Web Services. Aspectos avançados.

Publicando serviços SOAP com Genexus.

GeneXus™

Neste vídeo vamos nos concentrar na publicação, teste e personalização de serviços SOAP com GeneXus, como por exemplo atribuir um namespace a ele, ou incluir mais de um método em um único web service.

Publishing a procedure as SOAP web service

GetAttractionsByCountryWS

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

Subroutines

```
For each Attraction
      Where CountryId = &CountryId
      &OneAttraction.AttractionName = AttractionName
      &OneAttraction.AttractionPhoto = AttractionPhoto
      &OneAttraction.CategoryName = CategoryName
      &OneAttraction.CityName = CityName
      &OneAttraction.CountryName = CountryName
      &SDTAttractions.Add(&OneAttraction)
      &OneAttraction = New()
Endfor
&Attractions = &SDTAttractions.ToJson()
```

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

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

Name	Type
& Variables	
& Standard Variables	
& Autodefined Variables	
Attractions	LongVarChar(2M)
OneAttraction	SDTAttractions.SDTAttractionsItem
SDTAttractions	SDTAttractions

Procedure: GetAttractionsByCountryWS	
Name	**GetAttractionsByCountryWS**
Description	Get Attractions By Country WS
Module/Folder	Root Module
Main program	False
Call protocol	Internal
Execute in new LUW	False
Qualified Name	GetAttractionsByCountryWS
Object Visibility	Public
Interoperability	
Enable MTOM	False
Expose as Web Service	**True**
Web Service Protocol	
SOAP Protocol	True
Use Native Soap	Use Environment property value
Web Service schema vali	Use Environment property value

Name	Type	Is Collection
SDTAttractions		☑
SDTAttractionsItem		
AttractionName	Attribute:AttractionName	☐
AttractionPhoto	Attribute:AttractionPhoto	☐
CityName	Attribute:CityName	☐
CountryName	Attribute:CountryName	☐
CategoryName	Attribute:CategoryName	☐

Name	Type
Attraction	Attraction
AttractionId	Id
AttractionName	Name
AttractionAddress	Address, GeneXus
AttractionPhoto	Image
CityId	Id
CityName	Name
CountryId	Id
CountryName	Name
CategoryId	Id
CategoryName	Name

Name	Type
Country	Country
CountryId	Id
CountryName	Name
City	City
CityId	Id
CityName	Name

Name	Type
Category	Category
CategoryId	Id
CategoryName	Name

Vamos publicar um objeto procedimento como serviço.
Este serviço acessará a base de dados da aplicação de uma Agência de Viagens e retornará a coleção de atrações turísticas registradas pela agência, que pertencem a um determinado país.

Para isto criamos um objeto procedimento e o nomeamos GetAttractionsByCountryWS. Quando publicamos um web service, é importante escolher um nome que ajude a identificar qual função cumpre o serviço, de forma que fique claro para quem o consuma.

O procedimento recebe por parâmetro o identificador de país e retorna em uma string uma estrutura JSON com a lista de atrações que pertencem ao país recebido.

Para definir a estrutura criamos um SDT coleção com os dados das atrações que queremos retornar e na proc criamos uma variável SDTAttractions do tipo de dados do SDT coleção, uma variável OneAttraction do tipo do item da coleção e uma variável Attractions do tipo LongVarChar que conterá o JSON que retornará o procedimento.

No source, o For Each navega a tabela Attraction (associada à transação base Attraction) filtrada pelo país recebido por parâmetro, para cada atração encontrada carrega os valores em um item e então é adicionado o item à coleção. Por último, a variável SDT coleção é serializada em um JSON e é atribuída à variável do parâmetro de saída.

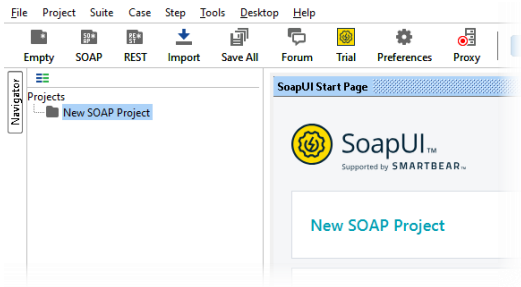
Para expor o procedimento GetAttractionsByCountryWS como serviço SOAP, colocamos a propriedade Expose as Web Service no valor True e definimos a propriedade SOAP Protocol como True e REST Protocol como False.

