Filtering collections

When requesting a collection it's pobbile to filter the output. The filtering system is quite flexible and allows to contstruct complex queries. The module translates your query to an actual WHERE clause that is added to the database query that is executed to get your collection. In that sense it's best to see the jsonapi filtering system as a query builder. You will find some basic and advanced examples below.

Basic setup and notation

To add a filter to a query you add an array to the querystring of your request. There are two available notations. There's a common notation and a short version. The short notation allows you to add a single value filter to a single field. Below is the most basic example in both notations.

Array Keys in italics are keywords required by jsonapi module

Short notation

```
GET drupal8.dev/api/node/article?_format=api_json
&filter[nid][value]=1
```

Common notation

```
GET drupal8.dev/api/node/article?_format=api_json
&filter[nid-filter][condition][field]=nid
&filter[nid-filter][condition][value]=1
```

As you can see for really basic requests it's easier to add a short notated

filter. If you need more functionality you will probably use the common notation

The common filter notation is an array where each key can be a different filter. To allow grouping the filters require an ID (filter[filter-id]). This ID can be numeric or any string. For demonstration I'll use an appropriate string as filter id.

Filter conditions

When adding a filter condition you can add the following parameters

- field: The field name, or nested field name (See example 2).
- value: The value as a single string / integer or an array
- operator: The operator you want this filter to use. (=, <, >, <>, IN, NOT IN, BETWEEN) (see example 3)
- group: The group this filter is a child of. (see example 4)

Grouping filters

It's possible to add multiple filters and create complex groups with them as you'd be able to do with a database query.

When adding a filter you can add a group key and a conjunction key. The conjunction key defines wether the child filters of this group will we combined by AND or by OR.

filter[filter-id][group][conjunction] = AND|OR

It's also possible to add a group to another group. filter[filter-id][group][group] = "parent-group-id"

For more see example 4 and 5

Examples

1. Only get nodes that are published

A very common scenario is to only load the nodes that are published. This is a very easy filter to add.

```
SHORT
filter[status][value]:1

NORMAL
filter[status-filter][condition][field]:status
filter[status-filter][condition][value]:1
```

2. Nested Filters: Get nodes created by user admin

It's possible to filter on fields from referenced entities like the user, taxonomy fields or any entity reference field. You can do this easily but just using the the following notation. reference_field.nested_field. In this example the reference field is uid for the user and name which is a field of the user entity.

```
SHORT
filter[uid.name][value]:admin

NORMAL
filter[name-filter][condition][field]:uid.name
filter[name-filter][condition][value]:admin
```

3. Filtering with arrays: Get nodes created by users [admin, john].

You can give multiple values to a filter for it to search in. Next to the field and value keys you can add an operator to your condition. Usually it's "=" but you can also use "IN", "NOT IN", ">", "<", "<>", BETWEEN".

For this example we're going to use the IN operator. Not that I added two square brackets behind the value to make it into an array.

```
NORMAL
filter[name-filter][condition][field]:uid.name
filter[name-filter][condition][operator]:IN
filter[name-filter][condition][value][]:admin
filter[name-filter][condition][value][]:john
```

4. Grouping filters: Get nodes that are published and create by admin.

Now let's combine some of the examples above and create the following scenario. WHERE user.name = admin AND node.status = 1;

```
filter[and-group][group][conjunction]:AND
filter[name-filter][condition][field]:uid.name
filter[name-filter][condition][value]:admin
filter[name-filter][condition][group]:and-group
filter[status-filter][condition][field]:status
filter[status-filter][condition][field]:1
filter[status-filter][condition][group]:and-group
```

You don't really have to add the and-group but I find that a bit easier usually.

5. Grouping grouped filters: Get nodes that are promoted or sticky and creatd by admin

Like mentioned in the grouping section, you can put groups into other groups. WHERE (user.name = admin) AND (node.sticky = 1 OR node.promoted = 1)

To do this we put sticky and promoted into a group with conjuction OR. Create a group with conjunction AND and put the admin filter, and the promoted/sticky OR group into that.

```
# Create an AND and an OR GROUP
filter[and-group][group][conjunction]:AND
filter[or-group][group][conjunction]: OR

# Put the OR group into the AND GROUP
filter[or-group][group][group]: and-group

# Create the admin filter and put it in the AND GROUP
filter[admin-filter][condition][field]:uid.name
filter[admin-filter][condition][value]:admin
filter[admin-filter][condition][group]:and-group

# Create the sticky filter and put it in the OR GROUP
filter[sticky-filter][condition][field]:sticky
filter[sticky-filter][condition][field]:1
filter[sticky-filter][condition][group]:sticky-filter

# Create the promoted filter and put it in the OR GROUP
filter[promote-filter][condition][field]:promote
filter[promote-filter][condition][field]:1
filter[promote-filter][condition][group]:and-group
```