WFS primer

In this primer we will demonstrate how to obtain a small set of data from a large datasets with the OGC Web Coverage Service (WFS) protocol. We will show that ordering data through WFS is just as easy as buying an ice cream.

Find a WFS server

Find a data web source that hosts a WFS server (*go to an ice cream vendor*). We will use the Marbound datasets hosted at the Flanders Marine Institute (VLIZ) as example in this primer. There are several WFS services to be found, for a selection follow the link. If you have more send them to the OpenEarth team.

Request an overview of the content of a WFS server

Ask for what the server has to offer (*see which flavours he has and which kind of cups*). You need to add the following mandatory <keyword,value> pairs to the base server url, separated by an &, e.g.: `?service=WFS&request=GetCapabilities`.

keyword	value	source
service	WFS	Mandatory WFS standard value
request	GetCapabilities	Mandatory WFS standard value

This procedure works for all subsequent <keyword,value> pairs in this primer:http://geo.vliz.be/geoserver/wfs?service=WFS&request=GetCapabilities

This url will return an xml file that contains an inventory of the available datasets. (You can also request available datasets for one WFS version only by appending the optional `version` keyword.)

Inspect the overview of the content of a WFS server

Look at what versions of WFS the server has to offer (*check whether the ice cream is fresh*). For each version there is a tag `WFS_Capabilities` with attribute version, e,.g.

```
<wfs:WFS_Capabilities version="1.1.0" ...>
...
</wfs:WFS_Capabilities>
```

In this case the VLIZ hosts a lot of dataset. The GetCapabilities request gives a complete list of data. Select a dataset (ice cream flavour) from the list through the `FeatureType` tag. The name of the dataest is the `name` tag, for instance `World:worldcities`.

```
<FeatureType>
<Name>World:worldcities</Name>
<Title>World Cities</Title>
<Abstract>World Cities represents the locations of major cities of the world.</Abstract>
<ows:Keywords>
<ows:Keyword>worldcities</ows:Keyword>
<ows:Keyword>World</ows:Keyword>
</ows:Keywords>
<DefaultSRS>urn:x-ogc:def:crs:EPSG:4326</DefaultSRS>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-176.152 \-54.792</ows:LowerCorner>
<ows:UpperCorner>179.222 78.2</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>
```

Request some content of a WFS server

Now we can actually get a subset from the dataset want by using `request=GetFeature` instead of the `request=GetCapabilities` we used above to obtain the meta-data (*order ice cream*). The following <keyword,value> pairs are mandatory.

keyword	value	source
service	WFS	Mandatory WFS standard value
request	GetFeature	Mandatory WFS standard value
version	1.0.0	One of the mandatory WFS standard values returned by returned by the `request=GetCapabilities`.

typename	value inside one of the <FeatureType><Name> /Name><FeatureType> tags.	One of the features returned by returned by the `request=GetCapabilities`. In the example above it is `World:worldcities`
BBOX (optional)	0,50,10,55	bounding box: min(longitude),min(latitude),max(longitude),max(latitude) . This is optional and return only the part of {{typename }} that is inside the bounding box. WFS is know to be a hotchpotch for coordinate order, this particular server configuration seems to prefer lat-lon swapped: 50,0,55,10).
Filter (optional)	Amsterdam	It is possible to add filters to a request, using the description returned by the `request=DescribeFeatureType`, for instance `http://geo.vliz.be/geoserver/wfs?service=WFS&request=GetFeature&typeName=World:worldcities&filter=<PropertyIsEqualTo><PropertyName>World:city_name</PropertyName><Literal>Amsterdam</Literal></PropertyIsEqualTo>`

```
http://geo.vliz.be/geoserver/wfs?service=WFS
&request=GetFeature
&typeName=World:worldcities
&bbox=51,4,52,5
```

or

```
http://geo.vliz.be/geoserver/wfs?service=WFS
&request=GetFeature
&typeName=World:worldcities
&filter=<PropertyIsEqualTo><PropertyName>World:city_name</PropertyName><Literal>Amsterdam</Literal><
/PropertyIsEqualTo>
```

Actually various types of filters can be combined. For more information visit the user manual of OGC WFS.

WFS features can be visualised and filtered very easy with QuantumGIS, just paste http://geo.vliz.be:80/geoserver/wfs?service=WFS in the WFS service menu of QuantumGIS.

See also: geoserver WFS, opengeo WFS

Accessing WFS by Python OWSLib library

simple syntax to get list of available layers offered by WFS server

```
# carry out necessary import (to install OWSLib, pip install owslib)
from owslib import wfs,wms

# give URL, this is the URL of Deltares Data Portal
url = 'http://deltaresdata.openearth.nl/geoserver/ows?'

# construct wfs
awfs = wfs.WebFeatureService(url,version='1.0.0')

# print available layers
for layer in awfs.contents.keys():
    print layer
```

This example is based on the Deltares Data Portal geoserver contents. Every known URL can be used, like the 'http://geo.vliz.be/geoserver/wfs?' as used in the above primer examples.
Off course more information can be retrieved. Basic documentation on OWSLib can be found on geopython page and more detailed on the doctest section of this github repository.

Accessing WFS by R

This is an example of the Marine projects geoserver. As above, this can be used with any WFS server.

Access WFS with R

```
library(gdalUtils)
library(rgdal)
dsn <- "http://marineprojects.openearth.nl/geoserver/ows?service=WFS&request=GetCapabilities"
ogrinfo(dsn, so=TRUE)
```

There are no images attached to this page.