Para que o serviço seja publicado no servidor web, devemos fazer um Build.

Testing the procedure published as SOAP web service

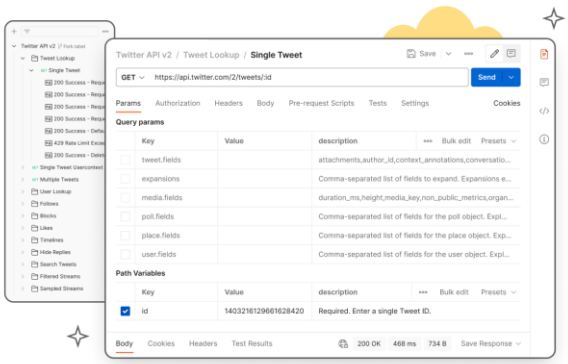
SoapUI
Supported by SMARTBEAR

https://www.soapui.org/



POSTMAN

https://www.postman.com/

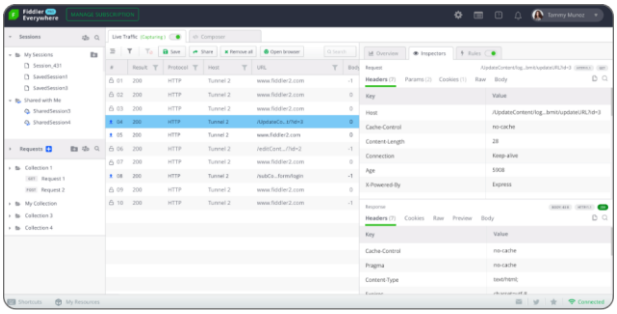


Agora vamos verificar se o procedimento ficou corretamente exposto como serviço SOAP.

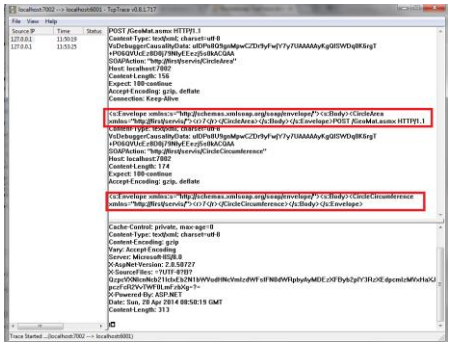
Para testá-lo, podemos usar várias ferramentas, como SOAP UI ou POSTMAN. Estas ferramentas permitem consumir o web service como se fossem clientes, para ver se o serviço funciona corretamente e obter informação detalhada sobre o processo, tanto para web services SOAP quanto REST.

