Diseño de URL para Filtros Avanzados

REGER

Mapeo de URL a Sentencias SQL

MTI Consultoría
Versión: 2018.12.17

Control del Documento

Rol Propietario Líder de Arquitectura

Roles involucrados Arquitecto, Líder Técnico, Consultor

Localización

Confidencialidad La información contenida en este documento no deberá ser divulgada, duplicada o dada a conocer, parcial o totalmente, fuera de la Organización sin una autorización por escrito.

Registro de cambio

Versión	Autor	Referencia de cambios
2020.06.12	Cristian Ariel Aguilar Santana	Creación del documento.

Revisores

Versión	Revisor	Posición
2020.07.23	Juan Carlos Piña Lugo	Consultor Snr.

Referencias

Nombre	Localización	Descripción

Contenido

Introducción

Objetivos

El presente documento pretende mostrar una información de alto nivel del diseño de los parámetros para filtros avanzados y su mapeo en sentencias SQL. Mantener un diseño simple y transparente para los usuarios y desarrolladores.

Estándares y diseño de la URL

¿Qué es URL?

URL son siglas en inglés de Uniform Resource Locator que en español se traduce como Localizador Uniforme de Recursos, esto significa que es una dirección específica y única que se asigna a un recurso disponible en la red con el objetivo de que puedan ser identificados.

Parámetros en la URL

Un parámetro es una propiedad que se agrega directamente a la URL con el signo de interrogación (?). Cada parámetro debe contener un nombre y el valor separados por el signo igual (=).

Ejemplo:

<protocolo>://<urlPersonalizada>/modulo/catalogo?param=value

En caso de tener más parámetros, agregue un símbolo et (&) entre cada uno:

<protocolo>://<urlPersonalizada>/modulo/catalogo?param=value&param2=value2

Nota: El orden de los parámetros es irrelevante, pero si se quiere seguir un estándar de buenas prácticas si es necesario un orden para que sea legible tanto desarrolladores como usuarios.

El nombre de los parámetros está compuesto por el código ASCII y pueden tener guión bajo (_) o guión (-) en cualquier posición que no sea la primera. Por estándar de buenas prácticas en API REST estos parámetros cuando están unidos por dos o más palabras deben de usar el estándar Camel case.

Ejemplo:

<MALA PRÁCTICA>

?type_key=value

<MALA PRÁCTICA>

?type-key=value

<BUENA PRÁCTICA>

?typeKey=value

Codificación de la URL

Al enviar un parámetro a la URL, este debe ser codificado en tiempo de ejecución. La codificación de los parámetros garantiza el escapado de caracteres inválidos y los sustituye por % , seguido de su número en hexadecimal.

Un ejemplo claro de un error por caracteres inválidos:

<INCORRECTO>

?param="%p%"

<CORRECTO>

?param="%25p%25"

Es muy importante hacer pruebas en postman para ver si su cadena es correcta por URL o necesita una codificación antes de mandarla al servicio.

En esta página se puede apreciar el carácter y su valor en %<Valor en Hexadecimal>:

https://www.urlencoder.io/learn/

El estándar de buenas prácticas para los valores al momento de enviar el parámetro a la URL, es:

- **Codifique cada valor por separado**

  Correcto

  ?param=Av%20Roja,Calle%20Viva

  Incorrecto

  No se debe de codificar la coma porque nos sirve para separar valores.

  ?param=Av%20Roja%2CCalle%20Viva

- **No incluir espacios entre valores**

  Correcto

  ?param=Juan%20Carlos%20Piña

  Incorrecto

  ?param=Juan Carlos Piña

Errores de caracteres especiales en Postman

Si existe algún carácter ilegal al momento de hacer una petición es muy importante saber que Postman no acepta algunos caracteres por URL, un ejemplo claro es el porcentaje (%) que debe cambiarse por (%25).

Estructura del parámetro de filtrado avanzado

Estructura de la consulta para filtros avanzados

El nombre del parámetro que se debe transmitir del frontend al backend es (q) y su valor debe ser un array de objetos. Por ejemplo

?q=[{ "columnName":value, "filter": [ ] }]

Estructura del objeto

Existen dos estructuras similares, pero diferentes. La primera es el padre, que es la columna donde se efectuarán los filtros correspondientes.

