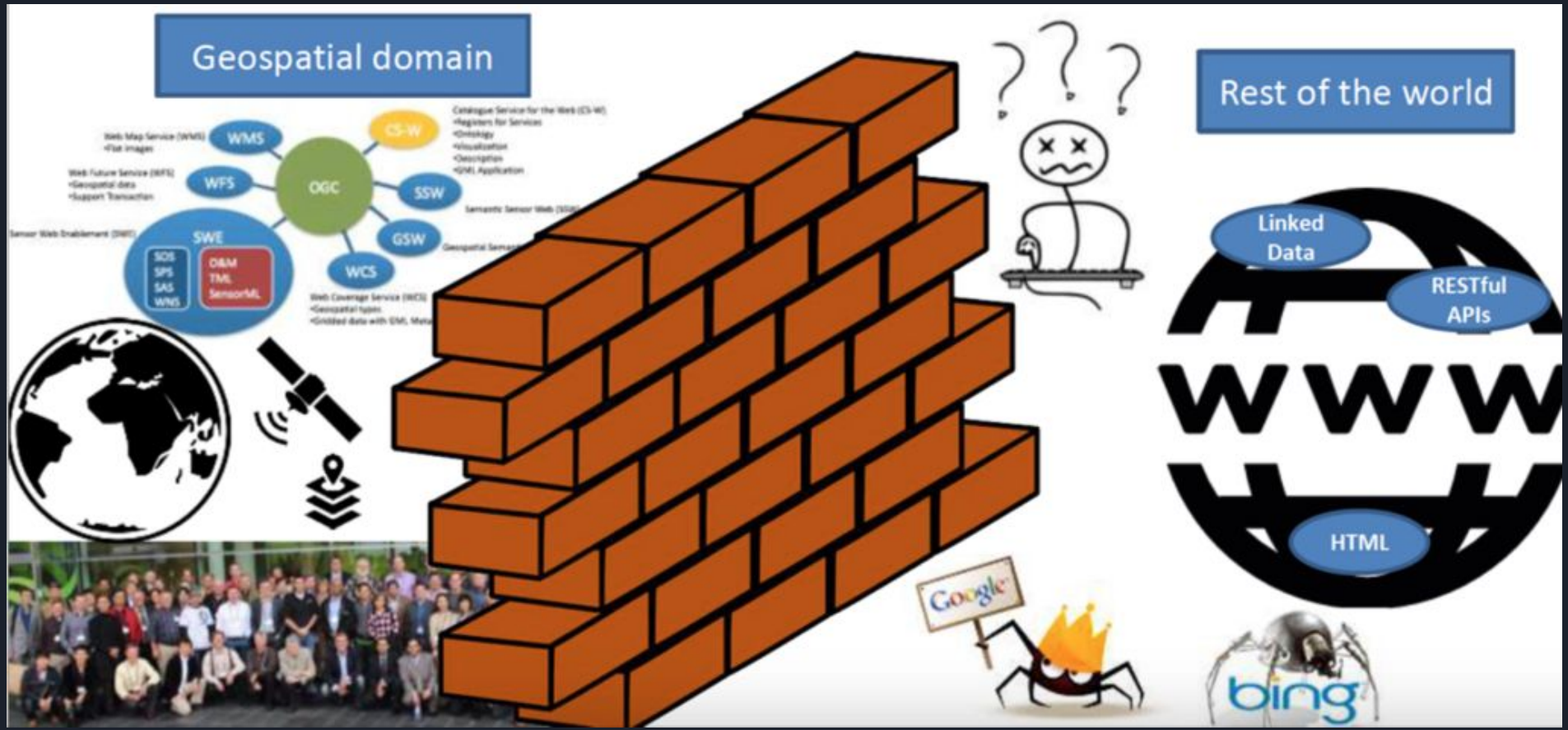
pygeoapi, an OGC API and STAC implementation