Testing the procedure published as SOAP web service

Progress® Telerik® Fiddler™ https://www.telerik.com/fiddler



TcpTrace https://sourceforge.net/projects/open-tcptrace/

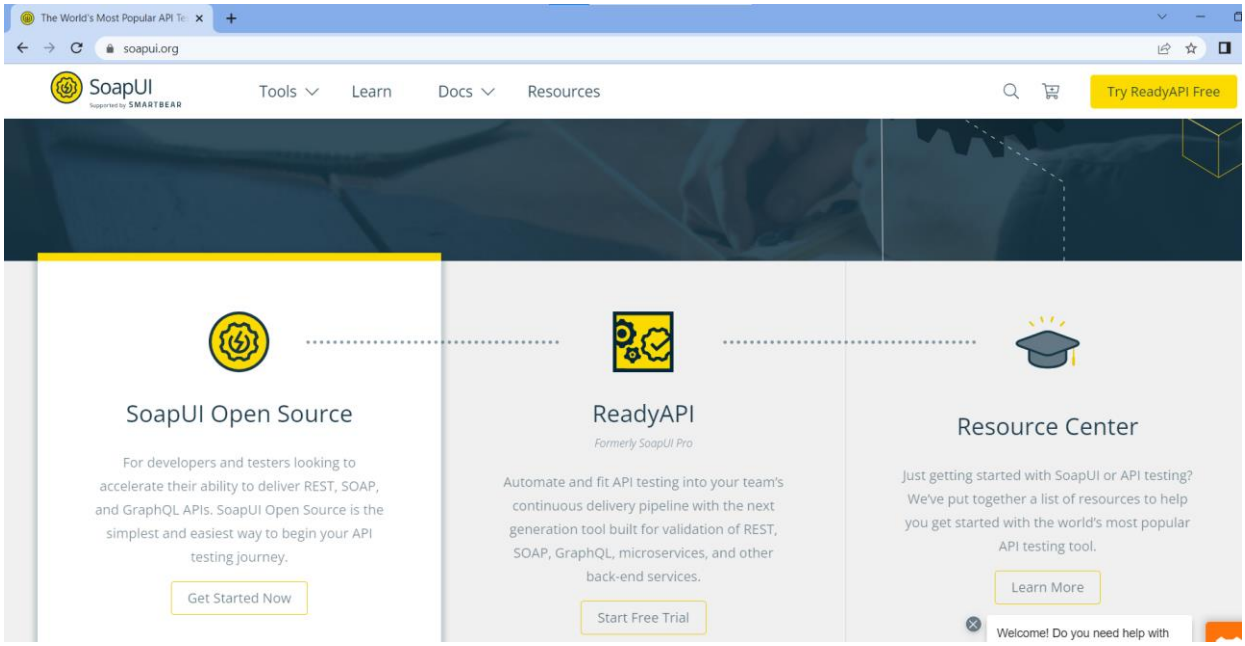


También é útil o uso de ferramentas que permitam visualizar o fluxo do serviço, ou seja, como foi a solicitação de execução ao servidor (Request) e como foi a resposta do serviço (Response).

Duas das mais conhecidas são Fiddler e TcpTrace. TcpTrace é gratuita (open source), mas permite apenas fluxo http, para https deve ser usada Fiddler.

Estas ferramentas ficam no meio entre a aplicação que consome e o servidor web, como se fosse um proxy, e exibem o Request e o Response entre o cliente e o serviço.

Installing SoapUI Open Source

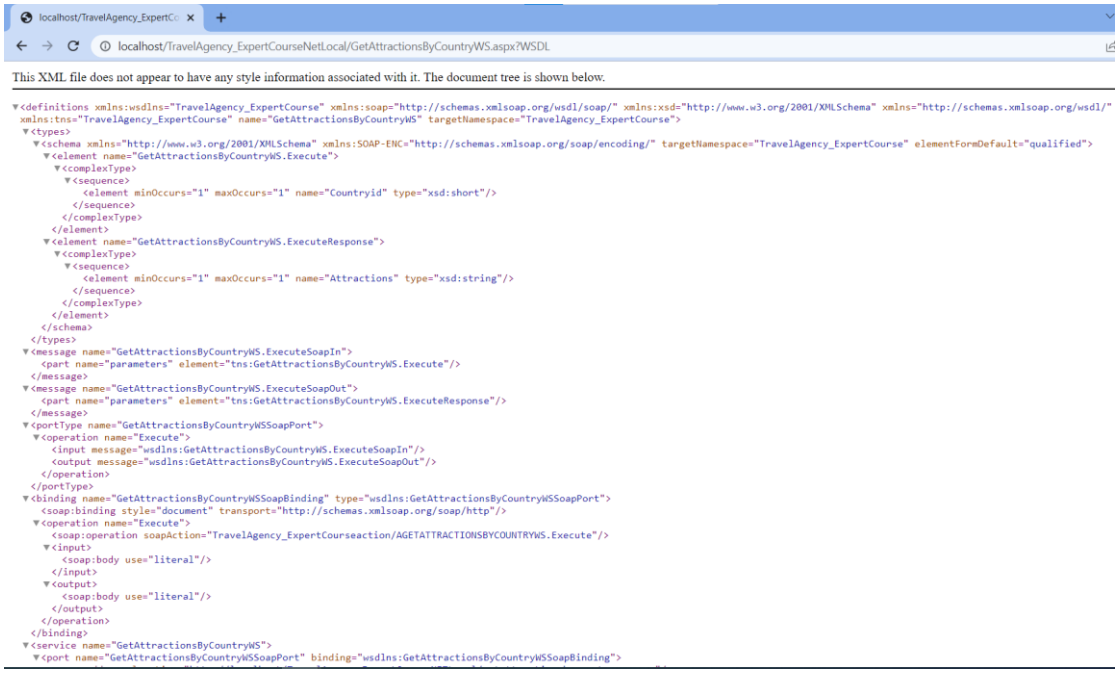


Vamos utilizar a ferramenta SoapUI para verificar nosso serviço. Abrimos a página web soapui.org e baixamos SoapUI OpenSource.
A instalamos e quando executamos aparece a Start Page. Se em Resources clicarmos em Test a SOAP API, é aberta uma página com instruções, então vamos segui-las.

