Web Services. Aspectos avanzados.

Publicando servicios REST con GeneXus

GeneXus™

Hasta ahora publicamos únicamente servicios SOAP, veamos ahora cómo publicar servicios REST con GeneXus utilizando el mecanismo general y también a través del objeto API.

Publishing a REST web service

- Procedure
- Business component
- Data Provider

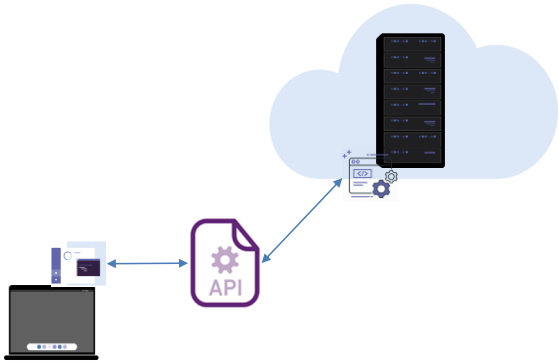
+

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



En GeneXus hay dos formas de publicar un servicio REST.

Una que ya vimos, que es exponer objetos procedimientos, data providers o business components utilizando la propiedad Expose as Web Services en el valor True y seteando la propiedad REST Protocol en el valor True.

Otra forma es usando el objeto API, que nos sirve para exponer objetos procedimientos o data providers. Este objeto API permite exponer servicios con protocolos REST o gRPC, permitiendo agrupar varios servicios que estén semántica o funcionalmente relacionados

El objeto API agrega una capa intermedia que separa la interfase, de los detalles de implementación, lo que permite que futuros cambios en la programación en los objetos, no afecten la forma en que son invocados por las aplicaciones externas.

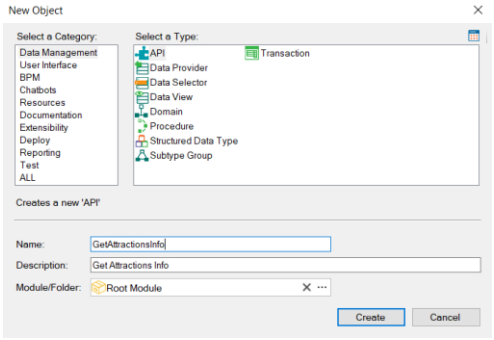
Esto permite hacer un mapeo entre el nombre interno del objeto en la KB y el nombre con el que el mismo es expuesto como servicio y también hace posible que se puede cambiar el tipo y nombre de los parámetros o cambiar el path de acceso, sin que eso afecte a la forma en que el servicio es invocado.

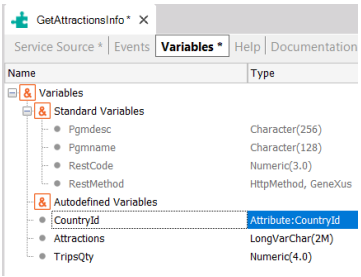
Esta abstracción provee una flexibilidad importante porque podemos evolucionar nuestros servicios, sin obligar a que las aplicaciones que los usan deban cambiar su código para adaptarse a los cambios.

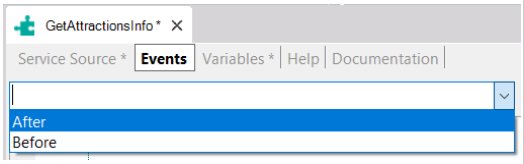
Por esa razón, para exponer Procedimientos y Data Providers como servicios REST, se recomienda fuertemente usar siempre el objeto API, en lugar de las

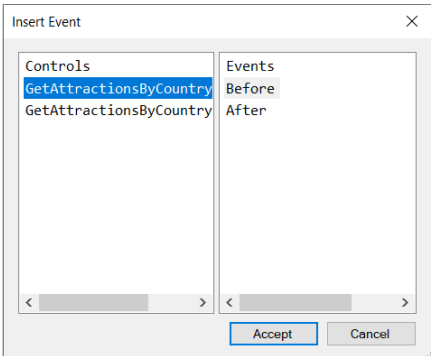
propiedades Expose as Web Service y Rest Protocol.

