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

CalkInctionsByCountryWS Source Layout Rules Conditions Variables Help Documentation Call protocol Mane Call protocol Mane Call protocol Mare CountryId Mare CountryIde CountryIde Mare CountryIde Mare CountryIde Mare CountryIde Mare CountryIde Mare Conditions Variables Help Documenta Mare Conditions Variables Help Documenta Mare Conditions Surfactorions); Mare Conditions Surfactorions; Mare				Procedure: GetAttractions	ByCountryWS		-
Source Layout Rules Conditions Variables Help Documentation Control Main program False For each Attraction Main program False Call protocol Internal Execute in new LUW False Citylante Name SoneAttraction.CategoryName CategoryName CategoryName Citylante Name SoneAttractions CountryName Citylante Name Citylante Name SoneAttractions SoneAttractions.CategoryName Citylante Name Citylante Name SoneAttractions CountryName Citylante Name Citylante Name SoneAttractions CountryName Citylante Name Citylante Name SoneAttractions Name CountryName Citylante Name CountryName SoneAttractions Veb Service True Veb Service Name CountryName Name Source Layout Name Type Is Collection Citylante Name Source Layout Name Type Source	GetAttractionsByCountryWS ×			Name	GetAttractionsByCountryWS	Name Attraction	Attraction
Source Layout Rules Conditions Variables Mame Type Is collection Attraction Name 1 Parm(in:&CountryId, out:&Attractions); Name Type Is collection For each Attractions Attraction 3 Module/Folder Root Module Main program Faise Attraction	Source Layout Rules Conditions Variables Help Documentation			Description	Get Attractions By Country WS	AttractionId	Id
Address Addres Address Address	Subroutines	~		Module/Folder	Root Module	AttractionName	Name
All Mere CountryId Image: ContryId Image: ContryId Image: ContryId Image: Contry: Contry: Contry Contry Contry Contry Contry: Contry Co	1 For each Attract	ion		Main program	False	AttractionAddress AttractionPhoto	Address, GeneXus
Address of the second of th	2 Where Count	ryId = &CountryId		Call protocol	Internal	- > CityId	Id
Source Layout Rules Conditions Variables Help Documente Name Type Source Layout Rules Conditions Variables Help Documente Source Conditions Variables Help Documente Source Conditions Variables Help Documente Name Type Source Layout Rules Conditions Variables Help Documente Name Type Source ContryName Name Type Source Layout Rules Conditions Variables Help Documente Name Type Source ContryName Name Type Source Layout Rules Conditions Variables Help Documente Source ContryName Name Type Source ContryName Name Type Source Layout Rules Conditions Variables Help Documente Source ContryName Name Type Source ContryName Name Type Source Layout Rules Conditions Variables Source Type Source ContryName Name Type Source Layout Rules Conditions Variables Mame Type Source ContryName Name Name Source Layout Rules Conditions Variables Mame Type Source ContryName Name Name Source Layout ContryName Type Source ContryName Name Type Source ContryName Name Sourd	4 &OneAttract	ion.AttractionPhoto = Attractio	onPhoto	Execute in new LUW	False	🛛 🖌 CityName	Name
addreduced traction. Survivaliance = Cultyriality Object Visibility Public Country Value Category Value 8 Source tayout Rules Conditions Variables Help Documenta Object Visibility Public Name Type Source tayout Rules Conditions Variables Help Documenta Veb Service Protocol Source country Value Name Type Source tayout Rules Conditions Variables Help Documenta Veb Service schema val Use Environment property value Name Type Name Type SofTattractionsEtem Attrabute: Attrabute: AttractionName Attrabute: AttractionName Source tayout Rules LongVarchar(2M) SofTattractionsEtem Attrabute: CountryName Name Source tayout Rules SofTattractionsStem Attrabute: CountryName Name Type Source tayout Rules SofTattractionsEtem Attrabute: CountryName Name Name * SofTattractions SofTattractionsEtem Attrabute: CountryName Id Name * Attractorine SofTattractionsStem Attrabute: CountryName Id CategoryName Name * SofTattractions SofTattractionsRtem Attrabute: CountryName Id CategoryName Name <tr< td=""><td>5 &OneAttract</td><td>ion.CategoryName = CategoryNam</td><td>e</td><td>Qualified Name</td><td>GetAttractionsByCountryWS</td><td rowspan="2"> </td><td>Id</td></tr<>	5 &OneAttract	ion.CategoryName = CategoryNam	e	Qualified Name	GetAttractionsByCountryWS	 	Id
8 SOTAttractions. Add(&OneAttraction) Name Interoperability Interoperability 10 Endbe MTOM False 11 &Attractions = &SOTAttractions. ToJson() Enable MTOM False Source Layout Rules Conditions Variables Veb Service Protocol 1 Parm(in: &CountryId, out: &Attractions); SOAP Protocol True CountryId Id Name Type Is Collection Veb Service schema vali Use Environment property value Veb Service schema vali Use Environment property value Name Type Sot Attractions Sot Attractions Veb Service is Collection Veb Service Schema vali Veb S	7 &OneAttract	ion.CountryName = CountryName		Object Visibility	Public		Id
10 Endfor 11 BAttractions = &SDTAttractions.ToJson() Enable MTOM False Expose as Web Service True Web Service Protocol Source Layout Rules Conditions Variables Help Documenta 1 Parm(in:&CountryId, out:&Attractions); Name Type SotAttractions SotAttractions <td>8 &SDTAttract</td> <td>ions.Add(&OneAttraction)</td> <td></td> <td colspan="2">✓ Interoperability</td> <td>CategoryName</td> <td>Name</td>	8 &SDTAttract	ions.Add(&OneAttraction)		✓ Interoperability		CategoryName	Name
