
Introduction to the ArchivesSpace API

ArchivesSpace Member Forum
Portland State University
July 25, 2017

Lora Woodford

Introduction

The Odd Couple

We are stronger and smarter together THOUGH YOU ONLY GET ME!

Lora

- Digital Archivist
- Never met a task she didn't want to solve with Python/Ruby
- Thinks hanging with the Bmore on Rails group sounds like a fun night out
- Helped migrate two institutions to ArchivesSpace and found it thrilling
- Enjoys craft beer, cross stitch, and the Pittsburgh Steelers
- Knows how to programmatically manipulate data
- CATS YAY!



Valerie

- (Former) Photo Archivist
- Never met a task she didn't want to solve with an MS Access database
- Knows rails are either “narrow gauge” or “standard” because ❤️ steam trains
- Helped migrate to ASpace and felt like a fish climbing a tree
- Enjoys geocaching, hiking, and planning ambitious camping trips
- Knows data modeling and systematic analysis of the *really hard* problems
- CATS YAY!



The 1-up learning experience

You can't learn it all and we can't teach it

Are you a 1?

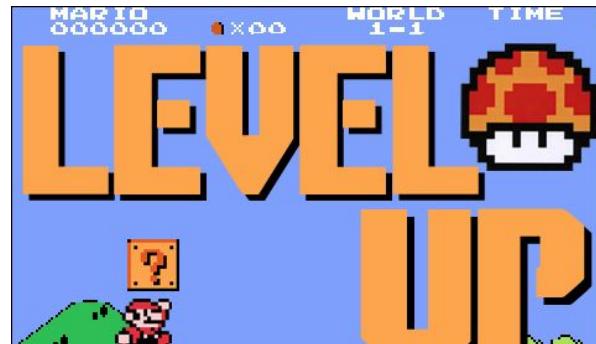
We hope you'll leave a
2.

Are you a 2?

We hope you'll leave a
3.

Are you a 3?

We hope to give you
new ideas, scripts, and
momentum.



Workshop aims

1.

Practical, real-life application

...including the bad parts of real-life

Resource-packed GitHub

These voluminous slides

2.

Assurances:

You are not alone

I didn't become an archivist for this either (but some of you may have!)

This is difficult and frustrating

You didn't miss a flight; [the airport turned into a space launch while you were in the parking lot](#)

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 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.

That's not at all what I thought an API was!

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>

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.

Vocabulary pitstop: API Terms

- **GET**, **POST**, and **DELETE** are the three cornerstone commands for a RESTful API
- We will use these terms throughout
- Think of them as View, Save, and of course, Delete
- All APIs allow GETs, some let you POST, and few allow you to Delete
 - ASpace does all three, but allows you to tailor permissions for each

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.

API possibilities

Get data out → Do something to it → Put it back in

JSON
MARC21
Any standardized
data

Access
OpenRefine
XSLT
Custom script (your choice)
Find and Replace
Hand encoding, copy and
pasting, glue and popsicle
sticks, *whatever it takes!*

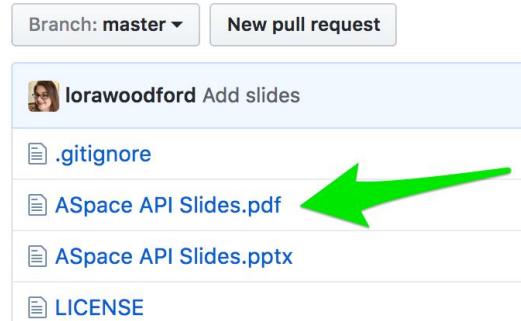
*needle coming off
the record*

This is the tough
part.

Questions?

Resource Pitstop - Get these slides

1. Navigate to https://github.com/jhu-archives-and-manuscripts/ASpace_API_Workshop
2. Once in the repo, bookmark! You'll need this later.
3. Click "ASpace API Slides.pdf"



4. Download and open. Leave these open; you will be using the slides on your own throughout the day.



Setting up your tech: A long-ish pitstop

(We promise you're in the right workshop)

Technical pitstop: The (FREE) Applications



Atom

- A text editor that is handy for interacting with JSON, scripts, and all sorts of structured data
- Can utilize additional packages to customize to your needs (e.g. a JSON “linter”)



Postman

- A GUI application for interacting with APIs

Technical Pitstop: Scripting set-up

Technical Pitstop - Scripting set-up

We are about to:

- Show you a quick **shortcut** for opening the terminal/command prompt
- Get some **important packages installed** on your machines
- **Clone our GitHub repository** (download some scripts to your computer)

Caveats:

- This *is* new, **we will lose some of you**
- You *do* need to know this, but **we have other tofu to fry**
- Use these slides if you need to set up your own workstations **back at the office**

Housekeeping:

- With 15+ people in this workshop, we've got 15+ different environments to troubleshoot and 15+ different opportunities to fail - **please be patient!**
- If, at any point for the remainder of the workshop, you need **assistance**, please place a **hot pink post-it** on the back of your laptop screen

Please follow along starting from this slide

This terrible shade of yellow should be easy to find.

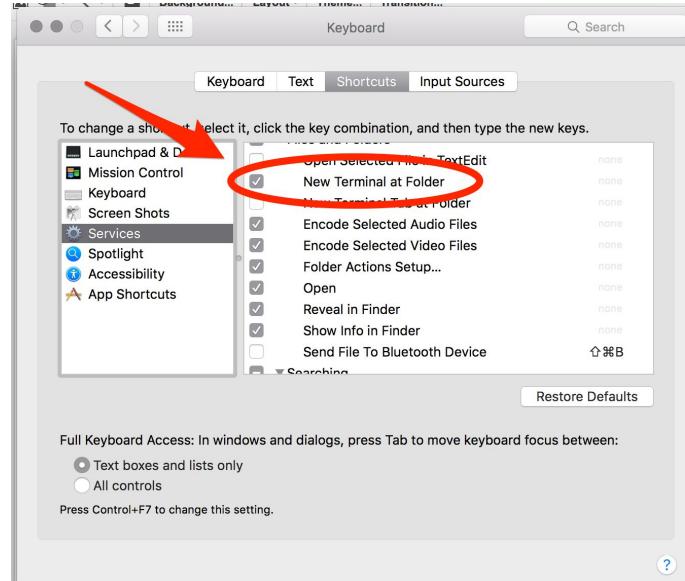
Everyone start with a **Green post-it** = good to go
Red/pink post-it = please assist!

Technical Pitstop - Scripting set-up

This is a handy shortcut that you'll need later

Mac

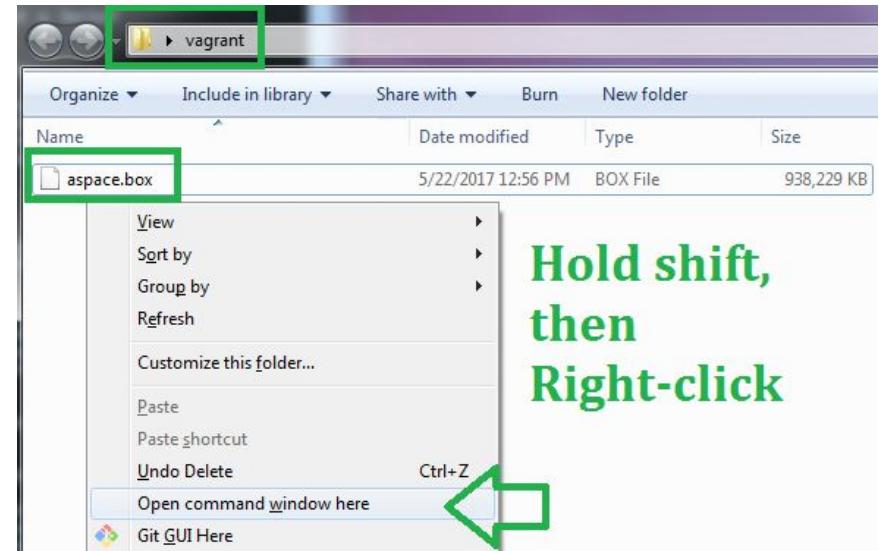
System Preferences > Keyboard > Shortcuts tab >
Services



Note: Some (especially older) Mac OSes may not have this option! If so, no harm. Alert an instructor and we'll walk you through this manually.

PC

Practice: Open any folder, then hold shift + right click anywhere in the window



This is just practice, we don't actually need this window now.

Technical Pitstop - Scripting set-up

Mac

1. Please open the terminal:
 - Use spotlight search to search for “Terminal”
 - OR, open your Applications folder, then open the Utilities folder. Open the Terminal application.



PC

2. Bring up the Cygwin installer you downloaded as part of your pre-workshop homework



Technical Pitstop - Scripting set-up

Mac

1. Type `gcc --version` and hit enter

If you are prompted to install, hit Install!
Once complete, type `gcc --version` again.

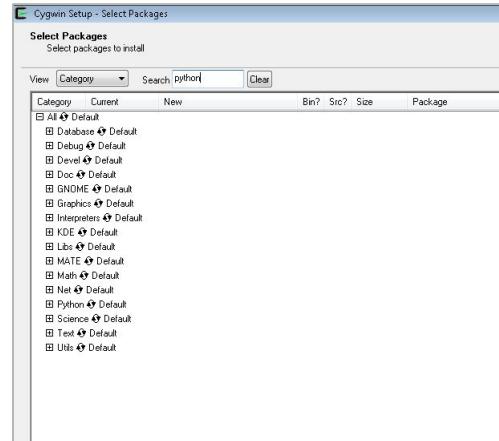
If you see the following, you're good to go.
Leave the terminal open.

```
$ gcc --version
# After installing, check the gcc --version again.
# You should get the below message:

Configured with: --prefix=/Applications/Xcode.app/Contents/Developer/usr --with-gxx-include-dir=/usr/include/c++
Apple LLVM version 8.0.0 (clang-800.0.38)
Target: x86_64-apple-darwin15.6.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin
```

PC

1. Start downloading. Say yes to everything
2. Pick any site to install from and continue
3. Stop when you get to this screen:



Technical Pitstop - Scripting set-up

Mac

1. Open a browser and navigate to:
`brew.sh`
(yup that's a webpage)
2. Copy the long command and paste into terminal. Hit enter.