Publishing a service with the API object









Para probar lo que vimos, vamos a crear un objeto API para publicar los 2 métodos que teníamos para obtener datos de atracciones por país, como servicios REST. Llamamos GetAttractionsInfo al API object.

El objeto API tiene tres partes: Service Source, Events y Variables. En la sección Service Source se define, mediante una sintaxis declarativa, el nombre de cada servicio a ser publicado, con los parámetros que correspondan, especificando si cada parámetro es de entrada o de salida y el nombre del objeto GeneXus en nuestra KB que estamos exponiendo como servicio, con sus respectivos parámetros.

Aquí estamos exponiendo al servicio GetAttractionByCountry en base al procedimiento GetAttractionsByCountryWS. Los parámetros coinciden.

Análogamente, más abajo estamos exponiendo al procedimiento GetAttractionsByCountryWithTripsWS como servicio, con el nombre GetAttractionsByCountryAndTrips y los parámetros que exponemos, son los mismos que los del objeto procedimiento.

Esta definición nos permite en el futuro cambiar el nombre o los parámetros del objeto GeneXus, sin que cambie la definición de cómo está publicado este objeto.

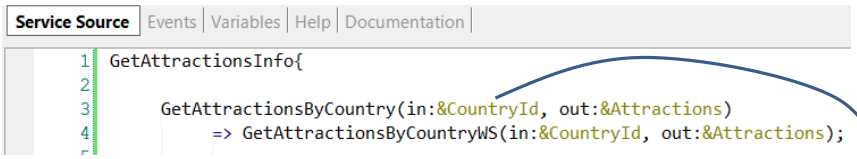
En los eventos disponemos del evento Before y After, con el que podemos realizar acciones que se ejecuten antes o después de la invocación al objeto como servicio. También disponemos de los eventos Before y After para cada servicio expuesto. En estos eventos no se puede acceder a la base de datos, en caso de requerirse, debe

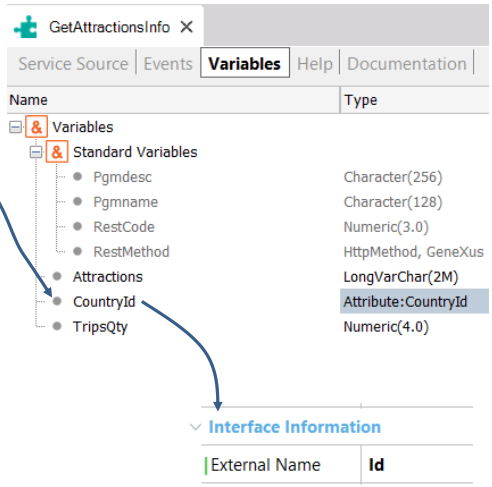
incluirse en el código del procedimiento o data provider expuesto como servicio.

En las variables hay algunas bajo el grupo de variables estándar, que nos permiten conoce o cambiar características del objeto API en tiempo de ejecución. La variable &Pgmname contiene el nombre del objeto, &Pgmdesc la descripción del objeto, &RestMethod contiene el método HTTP con el que se llamó al objeto. Esta variable queda vacía cuando se llama al objeto utilizando un protocolo que no sea REST. Y la variable &RestCode se utiliza para setear el código de estado HTTP, dependiendo de lo que devuelva la invocación al servicio. En nuestro ejemplo, si no encontramos ninguna atracción para el país dado, podríamos asignar a la variable &RestCode el valor 404 (Not found).

Customizing a REST web service with the API object

1) Parameter customization





URL: …GetAttractionsInfo/GetAttractionsByCountry?CountryId=3

URL: …GetAttractionsInfo/GetAttractionsByCountry?Id=3

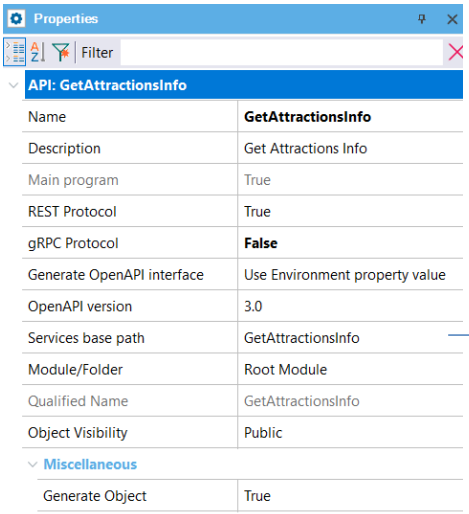
Como dijimos antes, el objeto API nos permite cambiar el nombre de un parámetro. Para eso debemos asignar la propiedad External Name de las variables usadas en los parámetros del servicio expuesto.

