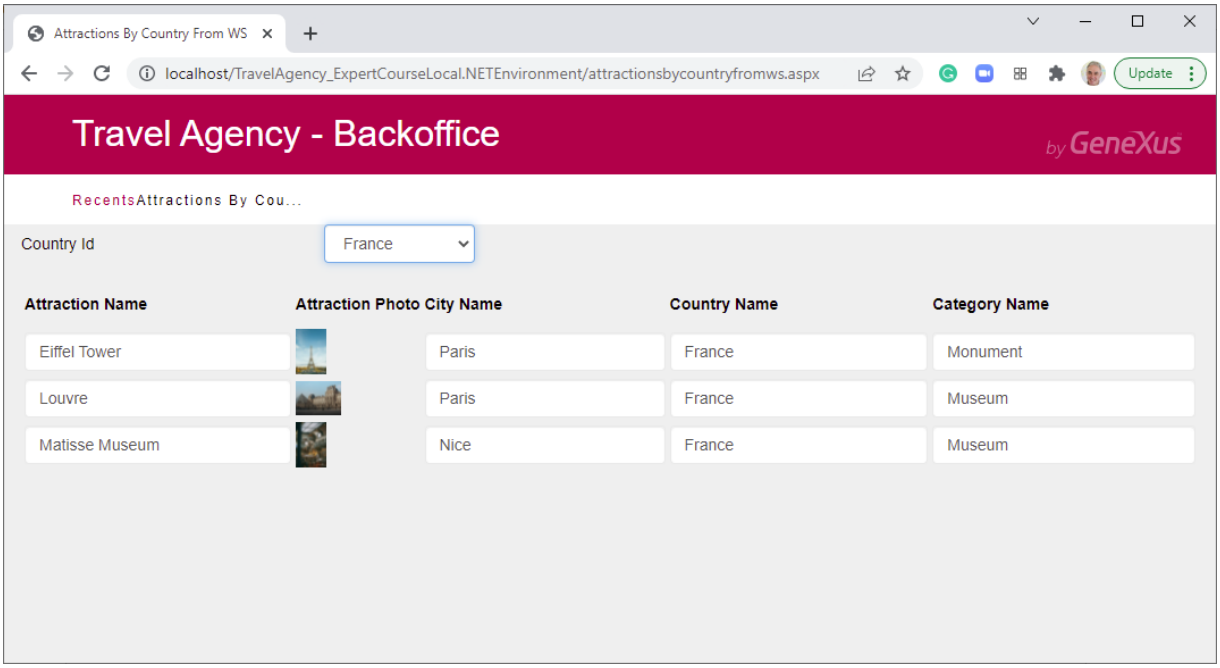


Agora, para testar o web service que publicamos, criamos um web panel chamado AttractionsByCountryFromWS que definimos como main e em sua seção variables definimos a variável &CountryId, e uma variável SDTAttractions que fica baseada no SDT coleção de mesmo nome. Também criamos **uma variável com o mesmo nome do objeto externo para que fique desse tipo.**

A variável &CountryId no form definimos como Dynamic Combobox com a propriedade Item Descriptions como CountryName e a propriedade Empty Item como True.

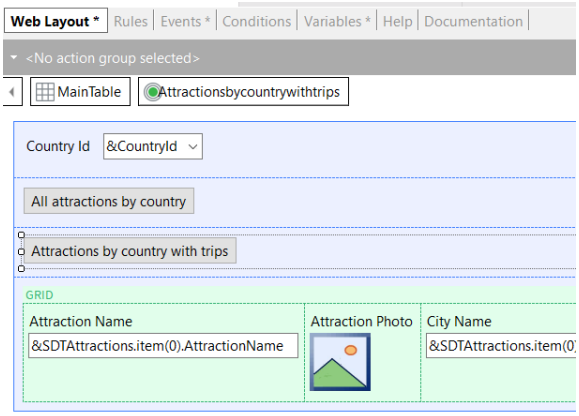
Nos eventos, programamos o evento ControlValueChanged do Dynamic combobox, de forma que ao escolher um país, seja disparado o evento. Dentro do evento escrevemos a invocação ao web service usando a variável &GetAttractionsByCountryWS, ponto, Execute e passando como parâmetro o identificador de país.
E esta invocação colocamos dentro dos parênteses do método FromJson da variável &SDTAttractions, que conterá a coleção de atrações retornadas por nosso web service.

