Herbert, AV, Haberle, SG, Munack, H and Codilean, ATC 2024. Collaborative effort towards a FAIR and OPEN Indo-Pacific Pollen Database (IPPD).

Supplemental material: Alternative ways of OCTOPUS db data access

To retrieve data from OCTOPUS database[1], users will not talk to the database directly, but to the GeoServer application, which serves OCTOPUS' spatial data using the Web Feature Service (WFS).

The WFS is an OGC (Open Geospatial Consortium; responsible for the global development and standardization of geospatial and location information) standard[2] that allows for the platform-independent exchange of spatial features over the internet using Geography Markup Language (GML), a type of XML. Unlike the Web Map Service (WMS), which delivers static map tiles, WFS enables the querying, modification, and spatial analysis of vector data. WFS supports three primary operations: *GetCapabilities*, which provides a human-readable description of a WFS service and its capabilities; *DescribeFeatureType*, offering details on supported feature types; and *GetFeature*, which retrieves features, including their geometries and attributes.

Here we describe **two WFS ways to access the IPPD@OCTOPUS, (1) via the R software[3] and (2) using QGIS**[4], on the example of the Indo-Pacific Pollen Database (IPPD) collection.

1. An R way

1.1 Loading required libraries

```
# install.packages(c("sf", "httr", "ows4R","tidyverse"), dependencies = TRUE)
library(sf) # Provides support for handling geospatial data in R
library(httr) # A package to work with HTTP, useful for making web requests
library(ows4R) #  An interface for OGC web services, incl. WFS
```

1.2 Setting up the WFS client

Store URL in an object ...

```
OCTOPUSdata <- "http://geoserver.non-prod.octopusdata.org/geoserver/wfs"
```

and set up database connection

```
OCTOPUSdata_client <- WFSClient$new(OCTOPUSdata, serviceVersion = "2.0.0")
```

1.3 Querying the WFS server

Now retrieve and print all available feature types (layers)

```
OCTOPUSdata_client$getFeatureTypes(pretty = TRUE)
```

[1]Codilean et al. 2022. OCTOPUS database (v.2). ESSD, 3695–3713. https://doi.org/10.5194/essd-14-3695-2022.
[2]https://www.ogc.org/standards/ (accessed 28-07-2024).
[3]https://www.r-project.org (accessed 28-07-2024).
[4]https://qgis.org/ (accessed 28-07-2024).

1.4 Building a custom request URL

Parse the WFS base URL into components ...

```
url <- parse_url(OCTOPUSdata)
url$query <- list(service = "wfs",
                  version = "2.0.0",
                  request = "GetFeature",
                  typename = "be10-denude:ippd_baseline_pollen",
                  srsName = "EPSG:900913") # set parameters for url$query
```

and construct full request URL from above components

```
request <- build_url(url)
```

1.5 Retrieving, reading, filtering and saving

Now use the constructed request URL to retrieve ippd_baseline_pollen

```
ippd_baseline_pollen <- read_sf(request) # Takes few seconds ...
```

Convert ippd_baseline_pollen to a regular data frame ...

```
ippd_pollen_baseline_df <- st_drop_geometry(ippd_baseline_pollen) # Drops geometry column
```

and write the data frame to a CSV file

```
write.csv(ippd_pollen_baseline_df, "ippd_baseline_pollen.csv", row.names = FALSE)
```

Summarising the above, the entire procedure of data retrieval (in this case ippd_baseline_pollen) from OCTOPUS database via WFS using R looks as follows

```
# Save the below script (for instance as ippd_baseline_pollen.R in your downloads folder),
# open it in R Studio and SET WORKING DIRECTORY > TO SOURCE FILE LOCATION
library(sf)
library(httr)
library(ows4R)
OCTOPUSdata <- "http://geoserver.non-prod.octopusdata.org/geoserver/wfs"
OCTOPUSdata_client <- WFSClient$new(OCTOPUSdata, serviceVersion = "2.0.0")
OCTOPUSdata_client$getFeatureTypes(pretty = TRUE)
url <- parse_url(OCTOPUSdata)
url$query <- list(
  service = "wfs",
  version = "2.0.0",
  request = "GetFeature",
  typename = "be10-denude:ippd_baseline_pollen",
  srsName = "EPSG:900913")
request <- build_url(url)
ippd_baseline_pollen <- read_sf(request) # Takes few seconds ...
ippd_pollen_baseline_df <- st_drop_geometry(ippd_baseline_pollen)
write.csv(ippd_pollen_baseline_df, "ippd_baseline_pollen.csv", row.names = FALSE)
```

Addendum

Alternatively, to keep the geometry data as well-known text (WKT)

```
ippd_pollen_baseline_df <- st_as_text(st_geometry(ippd_baseline_pollen))
```

2. The QGIS way

1. Open QGIS and start a new project: `Project > New`
2. In the *Browser pane*, select (right click) `WFS/OGC API Features > New Connection`
3. Name the new connection and insert `http://geoserver.octopusdata.org/geoserver/wfs` in the URL field and confirm. Once a connection is established, all available OCTOPUS collections will appear in the *Browser pane*.
4. To add a collection of interest, double click it (or drag it to the *Layers pane*). Your selection will appear in the *Layers pane*.
5. To locally store a collection, select `Export Layer > Save Features As ...`. Choose file format (e.g. CSV) and destination.

More details about WFS and OCTOPUS db can be found in the database online documentation[5].

---

[5]Munack et al. 2023. OCTOPUS database documentation (online). https://octopus-db.github.io/documentation/ (accessed 28-07-2024).