CAP API - Filtering

Filtering

Filtering is a mechanism using which CAP API users can filter out the arrays in any JSON document.

For example in "profile" document the CAP API user might be interested in only getting back publications that are featured instead of all publications. The CAP API user could be requesting all profiles in an org or could be requesting one particular profile based on sunetId - in either of these cases the CAP API user can specify the filters.

Filter is different from where clause.

Where clause returns back documents (profile, org etc) that match the criteria, which means the total entries returned back will differ. Where as a filter will not affect the number of documents that will be returned to the user. It will affect the number of entries in the arrays based off of the filter criteria. The intention of filters is to decrease the download size of the document.

How will an endpoint look with filter parameter?

```
GET /api/profiles/v1/russ-altman?
filter=fieldName:operator:value
&filter=fieldName:operator:value
&filter=fieldName:operator:value
&filter=fieldName:operator:value
```

or

```
GET /api/profiles/v1/russ-altman?
filter=fieldName:operator:value,fieldName:operator:value,fieldName:
operator:value,filter=fieldName:operator:value
```

The filters are separated by a "," (comma) in the swagger console, and in Postman/CURL end point both comma and & will work.

Format

Each filter has to specify a filedName followed by a colon followed by an operator followed by a colon and followed by a value.

fieldName:operator:value

Validations will be performed to see if the filter parameter adheres to this format.

FieldName

filter=publications.featured:equals:true

Dot (.) is used for fieldName qualification
The fieldName will be qualified by a dot, the first part of the qualifier always has to be an array, if not - then a validation error will be shown to the user.

Example:
filter=names:equals:John

The full fieldName will be split up into a array name and the path to the attribute. This will be validated against the Json schema and if that particular attribute is not present with the specified path under the specified array - then a validation error will be shown to the user.

Example:

```
filter=names.legal.firstName:equals:John
filter=publications.featurewithtypo:equals:true
```

Validations

As of right now CAP API will not support filtering on all arrays, because of index constraints etc,so if the end-user specifies a fieldName which is not in the CAP API AllowedProfileAttributesForFiltering Enum - then a validation error will be shown to the user informing the end-user on which arrays can be used for filtering.

The array part in the fieldName specified in the filter also has to be present in the whitelist (if there is a whitelist) and if the user specified an array in fieldName in the filter, which is not in the whitelist (if there is a whitelist) - then a validation error will be shown to the user. (the user will get back a JSON with validation error message with status code as BAD REQUEST)
As of right now the Enum has all arrays, but will short listed based off of requirements and performance. Based off of current testing we do not see a performance degradation.

Operator

As of right now CAP API supports only the following operators

1. equals
2. lt
3. gt
4. lte
5. gte
6. in
7. contains
8. exists
9. ne

Validations

For each filter the user has to specify the operator.
And this operator has to be one of the operators that are currently beign supported by CAP API, if it is not supported - then a validation error will be shown to the user.

Value

Internally CAP API will deduce the type of the value based off of the fieldName, and perform the operation by converting the value to that type. If CAP API is not able to deduce for some reason
then it will be considered as "String"

Validations

If the filter specified has the "in" operator then the value should be separated by ";" and there should be at least 2 values separated by "; " and if that is not the case - then a validation error will be shown to the user.

Other generic validations

1. Based on the query performance analysis in Mongodb, CAP API will have a restriction as to the number of filters that can be specified. As of right now CAP API can support only a max of 4 filters.
    1. (Based off of the initial testing if the query has more than 4 filters, the mongoDB query is not returning back).
    2. So if the number of filters is more than 4 - then a error will be shown to the user.
2. Validations are performed on all the filters specified by the user, and CAP API will return back 1 error message - with all the validation errors found on each filter specified.
3. As or right now CAP API's findAll() end point will not support filtering. If the user specifies filter for the findAll end point - then a clear message informing this constraint will be shown to the user.
4. The filtering is in the order that the user specified.

