

Web Services. Aspectos avançados.

Publicando serviços REST com GeneXus

GeneXus™

Até agora só publicamos serviços SOAP, vejamos agora como publicar serviços REST com GeneXus utilizando o mecanismo geral e também através do objeto API, com todas as vantagens que este objeto nos proporciona.

Publishing a REST web service

- Procedure
- Business component
- Data Provider

+

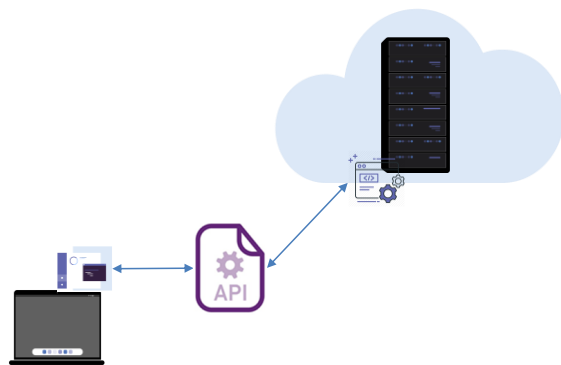
Expose as Web Service	True
Web Service Protocol	
SOAP Protocol	False
REST Protocol	True
Generate OpenAPI interface	Use Environment property value

- Procedure
- Data Provider

+



API object



Em GeneXus existem duas maneiras de publicar um serviço REST.

Uma que já vimos, que é expor objetos procedimentos, data providers ou business components utilizando a propriedade Expose as Web Service no valor True e definindo a propriedade REST Protocol no valor True.

Outra maneira é usando o objeto API, que serve para expor objetos procedimentos ou data providers. Este objeto API permite expor serviços com protocolos REST ou gRPC, permitindo agrupar vários serviços que estejam semanticamente ou funcionalmente relacionados

O objeto API adiciona uma camada intermediária que separa a interface dos detalhes de implementação, o que permite que futuras alterações na programação nos objetos não afetem a forma como são invocados pelas aplicações externas.

Isto permite fazer um mapeamento entre o nome interno do objeto na KB e o nome com o qual ele está exposto como serviço e também possibilita alterar o tipo e nome dos parâmetros ou alterar o path de acesso, sem que isso afete a forma como o serviço é invocado.

Esta abstração fornece uma flexibilidade importante porque podemos evoluir nossos serviços, sem forçar as aplicações que os utilizam a alterar seu código para se adaptar às mudanças.

Por esse motivo, para expor Procedimentos e Data Providers como serviços REST,

é altamente recomendável sempre usar o objeto API, em vez das propriedades Expose as Web Service e Rest Protocol.

Publishing a service with the API object

The image displays four screenshots from the GeneXus IDE illustrating the process of creating and configuring an API object:

- Top Left:** The "New Object" dialog box. The "Select a Type:" dropdown is set to "API". The "Name:" field contains "GetAttractionsInfo", and the "Description:" field contains "Get Attractions Info".
- Top Right:** The "Service Source" editor for the "GetAttractionsInfo" object. It shows the following code:


```

1 GetAttractionsInfo{
2
3     GetAttractionsByCountry(in:&CountryId, out:&Attractions)
4         => GetAttractionsByCountryWS(in:&CountryId, out:&Attractions);
5
6     GetAttractionsByCountryAndTrips(in:&CountryId, in:&TripsQty, out:&Attractions)
7         => GetAttractionsByCountryWithTripsWS(in:&CountryId, in:&TripsQty, out:&Attractions);
8 }
9
      
```
- Bottom Left:** The "Variables" editor. It shows a tree view with "Autodefined Variables" expanded, listing:
 - CountryId: Attribute:CountryId
 - Attractions: LongVarChar(2M)
 - TripsQty: Numeric(4,0)
- Bottom Right:** The "Insert Event" dialog box. It shows a table with two columns: "Controls" and "Events".

Controls	Events
GetAttractionsByCountry	Before
GetAttractionsByCountry	After

Para testar o que vimos, vamos criar um objeto API para publicar os 2 métodos que tínhamos para obter dados de atração por país, como serviços REST. Chamamos GetAttractionsInfo o API object.