Angelos Tzotsos, Paul van Genuchten, Tom Kralidis
INSPIRE Conference, June 2020

Overview

- Geospatial API Evolution
- OGC API Common and pygeoapi
- Discovery via HTML and Schema.org

- CSW and OGC API - Records implementation
- STAC
- Demo


Geospatial domain
WMS
WFS
OGC
CS-W
SSW
GSW
WCS
SWE
SOS
SPS
SAS
WNS
O&M
TML
SensorML
Rest of the world
Linked Data
RESTful APIs
WWW
HTML
Google
bing

Geospatial API Evolution

1990s:

- Spirit of XML-RPC/CORBA
- SOAP/WSDL/UDDI
- Service Oriented Architecture (SOA)
- Strong concept of RDBMS as the backend
- OGC WMS (1999)

2000s:

- Web 2.0
- JavaScript/AJAX
- Google Maps, Slippy maps, tiles
- OGC WFS (2002), WCS (2003), WPS (2005), CSW (2007)

2020s:

- Web 3.0
- Structured data
- Machine learning / Artificial intelligence
- Democratising software development
- OGC API’s

Challenges in traditional OGC services

OGC+W3C spatial data on the web interest group

- Hard to access by non-specialised user
- Search engine can not access records/features
- No use of URI’s and HTTP verbs

Our overarching goal is to enable spatial data to be integrated within the wider Web of data; providing standard patterns and solutions that help solve problems

Spatial data on the web best practice on top of W3C data on the web best practice

OGC-API - Common

OGC API - Features was the first activity in this new generation of OGC Standards

OGC API - Common was defined as a basis new developments

OGC API - Records, OGC API - Maps, OGC API - Coverages, OGC API - Processes are other initiatives

OGC API - Common implements spatial specialisation on top of common web standards

- HTTP verbs, content negotiation
- OpenAPI specification
- HTML, GeoJSON as encodings
- Specifications developed on GitHub (Issue tracking/discussion/collaboration)

Design Patterns

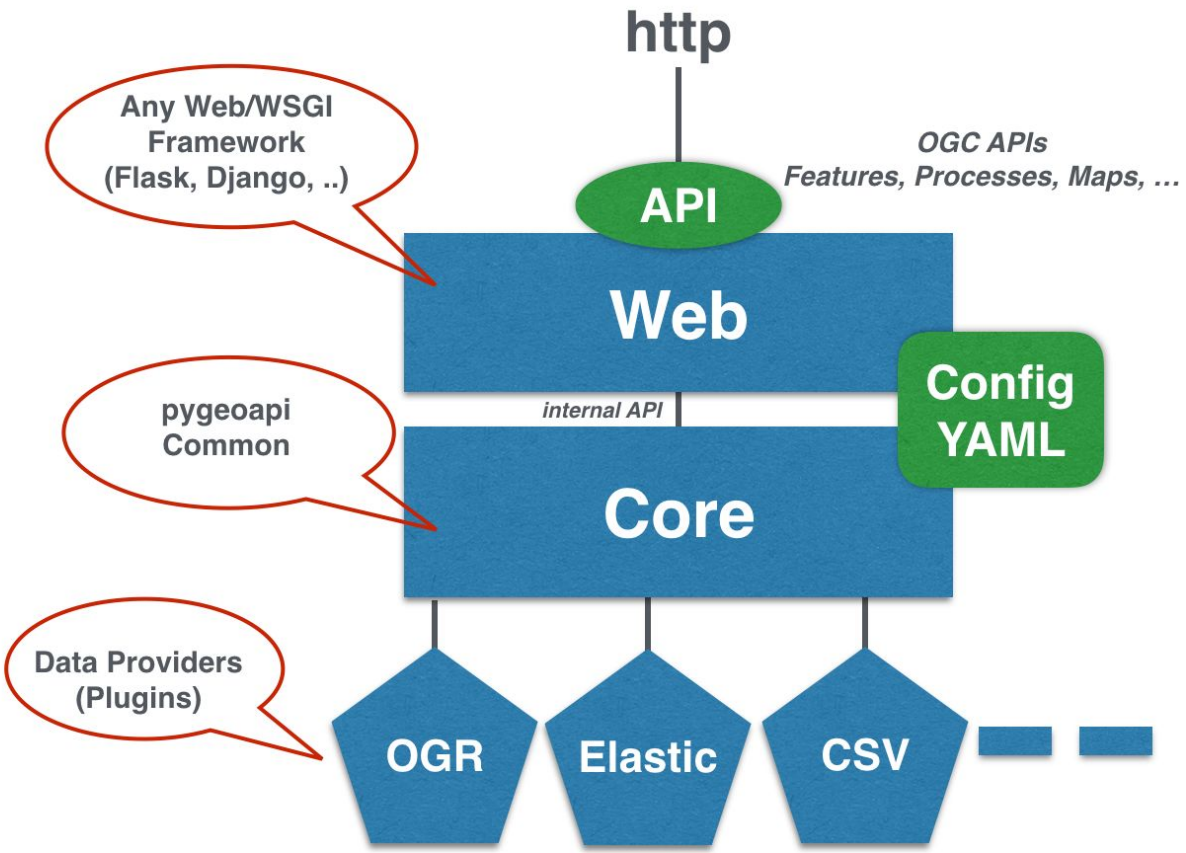
- Developer friendly
- Lightweight specification development
- Removing HTTP use as a tunnel
  - ~~/ows?request=GetFeature&typename=roads&featureid=5~~
  - /api/collections/roads/items/5
- Modular specification development
- Core and extensions