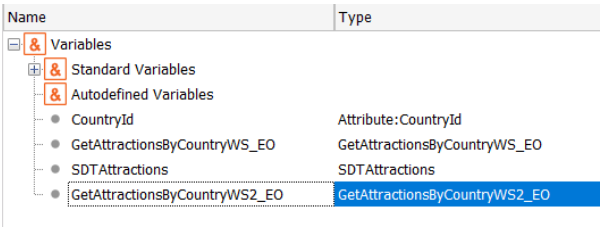
Testing the procedure published as SOAP web service with GeneXus

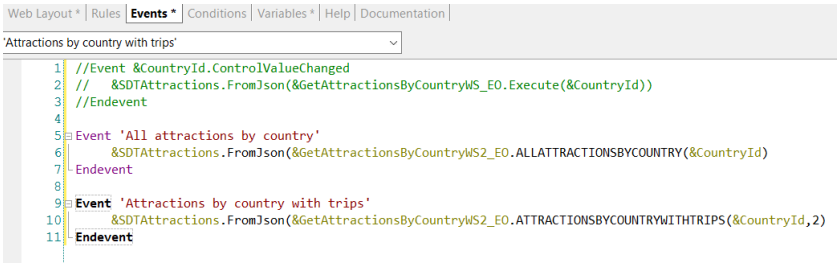


Se executamos o web panel, vemos que ao escolher um país, podemos ver todas as atrações turísticas daquele país, retornadas por nosso web service, que acessou a base de dados para obtê-las.

Web service SOAP with more than one method







Vamos testar também o serviço que publicamos com dois métodos.

Para invocar o serviço, fizemos uma cópia do web panel que criamos. Adicionamos dois botões com as captions “All attractions by country” e “Attractions by country with trips”.

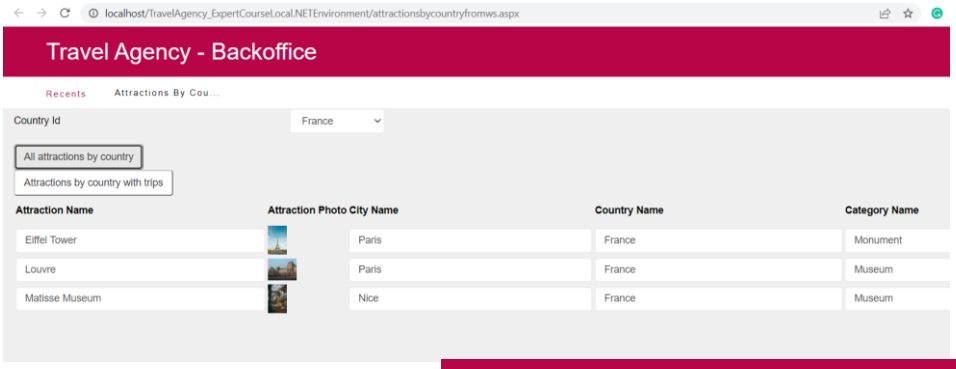
Criamos a variável com o mesmo nome do objeto externo e comentamos o evento ControlValueChanged.

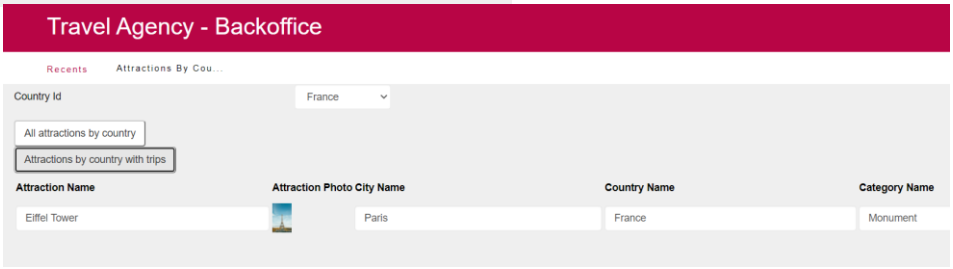
No evento do primeiro botão, realizamos uma invocação semelhante à que tínhamos, usando a nova variável e invocando o método AllAttractionsByCountry, passando como parâmetro o CountryId.

Fazemos o mesmo para o evento do segundo botão, invocamos AttractionsByCountryWithTrips, passando como parâmetro o CounryId e o valor 2, para que recupere apenas as atrações do país selecionado que tenham pelo menos 2 viagens.

Executamos o web panel porque é main.