Criamos um Projeto SOAP novo, no nome do Projeto colocamos GetAttractionsByCountry e pressionamos OK.
Agora clicamos com o botão direito sobre o Projeto e escolhemos Add WSDL.

Este arquivo WSDL foi gerado pelo GeneXus seguindo a especificação Web Services Description Language e contém a informação sobre como está estruturado nosso web service, como por exemplo os métodos que tem, os parâmetros de cada método, etc.

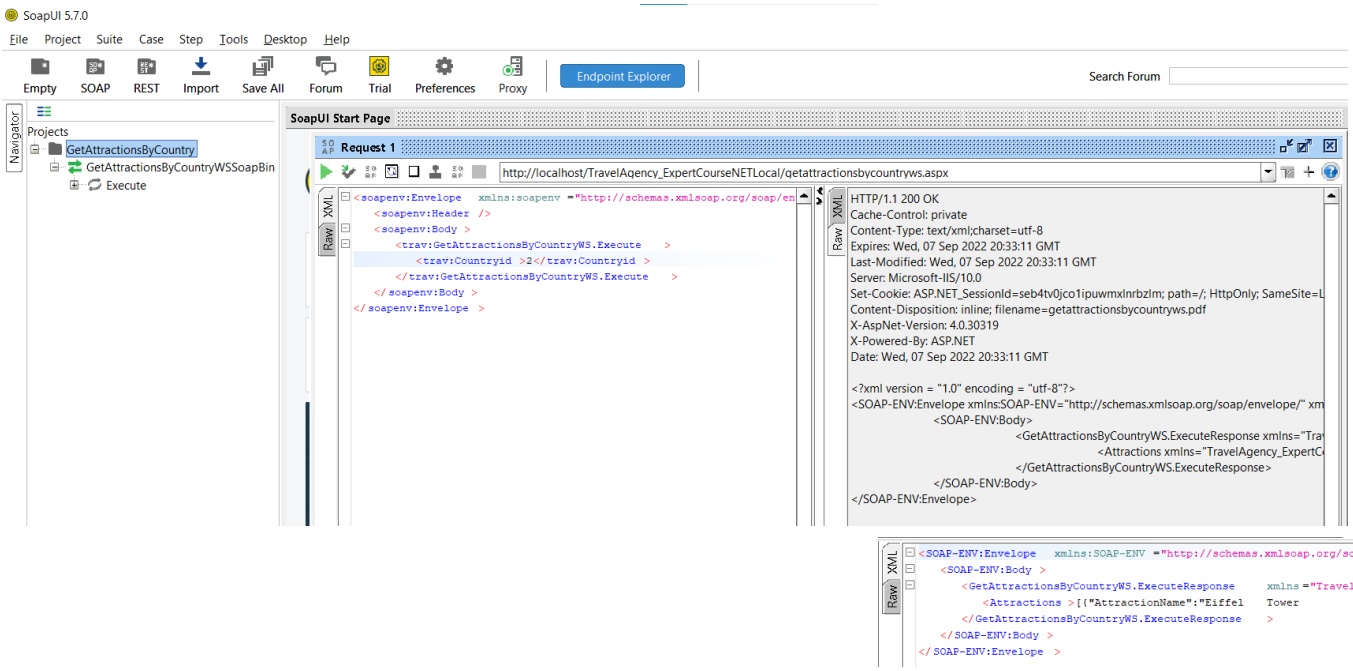
Using de browser to discover WSDL structure



Para ver o conteúdo deste arquivo, abrimos uma janela do browser e digitamos a URL do nosso serviço, que como estamos gerando em .NET, digitamos a URL que formamos concatenando a URL da propriedade Web root do gerador, o nome do nosso procedimento exposto, depois .aspx, adicionando depois um ponto de interrogação e as letras WSDL. Se o gerador fosse Java, adicionaríamos /servlet antes do nome do objeto e não vai a extensão aspx.
Se o objeto exposto for um Business Component, o nome do BC estará seguido de "_BC" se o gerador for .NET e "_BC_WS" se o gerador for Java.
Pressionamos Enter e vemos que o browser nos mostra a estrutura de nosso web service, onde identificamos algumas coisas como o nome, o método Execute que requer o parâmetro CountryId, etc.