3. Enter your password if needed

PC

1. Look for View: at the top of the screen and select Category
2. Search for “python2” (without quotes) but do not hit enter; the search happens as soon as you type



Technical Pitstop - Scripting set-up

Mac

1. Type `brew install python` and hit enter
2. Type `python --version` and hit enter to be sure Python appears:

```
[Loras-MacBook-Pro:~ lorajdavis$ python --version
Python 2.7.10
Loras-MacBook-Pro:~ lorajdavis$ ]
```

3. Type `pip install requests`

(we will not remind you to hit enter after commands from now on)

PC

Go to next slide...

Windows users only: Install packages

1. After you have searched for python2 locate and *unskip* the following (the list is alphabetical and the kilobyte counts help):
 - a. python2: Python 2 language interpreter **5,873k**
 - b. python2-requests: Python HTTP/1.1 request module **84k**
2. Now search for git (but don't hit enter), expand the Devel heading, *unskip*:
 - a. git: Distributed version control system **5,387k**
3. Finally, search for openssh (don't hit enter), expand the Net heading, *unskip*:
 - a. openssh: The OpenSSH server and client programs **750k**
4. Once all four have been unskipped, proceed with install: Next > Next

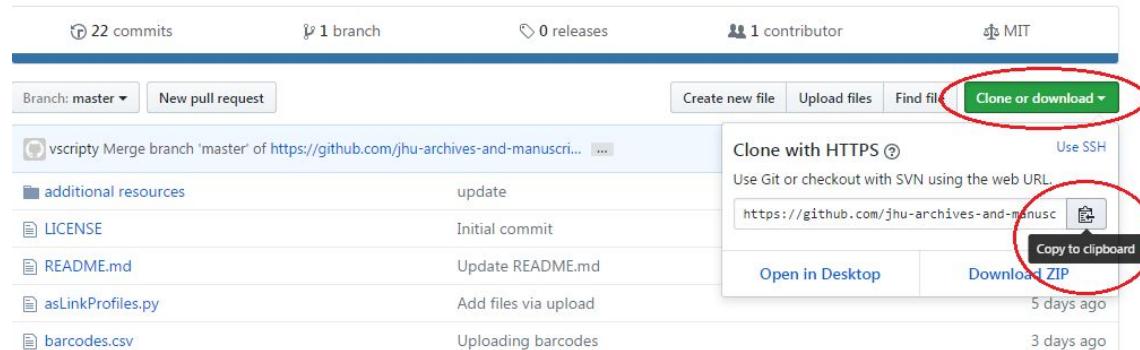
This screenshot shows what an “unskipped” line looks like

Net 			
Python 			
<input type="checkbox"/> Skip	n/a	n/a	38k gnome-python: Python GNOME platform bindings (meta-package)
<input type="checkbox"/> Skip	n/a	n/a	33k gnome-python-desktop: Python GNOME Desktop bindings
<input type="checkbox"/> Skip	n/a	n/a	16k gnome-python-extras: Python GNOME extras bindings
<input type="checkbox"/> Skip	n/a	n/a	21k net-snmp-python: Net-SNMP (python)
<input checked="" type="checkbox"/> 2.7.10-1	<input type="checkbox"/>	<input type="checkbox"/>	5,926k python: Python language interpreter
<input type="checkbox"/> Skip	n/a	n/a	5k python-avahi: mDNS/DNS_SD/Zeroconf implementation (Python)
<input type="checkbox"/> Skip	n/a	n/a	243k python-avogadro: Molecular editor and modeling system (Python)
<input type="checkbox"/> Skip	n/a	n/a	6k python-backports.ssl_match_hostname: SSL hostname verification

Technical Pitstop - Scripting set-up

Everyone

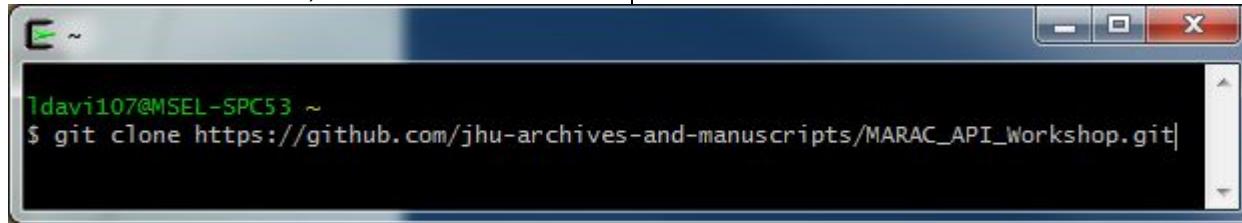
1. Go back to our GitHub repo, which you bookmarked earlier.
2. Then click the green button, click the little clipboard icon, and **Copy to clipboard**



Technical Pitstop - Scripting set-up

Mac

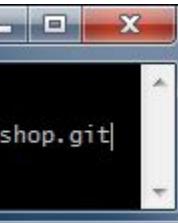
1. Open Terminal
2. Type `cd Desktop` and hit enter
3. Type `git clone` [command+v to paste]
↑
(don't type this, we mean an action)
4. Hit enter



A screenshot of a Mac OS X terminal window. The window title bar says "Terminal". The command line shows the user's name and host followed by a tilde (~). Below that, the command `$ git clone https://github.com/jhu-archives-and-manuscripts/MARAC_API_Workshop.git` is being typed. The cursor is at the end of the command.

PC

1. Open Cygwin
2. Type `git clone` [right-click then paste]
↑
(don't type this, we mean an action)
3. Hit enter



Now you have a folder (either on your Mac's Desktop, or in C:/Cygwin/home/[username]) that contains all the materials you'll need for the rest of today's workshop.

This folder, titled “ASpace_API_Workshop,” is a *direct clone* of what you see in your browser on [github.com](https://github.com/jhu-archives-and-manuscripts/MARAC_API_Workshop).

Command line bootcamp

- Some very simple Unix commands are necessary in this workshop
 - But more important is being able to use them effectively
 - Mac users, and PC users in Cygwin, will be using the same commands...
 - ...but will be working in different directories.
 - So navigating your own way is super important.
-

Command line bootcamp: Navigation

Where are you, and where do you want to go?

Mac

PC

1. In the Finder navigate to your ASpace_API_Workshop directory
2. Ctrl+click the ASpace_API_Workshop directory, and select “New Terminal at Folder”

1. Open Cygwin

Command line bootcamp: Navigation

Where are you?

Everyone type `pwd` and then hit enter.

Mac

Mac users should see something like this:

```
[Loras-MacBook-Pro:MARAC_API_Workshop lorajdavis$ pwd  
/Users/lorajdavis/Desktop/MARAC_API_Workshop  
Loras-MacBook-Pro:MARAC_API_Workshop lorajdavis$ ]
```

PC

PC users should see something like this:



```
vaddoni2@MSEL-SPC32 ~  
> pwd  
/home/vaddoni2  
vaddoni2@MSEL-SPC32 ~  
$
```

Note: There will be more screenshots for Windows users than Macs for the next few slides as we help PC users determine where they are. If your work computer is Windows, this will eventually matter to you.

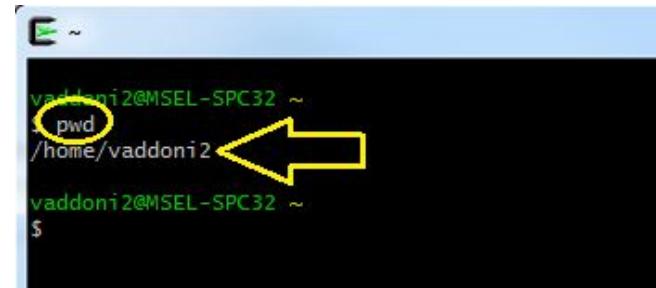
Command line bootcamp: Navigation

Where are you, and where do you want to go?

PC

Windows users will ask: but where is that? This is non-intuitive, but *you're already in C:\cygwin* because you're using the Cygwin window, so

This location:



```
vaddoni2@MSEL-SPC32 ~
$ pwd
/home/vaddoni2
vaddoni2@MSEL-SPC32 ~
$
```

Is this location:



Unix commands for Mac and Cygwin

Where am I?

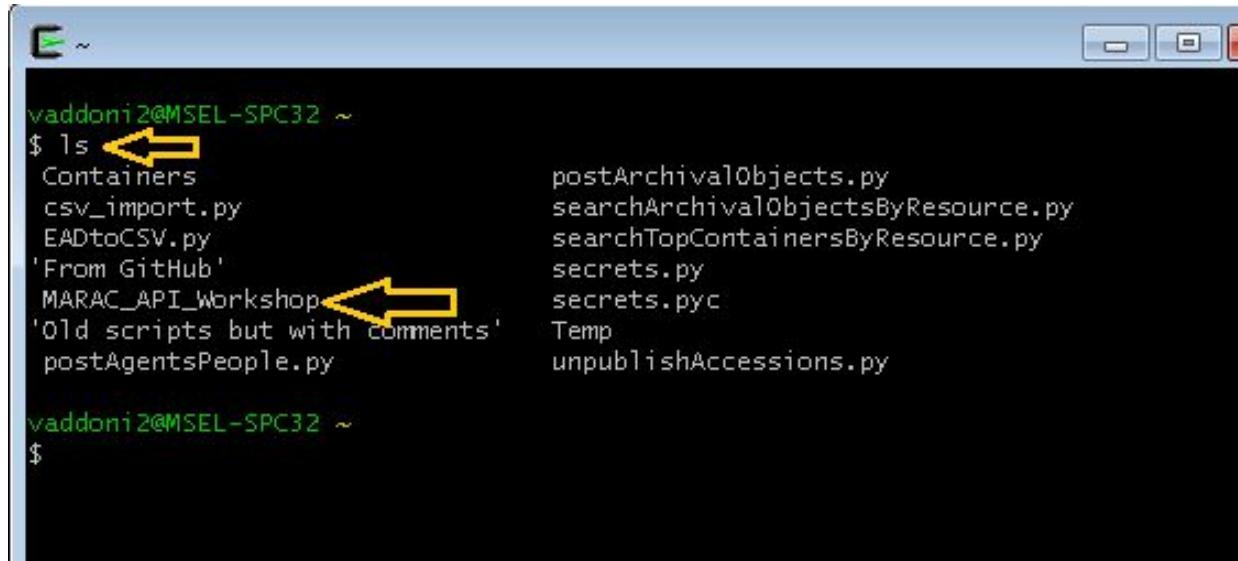
“print working directory”

`pwd`

Command line bootcamp: ls

What is here?

