Web Services

✦ What is a “web service” (and why do we want to use one)?
✦ RPC
✦ REST

RPC vs REST

✦ There are two main styles of web service
  - Remote Procedure Call (RPC)
  - Representational State Transfer (REST)
✦ RPC is older, more well-established
  - called “big web services” by author of book on REST
  - used by Amazon.com and many other services
  - may be hard to get started with a service -- not always clear what they provide
✦ Almost all services now support REST, some older ones may still use RPC

HTTP

✦ Both types of service are based on the HTTP protocol
✦ Human web: documents are typically formatted in HTML
✦ Programmable web:
  - XML (extensible markup language)
    - very similar format, but tags just indicate structure, have no meaning
  - CSV (comma-separated values)
    - plain text, use commas to separate fields
  - JSON (Javascript Object Notation)
    - very similar to Python dictionaries and lists, easily converted into Python objects

Screen scraping is tedious, error-prone, unreliable, and (depending on the application) unethical

**screen scraping:** v.
The act of capturing data from a system or program by snooping the contents of some display that is not actually intended for data transport or inspection by programs... Nowadays it often refers to parsing the HTML in generated web pages with programs designed to mine out particular patterns of content. In either guise screen-scraping is an ugly, ad-hoc, last-resort technique that is very likely to break on even minor changes to the format of the data being snooped.

The Jargon File (aka The New Hacker's Dictionary)

Instead of trying to extract data from a document intended to be read by humans (via a web browser) get the data from a source that provides easily parsed formats

Some good (but biased) background:

L. Richardson and S. Ruby, 2007

Mostly for web service developers, but good sections on writing clients
HTTP Protocol

- Compared to other protocols, HTTP is very simple
  - it is stateless -- neither side needs to remember what happened earlier in a conversation
  - a client sends a request, and the server sends a response

- HTTP requests have:
  - a verb that defines the request (GET, POST, ...)
  - a path (on the server)
  - message headers
  - message body

```
GET classes/10S HTTP/1.1
Host: www.cs.uoregon.edu
...
```

- HTTP responses have the same general structure
  - the response code (200 = OK, 404 = not found, 500 = server error, etc)
  - response headers
  - response body

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Date: Fri, 21 May 2010 21:49:44 GMT
Expires: -1
Server: gws
<!doctype html><head><title>Waffle - Google Search</title><script>...
```

Remote Procedure Call

- The idea behind RPC is that objects can reside on remote systems
- All we really need is to be able to refer to an object
- Example: gene sequence at NCBI
  - if seq is a reference to an object (either remote or local) an expression like len(seq) should return the number of letters in the sequence
  - one form of web service is based on a remote procedure call
  - “procedure” is an old name for a subroutine or function

```
seq = download.ecoli.gbk
process.rb(seq)  # RPC call
```

Example: Geocoder

- A geocoder is a program that associates place names or addresses with geographic coordinates (latitude and longitude)
- If you have an application that needs coordinates, you can call a web service to get the coordinates, instead of keeping a local database:

```
ws.geocode("1477 W. 13th Ave., Eugene, OR")
```

- The result returned by the web service is an object (named loc in this example):

```
ws is a reference to an object that resides on the Internet, at geocoder.us
ws.geocode("1477 W. 13th Ave., Eugene, OR")
>>> loc.lat
44.045746
>>> loc.long
-123.115144
>>> loc.zip
97402
```

ws is a reference to an object that resides on the Internet, at geocoder.us
the call to geocode is a remote procedure call, executed on a server at geocode.us
RPC in Python

- Since all communication with a web service is done with HTTP, we can use any of Python’s libraries that operate with HTTP
- Net::HTTP
  - basic library, has methods to get message, look at response headers, etc
- XMLRPC or SOAP
  - libraries that make “proxy” objects, so getting data from a remote object is just like getting data from a local object
- Many of these are out of date and not supported any more — we’ll use REST instead

REST

- The biggest difference between REST and RPC is that in REST everything is included in the URL
  - with SOAP the URL is for the “endpoint” that will process the request
  - the service looks in the message body to find the parameters that were passed
- With REST, the URL contains all the necessary information
- To fetch a document from a web service, include the specs as part of the URL:

  http://geocoder.us/service/csv/geocode?address={address}

REST Access to Geocoder

- Here is an example of how to get information from geocoder.us using their REST web service:
  - the documentation says the first part of the URL is http://geocoder.us/service
  - the next part of the path specifies a format (csv, json, etc)
  - next is the name of the service you want (corresponds to the method name in RPC):
    - geocode, geocode_address, or geocode_intersection
  - specify any arguments after the path
- Example:

  http://geocoder.us/service/csv/geocode?address=1477+E+13th+Ave,+Eugene,+OR

Testing a Web Service: curl

- A command line program named curl will fetch a document and display it on the console
- We can use curl to test a REST service
- To see what geocoder.us does:

  $ curl 'http://geocoder.us/service/csv/geocode?address=1477+E+13th+Ave,+Eugene,+OR' 44.045547,-123.070270,1477 E 13th Ave,Eugene,OR,97401

  Note URL enclosed in quotes on the command line

- The output is in CSV (comma separated values) format
  - note how the format is specified as part of the URL
REST with Python

- A Python program that uses a REST web service just needs to fetch a page using `urllib`

```python
from sys import argv
from urllib.request import *

addr = argv[1]

site = 'http://geocoder.us'
service = '/service/csv/geocode?address='
url = site + service + addr.replace(' ', '+')
print('fetching from', url)

content = urlopen(url).read().decode()
for x in content.split(', '):
    print(x)
```

Output from geocode.py

- Here is a sample execution of the Python code on the previous slide:

```bash
% python3 geocode.py '1477 E 13th Ave, Eugene, OR'
fetching from http://geocoder.us/service/csv/geocode?address=1477+E+13th+Ave,+Eugene,+OR
44.045547
-123.070270
1477 E 13th Ave
Eugene
OR
97401
```

JSON

- JSON = Javascript Object Notation
  - described at json.org (official website)
  - see also json.com (blog post and discussion site)
- Almost (but not exactly) identical to Python syntax
- Previous example, in JSON format:

```json
curl 'http://geocoder.us/service/json/geocode?address=1477+E+13th+Ave,+Eugene,+OR' |
{   
  "number" : "1477",
  "street" : "13th",
  "lat" : "44.045547",
  "state" : "OR",
  ...
}
```

JSON Parser

- To handle the slight differences use a JSON parser to convert a string returned from a web service into a Python object
  - the parser is a function named `loads`

```python
import json
content = urlopen(url).read().decode()
x = json.loads(content)
```