[http://localhost/TravelAgency_ExpertCourseNETLocal/GetAttractionsByCountryWS.aspx?WSDL]

Testing our web service in SoapUI



Agora que sabemos que o WSDL foi aberto corretamente, vamos inserir a mesma URL na janela do projeto SoapUi e pressionamos OK.

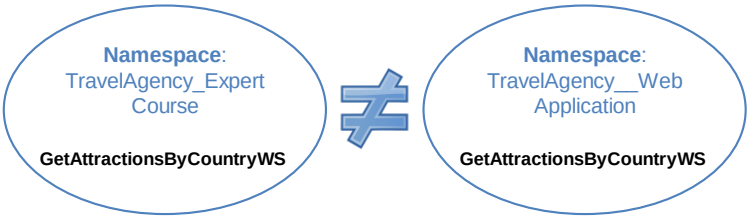
Vemos que sob o projeto que criamos, aparece uma entrada com o nome de nosso serviço e o método Execute. Se pressionarmos o botão + veremos que foi gerado automaticamente um Request para invocar o serviço. Clicamos duas vezes e vemos que é aberto o Request Editor, onde do lado esquerdo aparece um template do XML para a invocação e do lado direito aparecerá a resposta dada pelo serviço ao invocá-lo.

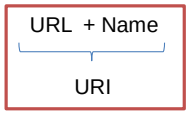
Na janela do request, no método Execute identificamos o parâmetro CountryId, então substituímos o ponto de interrogação pelo identificador do país do qual queremos obter informação de suas atrações. Escrevemos 2 que é o Id da França.

Agora pressionamos o botão Play e vemos que à direita apareceu a estrutura do response. Vemos que no nó Attractions, movendo para a direita, aparece entre colchetes a coleção das atrações turísticas da França.
Também podemos visualizar esta informação como XML e verificamos que nosso serviço GetAttractionsByCountry está funcionando perfeitamente.

Use of Namespaces in a SOAP web service

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	Use Environment property value
Web Service schema validation	Use Environment property value
REST Protocol	False





Vejamos agora o conceito de namespace e como isto pode ser útil na hora de publicar um serviço.

Um espaço de nomes é um contêiner de nomes onde o mesmo nome não pode ser repetido. No entanto, o mesmo nome pode estar presente em mais de um namespace.

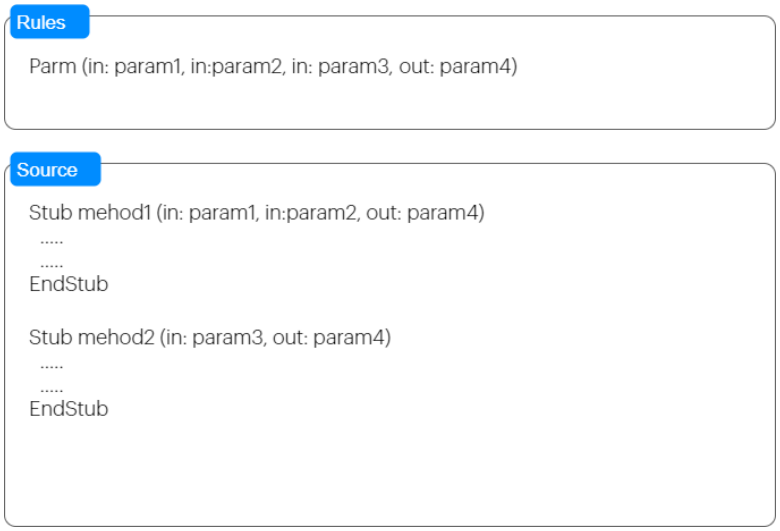
A propriedade **Exposed namespace** nos permite atribuir um espaço de nomes a um serviço.