Everyone type `ls` and then hit enter (that is L as in List)



```
vaddoni2@MSEL-SPC32 ~
$ ls
Containers
csv_import.py
EADtoCSV.py
'From GitHub'
'MARAC_API_Workshop' <-- Yellow arrow points here
'Old scripts but with comments' <-- Yellow arrow points here
postAgentsPeople.py

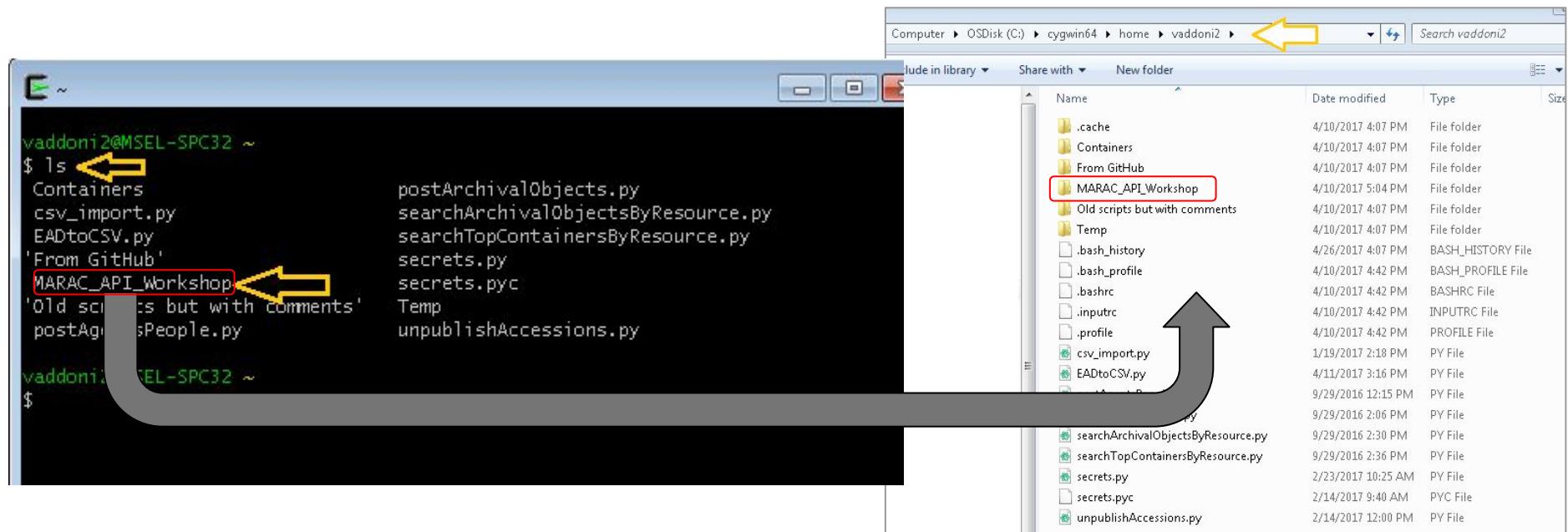
postArchivalObjects.py
searchArchivalObjectsByResource.py
searchTopContainersByResource.py
secrets.py
secrets.pyc
Temp
unpublishAccessions.py

vaddoni2@MSEL-SPC32 ~
$
```

Command line bootcamp: ls

PC

`ls` shows the same list of contents that I see if I navigate to `C:\cygwin64\home\[user name]` in Windows (this is a screenshot from Valerie's PC, you won't have all these files):



Unix commands for Mac and Cygwin

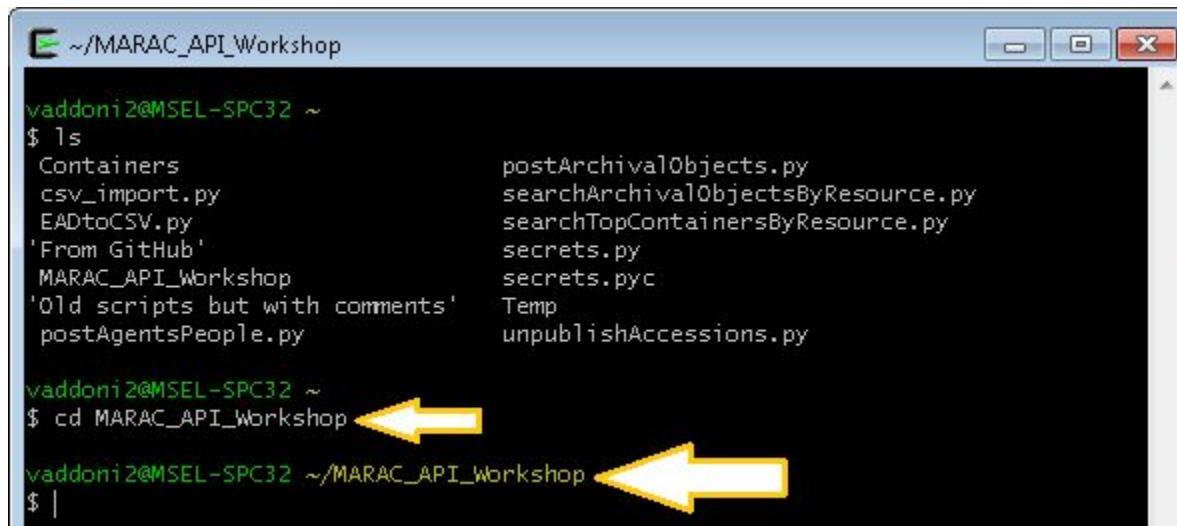
Where am I?	“print working directory”	pwd
What is here?	“list” (remember L as in List)	ls

Command line bootcamp: Navigation

Where are you, and where do you want to go?

Now you're going to move *from* where you are *into* the MARAC API Workshop clone folder:

To move into that folder type `cd` (change directory), leave a space, and then the name of the directory you want to go into: `cd ASpace_API_Workshop`



The screenshot shows a terminal window with the following text:

```
vaddoni2@MSEL-SPC32 ~
$ ls
Containers          postArchivalObjects.py
csv_import.py       searchArchivalObjectsByResource.py
EADtoCSV.py        searchTopContainersByResource.py
'From GitHub'
MARAC_API_Workshop secrets.py
'Old scripts but with comments' secrets.pyc
postAgentsPeople.py Temp
unpublishAccessions.py

vaddoni2@MSEL-SPC32 ~
$ cd MARAC_API_Workshop
```

Two yellow arrows point from the bottom right towards the command `cd MARAC_API_Workshop`. Another yellow arrow points from the bottom right towards the prompt `$ |`.

Command line bootcamp: Navigation

Where are you, and where do you want to go?

PC

Happily, the directory you're in now is more obvious with that handy yellow text.

So remember:

- The path in Windows is: C:\cygwin64\home\[user name]\ASpace_API_Workshop
(but this varies by user)
- And the *same path* in Cgywin looks like the new prompt, below:



```
vaddoni2@MSEL-SPC32: ~/MARAC_API_Workshop
```

A screenshot of a terminal window on a black background. The text "vaddoni2@MSEL-SPC32: ~/MARAC_API_Workshop" is displayed in white. A large yellow arrow points from the right towards the path " ~/MARAC_API_Workshop".

Unix commands for Mac and Cygwin

Where am I?	“print working directory”	<code>pwd</code>
What is here?	“list” (remember L as in List)	<code>ls</code>
How do I move from here to there?	“change directory”	<code>cd [type the name of the directory]</code> ↑ (don't type this, we mean an action)

Command line bootcamp: Navigation

Now you're going to move *from* the MARAC API Workshop clone folder back to the Cgywin home directory/Mac desktop:

Why? To demonstrate a simple command that means “go up one”

`cd ..` = “go up one”

```
vaddoni2@MSEL-SPC32 ~
$ cd MARAC_API_Workshop You were here
vaddoni2@MSEL-SPC32 ~/MARAC_API_Workshop You went here
$ cd ..
vaddoni2@MSEL-SPC32 ~      And you went back
$
```

Unix commands for Mac and Cygwin

Where am I?	“print working directory”	<code>pwd</code>
What is here?	“list” (remember L as in List)	<code>ls</code>
How do I move from here to there?	“change directory”	<code>cd</code> [type the name of the directory] ↑ (don't type this, we mean an action)
Move up one level		<code>cd ..</code>

Command line bootcamp: Navigation

Lastly, let's go back into the ASpace_API_Workshop directory, because that's where we need to be.

This is a good time to try the up-arrow on your keyboards to get back to a command you already issued:

- Try hitting the up-arrow a few times
- Pick the command that you need in order to get back into the ASpace_API_Workshop directory
- Use a command that will confirm where you are
- You may need to do this again, you have your handy cards to help you!

Unix commands for Mac and Cygwin

Where am I?	“print working directory”	<code>pwd</code>
What is here?	“list” (remember L as in List)	<code>ls</code>
How do I move from here to there?	“change directory”	<code>cd [type the name of the directory]</code> ↑ (don't type this, we mean an action)
Move up one level		<code>cd ..</code>
Repeat command		Up arrow on keyboard

These are called Unix commands, so Google “unix commands” for other commands that will work on Macs and in Cygwin.

To make your life harder, remember that these same commands do not work in the Windows command prompt; those are MS-DOS commands.