O objeto API tem três partes: Service Source, Events e Variables. Na seção Service Source é definido, através de uma sintaxe declarativa, o nome de cada serviço a ser publicado, com os parâmetros que correspondam, especificando se cada parâmetro é de entrada ou de saída, e o nome do objeto GeneXus em nossa KB que estamos expondo como serviço, com seus respectivos parâmetros.

Aqui estamos expondo o serviço GetAttractionsByCountry com base no procedimento GetAttractionsByCountryWS. Os parâmetros correspondem.

Analogamente, mais abaixo estamos expondo o procedimento GetAttractionsByCountryWithTripsWS como serviço, com o nome GetAttractionsByCountryAndTrips e os parâmetros que expomos, são os mesmos do objeto procedimento.

Esta definição permite no futuro alterar o nome ou os parâmetros do objeto GeneXus, sem que altere a definição de como está publicado este objeto.

Nos eventos dispomos do evento Before e After, com o que podemos realizar ações que sejam executadas antes ou depois da invocação do objeto como serviço. Também dispomos dos eventos Before e After para cada serviço exposto. Nestes eventos não é possível acessar a base de dados, se necessário, deve ser incluído no código do procedimento ou data provider exposto como serviço.

Nas variáveis existem algumas sob o grupo de variáveis padrão, que nos permitem conhecer ou alterar características do objeto API em tempo de execução. A variável `&Pgmname` contém o nome do objeto, `&Pgmdesc` a descrição do objeto, `&RestMethod` contém o método HTTP com o qual foi chamado o objeto. Esta variável fica vazia quando é chamado o objeto utilizando um protocolo que não seja REST. E a variável `&RestCode` é utilizada para definir o código de status HTTP, dependendo do que retorna a chamada do serviço. Em nosso exemplo, se não encontramos nenhuma atração para o país dado, poderíamos atribuir à variável `&RestCode` o valor 404 (Not found).

Customizing a REST web service with the API object

1) Parameter customization

Service Source | Events | Variables | Help | Documentation

```

1 GetAttractionsInfo{
2
3   GetAttractionsByCountry(in:&CountryId, out:&Attractions)
4   => GetAttractionsByCountryWS(in:&CountryId, out:&Attractions);

```

GetAttractionsInfo X

Service Source | Events | **Variables** | Help | Documentation

Name	Type
& Variables	
& Standard Variables	
● Pgmdesc	Character(256)
● Pgmname	Character(128)
● RestCode	Numeric(3.0)
● RestMethod	HttpMethod, GeneXus
● Attractions	LongVarChar(2M)
● CountryId	Attribute:CountryId
● TripsQty	Numeric(4.0)

Interface Information

External Name	Id

URL: ...GetAttractionsInfo/GetAttractionsByCountry/?CountryId=3

↓

URL: ...GetAttractionsInfo/GetAttractionsByCountry/?Id=3

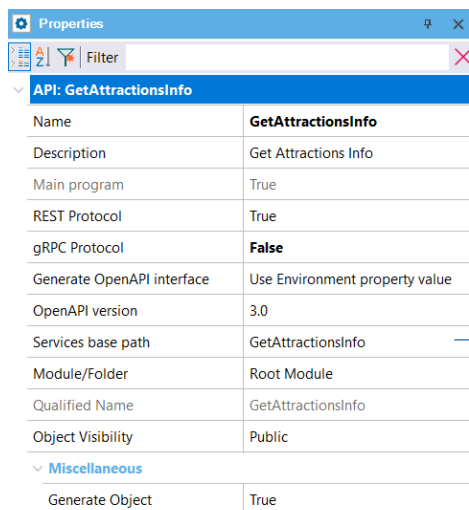
Como dissemos antes, o objeto API nos permite alterar o nome de um parâmetro. Para isso devemos atribuir a propriedade External Name das variáveis usadas nos parâmetros do serviço exposto.

No exemplo que vimos se formos para as variáveis do objeto API GetAttractionsInfo, selecionamos a variável CountryId e na propriedade ExternalName escrevemos Id.

A URL com a qual será invocado o serviço em vez de usar o nome CountryId para o parâmetro, usará o nome que definimos como external name e aparecerá a palavra Id em vez de CountryId.