Esta propriedade contém uma string que ajuda a identificar o web service. A combinação do namespace com o nome do web service deve ser única, de forma que se tivermos dois serviços com o mesmo nome, mas que pertençam a aplicações diferentes, alterando o namespace sejam corretamente identificados.

Por padrão, a propriedade Exposed namespace tem o nome da KB e isso não causa problemas se estivermos na etapa de prototipação, mas quando passamos o serviço para ambiente de produção, devemos escrever nesta propriedade uma URL que identifique a empresa ou o projeto ao qual pertence o serviço, caso contrário alguns consumidores não poderão processá-lo.

O fato de que este URI (Uniform Resource Identifier), composto pela URL e o nome do serviço, esteja bem formado e seja único, é de vital importância quando é adicionada segurança aos web services.

Procedure exposed as Web service SOAP, with more than one method



Stubs use: Only in SOAP web services

Uma coisa que pode surgir ao expor um serviço é como podemos incluir vários métodos dentro do mesmo serviço.

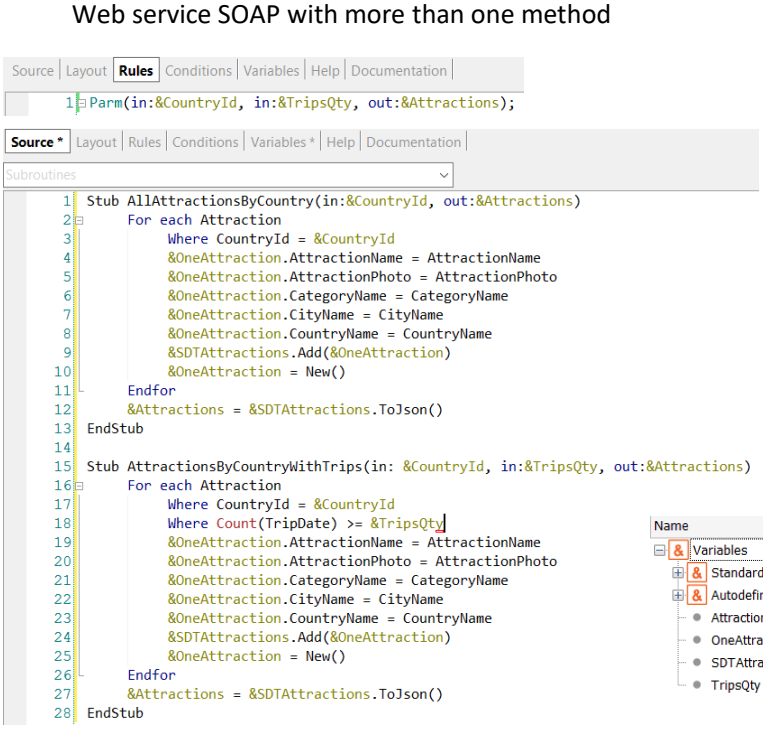
Como já confirmamos, se expormos um objeto procedimento como web service, o serviço inclui um único método: Execute.

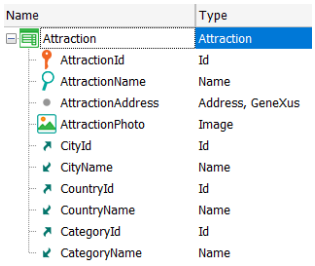
Se for requerido definir mais de um método no mesmo web service, deve ser feito uso de stubs no source do procedimento.

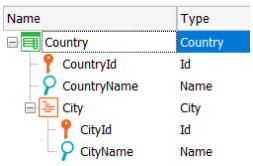
Os stubs são cláusulas que, dentro do source de um objeto procedimento, nos permitem definir um bloco de código associado a um nome e, em seguida, executar o código invocando esse nome. O conceito é similar ao de um subprograma ou sub-rotina e cada stub pode ter seus próprios parâmetros.

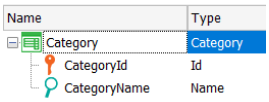
Algo importante a esclarecer é que o uso de stubs só é válido quando expomos o procedimento com protocolo SOAP, os procedimentos expostos como REST não suportam esta funcionalidade.