JUST
breathe

GET

API possibilities

Get data out → Do something to it → Put it back in



GET with GUI - Chronicling America

(and a bit about web searches versus APIs)

Vocabulary pitstop: GUI

- GUI (gooey) stands for Graphic User Interface: *every program* you use has a GUI
- But in the programming/scripting world, there is also the command line/terminal
- We will be using both: Postman is a GUI

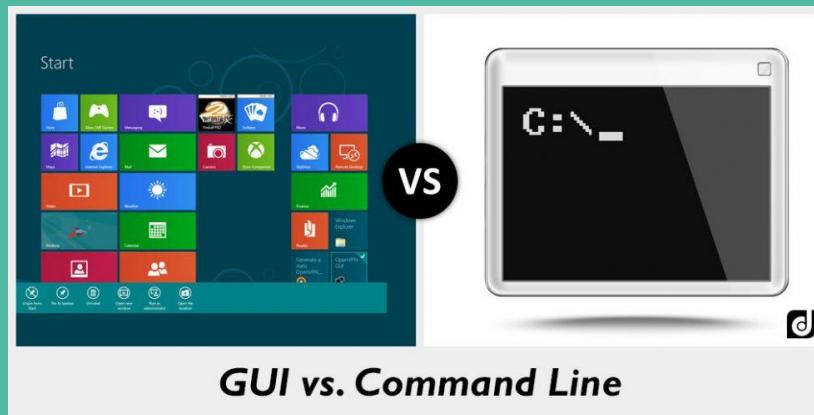


Image from <http://www.differencebtw.com/difference-between-gui-and-command-line/>

Web search versus API

Scenario: You wish to link to every digitized edition of a certain newspaper hosted in Chronicling America.

1. Navigate to ChroniclingAmerica.loc.gov
2. Search for “the times dispatch” in quotes

The screenshot shows the Chronicling America homepage. At the top, there's a search bar with the query "the times dispatch" highlighted by a green oval. Below the search bar, the results page displays three newspaper front pages from different publications: "The Ogden Standard", "The Brattleboro Daily Reformer", and "Hickory Daily Record". Each newspaper is shown with its title, location, and the number of pages. On the left side of the results page, there's a sidebar with links like "About Chronicling America", "About the Site and API" (which has a green arrow pointing to it), and "More Resources".

CHRONICLING AMERICA
Historic American Newspapers

Search Pages Advanced Search All Digitized Newspapers 1789-1924

All states + from 1789 to 1924 + "the times dispatch" GO

Pages Available: 11,845,995

100 Years Ago Today: 4/18/1917 (93 issues)

About Chronicling America
About the Site and API 
Recommended Topics
Help

More Resources

- › National Digital Newspaper Program
- › NDNP Award Recipients
- › Newspaper and Current Periodicals Reading Room
- › Ask LC Newspaper & Current Periodicals Librarian
- › Historic Newspapers on Flickr (part of the LC Flickr Commons photostream)

The Ogden Standard
NAVAL GUNS HEARD NEAR BOSTON
The Ogden standard. (12pp.)
Ogden City, Utah

The Brattleboro Daily Reformer
(8pp.)
Brattleboro, Vt.

HICKORY DAILY RECORD
The Hickory daily record. volume (4pp.)
Hickory, N.C.

Web search versus API

Scenario: You wish to link to every digitized edition of a certain newspaper hosted in Chronicling America.

1. Click any record
 2. Click All Issues



Web search versus API

This lists every issue, but not that helpful. I wonder if there's another way.

Browse Issues: The times dispatch.						
Richmond, Va. (1903-1914)						
Browse Issues About Libraries that Have It MARC Record						
Issues for: 1903 ▾						
S	M	T	W	T	F	S
Show all front pages						
Single edition: dates in bold .						
Multiple editions: dates in <i>bold italic</i> .						
January, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
February, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
March, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
April, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
May, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
June, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
July, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
August, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
September, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
October, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
November, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
December, 1903						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

GET with GUI - Chronicling America

Does Chronicling America have an API we can use to access this information we're seeing in our browsers?

YES!

To Postman!

About the Site and API

Introduction

Chronicling America provides access to information about historic newspapers and select digitized newspaper pages. To encourage a wide range of potential uses, we designed several different views of the data we provide, all of which are publicly visible. Each uses common Web protocols, and access is not restricted in any way. You do not need to apply for a special key to use them. Together they make up an extensive application programming Interface (API) which you can use to explore all of our data in many ways.

Details about these interfaces are below. In case you want to dive right in, though, we use HTML link conventions to advertise the availability of these views. If you are a software developer or researcher or anyone else who might be interested in programmatic access to the data in Chronicling America, we encourage you to look around the site, "view source" often, and follow where the different links take you to get started. When describing Chronicling America as the source of content, please use the URL and a Web site citation, such as "from the Library of Congress, Chronicling America: Historic American Newspapers site".

For more information about the open source Chronicling America software please see the [LibraryOfCongress/chronam](#) GitHub site. Also, please consider subscribing to the [ChronAm-Users](#) discussion list if you want to discuss how to use or extend the software or data from its APIs.

The API

Jump to:

- [Search](#) the newspaper directory and digitized page contents using OpenSearch.
- [Auto Suggest](#) API for looking up newspaper titles
- [Link](#) using our stable URL pattern for Chronicling America resources.
- [JSON](#) views of Chronicling America resources.
- [Linked Data](#) views of Chronicling American resources.
- [Bulk Data](#) for research and external services.
- [CORS](#) and [JSONP](#) support for your JavaScript applications.

Searching the directory and newspaper pages using OpenSearch

The [directory of newspaper titles](#) contains nearly 140,000 records of newspapers and libraries that hold copies of these newspapers. The title records are based on MARC data gathered and enhanced as part of the NDNP program.

GET with GUI - Chronicling America

Scenario: You wish to link to every digitized edition of a certain newspaper in Chronicling America.

The screenshot shows a REST client interface with the following details:

- URL: `http://chroniclingamerica`
- Method: `GET` (highlighted with a green oval)
- Request URL: `http://chroniclingamerica.loc.gov/lccn/sn85038615.json` (highlighted with a green rectangle)
- Environment: `No Environment`
- Params: None
- Send button: `Send` (highlighted with a green oval)
- Status: `200 OK`
- Time: `1425 ms`

`http://chroniclingamerica.loc.gov/lccn/sn85038615.json`

GET with GUI - Chronicling America

Scenario: You wish to link to every digitized edition of a certain newspaper in Chronicling America.

The screenshot shows a POSTMAN interface with the following details:

- Method:** GET
- URL:** <http://chroniclingamerica.loc.gov/lccn/sn85038615.json>
- Authorization:** No Auth
- Status:** 200 OK
- Time:** 1228 ms
- Body:** (Pretty, Raw, Preview, JSON) - The JSON response is displayed below.

```
1 {  
2   "place_of_publication": "Richmond, Va.",  
3   "lccn": "sn85038615",  
4   "start_year": "1903",  
5   "place": [  
6     "Virginia--Richmond"  
7   ],  
8   "name": "The times dispatch.",  
9   "publisher": "Times-Dispatch Co.",  
10  "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615.json",  
11  "end_year": "1914",  
12  "issues": [  
13    {  
14      "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-27/ed-1.json",  
15      "date_issued": "1903-01-27"  
16    },  
17    {  
18      "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-28/ed-1.json",  
19      "date_issued": "1903-01-28"  
20    }  
21  ]  
22}
```

Re-purposing API data

```
1  [
2    {
3      "place_of_publication": "Richmond, Va.",
4      "lccn": "sn85038615",
5      "start_year": "1903",
6      "place": [
7        {
8          "name": "The times dispatch",
9          "publisher": "times-Dispatch Co.",
10         "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615.json",
11         "end_year": "1914",
12         "issues": [
13           {
14             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-27/ed-1.json",
15             "date_issued": "1903-01-27"
16           },
17           {
18             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-28/ed-1.json",
19             "date_issued": "1903-01-28"
20           },
21           {
22             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-29/ed-1.json",
23             "date_issued": "1903-01-29"
24           },
25           {
26             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-30/ed-1.json",
27             "date_issued": "1903-01-30"
28           },
29           {
30             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-31/ed-1.json",
31             "date_issued": "1903-01-31"
32           },
33           {
34             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-01/ed-1.json",
35             "date_issued": "1903-02-01"
36           },
37           {
38             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-03/ed-1.json",
39             "date_issued": "1903-02-03"
40           },
41           {
42             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-04/ed-1.json",
43             "date_issued": "1903-02-04"
44           },
45           {
46             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-05/ed-1.json",
47             "date_issued": "1903-02-05"
48           },
49           {
50             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-06/ed-1.json",
51             "date_issued": "1903-02-06"
52           },
53           {
54             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-07/ed-1.json",
55             "date_issued": "1903-02-07"
56           },
57           {
58             "url": "http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-08/ed-1.json",
59             "date_issued": "1903-02-08"
60           }
61         ]
62       }
63     ]
64   ]
```

A	B
1 url	date_issued
2 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-27/ed-1.json	1903-01-27
3 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-28/ed-1.json	1903-01-28
4 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-29/ed-1.json	1903-01-29
5 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-30/ed-1.json	1903-01-30
6 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-01-31/ed-1.json	1903-01-31
7 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-01/ed-1.json	1903-02-01
8 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-03/ed-1.json	1903-02-03
9 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-04/ed-1.json	1903-02-04
10 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-05/ed-1.json	1903-02-05
11 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-06/ed-1.json	1903-02-06
12 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-07/ed-1.json	1903-02-07
13 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-08/ed-1.json	1903-02-08
14 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-10/ed-1.json	1903-02-10
15 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-11/ed-1.json	1903-02-11
16 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-12/ed-1.json	1903-02-12
17 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-13/ed-1.json	1903-02-13
18 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-14/ed-1.json	1903-02-14
19 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-15/ed-1.json	1903-02-15
20 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-17/ed-1.json	1903-02-17
21 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-18/ed-1.json	1903-02-18
22 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-19/ed-1.json	1903-02-19
23 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-20/ed-1.json	1903-02-20
24 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-21/ed-1.json	1903-02-21
25 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-22/ed-1.json	1903-02-22
26 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-24/ed-1.json	1903-02-24
27 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-25/ed-1.json	1903-02-25
28 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-26/ed-1.json	1903-02-26
29 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-27/ed-1.json	1903-02-27
30 http://chroniclingamerica.loc.gov/lccn/sn85038615/1903-02-28/ed-1.json	1903-02-28

Converting these JSON search results to a CSV (spreadsheet) took less than 10 seconds using an online converter (we just googled “JSON to CSV converter” and picked one)

Vocabulary pitstop: JSON

- JSON (jason) is the most typical data transmission standard in APIs
- It is lightweight and easy to read and NOT scary
- Consists of key-value pairs, “key”: “value”

```
<unittitle>Johns Hopkins University library records</unittitle>
```

```
“Title”: “Johns Hopkins University library records”
```

Questions???

