



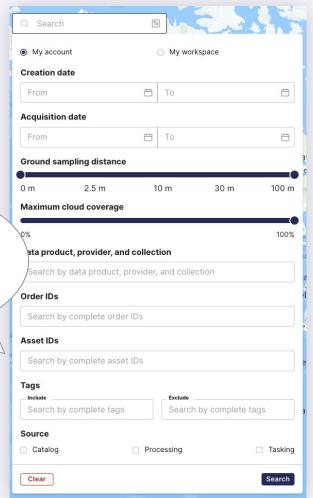
Hi, I'm Noam

- → Senior Front-End Engineer at UP42
- → Based in Berlin
- → My favourite ice cream flavour is Chocolate

Why visualize queries?

We want to make exploring geospatial datasets easier.

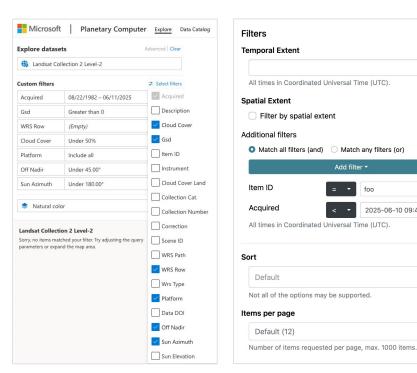
Web interfaces are good for that!



Why visualize queries?

How to translate the form to a query? Unfortunately, there was no JavaScript library to help me...

So I made one.



Planetary Computer

STAC Browser

Add filter ▼

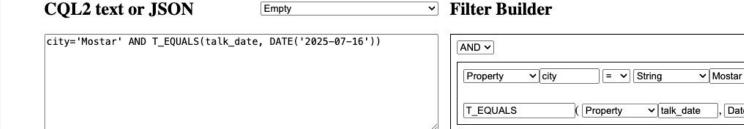
2025-06-10 09:4

0



CQL2-filters-parser library

- Unified way to parse CQL2 Text and JSON encodings
- Runs in **Browsers** and **JavaScript** hosts
- **Extendable** to your needs



Convert to rule

Convert to AND / OR

16.07.2025

Date

CQL2 filters

What is CQL2 and why is it helpful?

STAC vs CQL2

STAC

SpatioTemporal Asset Catalog

a way to organize geospatial data

- STAC Item
- STAC Catalog

Spec

CQL2

Common Query Language

a language to describe filter expressions for spatial and temporal data

Spec

Two encodings

CQL2 comes in two encodings,

Text and JSON

- Query language
- Only operators and operands
- No flow control, loops, recursion

```
// CQL2 Text encoding
city = 'Mostar'
// CQL2 JSON encoding
  "op": "=",
  "args": [
    { "property": "city" },
    "Mostar"
```

Two encodings

Differences:

- Audience
- Usage

```
// CQL2 Text encoding
city = 'Mostar'
// CQL2 JSON encoding
  "op": "=",
  "args": [
    { "property": "city" },
    "Mostar"
```

Building blocks

- **Literals:** strings, numbers, booleans, etc.
- Properties: the variables of CQL2
- **Operators**: logical, comparison, arithmetic
- **Combining**: binary expression, function, etc.
- Spatial: bbox, geometries
- **Temporal:** timestamp, date, interval

All of these nodes expressions in a tree data structure.

```
// CQL2 Text encoding
city = 'Mostar'
// CQL2 JSON encoding
  "op": "=",
  "args": [
    { "property": "city" },
    "Mostar"
```

Parsing Text, Parsing JSON



Parsers

Text → tokenizer → parser → Expression tree

JSON → depth first parser → Expression tree

parse() → Expression tree



Parsers

```
import { parse } from "cql2-filters-parser";
const { encoding, expression } = parse("city='Mostar'");
console.log(encoding); // -> Text
console.log(expression.toText()); // -> "city = 'Mostar'"
console.log(expression.toJSON()); // ->
// {
// op: '=',
// args: [ { property: 'city' }, 'Mostar' ]
// }
```

Visitors Especially for this library:) Welcome

Visitor design pattern

Separates the operation from the object.
Allows defining new operations on data structure.



Visitor design pattern

Each node type has a corresponding visit function:

- Literal → visitLiteralExpression()
- Operator → visitOperatorExpression()
- Property → visitPropertyExpression()
- Binary expression → visitBinaryExpression()

The visitor object is something that implements visit functions

HTML Builder Visitor

The visitor object is something that implements visit functions

```
const HTMLBuilderVisitor = {
 visitLiteralExpression(expr) { /* ToDo */ },
 visitOperatorExpression(expr) { /* ToDo */ },
 visitPropertyExpression(expr) { /* ToDo */ },
 visitBinaryExpression(expr) { /* ToDo */ },
};
const { expression } = parse("city='Mostar'");
const builderForm = expression.accept(HTMLBuilderVisitor);
document.getElementById("builder").appendChild(builderForm);
```



HTML Builder Visitor

```
const HTMLBuilderVisitor = {
 visitLiteralExpression(expr) {
    return createInputElement(expr.value, expr.type);
 },
 visitOperatorExpression(expr) {
    return createOperatorSelectElement(expr.text);
 },
 visitPropertyExpression(expr) {
    return createInputElement(expr.name, "text");
 },
 visitBinaryExpression(expr) {
    const left = expr.left.accept(BuilderVisitor);
    const op = expr.operator.accept(BuilderVisitor);
    const right = expr.right.accept(BuilderVisitor);
    return createBinaryPairElement(left, op, right);
 },
};
const builderForm = expression.accept(HTMLBuilderVisitor);
```



HTML Builder Visitor

```
const HTMLBuilderVisitor = {
  visitLiteralExpression(expr) {
    return cre
  visitOperat
                    CQL2 Text
    return ci
                    city='Mostar' AND talk_duration > 20
  visitProper
    return ci
  },
  visitBinary
    const let
                    Filter Builder
    const op
    const ric
                     city
                                 =
                                          Mostar
                                                      and
                                                               talk duration
    return ci
```

const builderForm = expression.accept(HTMLBuilderVisitor);

Wrapping up Something funny to add here



CQL2-filters-parser library

- → Unified way to parse CQL2 Text and JSON encodings
- → Runs in **Browsers** and **JavaScript** hosts
- **Extendable** to your needs



GitHub repository

CQL2 text or JSON Empty Filter Builder



AND 🗸					Conv	ert to rule
Property	∨ city	= •	String	✓ Mostar	Convert to A	ND / OR
T_EQUALS		(Property	✓ [talk_	date , Date	~ [16.07.2025	

CQL2 Playground - https://noamra.github.io/ogc-cql2-filters
GitHub repository - https://github.com/NoamRa/ogc-cql2-filters
npm i cql2-filters-parser