11 &Attractions = &SDTAttractions.ToJson() Expose as Web Service True Source Layout Rules Conditions Variables Web Service Protocol Source Layout Rules Conditions Variables Use Native Soap Use Environment property value 1 Parm(in:&CountryId, out:&Attractions); Name Type Use Environment property value Name Type Sol Attractions Sol Attractions Id Image: Sol Attractions Sol Attractions Image: Sol Attractions Image: Sol Attractions Image: Sol Attractions Sol Attractions AttractionName AttractionName Image: Sol AttractionName Image: Attractions Sol AttractionsName Attractic:ChyName Attribute:AttractionPhoto Id Image: Attractions Sol AttractionsName Attribute:CountryName Id Id Id Image: Sol Attractions Sol AttractionsName Attribute:CountryName Image: CountryName Id Image: Sol Attractions Sol AttractionsName Attribute:CountryName Id Id Id Image: Sol Attractions Sol AttractionName Attribute:CountryNam	10 - Endfor			Enable MTOM	False		
Source Layout Rules Conditions Variables SOAP Protocol True 1 Parm(in:&CountryId, out:&Attractions); SOAP Protocol True Use Environment property value Name CountryId Id Name Type Is Collection Veb Service schema vali Use Environment property value Name City City City City City City/ City/diant Name Image: Stratard Variables Image: Stratardions StratarctionName AttractionName AttractionName AttractionName Type CategoryId Id Id CategoryId Id Id CategoryId Id CategoryId Id C	<pre>11 &Attractions = &SDTAttractions.ToJson()</pre>			Expose as Web Service	True	Name	Type
Source Layout Rules Conditions Variables SOAP Protocol True 1 Parm(in:&CountryId, out:&Attractions); Use Environment property value Use Environment property value Name Type Name Type Id Is Sold Attractions Sold Attractions Id Provide Image: Sold Attractions Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions Image: Sold Attractions <td< td=""><td></td><td></td><td></td><td colspan="2">V Web Service Protocol</td><td>Country</td><td>Country</td></td<>				V Web Service Protocol		Country	Country
Source Layout Kules Conditions Variables Help Documenta Use Native Soap Use Environment property value Name Name 1 Parm(in:&CountryId, out:&Attractions); Web Service schema vali Use Environment property value Web Service schema vali Use Environment property value P CountryName Name Name Type Name Type Is Collection P Cityl Mame Name & Variables SDT AttractionsEem Autodefined Variables AttractionPhoto AttractionPhoto AttractionPhoto AttractionPhoto CategoryId Id • Attractions SDT Attractions SDT AttractionsItem • CountryName Attribute:ChayName CategoryId Id • Attractions SDT Attractions SDT AttractionsItem • CategoryName Attribute:CountryName CategoryId Id • SDT Attractions SDT Attractions CategoryName Attribute:CountryName Mame Name				SOAP Protocol	True	CountryId CountryName	Id
1 Parm(in:&CountryId, out:&Attractions); Web Service schema vali Use Environment property value Image: City display Id Name Type Name Type Is Collection Image: City display Image: City display Id Name Image: City display Image: City display Name Type Image: City display Image: City display Name Name Image: City display Image: City display Image: City display Name Image: City display Image: City display Image: City display Name Image: City display Image: City display Image: City display Name Image: City display Image: City display Image: City display Name Image: City display Image: City displa	Source Layout Rules Cond	itions Variables Help Documenta	3	Use Native Soap	Use Environment property value		Name
Name Type Is Collection Image: Stratule strate in S	<pre>1 Parm(in:&CountryId, out:&Attractions);</pre>			Web Service schema va	i Use Environment property value	⊟ 🚔 City	City
Name Type Is Collection						CityName	Name
Structure Structure Name Type	Name	Туре	Name	Туре	Is Collection		
Image: Standard Variables Name Type Image: Standard Variables Image: Standard Variables <td>- & Variables</td> <td></td> <td>SDTAttractions</td> <td></td> <td></td> <td></td> <td></td>	- & Variables		SDTAttractions				
• Attractions • Attractions • AttractionsItem • AttractionsItem • OneAttractions • SDTAttractions • SDTAttractions • SDTAttractions • SDTAttractions • CategoryName	8 Standard Variables		BTAttractions	Item		Name	Туре
Attractions LongVarChar(2M) AttractionPhoto AttractionPhoto AttractionPhoto AttractionPhoto AttractionPhoto OneAttraction SDTAttractions.SDTAttractionsItem SDTAttractions SDTAttracting SDTAttractions SDTAttractions SDTAttr			 AttractionNa 	me Attribute:AttractionNan	ie 🗌	Category	Category
OneAttractions SDTAttractions.SDTAttractions.Item SDTAttractions SDTAttracti	 Attractions 	LongVarChar(2M)	 AttractionPh 	oto Attribute:AttractionPho	o 🗌	🗧 📍 CategoryId	Id
SDTAttractions SDTAttractions SDTAttractions SDTAttractions SDTAttractions SDTAttractions SDTAttractions Attribute:CountryName Attribute:CategoryName	 OneAttraction 	SDTAttractions.SDTAttractionsItem	 CityName 	Attribute:CityName		🗆 ዖ CategoryName	Name
 CategoryName Attribute:CategoryName 	 SDTAttractions 	SDTAttractions	 CountryNam 	e Attribute:CountryName			
			CategoryNa	me Attribute:CategoryNam			