En el ejemplo que vimos si vamos a las variables del objeto API GetAttractionsInfo, seleccionamos la variable CountryId y en la propiedad ExternalName escribimos Id.

La URL con la que se invocará al servicio en lugar de usar el nombre CountryId para el parámetro, usará el nombre que definimos como external name y aparecerá la palabra Id en lugar de CountryId.

Customizing a REST web service with the API object

2) Service URL customization



URL:

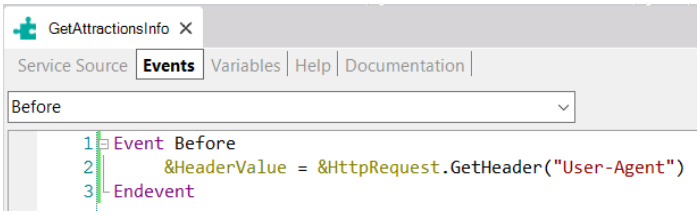
…GetAttractionsInfo/GetAttractionsByCountry?CountryId=3

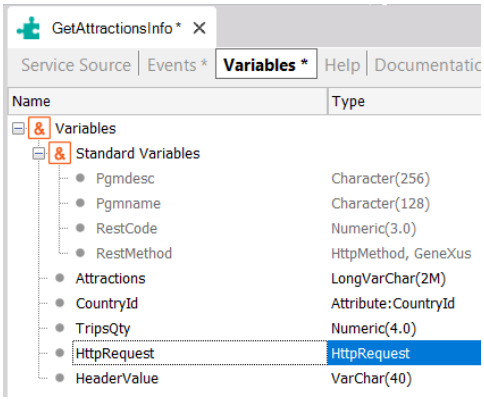
Si queremos cambiar la URL por defecto de un objeto API, podemos hacerlo cambiando el valor de su propiedad Services base path.

Por defecto esta propiedad tiene el nombre del objeto API, si ponemos otro texto, ese texto aparecerá en la URL en lugar del nombre del objeto.

Customizing a REST web service with the API object

3) Reading parameters from HTTP header Request





Otra cosa que podemos hacer con el objeto API, es leer los datos del encabezado de la invocación a nuestro servicio REST.

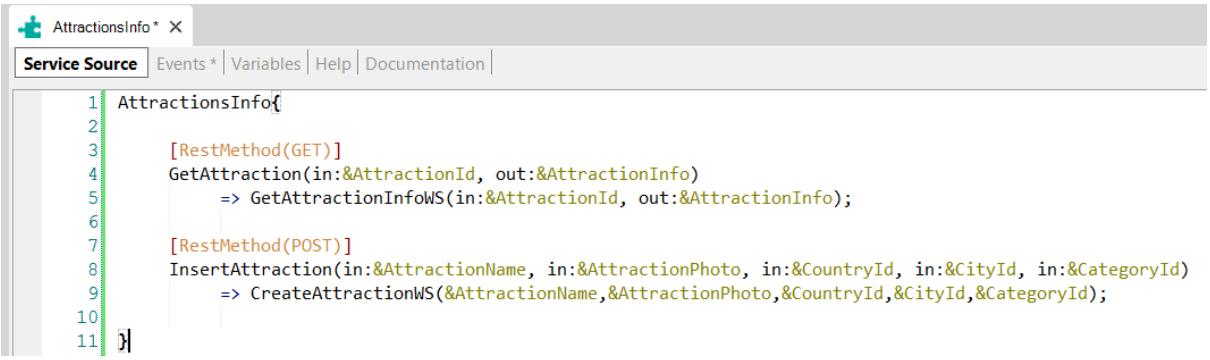
Si en el header HTTP del servicio se recibe algún parámetro, usando el evento Before, se puede leer el header que es enviado por la aplicación que invoca al servicio cuando hace el Request.