Columna Padre

Este objeto es el principal en la estructura y contiene campos muy simples como:

columnName: Nombre de la columna que se aplicarán los filtros necesarios. Debe estar entre comillas el valor.

filter: Es un object Array de filtros especiales que se aplican a la columna.

operator: Es un operador lógico AND u OR. Este solo se aplica si existe en seguida otra columna padre con su filtro, en caso de no haber un siguiente elemento puede que marque un error la estructura. Por default es AND.

```
[{
"columnName":value,
"filter": [ ],
"operator": value
}]
```

Filtro

Este objeto contiene los filtros que se van aplicar y tiene una estructura similar a columna padre, sin embargo, existen ciertas diferencias como:

columnName: Nombre de la columna que se aplicarán los filtros necesarios. Debe estar entre comillas el valor. Por default toma el valor de la columna padre.

comparison: Contiene un operador de comparación.

value: Es el valor con el que se va a comparar, acepta valores de tipo String (""), Integer (1) y arreglos de ambos tipos de datos ([1,2,3] y ["USA", "MEXICO"]).

operator: Es un operador lógico AND u OR. Este solo se aplica si existe en

seguida otro filtro, en caso de no haber un siguiente elemento puede que marque un error la estructura. Por default es AND.

```
filter = [{
   "columnName":value,
   "comparison": value,
   "operator":value
}]
```

Operadores lógicos y de comparación

La siguiente lista muestra los operadores que se encuentran disponibles y su valor que se debe de mandar por URL en los filtros avanzados.

Operadores Lógicos válidos para el campo operator:

Operador	Valor
AND	$and
OR	$or

Operadores de comparación válidos para el campo de comparison en el filtro:

Comparación	Valor
>	$gt
>=	$gte
<	$lt
<=	$lte
=	$eq
<>	$ne
LIKE	$like
NOT LIKE	$nlike
BETWEEN	$between
IN	$in
NOT IN	$nin
IS NULL	$isNull
NOT NULL	$notNull

Mapeado de la URL a SQL

Filtros Simples

Los tipos de filtrado de información que el usuario tiene disponible son de dos tipos: filtros generales o simples y filtros avanzados.

Los filtros simples pueden ser una igualación o tener una sentencia WHERE lo más simple que se pueda. Un ejemplo:

El cliente X requiere que obtengas los registros cuyo importe sea mayor a 5000 en la tabla base template.

SQL:

```
SELECT *
  FROM dev_base_template
WHERE amount > 5000
```

URL:

```
?q=[ {
 "columnName":"amount",
 "filter":[
  {
        "comparison":"$gt",
        "value":5000
  }]
}]
```

Otra manera de aprovechar la estructura del json para tener un rango con un campo, es poner más filtros sin el columnName en el filtro, puesto que se agregará al del padre por defecto.

SQL:

```
SELECT *
  FROM dev_base_template
WHERE (base_template_key > 3 AND base_template_key < 12 )
```

URL:

```
?q=[ {
 "columnName":"base_template_key",
 "filter":[
  {
        "comparison":"$gt",
```

```
        "value":3
  },
  {
        "comparison":"$lt",
        "value":12
  }]
}]
```

También se pueden anexar operadores lógicos o consultas que ambas condiciones se cumplan, por ejemplo:

La empresa Z requiere que se obtengan los registros dónde la fecha de cumpleaños sea '2020-06-03' y que su clave única sea mayor que 5.