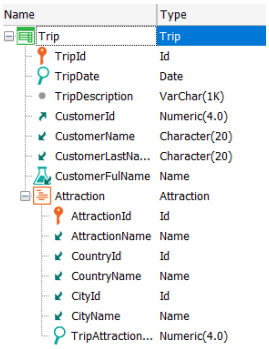
Web service SOAP with more than one method

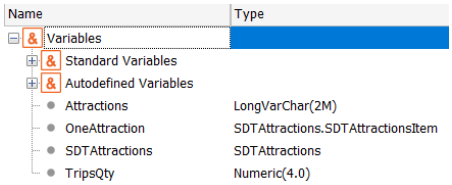












Vejamos um exemplo de expor um procedimento SOAP com mais de um método.

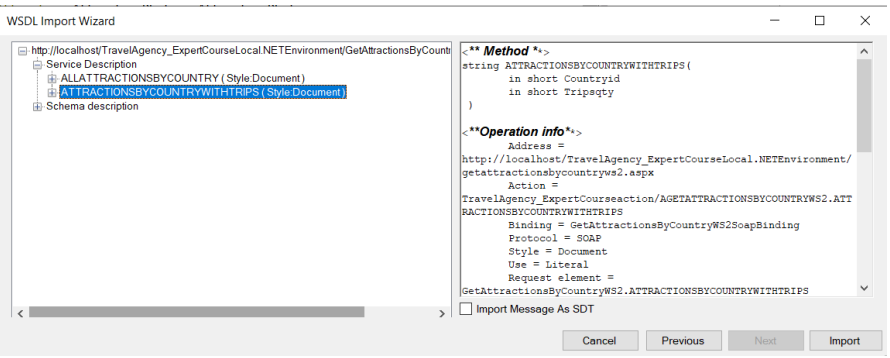
Suponhamos que o procedimento GetAttractionsByCountryWS que expusemos como web service, queremos que tenha dois métodos, um para nos trazer todas as atrações turísticas de um país e outro que retorne apenas as atrações de um país que já teve uma quantidade de visitas (trips) maior ou igual a um número dado.

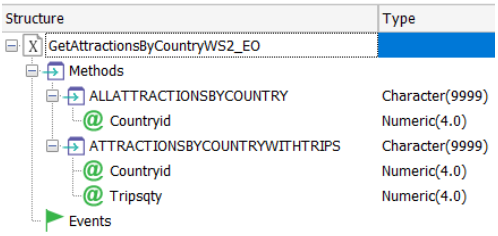
Salvamos o procedimento GetAttractionsByCountryWS como GetAttractionsByCountryWS2 e modificamos sua regra parm adicionando a variável &TripQty como parâmetro de entrada.

No source criamos 2 stubs, um com o nome AllAttractionsByCountry, que recebe como parâmetro o CountryId e retorna um JSON com todas as atrações daquele país (tenham ou não trips), e outro stub de nome AttractionsByCountryWithTrips, que recebe como parâmetros o CountryId e a quantidade de trips pela qual queremos filtrar, e retorna um JSON com as atrações encontradas que possuem a mesma quantidade ou mais viagens que o valor da variável &TripsQty.

Vamos fazer um build all para que o serviço seja publicado no server.

Web service SOAP with more than one method





Vamos importar o novo web service que criamos. Executamos o wizard, escrevemos a nova URL com o nome do novo objeto e pressionamos Next.

Modificamos o nome do objeto externo sugerido adicionando _EO, escrevemos o nome de um folder de destino e pressionamos Next.

Agora, se abrirmos o nó Service Description, veremos que não está mais lá o método Execute e existem os dois métodos que correspondem aos stubs que criamos.

Se selecionarmos cada método, vemos que na janela da direita podemos ver os parâmetros, que coincidem com aqueles que definimos previamente nos stubs.

Se por motivos de compatibilidade é requerido que o método Execute continue sendo exposto, deverá ser criado expressamente um stub com o nome Execute.

Continuando com o wizard, pressionamos Import e verificamos que foi criado o objeto externo dentro do folder que já tínhamos. Se o abrirmos verificamos que tem os 2 métodos que definimos.

Desta forma verificamos a flexibilidade de incluir vários métodos em um mesmo serviço, o que é bastante útil para quem consome nosso web service SOAP.

[http://localhost/TravelAgency_ExpertCourseNETLocal/GetAttractionsByCountryWS2.aspx?WSDL]

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