Web service SOAP with more than one method



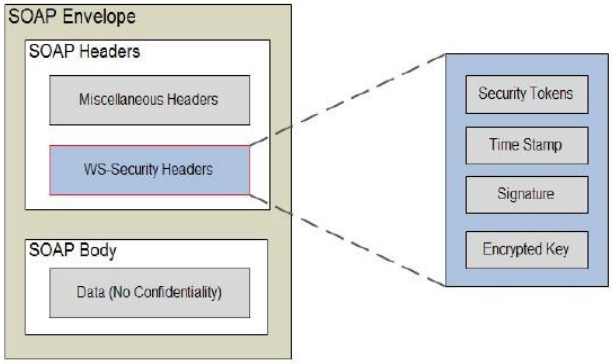


Selecionamos o país France, pressionamos o primeiro botão e vemos que nos traz todas as atrações da França e se pressionamos o segundo botão vemos que nos traz a atração Torre Eiffel e o Museu Matisse, que são as únicas que possuem mais de uma viagem registrada.

Se abrimos o Work With Trips, verificamos que apenas estas atrações da França possuem mais de uma viagem registrada.

Consuming secure SOAP web services with GeneXus: WS-SECURITY

WS-SECURITY (WSS)



Typical Message Flow with WS-Security [WSS2]

Um tema necessário a ser considerado nos serviços web é a questão da segurança da informação transmitida.

Em geral, quando é requerido que a estrutura dos dados e as operações não alterem e se requer a autenticação ou sessões, é utilizado o protocolo SOAP.

Para acrescentar segurança à mensagem SOAP, muitos provedores de serviços utilizam o protocolo WSS (WS-Security), que adiciona uma assinatura digital, criptografia e métodos de autenticação, que garantem a integridade da mensagem (ou seja, que a mensagem não foi modificada durante a transmissão), sua confidencialidade (que a mensagem não seja vista por terceiros) e que a autenticação seja segura (que não sejam violadas as credenciais da mensagem).

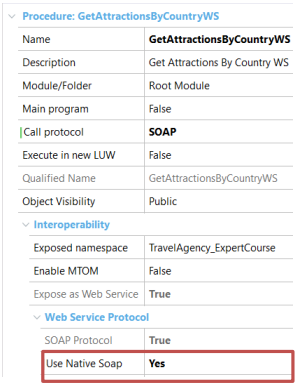
Esta segurança é adicionada no nível do servidor web, através de ferramentas de terceiros específicas.

O mecanismo é baseado em modificar as mensagens SOAP, são incluídos cabeçalhos SOAP específicos e é modificado o corpo SOAP com informação que é utilizada para garantir a integridade e confidencialidade. Estas modificações envolvem:

- Adicionar um token de segurança que especifique as credenciais do autor da mensagem.
- Adicionar uma descrição de como é assinada a mensagem (chave utilizada, algoritmo, qual parte da mensagem é assinada) e sua assinatura.
- Adicionar uma descrição de como é criptografada a mensagem (chave de

criptografia, algoritmo e qual parte da mensagem está criptografada).

Consuming secure SOAP web services with GeneXus: WS-SECURITY



Procedure: GetAttractionsByCountryWS

Name	**GetAttractionsByCountryWS**
Description	Get Attractions By Country WS
Module/Folder	Root Module
Main program	False
Call protocol	**SOAP**
Execute in new LUW	False
Qualified Name	GetAttractionsByCountryWS
Object Visibility	Public
Interoperability	
Exposed namespace	TravelAgency_ExpertCourse
Enable MTOM	False
Expose as Web Service	**True**
Web Service Protocol	
SOAP Protocol	**True**
Use Native Soap	**Yes**

GeneXus WSSecurity Data Type:

https://wiki.genexus.com/commwiki/servlet/wiki?44552

GeneXus WSSecurity Data Type properties:

- WSSecurity
- WSSignature
- WSEncryption
- WSSecurityKeyStore

EXAMPLE:

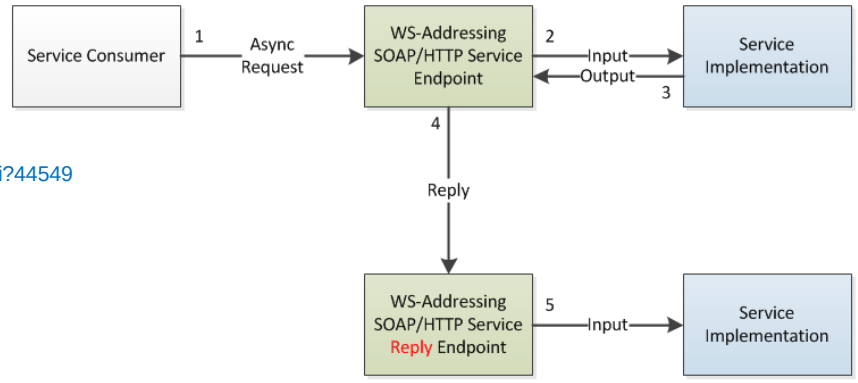
```
//Encryption
&WsSecurityKeyStore.Password = "prueba123"
&WsSecurityKeyStore.Type = WSSecurityKeyStore.JKS
&WsSecurityKeyStore.Source = "C:\temp\keystoreprueba.jks"

&wsEncryption.Alias = "epagos"
&wsEncryption.keyIdentifierType = WsSecurity.BINARY_SECURITY_TOKEN
&wsEncryption.Keystore = &WsSecurityKeyStore

&wssecurity.Encryption = &wsEncryption

&wssecurity.ExpirationTimeout = 5

//Signature
&wssecurity.Signature.Alias = "alias1"
&wssecurity.Signature.keyIdentifierType = WsSecurity.BINARY_SECURITY_TOKEN
&wssecurity.Signature.Keystore.Password = "PasswordAlias1"
&wssecurity.Signature.Keystore.Type = WSSecurityKeyStore.JKS
&wssecurity.Signature.Keystore.Source = "C:\temp\prueba2.jks"

&wssecurity.Signature.CanonicalizationAlgorithm = "CanonicalizationMethod.EXCLUSIVE"
&wssecurity.Signature.SignatureAlgorithm = "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"
&wssecurity.Signature.Digest = "DigestMethod.SHA256"

&location.WSSecurity = &wssecurity
```

Com GeneXus não publicamos web services com WS-Security, mas podemos consumir serviços de terceiros que cumpram o referido protocolo, usando funções nativas fornecidas por .NET ou Java, que habilitamos com a propriedade Use Native Soap.

O protocolo SOAP utiliza o padrão XML para a definição das mensagens de request e response e a propriedade Use Native Soap determina a forma em que serão gerados os XMLs. Se colocado como Yes, para a serialização do XML, são utilizadas funcionalidades fornecidas pela linguagem da plataforma, ou seja, que serão instruções do framework .Net ou da linguagem Java que resolvem a geração do XML em baixo nível. Se esta propriedade estiver como No (que é o valor padrão), GeneXus se encarregará de gerar os XMLs que usa o serviço.

GeneXus também nos oferece tipos de dados especiais para consumir web services WS-Security e também constantes predefinidas que facilitam o uso destes tipos de dados.

Tracking and routing SOAP web services with GeneXus: WS-ADDRESSING



GeneXus WSAddressing Data Type:

https://wiki.genexus.com/commwiki/servlet/wiki?44549

WSAddressing Properties

To	Character
Action	Character
MessageID	Character
From	WSAddressingEndPoint
ReplyTo	WSAddressingEndPoint
FaultTo	WSAddressingEndPoint

WSAddressingEndPoint Properties

Address	Character
PortType	Character
ServiceName	Character
Properties	Character
Parameters	Character

EXAMPLE:

```
&location = getLocation("EObjectName")
&wsaddressing.Action = "urn:antel:mdm:system:epagos:b2b:comercio:iniciarSolicitud"
&wsaddressing.MessageID = "uuid:e5403230-9bab-4152-a2ba-e4e47165f135"
&wsaddressing.To = "urn:antel:mdm:system:epagos"
&wsaddressing.From = &wsaddressingendpoint
&wsaddressing.ReplyTo = &otherwsadressingendpoint
&location.WSAddressing = &wsaddressing
```

Em determinadas aplicações, é muito útil que quem consumir um serviço publicado por nós possa enviar uma resposta ao servidor que publica nosso serviço.

GeneXus permite rastrear e direcionar web services SOAP através do WSAddressing, habilitando também a propriedade Use Native Soap.

WS-Addressing é um protocolo da World Wide Web Consortium (W3C) que especifica como um consumidor de serviços pode indicar o ponto de extremidade para o qual o serviço deve enviar sua resposta (e/ou enviar falhas SOAP), em uma invocação.