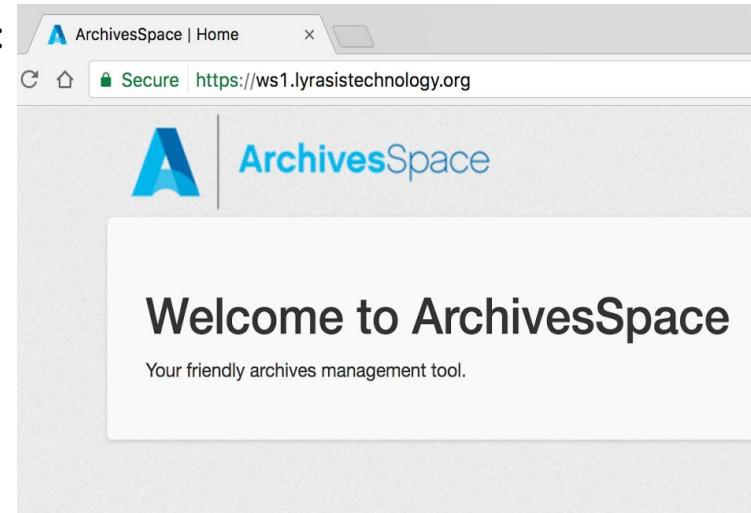


ArchivesSpace

(I know, finally right?)

ArchivesSpace!

- You've each been provided a test instance of ArchivesSpace by our gracious hosts at Lyrasis
- The address of the **staff interface** of your instance is:
[https://ws\[your#\].lyrasistechology.org/](https://ws[your#].lyrasistechology.org/)
- The address of the **API** of your instance is:
[https://ws\[your#\].lyrasistechology.org/api](https://ws[your#].lyrasistechology.org/api)
- Go check out the **staff interface** now!
username: admin
password: admin
- Your number can be found on a post-it at your seat!



ArchivesSpace! - The Scenario

Scenario: You have successfully migrated into -- or have begun to use -- an instance of ASpace at your institution, but...

- There's all this new functionality, what do I do with it?
 - I don't have barcodes for my containers, or I have faux codes
 - I do have barcodes, but they're not in ArchivesSpace. How do I get them in without ruining a student worker's semester?
- There are new fields where there were no fields before
 - I'd love to use URIs for Agents, but that's a lot of work
 - BARCODES, again with the barcodes
- We didn't use Archivists' Toolkit for accessions, how do I get them in now?
- Suppressing and unsuppressing, publishing and unpublishing, and how do I publish everything but not *those* things?

As some of you know, it's a huge undertaking and you might have dozens/hundreds/thousands of old and new problems.

ArchivesSpace! - The Scenario

Scenario: You have successfully migrated into -- or have begun to use -- an instance of ASpace at your institution, but this is a short workshop, so here are our problems for today:

1. We don't have **container profiles** in ArchivesSpace and would like to, so we need to create some
2. In following the migration instructions for 1.5, we had to add **faux codes**; we'd like to use our *actual* barcodes
3. Now that we have container profiles, we need to link them to *actual top containers*

Extra archivistry sidebar

- These are, in fact, all problems we've addressed (or are addressing) at Johns Hopkins. And this is exactly how we did (or will be) solving these issues.
- If you switch out "container profiles" for "agent records" or "subject headings" or "digital objects," the steps are similar and will likely transfer. Namely:
 - Create new records
 - Modify existing records
 - Link records

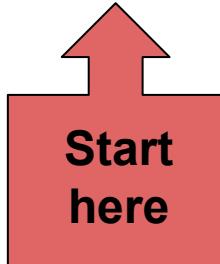
API possibilities

Get data out → Do something to it → Put it back in



API possibilities

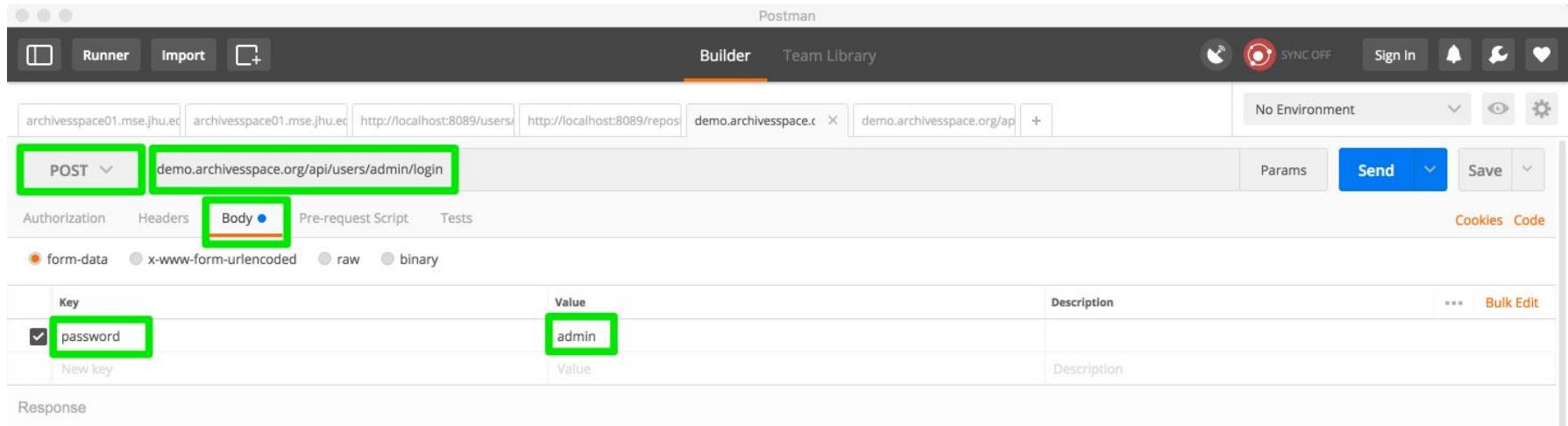
Get data out → Do something to it → Put it back in



Authenticate and GET - AS with GUI

Authenticate to AS with GUI

Before we start posting to AS, we need to authenticate, so let's do that and try a GET first:



The screenshot shows the Postman application interface. The top navigation bar includes 'Runner', 'Import', 'Builder' (which is selected), 'Team Library', and various status indicators like 'SYNC OFF', 'Sign In', and notifications. The main workspace shows a POST request to 'demo.archivesspace.org/api/users/admin/login'. The 'Body' tab is selected, showing a single key-value pair: 'password' with the value 'admin'. Other tabs like 'Authorization', 'Headers', 'Pre-request Script', and 'Tests' are visible but not selected. The bottom section shows the response table with columns for Key, Value, and Description.

Endpoint: [https://ws\[your#\].lyrasistechology.org/api/users/admin/login](https://ws[your#].lyrasistechology.org/api/users/admin/login)

password: admin

Authenticate to AS with GUI

The screenshot shows the Postman application interface. A POST request is made to `demo.archivesspace.org/api/users/admin/login`. The request body contains a form-data key `password` with value `admin`. The response status is 200 OK, with a response body containing JSON data. A green box highlights the `session` field in the JSON response, which is a long alphanumeric string. A green arrow points from the text instructions below to this highlighted field.

Postman

Builder Team Library

Runner Import

demo.archivesspace01.mse.jhu.edu demo.archivesspace01.mse.jhu.edu http://localhost:8089/users/ http://localhost:8089/repos/ demo.archivesspace.c demo.archivesspace.org/api/+ No Environment

POST demo.archivesspace.org/api/users/admin/login Params Send Save

Authorization Headers Body Pre-request Script Tests Cookies Code

form-data x-www-form-urlencoded raw binary

Key	Value	Description	... Bulk Edit
<input checked="" type="checkbox"/> password	admin		
New key	Value	Description	

Body Cookies Headers (9) Tests Status: 200 OK Time: 396 ms Size: 2.9 KB

Pretty Raw Preview JSON

```
1 {  
2   "session": "3fbf4887b333cdfc68bcab2b0306a8e0fc152fdc3342ee15a4ca4616aaa94eb",  
3   "user": {  
4     "lock_version": 26,  
5     "username": "admin",  
6     "name": "Administrator",  
7     "is_system_user": true,  
8     "create_time": "2016-08-22T18:48:50Z",  
9     "system_mtime": "2017-07-18T20:32:51Z",  
10    "user_mtime": "2017-07-18T20:32:51Z",  
11    "jsonmodel_type": "user",  
12    "groups": [],  
13    "is_admin": true,  
14    "uri": "/users/1",  
15    "agent_record": {  
16      "ref": "/agents/people/1"  
17    },  
18    "is_browsing": false  
19  }  
20}
```

Copy just the alphanumeric "session" not the quotation marks.

GET from AS with GUI

Type this

Paste this from clipboard

Hit the Send button to get a response.