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



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



https://www.telerik.com/fiddler





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

localheat?	102 ··· > local	wik6001	- TopTream v0.81717
File Ven 1	telp:	_	
Source P 127-8-01 127-9-0-1	Tene 11:50:19 11:53:25	Status	POST Ficked same of TTP1-13 POST Ficked same of TTP1-14 POST Ficked same of TTP1-14
			Le Can Carport Services "The global constanting and includes the service of the
			Co:Envelope xmlmcs="http://ichemas.xmloung.ang/anag/envelope?>Co:Body>ClindeCircumference xmlns="http://first/servis/?>Co?CircleCircumference?c4:Body>CircIneCircumference?
			Cada Canada Anton, na cagar O Canada Yan Canada Canada Anton Yang Kanada Sala Yang Kanada S

Também 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

♦ localhost/TravelAgency_ExpertCo x +
← → C O localhost/TravelAgency_ExpertCourseNetLocal/GetAttractionsByCountryWS.aspx?WSDL
his XML file does not appear to have any style information associated with it. The document tree is shown below.
<pre>cdefinitions xmlns:wslns="TravelAgency_ExpertCourse" xmlns:soap="http://schemas.xmlsoap.org/wsl/soap/" xmlns:xsd="http://wn.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsl/ xmlns:tns="TravelAgency_ExpertCourse" name="GetAttractionsByCountryKS" targetHamespace="TravelAgency_ExpertCourse" vfcypes> vfcchema name="GetAttractionsByCountryKS" targetHamespace="TravelAgency_ExpertCourse" vfcchema name="GetAttractionsByCountryKS" targetHamespace="TravelAgency_ExpertCourse" vfcchema name="GetAttractionsByCountryKS.texcute" vfcchemationsByCountryKS.texcute" vfcchemationsByCountryKS.texcute" vfcchemationsByCountryKS.texcute" vfcchemationsByCountryKS.texcute" vfcchemationsByCountryKS.texcute" vfcchemationsByCountryKS.texcuteResponse" vfcchemationsByCountryKS.texcuteResponse" vfcchemationCours=1" maxOccurs=1" m</pre>
<pre> </pre>
<pre>Y <pre>trype name="GetAttractionsByCountry/BSSoapPort"></pre></pre>
<pre><!--/sinding--> V(service name="GetAttractionsByCountryMS"> V(port name="GetAttractionsByCountryMSSoapPort" binding="wsdlns:GetAttractionsByCountryMSSoapBinding"> </pre>

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 Countryld, etc.