La expresión de texto que se use entre los paréntesis del método GetHeader, determina el nombre del header HTTP. Estos headers son invisibles al usuario final y definen cómo la información es enviada o recibida del servicio, por ejemplo quién llama al servicio, nombre del host, credenciales de acceso, el tipo de conexión, cookies, etc.

En este caso el header tiene de nombre “User-Agent”, y el valor devuelto es una cadena de texto que identifica al agente de usuario ante el servidor, en otras palabras el nombre de la aplicación cliente que está invocando al servicio.

Customizing a REST web service with the API object

4) HTTP access methods customization



Con el objeto API, si incluimos una anotación previa a la definición del servicio, podemos especificar el método HTTP de acceso.

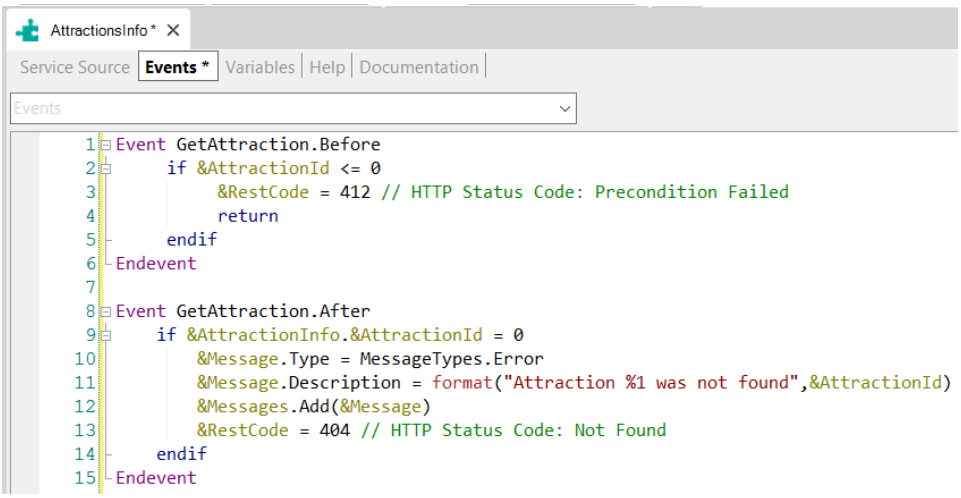
Supongamos que tenemos dos servicios, uno de nombre GetAttraction para obtener datos de una determinada atracción turística mediante un data provider y otro de nombre InsertAttraction, que nos permite crear una atracción nueva a la base de datos mediante un procedimiento.

Por ejemplo en el caso del servicio GetAttraction, estamos especificando que se usará el método GET para invocarlo, ya que estamos obteniendo información a través del servicio. La anotación va entre paréntesis rectos y adentro tendrá RestMethod y entre paréntesis el método HTTP a usar, en este caso GET.
En el ejemplo, el método interno que se ejecutará es el data provider GetAttractionInfoWS. Como los DP devuelven información son invocados por defecto como GET y con la anotación estamos haciendo esto en forma explícita.

El método InsertAttraction que inserta una atracción en la base de datos, lo invocamos con un POST, mediante la anotación RestMethod(POST).
El objeto CreateAttractionWS por ser un procedimiento, puede invocarse como GET o como POST, en este caso lo invocamos explícitamente como POST porque estará grabando información en la base de datos.

Customizing a REST web service with the API object

5) Service response customization



Mediante el uso de la variable estándar &RestCode del objeto API, podemos cambiar el código de estado HTTP.

Esto lo hacemos en los eventos y podemos hacerlo antes de invocar al servicio (en un evento Before) para setear un valor en caso de que ya se determine que no se puede invocar al servicio porque los datos recibidos no son correctos.
En el código estamos asignando el HTTP Status Code 412 Precondition Failed, si el AttractionId recibido como parámetro no es válido.

Y también lo podemos hacer luego de ejecutar al servicio, para establecer el resultado de la operación, por ejemplo si no se encontró la información requerida. En el ejemplo, la información recibida tiene el campo AttractionId vacío, por lo que no se encontró información de una atracción con el AttractionId pasado por parámetro, así que asignamos la variable &RestCode con el valor 404 (Not Found).

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