Strategy: DOCUMENTS_BY_SEARCH_FT_PAGED

Use this strategy to select documents from the database using full-text search.

Allowed methods

- GET
- POST

Parameters in routes.groovy:

- `databaseName(String databaseName)`: name of the database to read data from, use "`server!!path/database.nsf`" format; when omitted, current database is used

Both GET and POST calls expect these parameters to be passed:

- `count`: number or records to return, default is 10
- `start`: position in the view to start from, default value is 1
- `search`: full-text query to be executed

POST variant

```
router.POST('topics/search') {
   strategy(DOCUMENTS_BY_SEARCH_FT_PAGED) {
   }
   mapJson 'id', json:'id', type:'STRING', isformula:true, formula:'@DocumentUniqueID'
   mapJson "date", json:'date_created',type:'DATETIME',isformula:true, formula:'@Created'
   mapJson "topic", json:'topic', type:'STRING'
   mapJson "author", json:'author', type:'STRING',isformula:true,formula:'@Name([CN]; @Author)'
}
```

With route defined above, POSTing following JSON

```
{
    "search": "10000*",
    "start": 1,
    "count": 5
}
```

to URL `http://server.name/path-to/db.nsf/xsp/.xrest/topics/search` will return all documents from database containing "20000*", such as

```
{
    "entries": [
        {
            "date_created": "2017-03-03T11:01+00:00",
            "author": "Martin Jinoch",
            "id": "E044D66CFEA6735AC12580D8003F43C7",
            "topic": "10000"
        },
        {
            "date_created": "2017-03-03T11:01+00:00",
            "author": "Martin Jinoch",
            "id": "66671AF824F5A3D3C12580D8003DE435",
            "topic": "100002"
        },
        {
            "date_created": "2017-03-03T11:01+00:00",
            "author": "Martin Jinoch",
            "id": "F52C541EC8EF970EC12580D8003DE437",
            "topic": "100000"
        },
        {
            "date_created": "2017-03-03T11:01+00:00",
            "author": "Martin Jinoch",
            "id": "01ED9C403BE0ED4FC12580D8003DE436",
            "topic": "100001"
        },
        {
            "date_created": "2017-03-03T11:01+00:00",
            "author": "Martin Jinoch",
            "id": "393C379E0527C246C12580D8003DE434",
            "topic": "100003"
        }
    ],
    "start": 1,
    "total": 11,
    "count": 5
}
```


GET variant

```
router.GET('topics/search/{srchFor}') {
   strategy(DOCUMENTS_BY_SEARCH_FT_PAGED) {
     ftQuery('srchFor') //Available since Beta 4
   }
   mapJson 'id', json:'id', type:'STRING', isformula:true, formula:'@DocumentUniqueID'
   mapJson "date", json:'date_created',type:'DATETIME',isformula:true, formula:'@Created'
   mapJson "topic", json:'topic', type:'STRING'
   mapJson "author", json:'author', type:'STRING',isformula:true,formula:'@Name([CN]; @Author)'
}
```

For GET you can define `ftQuery` placeholder like shown above and use URL like `http://server.name/path-to/db.nsf/xsp/.xrest/topics/search/10000*` or you can be passing the `search` parameter in the URL like this `http://server.name/path-to/db.nsf/xsp/.xrest/topics/search?search=10000*` In this case do not include the placeholder in the router path definition, ie. `router.GET('topics/search')`

Note: `ftQuery` takes precedence, so if you add `?search=` to the URL, it will be ignored.

Format of the returned JSON is identical as for POST variant.

Related articles

- Strategy: CUSTOM
- Strategy: DOCUMENT_BY_UNID
- Strategy: DOCUMENT_FROM_VIEW_BY_KEY
- Strategy: DOCUMENTS_BY_FORMULA

- Strategy: DOCUMENTS_BY_FORMULA_PAGED
- Strategy: DOCUMENTS_BY_SEARCH_FT
- Strategy: DOCUMENTS_BY_SEARCH_FT_PAGED
- Strategy: DOCUMENTS_BY_VIEW
- Strategy: DOCUMENTS_BY_VIEW_PAGED
- Strategy: DOCUMENTS_FROM_VIEW_BY_KEY
- Strategy: DOCUMENTS_FROM_VIEW_BY_KEY_PAGED
- Strategy: VIEWENTRIES
- Strategy: VIEWENTRIES_BY_CATEGORY
- Strategy: VIEWENTRIES_BY_CATEGORY_PAGED
- Strategy: VIEWENTRIES_PAGED