- Geospatial data API framework
- OGC Compliant (already!)
- OSGeo Community Project
- pygeoapi project started at the first OGC API hackathon
- https://pygeoapi.io/

pygeoapi

- Leveraging common Python Web Modules, making it a relatively lightweight implementation.
- Core abstract API; web framework agnostic (default Flask)
- YAML configuration (metadata, dataset connections
- Automated OpenAPI document generation (data binding)
- Robust plugin framework (data connections, formats, processing)
- Easy deployment (pip install, Docker)
- Minimal dependencies

pygeoapi - Architecture

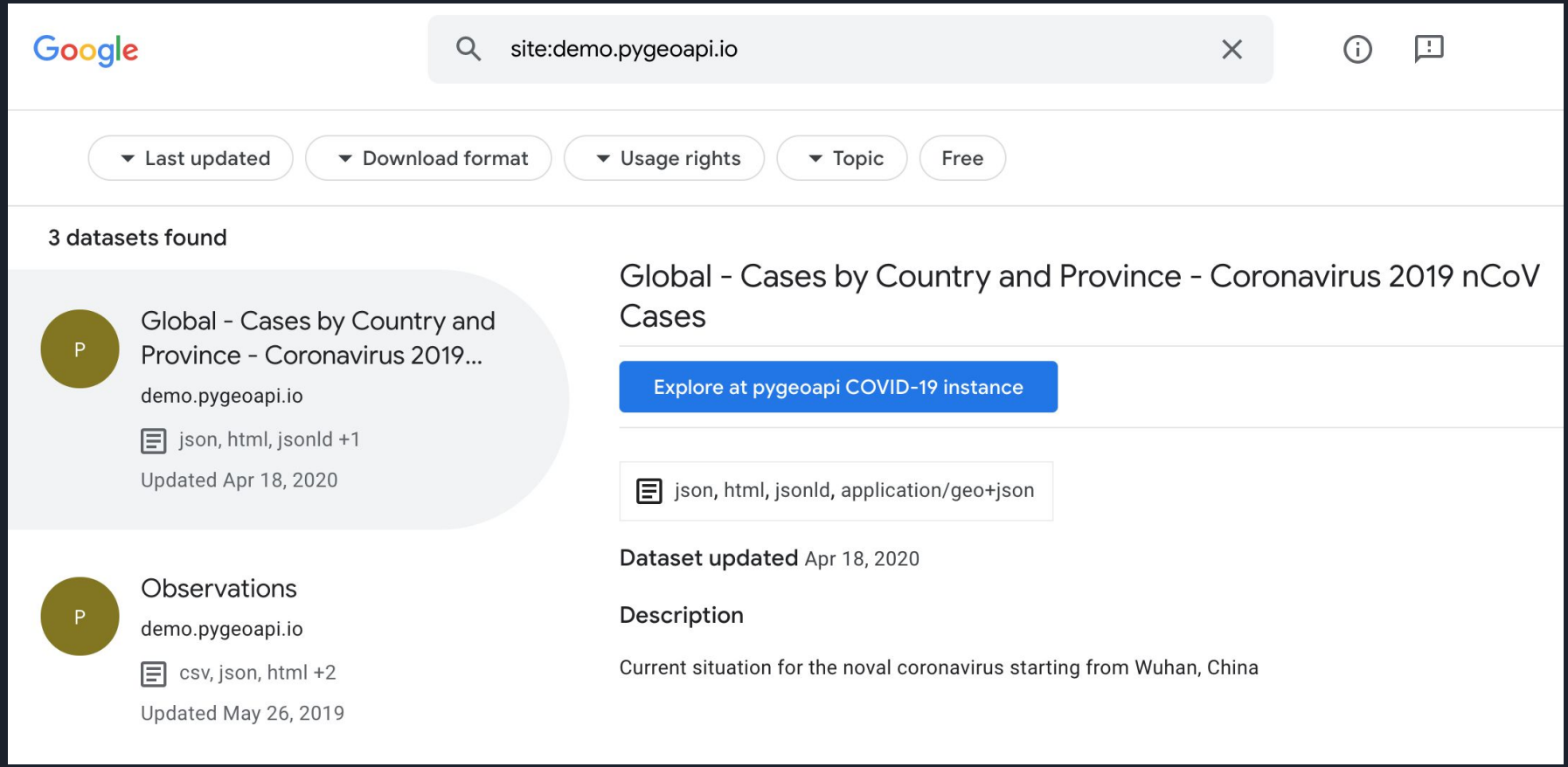


HTML and Schema.org

- HTML is a primary encoding in OGC API Common
- Datasets are discovered by search engines without the need of a catalogue
- Also individual features are crawled by search engines

- Schema.org is the technology that powers google dataset search

- pygeoapi implements HTML and schema.org for collections, records and features


Google
site:demo.pygeoapi.io
Last updated
Download format
Usage rights
Topic
Free
3 datasets found
P
Global - Cases by Country and Province - Coronavirus 2019...
demo.pygeoapi.io
json, html, jsonld +1
Updated Apr 18, 2020
P
Observations
demo.pygeoapi.io
csv, json, html +2
Updated May 26, 2019
Global - Cases by Country and Province - Coronavirus 2019 nCoV Cases
Explore at pygeoapi COVID-19 instance
json, html, jsonld, application/geo+json
Dataset updated Apr 18, 2020
Description
Current situation for the noval coronavirus starting from Wuhan, China

And what about catalogues?

OGC CSW

- Current OGC search/discovery (2007: version 2.0.2)
- OGC CSW 3.0.0 released in 2016
- Core profile: Dublin Core
- Application profiles: ISO, ebRIM, etc.
- Adoption by many organizations and tools
- QGIS MetaSearch, pycsw, GeoNetwork, deegree, OWSLib
- Challenges are balancing simple search with complex metadata (ISO)
- Web service find/bind a challenge (unless using ISO, which is complex)
- Mass market integration

OGCAPI - Records

- OGC API - Catalogues Standards Working Group Charter formed June 2019
- Specification renamed to OGC API - Records
- SWG is working on GitHub and has released a draft specification
- https://github.com/opengeospatial/ogcapi-records