SQL:

```
SELECT *
  FROM dev_base_template
WHERE (birthday = '2020-06-03' AND base_template_key > 5 )
```

URL:

```
?q=[ {
 "columnName":"birthday",
 "filter":[
  {
        "comparison":"$eq",
        "value":"2020-06-03"
  },
  {
        "columnName": "base_template_key",
        "comparison":"$gt",
        "value":5
  }]
}]
```

Si quieres cambiar el operador lógico por default AND por OR solo le tienes que agregar el campo operator a la primera comparación o filtro, por ejemplo:

SQL:

```
SELECT *
  FROM base_template
WHERE (status = '2020-06-03' OR base_template_key > 5 )
```

URL:

```
?q=[ {
 "columnName":"birthday",
 "filter":[
  {
        "comparison":"$eq",
        "value":"2020-06-03",
       "operator": "$or"
  },
  {
        "columnName": "base_template_key",
        "comparison":"$gt",
        "value":5
  }]
}]
```

Filtros avanzados

Los filtros avanzados pueden llegar a ser un poco más complejos de acuerdo a la necesidad del cliente. Un ejemplo:

Los clientes de la empresa “Y” quieren hacer un reporte que primero se cumpla que la llave primaria sea mayor de 3 y menor e igual que 12, después que se cumpla debe de filtrar por fecha de última modificación donde sea igual a ‘2020-06-03’.

SQL:

```
SELECT *
  FROM dev_base_template
WHERE
((base_template_key > 3 AND base_template_key <= 12 )  AND
(last_modified = ‘2020-06-03’))
```

URL:

```
?q=[ {
 "columnName":"base_template_key",
 "filter":[
  {
        "comparison":"$gt",
        "value":3
  },
  {
        "comparison":"$lte",
        "value":12
  }]
},
{
 "columnName":"last_modified",
```

```
 "filter":[
   {
         "comparison":"$eq",
         "value":"2020-06-03"
   }]
}]
```

Paréntesis un factor importante

Los paréntesis son un factor importante en la construcción de filtros complejos y se van creando conforme al json. El nivel de paréntesis es cuando tienes tu primer objeto columna padre.

?q=[{ }] esto es igual a WHERE ( )

?q=[{ columnaPadre1 }, { columnaPadre2 }] esto es igual a WHERE ((comparaciónColPadre1) AND (comparaciónColPadre1))

Ordenamiento por columnas

El ordenamiento es un factor muy importante y se le puede indicar en la URL con el parámetro order_by. Los valores que puede recibir este parámetro es el nombre de la columna y la forma en que se va a ordenar, es decir, si es descendente o ascendente. Por defecto es ascendente, por ejemplo:

- **Ordenar el código de forma ascendente**

  (Correcto)

  /simple-form?order_by=code

  (Correcto)

  /simple-form?order_by=code asc

- **Ordenar el código de forma descendente**

  /simple-form?order_by=code desc

  Otra ventaja de este parámetro es que puedes combinar columnas e indicar la forma en cómo se van a ordenar. Por ejemplo:

  /simple-form?order_by=code desc, base_template_key desc

  o

  /simple-form?order_by=code desc,base_template_key asc

Paginación LIMIT y OFFSET

Otros valores que puedes agregar son los de paginado para fragmentar la información en pequeños lotes y que el costo de búsqueda sea más barato hablando en cuestion de procesamiento.

En términos más simples el offset indica el corrimiento de filas en el que inicia tu paginación actual y el limit indica el límite de registros que

esperas recibir.

Los parámetros correspondientes tienen el mismo nombre.

/simple-form?offset=100&limit=500

La consulta anterior te trae los registros a partir de 100 y su límite será hasta 500. Los valores por default que contiene son offset = 0 y limit = 100. Entonces si no le asignas valores te traerá solamente los primeros 100 registros.

Combinaciones de parámetros

La combinación de parámetros se puede realizar en cualquier momento con los filtros y en cualquier orden, pero para seguir un estándar y sea más fácil la lectura de la URL es necesario tener en cuenta que el orden sí importa en este caso.

