26/09/2018

Nodejs REST API

Stored Procedures Bridge

Objetivo:

Se agrega una nueva funcionalidad a la API que consiste en la capacidad solicitar ejecuciones de Stored Procedures de la base de datos.

Implementación:

Se ha creado un nuevo endpoint para procesar estos “request”:

http://<miserver>/api/proc/<nombre>

El verbo a ejecutar es POST.

Los parámetros de ejecución se envían en el body.

<nombre>: nombre del “servicio” definido en la metadata.

Metadata:

Como los “recursos” aplicables a la REST API, los stored procedures también requieren metadata para efectuar la validación de los parámetros recibidos y el ordenamiento de los mismos en la invocación al SP (recordemos que los parámetros de un llamado a un SP son posicionales).

La diferencia importante con la metadata de “recursos” es que los parámetros no necesitan corresponderse con nombres de columnas de tablas de la base de datos.

Cada archivo de metadata para stored procedures puede contener “n” definiciones de parámetros, es decir metadata para más de un SP. Cada archivo comprende “lógicamente” a los procedimientos que afecten a un recurso en un sentido amplio.

La metadata para SPs es un archivo JSON con el siguiente formato:

```
{
    "resource”:<string> (nombre del recurso)
    "sps":<array> (de definiciones de SP)
      [
        {
            "servicio": <string> (nombre “externo” del SP),
            "type": <string> “C”,”R”,”U”, “D” (tipo de operación del SP)
            "sp": <string> (nombre “interno”, en la base, del SP),
            "params": <array> (de parámetros ordenados, para ejecución)
            [
                {
                  "name":<string> (nombre del parámetro),
                  "required": <string> “Y”/”N” (si es requerido o no)
                }, …
            ]
        }, …
    ],
    "columns":<arreglo> (de descriptores de columnas, parámetros)
      [
        {
            "name": <string> (nombre del parámetro)
            "type": <string> (tipo de dato, idem tablas),
            "length": <int> (longitud),
            "decimals": <int> (decimales
        },…
      ]
}
```

El arreglo “columns” permite las definiciones de tipo y longitud para los parámetros, mientras que el arreglo “params” indica, para cada stored procedure, que columna debe incluirse, en qué orden y la obligatoriedad de informarlo en el request.

Se adjunta un ejemplo de metadata para cuatro SPs sobre la tabla “producto” (producto_sp.json).

Los archivos de metadata de definición de SPs residirán en la misma carpeta (.\metadata) que los archivos de metadata de tablas.

Carga de Metadata:

Esta versión cambia el procedimiento de carga de la metadata, eliminando la necesidad de tocar el código del server cada vez que se agrega o quita un archivo de metadata.

Para esto se crea un catálogo de metadata en la misma carpeta (.\metadata), con el nombre “meta_catalogo.json”.

Este archivo de catálogo consiste de un arreglo bajo el nombre “catalog” donde cada elemento es un objeto con dos miembros:

- name: nombre del archivo de metadata a cargar (sin extensión)
- type: tipo de metadata (“T”:tabla, “V”:vista(view) y “S”:stored procedure)

Una vez creada la metadata para una tabla, vista o grupo de SPs, deberácrearse la entrada en el catálogo para que sea cargado en el server, cada vez que éste se reinicie.

Se envía el ejemplo actualizado para el conjunto de ejemplos.

De los Stored Procedures:

Los SPs deberán construirse con una estructura de códigos de retorno consistentes con el esquema general de retorno de la API, tanto para que la misma sepa en qué orden retornar estos códigos como para que la aplicación cliente pueda dialogar de una forma independiente de los procesos subyacentes.

Se adjuntan cuatro Stored Procedures (Insert, Delete, Update y Retrieve) para la tabla “productos” como ejemplo de estructura y retornos.

Estos SPs son coincidentes con el archivo de definición de los mismos (producto_sp.json)