[http://localhost/TravelAgency_ExpertCourseNETLocal/GetAttractionsByCountry WS.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 Countryld, então substituímos o ponto de interrogação pelo identificador do país do qual queremos obter informação de suas atrações. Escrevemos 2 que é o ld 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



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

Source Stub mehod1 (in: param1, in:param2, out: param4) EndStub Stub mehod2 (in: param3, out: param4) EndStub	Rules Parm (in: param1, in:param2, in: param3, out: param4)
Stub mehod2 (in: param3, out: param4) EndStub	Source Stub mehod1 (in: param1, in:param2, out: param4) EndStub
	Stub mehod2 (in: param3, out: param4) EndStub

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.

Name Туре Name Туре Attraction Country Source Layout Rules Conditions Variables Help Documentation AttractionId Τd CountryId Id AttractionNai Name CountryName 1 Parm(in:&CountryId, in:&TripsQty, out:&Attractions); Name AttractionAddress Address, GeneXus 📥 들 City City 🚵 AttractionPhoto Image Source * Layout Rules Conditions Variables * Help Documentation 💡 CityId Id CityId Id CityName Name CityName Name CountryId Τd Stub AllAttractionsByCountry(in:&CountryId, out:&Attractions) Name Туре 2 For each Attraction CountryName Name Category 3 Where CountryId = &CountryId CategoryId Id CategoryId Τd 4 &OneAttraction.AttractionName = AttractionName CategoryName Name Q CategoryName Name 5 &OneAttraction.AttractionPhoto = AttractionPhoto 6 &OneAttraction.CategoryName = CategoryName 7 &OneAttraction.CityName = CityName Name Туре 8 &OneAttraction.CountryName = CountryName - Trip 9 &SDTAttractions.Add(&OneAttraction) TripId &OneAttraction = New() TripDate Date 11 Endfor &Attractions = &SDTAttractions.ToJson() TripDescription VarChar(1K) 13 EndStub CustomerId Numeric(4.0) 14 CustomerName Character(20) 15 Stub AttractionsByCountryWithTrips(in: &CountryId, in:&TripsQty, out:&Attractions) CustomerLastNa... Character(20) 16 For each Attraction 💑 CustomerFulName 🛛 Name Where CountrvId = &CountrvId Attraction Attraction 18 Where Count(TripDate) >= &TripsOty Name Туре AttractionId Td &OneAttraction.AttractionName = AttractionName 19 🖃 🚷 Variables AttractionName Name 20 &OneAttraction.AttractionPhoto = AttractionPhoto ✔ CountryId Td 🗄 🚷 Standard Variables 21 &OneAttraction.CategorvName = CategorvName CountryName Name Autodefined Variables &OneAttraction.CityName = CityName ✔ CityId Τd LongVarChar(2M) 23 &OneAttraction.CountryName = CountryName Attractions 🖌 CityName Name 24 &SDTAttractions Add(&OneAttraction) OneAttraction SDTAttractions.SDTAttractionsItem P TripAttraction... Numeric(4.0) 25 &OneAttraction = New() SDTAttractions SDTAttractions 26 Endfor TripsQty Numeric(4.0) 27 &Attractions = &SDTAttractions.ToJson()

Web service SOAP with more than one method

28 EndStub

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 Countryld 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 Countryld 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

WSDL Import Wizard	- 🗆 X	Structure	Туре
thtp://localhost/TravelAgency_ExperiCourseLocal.NETEnvironment/GerAttractionsByCount Service Description #LATTRACTIONSBYCOUNTRY (Style Document) #ATTRACTIONSBYCOUNTRYWITHTRIPS (Style Document) Schema description	<pre><** Method **> string ATTRACTIONSBYCOUNTRYWITHTRIPS(in short Countryid in short Tripeqty } <**Operation info**> Address = htp://loalhost/TravelAgency_ExpertCourseLocal.NETEnvironment/ getattractionsbycountryw32.aspx Action = TravelAgency_ExpertCourseaction/AgETATTRACTIONSBYCOUNTRYWS2.ATT ActionsBycountryW52.SorpBinding Protocol = SoR Style = Document Use = talmant = GetAttractionsByCountry%2.ATTRACTIONSBYCOUNTRYWTHTRIPS import Message As SDT Cancel Previous Next Import</pre>	Countryid Count	Character(9999) Numeric(4.0) Character(9999) Numeric(4.0) Numeric(4.0)

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/GetAttractionsByCountry WS2.aspx?WSDL]



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