O cliente envia uma solicitação SOAP ao serviço e a infraestrutura do serviço sabe que deve tratar a solicitação de forma assíncrona (para que o cliente não espere pela resposta). A implementação do serviço gera uma resposta e a retorna ao componente de serviço que interpreta os cabeçalhos de WS-Addressing e então encaminha automaticamente o envelope SOAP de resposta para o URI mencionado no cabeçalho ReplyTo.

Isto é feito por meio de duas estruturas: Headers e EndPoint References.

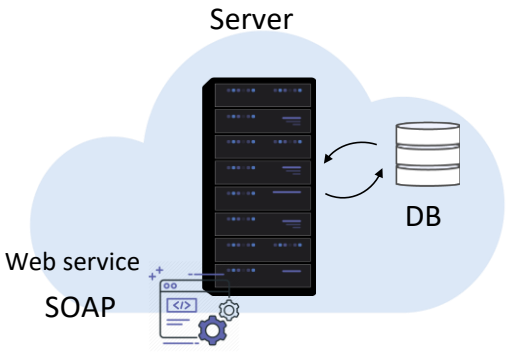
O poder do WS-Addressing é que o consumidor do serviço pode especificar para qual ponto de extremidade deve ser enviada a resposta. Não está codificado na implementação do serviço, é especificado pelo consumidor.

Na wiki encontraremos informação sobre o tipo de dados WSAddressing usado por GeneXus.

Location of a SOAP web service

LOCATION

- Host
- BaseUrl
- Port
- Secure
- Resource Name
- AuthenticationUser
- AutheticationPassword
- Timeout
- CancelOnError



Vejamos agora o que é a location de um web service.

Os web services são executados em um servidor web, portanto, cada web service é identificado por uma URL, de um determinado host e em uma porta determinada. Também possui outras características ao ser executado, como por exemplo a segurança que usa, o método de autenticação, as credenciais, etc.

Estes valores são atribuídos por padrão pelo GeneXus quando publicamos um web service e no caso em que o web service seja de um terceiro, o que fornece o serviço atribui estas características.

Este conjunto de propriedades que determinam onde e como é executado um serviço remoto, é denominada “location”.

Quando invocamos um web service do tipo SOAP, é possível alterar sua location e atribuir outros valores diferentes dos estabelecidos por padrão. Isto é especialmente útil quando, por exemplo, estamos executando um serviço em um servidor de desenvolvimento e queremos executá-lo em um servidor de produção. Embora o código que é executado seja o mesmo, ao alterar o endereço do host e outras propriedades, passa a ser considerado outro serviço diferente do primeiro.

Podemos alterar a location de um web service, em tempo de geração (para alterar os valores padrão atribuídos pelo GeneXus) ou em tempo de execução, por código, quando consumimos o web service.

Location of a SOAP web service

GetAttractionsByCountryWS

Source | Layout | **Rules** | Conditions | Variables | Help | Document

```
1  Parm(in:&CountryId, out:&Attractions);
```

GetAttractionsByCountryWS

Source | Layout | Rules | Conditions | Variables | Help | Documentation

Subroutines

```
1  For each Attraction
2        Where CountryId = &CountryId
3        &OneAttraction.AttractionName = AttractionName
4        &OneAttraction.AttractionPhoto = AttractionPhoto
5        &OneAttraction.CategoryName = CategoryName
6        &OneAttraction.CityName = CityName
7        &OneAttraction.CountryName = CountryName
8        &SDTAttractions.Add(&OneAttraction)
9        &OneAttraction = New()
10 Endfor
11 &Attractions = &SDTAttractions.ToJson()
```

AttractionsByCountryFromWS *

Web Layout | Rules | **Events *** | Conditions | Variables | Help | Documentation

Events

```
1  Event &CountryId.ControlValueChanged
2        //Service behaviour setting
3        &Location = GetLocation("GetAttractionsByCountryWS_EO")
4        &Location.Host = "www.servername.com"
5        &Location.BaseUrl = "/base URL/"
6        &Location.Port = 123456
7        &Location.Secure = 1 or 2
8        &Location.ResourceName = "ServiceName"
9        &Location.CancelOnError = 2 //Do not stop execution
10
11       //Web service invocation
12       &SDTAttractions.FromJson(&GetAttractionsByCountryWS_EO.Execute(&CountryId))
13       if GetSOAPErr() > 0
14            msg("Error: " + GetSOAPErrMsg())
15       endif
16 Endevent
```

Vamos voltar ao exemplo do procedimento GetAttractionsByCountryWS que publicamos como web service SOAP para obter os dados de atrações por país e que então consumimos a partir do web panel AttractionsByCountryFromWS, para testá-lo.

