LAB TEN

XML WEB SERVICES

CS1103, Winter 2021

LEARNING OUTCOMES

At the conclusion of the lab, students should be able to

- Explain the structure of a Web Services Definition Language document (file)
- Explore a WSDL and construct simple web service calls using python and the suds-community package.

BACKGROUND

Web services allows programmatic access to a host's services, facilitated through XML and JSON. The starting point for web service discovery (using UDDI) is the hosts's wsdl (web services definition language) file. A WSDL file is an XML document type, defining the datatypes particular to the web service (using a link to an XML schema document) as well as it's functionality including method name (aka endpoint), input parameters/types, and output types.

This lab is an overview of WSDL and exploration of a specific WSDL for Canadian tide data using the Python programming language.

UNDERSTANDING WSDL

Looking at the W3Schools page for XML WSDL beginning under the section "WSDL Documents" (https://www.w3schools.com/xml/xml_wsdl.asp), there are four sections to a WSDL XML document.

Briefly summarize this section in your own words

- What are the major elements of WSDL, and what do they hold?

WSDL EXAMPLE

Take a look at the WSDL file for tide web services from Environment Canada, available as https://ws-shc.qc.dfo-mpo.gc.ca/predictions?wsdl . Given that you know the structure of this file, look through the file and name two datatypes (types) defined by the file as well as two methods (operation within portType.

EXPLORING WSDL

GETTING READY

Download the Lab10.zip file from the LMS and unpack it on a lab machine as well as on your own computer so that you can look at the contents (files) in a browser. To prepare your Python3 session do the following in the directory that shows the env directory on the lab machine:

```
bash
source ./env/bin/activate

$ python3.6
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Now type (or copy and paste) the import statements to get ready:

```
import sys
sys.path.append(r'./env/lib/python3.5/site-packages/')
from suds.client import Client
```

EXPLORING THE WSDL DOCUMENT/FILE

Retrieve the WSDL document for a web service:

```
client = Client("https://ws-shc.qc.dfo-mpo.gc.ca/predictions?wsdl")
```

The WSDL document has now been downloaded into a Client object, named client. We can print the entire client by, well, printing it:

```
print(client)
```

Programmatically, here are some things we can easily discover:

```
# How many services are there?
print('Number of services: ',end='')
print(len(client.wsdl.services))
# How many ports (portTypes) are there?
print('Number of ports: ',end='')
print(len(client.wsdl.services[0].ports))
# How many methods are there?#
print('Number of methods: ',end='')
print(len(client.wsdl.services[0].ports[0].methods))
```

What's going on here:

- Python's print() prints a string. We use end=" to overwrite the default end parameter so that no newline character is printed.
- The len() function counts the number of things (in this case objects) inside the reference.
- Just like with System.in in Java, client.wsdl is an object within an object notation. Similarly, client.wsdl.services is a zero-based list (array, vector…) as is client.wsdl.services[0].ports . The last thing, client.wsdl.services[0].ports[0].methods is actually a dictionary variable, which is referenced by using a string as the index to lookup the item. Notice that ordering and naming of the objects follows the hierarchy of the WSDL document.

In Python, we can loop through a list/dictionary in this way to create a list of method names

```
methods = [method for method in client.wsdl.services[0].ports[0].methods]
print(methods)
```

What's going on here:

- The for-statement moves through each member of the list. The variable method holds the individual list item and changes each time through the loop. You see this in Java Script too.
- The method value is appended to the end of a local list variable, methods.
- Notice that references are not being returned from the methods dictionary variable, rather strings (indices) are. This is different than one would expect in Java.

Lastly, we can loop through the methods, listing the input parameters for each method. Notice that this is a bit ugly. Also know that code blocks in Python are identified by tabs, and the loops won't execute until a line with the "de-tabbing" is encountered.

```
for method in methods:
    print(method)
    fullMethod = client.wsdl.services[0].ports[0].methods[method]
    params = 
client.wsdl.services[0].ports[0].methods[method].binding.input.param_defs(clien
t.wsdl.services[0].ports[0].methods[method])
    for part in params:
        print('\t', end='')
        print(part)
```

Now look at the ExploreWSDL.py file. It's all of these sections, but uses a command line argument of the WSDL URL. Running this program. *Be sure to put the URL as a quoted string:*

```
$ ./ExploreWSDL.py <wsdl-url>
```

will give information for any WSDL document. How many methods are there in this web service?

USING THE WEB SERVICE

- Using the code TidePrediction.py code as a guide, do a search and report the predicted tide height between 8am and one second before midnight, on 6 April for North Head, New Brunswick
    - Station names can be found at http://www.tides.gc.ca/eng/station/list .
    - If you copy and paste the code upto and including the line 49 (data = rest.data), printing out data will show you the structure of the payload.
    - You can run this file as

```
$ ./TidePrediction.py
```

- Credit: Much of the code for this exercise is from https://www.sigmdel.ca/michel/program/python/tide_cdn_en.html

BENEFITS OF STANDARDIZING

Because the wsdl file follows the WSDL document standard, it can be understood programmatically. Take a look at http://www.soapclient.com/soaptest.html . As a simple example, use the wsdl file for a calculator web service: http://calculator-webservice.mybluemix.net/calculator?wsdl . Unfortunately the service is not setup, but you get the idea. Now try the tide data wsdl (https://ws-shc.qc.dfo-mpo.gc.ca/predictions?wsdl). Try out one of the methods that you identified in the WSDL Example section, and paste the results into your report.

SUBMISSION

Before 4:30pm on the day after this lab, students should submit online to the lms a zip/tar file containing

- The word-processor report as a pdf file, containing the answers to the questions.