OGCAPI - Records

- OGC API - Records provides discovery and access to metadata about geospatial data and services.
  - GET /collections/myCatalogue/items
  - GET /collections/myCatalogue/items?bbox=160.6,-55.95,-170,-25.89
- Data is returned in pageable chunks, with each response containing a next link pointing to the next set of response records.
- The core specification supports a basic set of filters roughly analogous to the OGC OpenSearch Geo/Time query parameters.
  - GET /collections/myCatalogue/items/{recordId}
- https://github.com/opengeospatial/ogcapi-records/blob/master/core/outline.adoc

OGC API - Records implementation

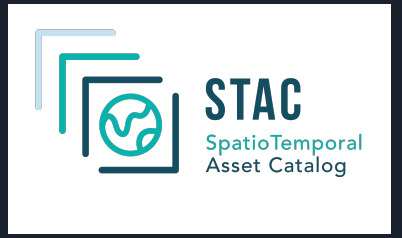
- pygeoapi project implements most of the current OGC API - Records draft
- pygeoapi extensibility will allow to bind to specific search backends (ES, SOLR) or metadata applications (pycsw, GeoNetwork, CKAN, etc.)
- A sample instance is deployed using discovery metadata from the Meteorological Service of Canada as part of the WMO Information System.
- https://github.com/opengeospatial/ogcapi-records/blob/master/implementations.md#pygeoapi
- https://dev.api.weather.gc.ca/msc-wis-dcpc

- DEMO

Findings from the OGC API - Records implementation

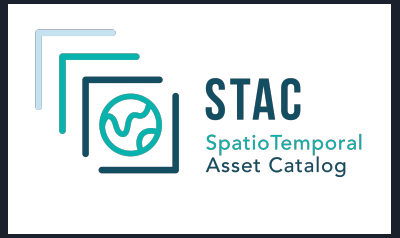
- What are common backends for storing metadata (PostgreSQL, Elasticsearch, TinyDB, GeoPackage, other document stores)
- How to facilitate full text search on a given backend
- OARec does not facilitate facet statistics yet, but it is a common catalogue use case
    - Candidate for an OGC API - Records extension
- Transforming records to alternative models is a common use case in catalogues, content-negotiation on output schema could be an interesting aspect for OARec

STAC



- The SpatioTemporal Asset Catalog (STAC) specification provides a common language to describe a range of geospatial information, so it can more easily be indexed and discovered.
- A 'spatiotemporal asset' is any file that represents information about the earth captured in a certain space and time.
- The goal is for all providers of spatiotemporal assets (Imagery, SAR, Point Clouds, Data Cubes, Full Motion Video, etc.) to expose their data as SpatioTemporal Asset Catalogs (**STAC**), so that new code doesn't need to be written whenever a new data set or API is released.
- https://stacspec.org/
- https://landsat.stac.cloud/?t=catalogs

STAC



The STAC Specification consists of 4 semi-independent specifications. Each can be used alone, but they work best in concert with one another.

- STAC Item is the core atomic unit, representing a single spatiotemporal asset as a GeoJSON feature plus datetime and links.
- STAC Catalog is a simple, flexible JSON file of links that provides a structure to organize and browse STAC Items. A series of best practices helps make recommendations for creating real world STAC Catalogs.
- STAC Collection is an extension of the STAC Catalog with additional information such as the extents, license, keywords, providers, etc that describe STAC Items that fall within the Collection.
- STAC API provides a RESTful endpoint that enables search of STAC Items, specified in OpenAPI, following OGC's WFS 3.

STAC in pygeoapi

- pygeoapi project implements the STAC Catalog specification
- pygeoapi implements STAC as an geospatial file browser through the FileSystem provider, supporting any level of file/directory nesting/hierarchy.
- A sample instance is deployed using data and discovery metadata from the Meteorological Service of Canada as part of the WMO Information System
- https://dev.api.weather.gc.ca/msc-wis-dcpc/stac
- DEMO

Demo

Download at pygeoapi.io

Try out pygeoapi at demo.pygeoapi.io

Join the development at github.com/geopython/pygeoapi

Come chat at https://gitter.im/geopython/pygeoapi