Customizing a REST web service with the API object

2) Service URL customization



API: GetAttractionsInfo	
Name	GetAttractionsInfo
Description	Get Attractions Info
Main program	True
REST Protocol	True
gRPC Protocol	False
Generate OpenAPI interface	Use Environment property value
OpenAPI version	3.0
Services base path	GetAttractionsInfo
Module/Folder	Root Module
Qualified Name	GetAttractionsInfo
Object Visibility	Public
Miscellaneous	
Generate Object	True

URL:

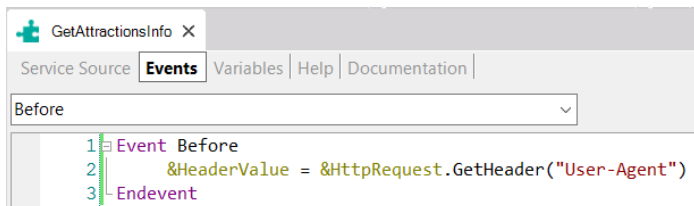
...GetAttractionsInfo/GetAttractionsByCountry?CountryId=3

Se quisermos alterar a URL padrão de um objeto API, podemos fazê-lo alterando o valor de sua propriedade Services base path.

Por padrão esta propriedade tem o nome do objeto API, se colocarmos outro texto, esse texto aparecerá na URL em vez do nome do objeto.

Customizing a REST web service with the API object

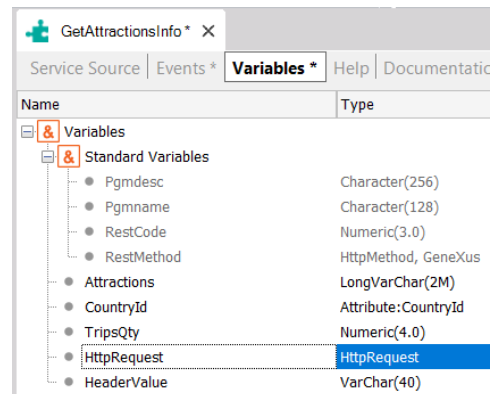
3) Reading parameters from HTTP header Request



```

1 Event Before
2   &HeaderValue = &HttpRequest.GetHeader("User-Agent")
3 Endevent

```



Name	Type
Variables	
Standard Variables	
PgmDesc	Character(256)
PgmName	Character(128)
RestCode	Numeric(3.0)
RestMethod	HttpMethod, GeneXus
Attractions	LongVarChar(2M)
CountryId	Attribute:CountryId
TripsQty	Numeric(4.0)
HttpRequest	HttpRequest
HeaderValue	VarChar(40)

Outra coisa que podemos fazer com o objeto API é ler os dados do cabeçalho da invocação ao nosso serviço REST.

Se no header HTTP do serviço é recebido algum parâmetro, usando o evento Before, é possível ler o header que é enviado pela aplicação que invoca o serviço quando faz o Request.

A expressão de texto usada entre os parênteses do método GetHeader determina o nome do header HTTP. Estes headers são invisíveis para o usuário final e definem como a informação é enviada ou recebida do serviço, por exemplo, quem chama o serviço, nome do host, credenciais de acesso, o tipo de conexão, cookies, etc.

Neste caso, o header é denominado "User-Agent", e o valor retornado é uma sequência de texto que identifica o agente de usuário para o servidor, em outras palavras, o nome da aplicação cliente que está invocando o serviço.

Customizing a REST web service with the API object

4) HTTP access methods customization

```

1 AttractionsInfo{
2
3     [RestMethod(GET)]
4     GetAttraction(in:&AttractionId, out:&AttractionInfo)
5         => GetAttractionInfoWS(in:&AttractionId, out:&AttractionInfo);
6
7     [RestMethod(POST)]
8     InsertAttraction(in:&AttractionName, in:&AttractionPhoto, in:&CountryId, in:&CityId, in:&CategoryId)
9         => CreateAttractionWS(&AttractionName,&AttractionPhoto,&CountryId,&CityId,&CategoryId);
10
11 }

```

Com o objeto API, se incluirmos uma anotação antes da definição do serviço, podemos especificar o método HTTP de acesso.