Key: X-ArchivesSpace-Session

Endpoint:

[https://ws\[your#\].lyrasistechology.org/api/repositories/2/resources/2](https://ws[your#].lyrasistechology.org/api/repositories/2/resources/2)

GET from AS with GUI

The screenshot shows the Postman application interface. The top navigation bar includes 'Runner', 'Import', 'Builder' (which is selected), 'Team Library', 'SYNC OFF', 'Sign In', and various status icons. The main workspace shows a 'Builder' tab with a 'GET' request to 'demo.archivesspace.org/api/repositories/2/resources/2'. The 'Headers (1)' tab is selected, containing a key-value pair: 'X-ArchivesSpace-Session' with value '3fbf4887b333cdfc68bcab2b0306a8e0fc152fdc3342ee15a4ca4616aaa94e6'. Below the headers, the 'Body' tab displays a JSON response with numerous fields, including 'lock_version', 'title', 'publish', 'restrictions', 'ead_id', 'ead_location', 'finding_aid_title', 'finding_aid_filing_title', 'finding_aid_date', 'finding_aid_author', 'finding_aid_language', 'created_by', 'last_modified_by', 'create_time', 'system_mtime', 'user_mtime', 'id_0', 'language', 'level', and 'finding_aid_description_rules'. The status bar at the bottom indicates a 200 OK status, 187 ms time, and 8.05 KB size.

Postman

Builder Team Library

No Environment

GET demo.archivesspace.org/api/repositories/2/resources/2

Headers (1)

X-ArchivesSpace-Session

3fbf4887b333cdfc68bcab2b0306a8e0fc152fdc3342ee15a4ca4616aaa94e6

Body Cookies Headers (9) Tests

Pretty Raw Preview JSON

```
1 {  
2   "lock_version": 3,  
3   "title": "Stump family business records and personal papers",  
4   "publish": true,  
5   "restrictions": false,  
6   "ead_id": "678.xml",  
7   "ead_location": "http://library.carpidiem.edu/ynhsc/YNHSC-MS-678.xml",  
8   "finding_aid_title": "Guide to the Stump family business records and personal papers <num>YNHSC.MS.678</num>",  
9   "finding_aid_filing_title": "Stump family business records and personal papers",  
10  "finding_aid_date": "October 22, 2002",  
11  "finding_aid_author": "Clement Samuels and Lola Amarillo.",  
12  "finding_aid_language": "Finding aid is in English.",  
13  "created_by": "admin",  
14  "last_modified_by": "bradw",  
15  "create_time": "2016-08-23T22:20:07Z",  
16  "system_mtime": "2017-04-19T00:48:47Z",  
17  "user_mtime": "2016-09-05T19:28:04Z",  
18  "suppressed": false,  
19  "id_0": "YNHSC.MS.678",  
20  "language": "eng",  
21  "level": "collection",  
22  "finding_aid_description_rules": "docs"
```

Status: 200 OK Time: 187 ms Size: 8.05 KB

GET from AS with GUI

The screenshot shows the ArchivesSpace web application interface. At the top, the URL in the browser bar is highlighted with a green box and contains the text: demo.archivespace.org/resources/2#tree::resource_2. The page title is "ArchivesSpace". The navigation bar includes links for okmarks, Feedly, Archive-It, DACS, EADiva, LC EAD, Relators, Chicago, JHU GitHub, JIRA, ArchivesSpace wiki, myJHU, and The Sheridan Libraries. A dropdown menu shows "Select Repository" set to "System" and "admin" logged in. Below the header, there are links for Home, Browse, Create, and a search bar. The main content area shows a collection record for "Stump family business records and personal papers". The record structure is as follows:

- Collection level:
 - Business records, 1878-1973 (Series)
 - Family papers, 1886-1979 (Series)

Below the record structure, a sidebar on the left lists navigation options: Basic Information, Dates, Extents, Finding Aid Data, Related Accessions, Agent Links, Subjects, Notes, and Classifications. The main content area has tabs for Edit, Calculate Extent, Add Event, Publish All, View Published, Export, Merge, Transfer, Suppress (highlighted in yellow), and Delete. The current tab is "Edit". The main title of the record is "Stump family business records and personal papers". A red banner at the bottom right of the screen displays the text "It's the same record!".

Basic Information

Title	Stump family business records and personal papers
Identifier	YNHSC.MS.678
Level of Description	Collection
Language	English
Published?	True

It's the same record!

POST to AS with GUI-Container profiles

Container profiles

What's a container profile?

ASpace offers “container modeling” for the first time in the archives world.

Every type of box (ex. record carton) in your library gets its own record (a profile), which records its height, width, and depth. This helps calculate space on a huge scale, and is a game-changer for some repositories.

So, we have boxes o'plenty ----->

But to use this feature, we need to get their profiles into AS.



API possibilities

Get data out → Do something to it → Put it back in



POST to AS with GUI - Container profiles

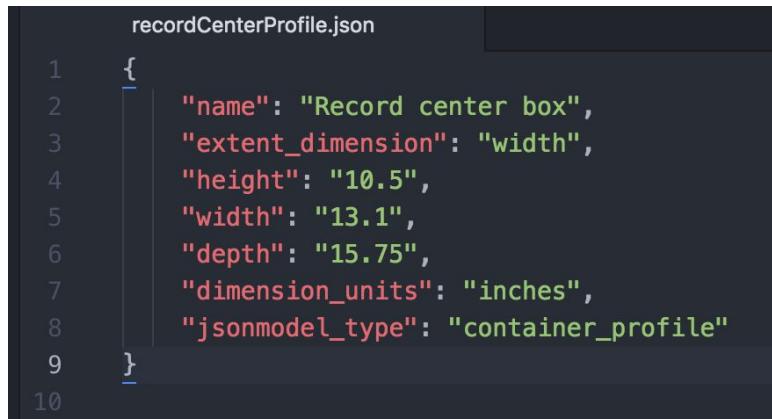
1. Navigate to the directory with our cloned GitHub repo

Mac users: Desktop

Windows users:

C:\cygwin64\home\[username]\ASpace_API_Workshop

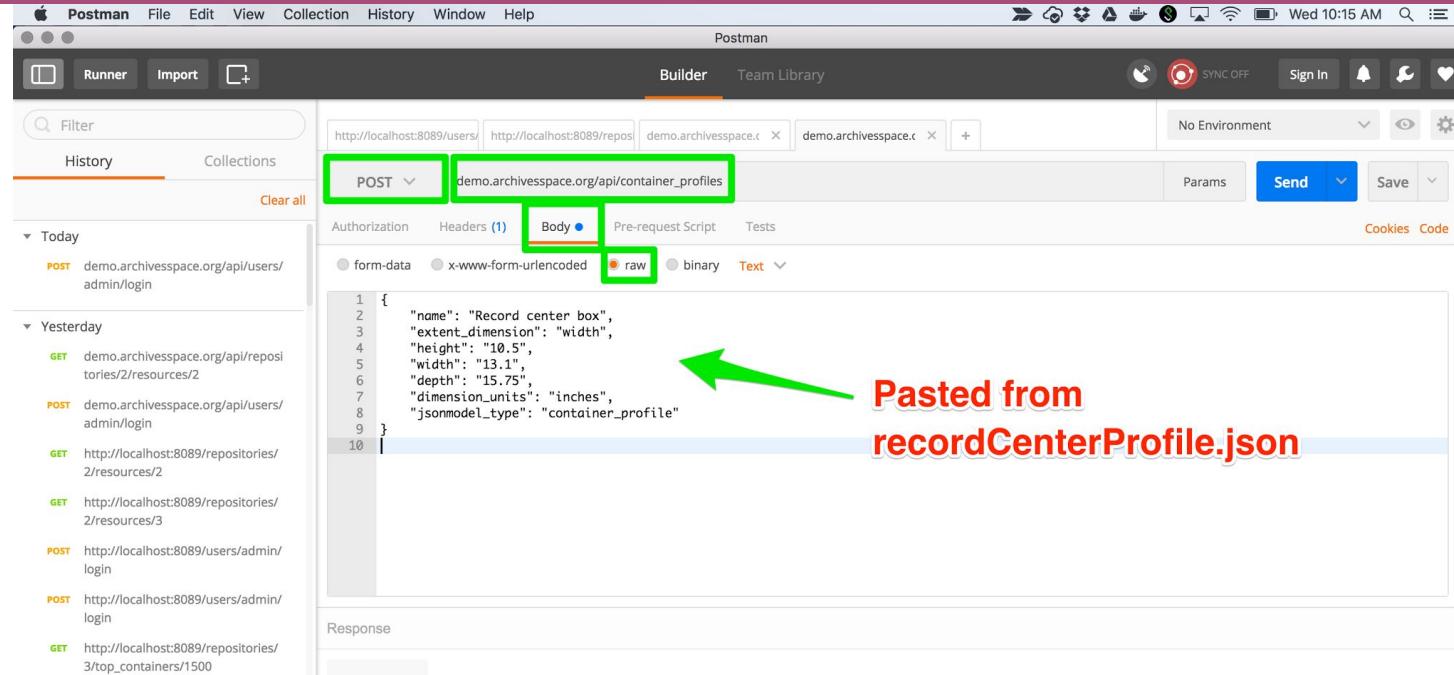
2. Open “recordCenterProfile.json” with Atom
3. Packages > Atom Beautify > Beautify
4. Here is the container profile for a record center carton in JSON, ready to go
5. Copy, and go back to Postman



A screenshot of the Atom code editor showing a JSON file named "recordCenterProfile.json". The file contains the following JSON object:

```
recordCenterProfile.json
1 {  
2   "name": "Record center box",  
3   "extent_dimension": "width",  
4   "height": "10.5",  
5   "width": "13.1",  
6   "depth": "15.75",  
7   "dimension_units": "inches",  
8   "jsonmodel_type": "container_profile"  
9 }  
10
```

POST to AS with GUI - Container profiles



The screenshot shows the Postman application interface. The top navigation bar includes File, Edit, View, Collection, History, Window, and Help. The main toolbar has icons for Runner, Import, and a new tab. The title bar says "Postman". The left sidebar shows a history of API calls, including a POST to demo.archivespace.org/admin/login and several GET requests to demo.archivespace.org/api/repositories. The central workspace shows a POST request to "demo.archivespace.org/api/container_profiles". The "Body" tab is selected, showing a raw JSON payload:

```
1 {  
2     "name": "Record center box",  
3     "extent_dimension": "width",  
4     "height": "10.5",  
5     "width": "13.1",  
6     "depth": "15.75",  
7     "dimension_units": "inches",  
8     "jsonmodel_type": "container_profile"  
9 }  
10 |
```

A green arrow points from the text "Pasted from recordCenterProfile.json" to the JSON code in the body panel. The status bar at the bottom indicates "Response".