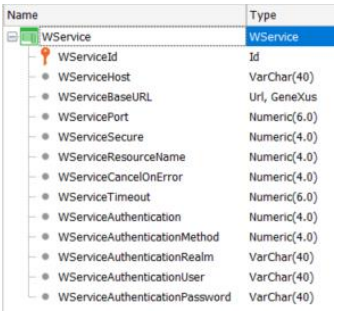
Nos eventos do webpanel onde mostramos os dados retornados pelo serviço, podemos obter a location do web service invocado atribuindo uma variável Location, do tipo Location, com o que é retornado pelo método GetLocation ao qual passamos como parâmetro o nome do External Object.

Isto nos permite atribuir diferentes valores às propriedades da location, por exemplo, alterar o host, a URL base, a porta, o nome do recurso e outros dados do serviço. Isto é muito útil quando temos um serviço executando em um servidor e então queremos que o serviço seja executado em outro servidor, por exemplo, quando passamos do servidor de desenvolvimento para o servidor de produção ou quando queremos reutilizar um serviço em outra aplicação e queremos que seja executado em outro server.

Observemos que após executar o web service, utilizamos os métodos GetSoapErr() e GetSoapErrMsg para obter informações sobre o resultado da execução. Aprofundaremos nestes conceitos mais adiante.

Customizing the location of a SOAP web service

Assigning Location variable from database



Name	Type
WService	WService
WServiceId	Id
WServiceHost	VarChar(40)
WServiceBaseURL	Url, GeneXus
WServicePort	Numeric(6.0)
WServiceSecure	Numeric(4.0)
WServiceResourceName	Numeric(4.0)
WServiceCancelOnError	Numeric(4.0)
WServiceTimeout	Numeric(6.0)
WServiceAuthentication	Numeric(4.0)
WServiceAuthenticationMethod	Numeric(4.0)
WServiceAuthenticationRealm	VarChar(40)
WServiceAuthenticationUser	VarChar(40)
WServiceAuthenticationPassword	VarChar(40)

```
Event &CountryId.ControlValueChanged
    //Service behaviour setting
    &Location = GetLocation("GetAttractionsByCountryWS_EO")
    For each WService
        Where WServiceId = &WSServiceId
        &Location.Host = WServiceHost
        &Location.BaseUrl = WServiceBaseURL
        &Location.Port =WServicePort
        &Location.Secure = WServiceSecure
        &Location.ResourceName = WServiceResourceName
        &Location.CancelOnError = WServiceCancelOnError
        Exit
    Endfor
    //Web service invocation
    &SDTAttractions.FromJson(&GetAttractionsByCountryWS_EO.Execute(&CountryId))
    if GetSOAPErr() > 0
        msg("Error: " + GetSOAPErrMsg())
    endif
Endevent
```

Location.xml

```
<GXLocations>
 <GXLocation name="GetAttractionsByCountryWS_EO"> // Name of the External Object
  <Common>
   <Host>"www.servername.com"</Host>   // Don't include protocol (i.e. HTTP/HTTPS)
   <Port>443</Port> // Port number
   <BaseUrl>/services/</BaseUrl>   // Start and end with a bar: /baseurl/
   <Secure>1</Secure> // 1 = HTTPS, 0 = HTTP
   <Proxyserverhost/>
   <Proxyserverport/>
   <Timeout/>
  </Common>
  <HTTP>
   <Authentication>
    <Authenticationmethod/>
    <Authenticationuser/>
    <Authenticationpassword/>
    <Authenticationrealm/>
   </Authentication>
   <Proxyauthentication>
    <Proxyauthenticationmethod/>
    <Proxyauthenticationrealm/>
    <Proxyauthenticationuser/>
    <Proxyauthenticationpassword/>
   </Proxyauthentication>s
  </HTTP>
 </GXLocation>
</GXLocations>
```

Os dados que atribuímos à location de um serviço podem ser parametrizados na base de dados. No exemplo, vemos que criamos uma transação WService e depois com um comando For Each acessamos a tabela que contém os valores dos atributos para atribuir as propriedades da location.