Examples

Example showing - getting profiles (or a single profile) with 1 filter

```
GET /api/profiles/v1/russ-altman?filter=publications.featured:equals:true
```

Example showing - getting profiles (or a single profile) with 2 filters

```
GET /api/profiles/v1/russ-altman?filter=publications.featured:equals:true,
filter=patents.title:equals:Patent 4
```

Example showing - getting profiles (or a single profile) with 4 filters

```
GET /api/profiles/v1/russ-altman?filter=publications.featured:equals:true,
filter=patents.title:equals:Patent 4,
filter=programs.affiliation:equals:FACULTY,
filter=honorsAndAwards.dates:equals:2010
```

Example showing - getting profiles (or a single profile) with filter using 'exists' operator

```
GET /api/profiles/v1/russ-altman?filter=publications.featured:equals:true,
filter=graduateAndFellowshipPrograms.url:exists:true
```

Example showing - getting profiles (or a single profile) with filter using 'like' operator

```
GET /api/profiles/v1/russ-altman?filter=publications.featured:equals:true,
filter=professionalOrganizations.role:like:gene
```

Example showing - getting profiles (or a single profile) with filter using 'ne' operator

```
GET /api/profiles/v1/russ-altman?filter=publications.featured:equals:true,
filter=professionalOrganizations.role:ne:Genetics
```

Example showing - getting profiles (or a single profile) with filter using 'gt' operator

```
GET /api/profiles/v1/russ-altman?filter=publications.featured:equals:true,
filter=postdoctoralAdvisees.profileId:gt:3000
```

Example showing - getting profiles (or a single profile) with filter using 'gte' operator

```
GET /api/profiles/v1/russ-altman?filter=publications.featured:equals:true,
filter=postdoctoralAdvisees.profileId:gte:3000
```

Example showing - getting profiles (or a single profile) with filter using 'lt' operator

```
GET /api/profiles/v1/russ-altman?filter=publications.featured:equals:true,
filter=postdoctoralAdvisees.profileId:lt:4000
```

Example showing - getting profiles (or a single profile) with filter using 'lte' operator

```
GET /api/profiles/v1/russ-altman?filter=publications.featured:equals:true,
filter=postdoctoralAdvisees.profileId:lte:4000
```

Example showing - getting profiles (or a single profile) with filter using 'in' operator

```
GET /api/profiles/v1/russ-altman?filter=publications.featured:equals:true,
filter=postdoctoralAdvisees.profileId:in:3000;4000
```

Where clause

How will an endpoint look with where-clause parameter?

GET /api/profiles/v1/....?where=fieldName:operator:value

?where=profileId:eq:4706
&where=professionalOrganizations.startYear.text:gt:2016
&where=profileId:eq:4706
&matches=[all/any]

When using this parameter - each parameter is separated by a comma, and each parameter should have a fieldName followed by a colon, followed by an operator, followed by a colon again followed by a fieldValue.

1. fieldName:operator:fieldValue,
2. fieldName:operator:fieldValue, fieldName:operator:fieldValue, fieldName:operator:fieldValue

Validations

1. If the fieldValue has a ":" which is a delimiter for a single where-clause element - then a validation error will be shown to the user.
    1. Example: http://localhost:8081/cap-api/api/profiles/v1/russ-altman?where=publicationTags.weight:in:13; 18
    2. Example : http://localhost:8081/cap-api/api/profiles/v1/russ-altman?where=publications.featured:equals:true&where=patents.title:equals:Patent 4
2. For each where-clause the user has to specify the operator. And this operator has to be one of the operators that are currently beign supported by CAP API, if it is not supported - then a validation error will be shown to the user.
3. The fieldName will be qualified by a dot, the first part of the qualifier does not necessarily have to be a array.
4. The fieldName with all the qualifiers - will be validated against the schema and if that particular attribute is not present with the specified path in the jsonschema - then a validation error will be shown to the user.
5. As of right now CAP API will support where-clause on all fields, but based off of analysis of index constraints etc, we might need to restrict it.
6. The fieldName specified in the where-clause need not be present in the whitelist (if there is a whitelist).
7. Based on the query performance analysis in Mongodb CAP API will have a restriction as to the number of where-clauses that can be specified.
8. Validations are performed on all the where-clauses specified by the user, and CAP API will return back 1 error message - with all the validation errors found on each where-clause specified.
9. If the where-clause specified has the "in" operator then the value should be separated by ";" and there should be at least 2 values separated by "; " and if that is not the case - then a validation error will be shown to the user.
10. As or right now CAP API's findAll() end point will support where-clause.