Pasted from
recordCenterProfile.json

Endpoint:

[https://ws\[your#\].lyrasistechology.org/api/container_profiles](https://ws[your#].lyrasistechology.org/api/container_profiles)

POST to AS with GUI - Container profiles

1. Navigate to the directory with our cloned GitHub repo
Mac users: Desktop
Windows users:
C:\cygwin64\home\[username]\ASpace_API_Workshop
2. Open “containerProfiles.json” with Atom
3. Packages > Atom Beautify > Beautify
4. Here are ALL the profiles, ready to go
5. Copy, and go back to Postman

```
[{  
    "name": "Flat box01",  
    "extent_dimension": "width",  
    "height": "3",  
    "width": "12",  
    "depth": "16",  
    "dimension_units": "inches",  
    "jsonmodel_type": "container_profile"  
},  
{  
    "name": "Flat box02",  
    "extent_dimension": "width",  
    "height": "3",  
    "width": "21",  
    "depth": "25",  
    "dimension_units": "inches",  
    "jsonmodel_type": "container_profile"  
},  
{  
    "name": "Flat box03",  
    "extent_dimension": "width",  
    "height": "3",  
    "width": "9",  
    "depth": "11",  
    "dimension_units": "inches",  
    "jsonmodel_type": "container_profile"  
},  
{
```

POST to AS with GUI - Container profiles

The screenshot shows the Postman interface with the following details:

- Method:** POST (highlighted with a green box)
- URL:** demo.archivesspace.org/api/container_profiles
- Body Type:** raw (highlighted with a green box)
- Body Content:** A JSON object containing two container profile definitions. The first profile is for "Flat box01" with dimensions 12x3x16 inches. The second profile is for "Flat box02" with dimensions 21x3x25 inches.

```
[{"name": "Flat box01", "extent_dimension": "width", "height": "3", "width": "12", "depth": "16", "dimension_units": "inches", "jsonmodel_type": "container_profile"}, {"name": "Flat box02", "extent_dimension": "width", "height": "3", "width": "21", "depth": "25", "dimension_units": "inches", "jsonmodel_type": "container_profile"}]
```

A red arrow points from the text "Pasted from containerProfiles.json" to the JSON code in the Postman body editor.

Pasted from
containerProfiles.json

Endpoint:

[https://ws\[your#\].lyrasistechology.org/api/container_profiles](https://ws[your#].lyrasistechology.org/api/container_profiles)

POST to AS with GUI - Container profiles

The screenshot shows a POST request to `localhost:8089/container_profiles`. The request body contains the following JSON:

```
89     "jsonmodel_type": "container_profile"
90   },
91   {
92     "name": "Postcard box",
93     "extent_dimension": "width",
94     "height": "5.5",
95     "width": "3.5",
96     "depth": "7",
97     "dimension_units": "inches",
98     "jsonmodel_id": "1c40a40e-40d0-40f0-8000-000000000000"
99   },
100  [
101    {
102      "name": "Photo album box",
103      "extent_dimension": "width",
104      "height": "10.5",
105      "width": "13.1",
106      "depth": "15.75",
107      "dimension_units": "inches",
108      "jsonmodel_type": "container_profile"
109    }
110  ]
```

The response body shows an error message:

```
1 [
2   {
3     "error": "can't convert String into Integer"
4   }
5 ]
```

Don't hate us: you cannot post multiple records through the GUI
This frustrating exercise will save you a month
(use your month wisely: take a vacation from computers)



JUST
breathe

Scripting

Scripting - Why?

Using a GUI application like **Postman** to interact with APIs can be a great way to *learn, explore, and troubleshoot*, but ultimately you'll hit a brick wall, because:

- It takes an **awful lot of clicks** to get out a small amount of data (relatively speaking)
- If you want to get multiple full records OUT you've got to run a GET as **many times** as there are records you want to retrieve
- While you can POST many one-off changes using a GUI like Postman, you can rarely get a GUI to make **intelligent, iterative POSTs at scale**
- **Manually authenticating** is a pain
- Though we told you that you will be sometimes playing the role of “application” in this API world, you don't *always* want to **be the application!**

Scripting - How?

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 you will use in this workshop are **Python** because: 1) Python (and, to a lesser degree, Ruby) is Lora's preferred hammer, and 2) unscientifically speaking, it seems that Python is the preferred language of archivists (which means there's more to steal/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

Scripting - No, really, *how*?

Remember all the legwork you did both at home and during the early part of this workshop? You've:

- Installed applications, including the **text editor Atom**
- Installed (or located) a **shell**, namely *Terminal* (Mac) or *Cygwin* (Windows)
- Installed (or confirmed installation of) **python**

Guess what? You've set up a **python development environment** already! Good work!

With that work complete, for the remainder of this workshop you should only need to type `python [name of script here].py` into Terminal/Cygwin, and you'll be **executing Python scripts!** Just remember:

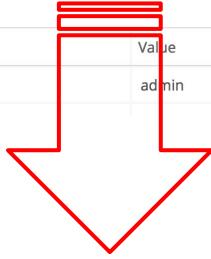
- You should be located in the same directory as the script (and any files it is reliant on) before you type your command (you can always `ls` to confirm the script is there!)

For more, see: <http://www.shubhro.com/2014/05/29/development-environment/> and/or
<http://python-guide-pt-br.readthedocs.io/en/latest/starting/install/osx/> (Mac specific)

POST to AS with script- Container profiles

POST to AS with script

Before we start posting to AS, we need to authenticate, so how do we do that with scripts?



The screenshot shows the Postman interface for a POST request to `http://localhost:8089/users/admin/login`. The 'Body' tab is selected, showing a single form-data entry named 'password' with the value 'admin'. A red arrow points from this entry to a code editor window below.

Key	Value
password	admin

The code editor window displays a Python script named `secrets.py` with the following content:

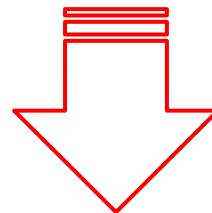
```
baseURL='http://localhost:8089'  
user='admin'  
password='admin'
```

```
baseURL='http://localhost:8089'  
user='admin'  
password='admin'
```

POST to AS with script

“Keep it secret, keep it safe.” - Gandalf

```
secrets.py  
1 baseURL='http://localhost:8089'  
2 user='admin'  
3 password='admin'
```



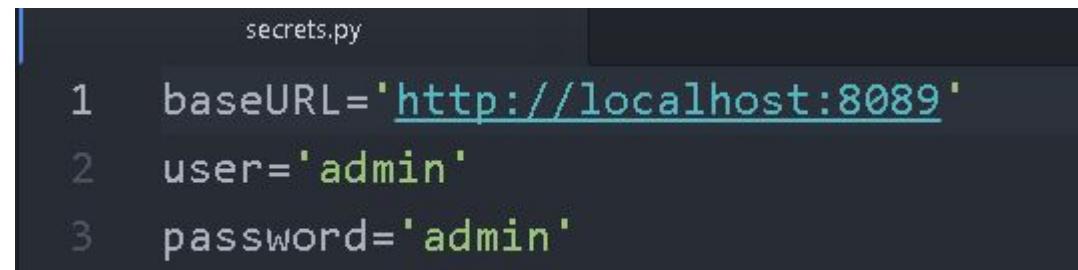
*This means no manual authenticating!
Learn to script just for that and call it a win!*

```
auth = requests.post(baseURL + '/users/' + user + '/login?password=' + password).json()  
session = auth["session"]  
headers = {'X-ArchivesSpace-Session': session}
```

POST to AS with script

- We're all connecting to different instances of ArchivesSpace (so we don't overwrite and/or clash with each others' work!), so we need to tell `secrets.py` where each of our individual instances live.
- Navigate to the `ASpace_API_Workshop` directory we cloned from GitHub earlier:
 - Mac users: This should be your Desktop
 - Windows users: This should be "C:\cygwin64\home\[username]\ASpace_API_Workshop"
- Open `secrets.py` in Atom
- Change the line:

baseURL='<http://localhost:8089>'
to
baseURL='[https://ws\[your#\]lyrasistechology.org/api/](https://ws[your#]lyrasistechology.org/api/)'



A screenshot of a code editor showing a file named `secrets.py`. The code contains three lines of Python code defining variables for baseURL, user, and password.

```
1  baseURL='http://localhost:8089'  
2  user='admin'  
3  password='admin'
```

POST to AS with script- Container Profiles

Mac

1. In the Finder navigate to your ASpace_API_Workshop directory
2. Ctrl+click the ASpace_API_Workshop directory, and select “New Terminal at Folder”
3. Type `ls` and examine the contents of that folder
4. Type `python postContainerProfiles.py` (case sensitive!)
5. Navigate back to AS in your browser ([https://ws\[your#\].lyrasistechology.org](https://ws[your#].lyrasistechology.org))

PC

1. Open Cygwin
2. Type `cd ASpace_API_Workshop` to enter the ASpace_API_Workshop directory
3. Type `ls` and examine the contents of that folder
4. Type `python postContainerProfiles.py` (case sensitive!)
5. Navigate back to AS in your browser ([https://ws\[your#\].lyrasistechology.org](https://ws[your#].lyrasistechology.org))

ArchivesSpace!

Scenario: You have successfully migrated into -- or have begun to use -- an instance of ASpace at your institution, but this is a short workshop, so here are our problems for today:

1. ~~We don't have container profiles in ArchivesSpace and would like to, so we need to create some~~
2. In following the migration instructions for 1.5, we had to add **faux codes**; we'd like to use our *actual* barcodes
3. Now that we have container profiles, we need to link them to *actual top containers*

POST to AS with script- Edit barcodes

API possibilities

Get data out → Do something to it → Put it back in



Barcodes/top_containers

Répète: ASpace offers “container modeling” for the first time in the archives world.

Every type of box (ex. record carton) in your library gets its own record (a profile), which records its height, width, and depth.

Every actual box in your collections *also* gets a record, and this is called a top container. Simply put, this is the thing you put a number on: Box 1.