Além de definir uma variável do tipo de dados Location e alterar os valores de suas propriedades por código como acabamos de ver, outra forma de atribuir os valores de uma location é utilizando um arquivo Location.xml que colocamos na pasta web de nosso target environment, por exemplo, a pasta Web em um modelo .NET ou web-inf em Java.

Quando GeneXus compila a aplicação, se encontra um arquivo chamado location.xml na pasta mencionada anteriormente, ele o lê e atribui os valores da location dinamicamente em tempo de execução.

O formato do arquivo location.xml é o que é exibido em tela.

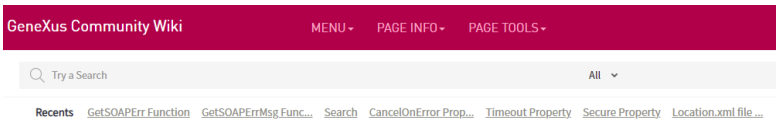
Em GXLocation Name atribuímos o nome do objeto externo associado ao nosso web service.
Host é o nome do servidor que hospedará o serviço, sem incluir o http ou https, ou seja, por exemplo, apenas www.servername.com
A porta é um valor numérico que depende do tipo de servidor, a url base se recomenda que seja incluída entre barras, por exemplo /services/.
Secure com o valor 1 indica que será utilizado um servidor seguro, ou seja, com o protocolo HTTPS e o valor 0 para HTTP.
Proxyserverhost e Proxyserverport são utilizados no caso de haver um servidor

Proxy entre o cliente e o servidor que fornece o serviço.
Timeout é atribuído com o tempo máximo em segundos que o sistema deve aguardar por uma resposta a ser enviada ao servidor após cada request. Se for atribuído um valor 0, significa que o tempo de espera é indefinido.
Então, se necessário, podem ser definidas as características de autenticação da comunicação HTTP.

As propriedades atribuídas em tempo de execução a uma variável do tipo de dados Location terão preferência sobre aquelas atribuídas em tempo de execução por meio de um arquivo location.xml, e estas últimas, por sua vez, terão preferência sobre aquelas atribuídas em tempo de geração. Isto permite mais dinamismo na configuração das localizações.

Errors handling of a SOAP web service invocation

```
Event &CountryId.ControlValueChanged
    //Service behaviour setting
    &Location = GetLocation("GetAttractionsByCountryWS_EO")
    &Location.CancelOnError = 2 //No stop execution on error

    //Web service invocation
    &SDTAttractions.FromJson(&GetAttractionsByCountryWS_EO.Execute(&CountryId))

    //Error handling
    &error = GetSOAPErr()
    If &error <> 0 Or null(&location.Host)
        Do Case
        Case &error = -20007
            &errorMsg = "Unknown error to set a web service, unknown location:" + &Location.ResourceName + newline() + GetSOAPErrMsg()
        Case &error > 0
            &errorMsg = "Unknown error to set a web service:" + &Location.ResourceName + newline() + GetSOAPErrMsg()
        Otherwise
            &errorMsg = "Error from unknown host to set a web service:" + &Location.ResourceName
        EndCase
    EndIf
Endevent
```



Quando invocamos um serviço SOAP e atribuímos valores a uma variável location, podemos atribuir um valor à propriedade CancelOnError, que nos permite obter o resultado da operação e tratar os erros. Aqui temos um exemplo de uso típico.

Dependendo do valor atribuído à propriedade CancelOnError, será o comportamento do serviço.

Por exemplo, se atribuirmos o valor 0, o programa que invoca o web service cancelará sempre a execução ao final da invocação. Este é o valor padrão.
Se atribuirmos o valor 1, o programa chamador cancelará a execução em caso de erro.
Se atribuirmos o valor 2, o programa chamador NÃO cancelará a execução em caso de um erro e para obter informação do erro podemos usar as funções GetSOAPErr() e GetSOAPErrMsg().

A função GetSOAPErr() retorna um valor numérico correspondente ao código de erro da última operação SOAP.
Se a função retorna 0, não ocorreu nenhum erro. Caso contrário, o código depende do tipo de erro e ele varia se o erro ocorreu no cliente ou no servidor.

A função GetSOAPErrMsg() retorna uma descrição do erro da última operação SOAP, o que nos permite dar uma mensagem mais amigável ao usuário.

No artigo da Wiki “Error Codes and Messages for Location Data Type” você encontrará uma tabela dos códigos de erros possíveis, tanto do cliente quanto do servidor.

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