APIs, Python, and Vagrant:

Or, how I taught my computer to be my processing archivist

Lora J. Davis
Digital Archivist
Johns Hopkins University

ljdavis@jhu.edu
@lorajdavis
https://github.com/lorajdavis
What does API stand for?

**Application**
As in a computer application, like Word or Chrome

**Programming**
As in computer “programming,” or taking steps to make a computer do something you want it to do

**Interface**
As in the the place where two systems meet
What do APIs do?

As the prior slide suggests, APIs make it possible for applications to interact (or interface) with one another.

APIs are not new, and there are many types of APIs.

When you copy content from a Word document to your clipboard, then paste that content into an Outlook e-mail, it works because your computer operating system, which both your versions of Word and Outlook are programmed to run on, uses an API to allow the interchange of information.

APIs tell software developers the rules of the road that they must follow if they want their applications to play well with others.

For more: http://readwrite.com/2013/09/19/api-defined/
Though anything that allows an interchange of information between two applications is *technically* a form of an API, what we typically mean today when we say “API” is a very specific thing.

That thing is a web API.
Ok, so what is a web API?

**Complicated:** A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

**Simple:** You access it over the web, using URL-like directions, and are limited to 3-4 simple commands or activities.

For more: [http://searchcloudstorage.techtarget.com/definition/RESTful-API](http://searchcloudstorage.techtarget.com/definition/RESTful-API)

**Extra nerdy sidebar:**

- Web APIs also come in several flavors, including SOAP and REST.
- We’re going to be exclusively working with RESTful APIs today, as they’re far more prevalent in archives/libraries technologies.
- REST stands for “representational state transfer” and was defined in 2000 in a doctoral dissertation by Roy Fielding.
- REST essentially dictates how an application should be able to textually interact with a web service.
I’m not an application, I’m an archivist! Why should I care?

As librarians and archivists with collection descriptions and/or collections themselves on the web, you probably do care about being able to access and meaningfully manipulate textual data on the web at scale.

In many of the exercises we will work through together today, you are, in fact, one of the “applications” interfacing with web-based data.
Yes, this is a huge barrier to entry for most users, but it can be mitigated:

- We (defined here as both archivists and developers) are a community that likes sharing!
  - Frankly, if you’re sitting down to write scripts from scratch, you’re doing it wrong
- There is no “one right language” to make this work
  - If you have any prior knowledge of a particular scripting language, start there
  - All the scripts I’m going to demonstrate are Python because: 1) Python (and, to a lesser degree, Ruby) is my preferred hammer, and 2) unscientifically speaking, it seems that Python is the preferred language of archivists (which means there’s more to steal or borrow)
  - But, if you want, you can use a Ruby or Perl or PHP or JavaScript shaped hammer!
- The Internet is full of helpful advice!
  - Just don’t feed the trolls
Scenario: At Hopkins, our Technical Services department hoped we could take the moment of our transition into ArchivesSpace as an opportunity to migrate our agent headings away from LCNAF to VIAF (this would keep our archival description in line with the linked-data driven description being done elsewhere in the library).
Example 1 - VIAF

The basic steps:

1. GET existing agent records out of ArchivesSpace
2. Convert the resulting ArchivesSpace JSON to a CSV
3. Run a Python script ([python viafReconciliationCorporate.py](https://github.com/ehanson8/viaf-dbpedia-reconciliation-python)) on the resulting CSV
4. **Manually review the results**
5. Following quality assurance, POST the CSV back into ArchivesSpace with [python postVIAFOrganizations.py](https://github.com/jhu-archives-and-manuscripts/MARAC_API_Workshop)

Scripts:
- [https://github.com/ehanson8/viaf-dbpedia-reconciliation-python](https://github.com/ehanson8/viaf-dbpedia-reconciliation-python)
- [https://github.com/jhu-archives-and-manuscripts/MARAC_API_Workshop](https://github.com/jhu-archives-and-manuscripts/MARAC_API_Workshop)
Example 1 - VIAF

Show & Tell
Example 2 - ArchivesSpace/Archive-It

Scenario: As my University’s web archivist, I want to make my Archive-It web crawls accessible to users who access our collections via ArchivesSpace, but I don’t want to manually create individual digital objects every time I run a new Archive-It crawl.
Technical Pitstop: vagrant install and vagrant up

(this is super awesome)

See Dallas Pillen’s excellent blog post for more on Archivagrant (from which this demo Vagrant was derived):
http://archival-integration.blogspot.com/2016/01/archivesspace-vagrant-archivagrant.html
Example 2 - ArchivesSpace/Archive-It

The basic steps:

1. Confirm an archival object (or many archival objects) exist in ArchivesSpace with the level “Other Level” and other level type “Web archive”
2. Modify line 37 of “archiveIt.py” to match the Archive-It collection ID number for the desired collection (in our case, 3181)
3. Run `python archiveIt.py`
4. Let’s review the results!

Scripts:
https://github.com/jhu-archives-and-manuscripts/MARAC_API_Workshop
Show & Tell
You WILL not succeed on the first try.
You WILL hit unanticipated snafus, oftentimes due to data models and/or poorly written documentation (aka, due to no fault of your own!).
You WILL be fitter, happier, and more productive if you start building a community now and asking questions.

A common ASpace “gotcha”:

Lock version - a value that incrementally increases every time an AS record is altered. In practice, this means work cannot and should not continue on the data in question, i.e. your team has to stop work

```
"lock_version": 5,
"title": "university history scrapbook collection",
"publish": true,
```
APIs, Python, and Vagrant:

Or, how I taught my computer to be my processing archivist

And YOU CAN TOO!

Lora J. Davis
Digital Archivist
Johns Hopkins University

ljdavis@jhu.edu
@lorajdavis
https://github.com/lorajdavis