Suponhamos que temos dois serviços, um denominado GetAttraction para obter dados de uma determinada atração turística através de um data provider e outro denominado InsertAttraction, que nos permite criar uma atração nova na base de dados através de um procedimento.

Por exemplo, no caso do serviço GetAttraction, estamos especificando que será usado o método GET para invocá-lo, pois estamos obtendo informação através do serviço. A anotação vai entre colchetes e dentro terá RestMethod e entre parênteses o método HTTP a ser usado, neste caso GET.

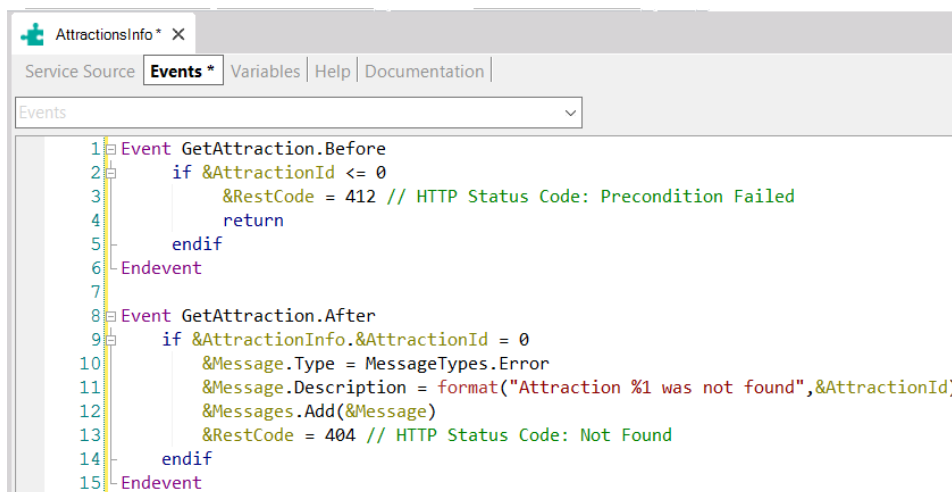
No exemplo, o método interno que será executado é o data provider GetAttractionInfoWS. Como os DP retornam informação, eles são invocados por padrão como GET e com a anotação estamos fazendo isto de forma explícita.

O método InsertAttraction, que insere uma atração na base de dados, o invocamos com um POST, usando a anotação RestMethod(POST).

O objeto CreateAttractionWS, por ser um procedimento, pode ser invocado como GET ou como POST, neste caso o invocamos explicitamente como POST porque estará gravando informação na base de dados.

Customizing a REST web service with the API object

5) Service response customization



```
1 Event GetAttraction.Before
2   if &AttractionId <= 0
3     &RestCode = 412 // HTTP Status Code: Precondition Failed
4     return
5   endif
6 Endevent
7
8 Event GetAttraction.After
9   if &AttractionInfo.&AttractionId = 0
10    &Message.Type = MessageTypes.Error
11    &Message.Description = format("Attraction %1 was not found",&AttractionId)
12    &Messages.Add(&Message)
13    &RestCode = 404 // HTTP Status Code: Not Found
14  endif
15 Endevent
```

Através do uso da variável padrão `&RestCode` do objeto API, podemos alterar o código de status HTTP.

Fazemos isto nos eventos e podemos fazê-lo antes de invocar o serviço (em um evento Before) para definir um valor no caso em que já se determine que não é possível invocar o serviço porque os dados recebidos não estão corretos. No código estamos atribuindo o HTTP Status Code 412 PreconditionFailed, se o `AttractionId` recebido como parâmetro não é válido.

E também podemos fazê-lo após executar o serviço, para estabelecer o resultado da operação, por exemplo, se não foi encontrada a informação requerida. No exemplo, a informação recebida tem o campo `AttractionId` vazio, portanto, não foi encontrada informação de uma atração com o `AttractionId` passado por parâmetro, por isso atribuímos a variável `&RestCode` com o valor 404 (Not Found).

Neste vídeo vimos a flexibilidade do objeto API para publicar serviços REST. Se quiser saber mais sobre este objeto, convidamos você a visitar a wiki.

GeneXus™

training.genexus.com

wiki.genexus.com

training.genexus.com/certifications