Exploring Web Feature Services

No fancy client needed, open standards and XML make it easy to explore WFS using Python.

Capabilities

We'll test against my two-bit WFS instance at

http://zcologia.com:9001/mapserver/members/

which has members of the next generation MapServer site as its sole feature type, and if port 9001 is forbidden in the UMN computer lab, we'll try

http://www.refractions.net:8080/geoserver/wfs/GetCapabilities

or another service from the catalog at

http://www.refractions.net/white_papers/ogcsurvey/index.php

Connecting

Start up the Python interpreter and define a GetCapabilities request URL:

```
>>> base =
'http://zcologia.com:9001/mapserver/members/capabilities.r
py'
>>> request = base +
'?service=WFS&request=GetCapabilities'
```

We'll use `urllib` to get a file on this URL and parse the file with an `ElementTree`

```
>>> import urllib
>>> u = urllib.urlopen(request)
>>> from elementtree.ElementTree import ElementTree
>>> tree = ElementTree()
>>> root = tree.parse(u)
```

This method returns the root Element of `tree`. Next print `root`

```
>>> print root
<Element {http://www.opengis.net/wfs}WFS_Capabilities at
99ed78>
>>>
```

the string representation of an Element includes the qualified name of the

element in the form *{uri}local*. In an XML file, we usually define a prefix for each URI, and write the element out like '<prefix:local />'.

Service Elements

Elements are list-like, so we have a simple Python-ic way to inspect the children of any element

```
>>> list(root)
[<Element {http://www.opengis.net/wfs}Service at 97a580>,
<Element {http://www.opengis.net/wfs}Capability at
99e878>, <Element {http://www.opengis.net/wfs}
FeatureTypeList at 9a5148>, <Element
{http://www.opengis.net/wfs}Filter_Capabilities at
9a53a0>]
```

Let's pick out the *Service*, or more accurately, the *{http://www.opengis.net/wfs} Service* element and print its children

```
>>> service = root[0]
>>> for e in service:
...     print '%s => %s' % (e.tag, e.text)
...
{http://www.opengis.net/wfs}Title => MapServer Site Member
Locations
{http://www.opengis.net/wfs}Name => members
{http://www.opengis.net/wfs}OnlineResource =>
http://zcologia.com:9001/mapserver/members
{http://www.opengis.net/wfs}Abstract => Demonstrating a
lightweight and low budget WFS server using ElementTree
and Twisted. Every 5 minutes we use RPC to mine the next
generation MapServer website for member locations. These
locations are rendered into GML for a WFS response.
{http://www.opengis.net/wfs}AccessConstraints => NONE
{http://www.opengis.net/wfs}Fees => NONE
{http://www.opengis.net/wfs}Keywords => WFS, ELEMENTTREE,
TWISTED, PYTHON
>>>
```

FeatureType Elements

The question we ask now is whether this service can give us any point type features. The first step towards the answer is to poke around our root's

FeatureTypeList element. This is at index 2.

```
>>> ftypes = root[2].getiterator
('{http://www.opengis.net/wfs}FeatureType')
>>> ftypes
[<Element {http://www.opengis.net/wfs}FeatureType at
9a5a80>]
>>> list(ftypes[0])
[<Element {http://www.opengis.net/wfs}Name at 9a59e0>,
<Element {http://www.opengis.net/wfs}SRS at 9a5ad0>,
<Element {http://www.opengis.net/wfs}LatLongBoundingBox at
9a5af8>]
>>> for e in ftypes[0]:
...     print '%s => %s' % (e.tag, e.text)
...
{http://www.opengis.net/wfs}Name => member
{http://www.opengis.net/wfs}SRS => EPSG:4326
{http://www.opengis.net/wfs}LatLongBoundingBox => None
>>>
```

Capability Elements

So, we have a feature type named 'member' ... does it have a point property? To answer this, we'll need to make a GetFeatureType request. The base URL for such a request is found by inspecting the root's Capability element:

```
>>> capability = root[1]
>>> list(capability)
[<Element {http://www.opengis.net/wfs}Request at 99ed00>]
>>> request = capability[0]
>>> list(request)
[<Element {http://www.opengis.net/wfs}GetCapabilities at
99eaf8>, <Element {http://www.opengis.net/wfs}
DescribeFeatureType at 9a52d8>, <Element
{http://www.opengis.net/wfs}GetFeature at 9a5580>]
>>> iter = request.getiterator
('{http://www.opengis.net/wfs}Get')
>>> for e in iter:
...     print '%s => %s' % (e.tag, e.items())
...
```

```
{http://www.opengis.net/wfs}Get => [('onlineResource',
'http://zcologia.com:9001/mapserver/members/capabilities.r
py')]
{http://www.opengis.net/wfs}Get => [('onlineResource',
'http://zcologia.com:9001/mapserver/members/description.rp
y')]
{http://www.opengis.net/wfs}Get => [('onlineResource',
'http://zcologia.com:9001/mapserver/members/features.rpy')
]
```

The second of these elements is the one we are after.

DescribeFeatureType

Now we make a DescribeFeatureType request and parse the response.

```
>>> description_base = iter[1].get('onlineResource')
>>> url = description_base +
'?service=WFS&request=DescribeFeatureType&typename=member'
>>> u = urllib.urlopen(url)
>>> dtree = ElementTree()
>>> droot = dtree.parse(u)
>>> list(droot)
[<Element {http://www.w3.org/2001/XMLSchema}import at
a3df30>, <Element {http://www.w3.org/2001/XMLSchema}
element at a3df58>, <Element
{http://www.w3.org/2001/XMLSchema}complexType at a3de40>]
```

Making sense of the schema is a bit beyond the scope of this humble hacking workshop. We'll just print the attributes of the schema elements and look for *location*, *position*, or *pointProperty* types and refs:

```
>>> elems = droot.getiterator
('{http://www.w3.org/2001/XMLSchema}element')
>>> for e in elems:
...     print e.items()
...
[('substitutionGroup', 'gml:_Feature'), ('type',
'member_Type'), ('name', 'member')]
[('type', 'string'), ('name', 'fid')]
[('type', 'string'), ('name', 'mid')]
[('type', 'string'), ('name', 'fullname')]
```

```
[('ref', 'gml:location')]
>>>
```

gml:location ... we have a point property.