El orden de los parámetros debe ser: el filtrado especial, el orden de las columnas, offset y limit, tal como se muestra en el siguiente ejemplo:

/simple-form?q=[{}]&order_by=value&offset=value&limit=value

Tabla de Mapeo de SQL A URL

SQL STATEMENT	URL STATEMENT

SELECT
code,
description,
amount
FROM
dev_base_template
WHERE
(amount= 500)
	[{
	"columnName":"amount",
	"filter":[
	{
	"comparison":"$eq",
	"value":500
	}]
	}]

SELECT
code,
description,
amount
FROM
dev_base_template
WHERE
(amount> 500 OR amount< 1000)
	[{
	"columnName":"amount",
	"filter":[
	{
	"comparison":"$eq",
	"value":500,
	"operator": "$or"
	},
	{
	"comparison":"$lt",
	"value":1000
	}]
	}]

SELECT
code,
description,
amount
FROM
dev_base_template
WHERE
((amount > 500 AND amount < 1000)
AND
(description LIKE '%Pepe%'))
	[{
	"columnName":"amount",
	"filter":[
	{
	"comparison":"$gt",
	"value":500
	},
	{
	"comparison":"$lt",
	"value":1000
	}]
	},
	{
	"columnName":"description",
	"filter":[
	{
	"comparison":"$like",
	"value":"%pepe%"
	}]
	}]

SELECT
code,
description,
amount
FROM
dev_base_template
WHERE
((amount> 500 OR amount < 1000)
AND
(description LIKE '%Pepe%'))
	[{
	"columnName":"amount",
	"filter":[
	{
	"comparison":"$gt",
	"value":500,
	"operator": "$or"
	},
	{
	"comparison":"$lt",
	"value":1000
	}]
	},
	{
	"columnName":"description",
	"filter":[
	{
	"comparison":"$like",
	"value":"%pepe%"
	}]
	}]

SELECT
code,
description,
amount
FROM
dev_base_template
WHERE
((amount> 500 OR amount < 1000)
OR
(description LIKE '%Pepe%'))
	[{
	"columnName":"amount",
	"filter":[
	{
	"comparison":"$gt",
	"value":500,
	"operator": "$or"
	},
	{
	"comparison":"$lt",
	"value":1000
	}],
	"operator": "$or"
	},
	{
	"columnName":"description",
	"filter":[
	{
	"comparison":"$like",
	"value":"%pepe%"
	}]
	}]

SELECT
code,
description,
amount
FROM
dev_base_template
WHERE
(amount BETWEEN 100 AND 200)
	[{
	"columnName":"amount",
	"filter":[
	{
	"comparison":"$between",
	"value":[100, 200]
	}]
	}]

SELECT
*****
FROM
dev_base_template
WHERE
(code IN (‘C-001’, ‘C-005’, ‘C-009’))
	[{
	"columnName":"code",
	"filter":[
	{
	"comparison":"$in",
	"value":[“C-001”, ”C-005”, “C-009”]
	}]
	}]

SELECT
code,
description
FROM
dev_base_template
WHERE
(code LIKE ‘%T%’ OR description LIKE ‘%Re%’)
	[{
	"columnName":"code",
	"filter":[
	{
	"comparison":"$like",
	"value":’%T%’,
	"operator": “$or**”**
	},
	{
	"columnName": "description"
	"comparison":"$like",
	"value":”%Re%”
	}]
	}]

SELECT
code,
description
FROM
dev_base_template
WHERE
(code IS NULL)
	[{
	"columnName":"code",
	"filter":[
	{
	"comparison":"$isNull"
	}]
	}]

Practicar un rato

Ejercicio

Los desarrolladores tienen la siguiente consulta en SQL:

```
SELECT base_template_key,
       code,
       description,
       birthday,
       last_modified
 FROM dev_base_template
WHERE ((last_modified BETWEEN '2020-06-03'
                          AND '2020-06-11')
       AND (    code LIKE '%001%'
            OR base_template_key <= 5
            AND icon_class IN ('fa fa-toggle-on')))
```

¿Cómo sería en URL?

Respuesta

```
[
 {
        "columnName":"last_modified",
        "filter":[
        {
        "comparison":"$between",
        "value":[
        "2020-06-03",
        "2020-06-11"]
        }
        ]
 },
 {
        "columnName":"code",
        "filter":[
        {
        "comparison":"$like",
        "value":"%001%",
        "operator":"$or"
        },
        {
        "columnName":"base_template_key",
        "comparison":"$lte",
        "value":5
```

```
            }]
        },
        {
            "columnName":"icon_class",
            "filter":[
            {
            "comparison":"$in",
            "value": ["fa fa-toggle-on"]
            }]
        }]
```