So, your archives might have hundreds or thousands of boxes called “Box 1”

Barcodes make that sane for AS. Hence, AS 1.5 requires some sort of unique code in every top container record.

Container 1: [31151030080422]		Top Container
Container Profile	Record center box	
Indicator	1	
Barcode	31151030080422	
Exported to ILS	Not exported	
Legacy	False	
Restricted?		

Barcodes/top_containers

Every marathon runner and every top_container must have a unique ID to participate.



Barcodes/top_containers

1. Navigate back to ASpace in your browser
([https://ws\[your#\].lyrasistechology.org](https://ws[your#].lyrasistechology.org))
2. Browse > Resources > Gérard Defaux papers > View > expand Research Materials > click on any file > scroll down to Instances > see fake barcode
3. These are the barcodes generated by the barcoder plugin. Hopkins has thousands of them.
4. Navigate to our [GitHub](#) and look at *barcodes.csv*

Barcodes/top_containers

If you're in ASpace, you will have some version of this problem, which is why we're featuring it.

Your top containers might:

- Have barcodes already! Well, this is still a lesson in editing records
- Have “faux codes,” like the ones in the AS vagrant
- Have nothing, and you have no idea where to start. We’ll have to refer you to the [AS 1.5 instructions](#) and [this plugin](#) by Chris Fitzpatrick

So let's fix our problem and imagine it working at scale.

Barcodes/top_containers

- Let's investigate the Gérard Defaux papers in your instance of ArchivesSpace
- Look at the instances attached to the archival objects under the “Research Materials” series
- These are “fauxcodes”
- We want our REAL barcodes!
- Luckily, we were able to map the fauxcode to the “real” barcode and generate a csv with this information
- Let's take a look at “barcodes.csv”

POST to AS with script- Edit barcodes

Mac

1. In the Finder navigate to your ASpace_API_Workshop directory
2. Ctrl+click the ASpace_API_Workshop directory, and select “New Terminal at Folder”
3. Type `ls` and examine the contents of that folder
4. Type `python postBarcodes.py` (case sensitive!)
5. Navigate back to AS in your browser ([https://ws\[your#\].lyrasistechology.org](https://ws[your#].lyrasistechology.org))

PC

1. Open Cygwin
2. Type `cd ASpace_API_Workshop` to enter the ASpace_API_Workshop directory
3. Type `ls` and examine the contents of that folder
4. Type `python postBarcodes.py` (case sensitive!)
5. Navigate back to AS in your browser ([https://ws\[your#\].lyrasistechology.org](https://ws[your#].lyrasistechology.org))

ArchivesSpace!

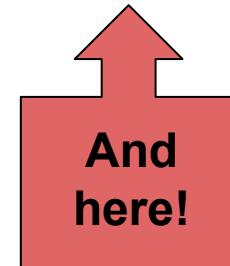
Scenario: You have successfully migrated into -- or have begun to use -- an instance of ASpace at your institution, but this is a short workshop, so here are our problems for today:

1. We don't have ~~container profiles~~ in ArchivesSpace and would like to, so we need to create some
2. In following the migration instructions for 1.5, we had to add ~~faux codes~~; we'd like to use our *actual* barcodes
3. Now that we have container profiles, we need to link them to *actual top containers*

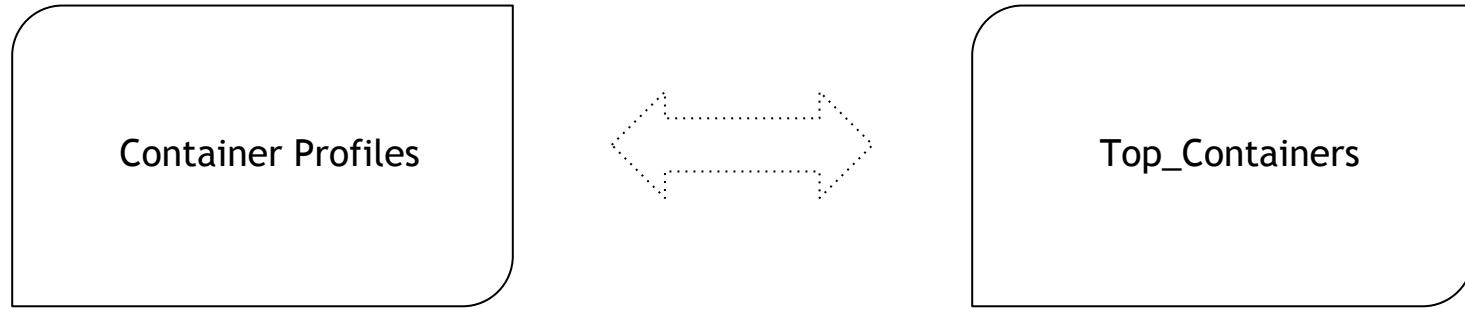
POST to AS with script- Link profiles

API possibilities

Get data out → Do something to it → Put it back in



Linking profiles to containers



POST to AS with script - Linking profiles

1. Type `ls` and examine the contents of that folder
2. Type `python asLinkProfiles.py` (case sensitive!)
3. You will be prompted for a resource id and a container id... how do you determine what you need to know?
4. Let's return to ASpace real quick

POST to AS with script - Linking profiles

The interface – just another lens on the same data – is helpful for constructing API requests.

View a resource record for its resource number:

A screenshot of the ArchivesSpace interface. The title bar says "ArchivesSpace | Resource". The address bar shows the URL "localhost:8080/resources/1#tree::resource_1". A red circle highlights this URL. Below the address bar, the ArchivesSpace logo and the word "ArchivesSpace" are displayed. The main content area shows a breadcrumb navigation: "Home / Resources / Gérard Defaux papers". Underneath, there's a tree view with "Gérard Defaux papers" expanded, showing "Research Materials" as a child node. At the bottom, there are "Browse" and "Create" dropdown menus, a "Search All Records" input field, and a "Record center box" containing "Basic Information" and "Edit" buttons.

View a container profile for its profile number:

A screenshot of the ArchivesSpace interface. The title bar says "ArchivesSpace | Container Profile". The address bar shows the URL "localhost:8080/container_profiles/12". A red circle highlights this URL. Below the address bar, the ArchivesSpace logo and the word "ArchivesSpace" are displayed. The main content area shows a breadcrumb navigation: "Home / Container Profiles / Record center box". Underneath, there's a "Record center box" containing "Basic Information" and "Edit" buttons. To the right, a separate panel titled "Record center box" shows the "Basic Information" tab.

ArchivesSpace! You did it!

Scenario: You have successfully migrated into -- or have begun to use -- an instance of ASpace at your institution, but this is a short workshop, so here are our problems for today:

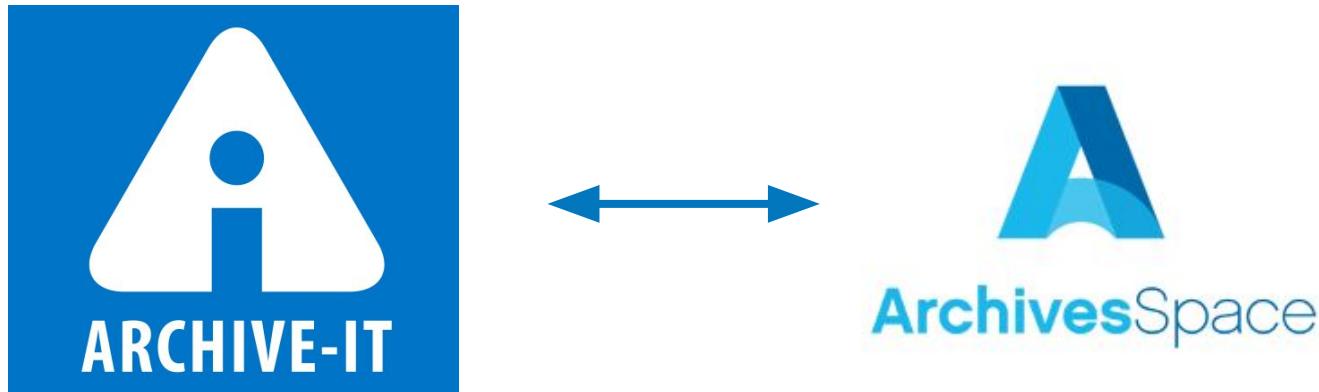
1. We don't have ~~container profiles~~ in ArchivesSpace and would like to, so we need to create some
2. In following the migration instructions for 1.5, we had to add ~~faux codes~~; we'd like to use our *actual* barcodes
3. Now that we have ~~container profiles~~, we need to link them to *actual* ~~top~~ containers



GET and POST across two applications with Python

App-to-app Communication

Scenario: As your university's web archivist, you wish to make your Archive-It web crawls accessible to users who access your collections via ArchivesSpace without having to individually create digital objects every time you run a new Archive-It crawl.



App-to-app Communication



ArchivesSpace

1. In ArchivesSpace, navigate to the “Records of the Johns Hopkins University Library” resource
2. Expand Subgroup 12: Library Website
3. Click on library.jhu.edu
4. Note that archival object’s level

App-to-app Communication

We now know that we can access ArchivesSpace's archival object records via the [ArchivesSpace API](#), right?



In fact, with a decent enough search we could probably even have a script return JUST those archival objects with the level "**“Web archive.”**"

Since we're going to want to keep programmatically working with/altering this data after we find it, we'll use a [Python script](#), instead of Postman to run this search.

```
39 # search AS for archival_object's with level "Web archive"
40 query =
41     '/search?page=1&filter_term[]={"primary_type":"archival_object"}&filter_term[]={"level":"Web
42     archive"}'
43 ASoutput = requests.get(baseURL + query, headers=headers).json()
44 print 'Found ' + str(len(ASoutput['results'])) + ' archival objects with the instance type "Web
45     archive."'
```

Code snippet from `archivelt.py`

App-to-app Communication

The screenshot shows the top navigation bar of the Archive-It website. It includes a logo with a stylized 'A' inside a blue square, followed by the text 'ARCHIVE-IT'. Below the logo are four main navigation links: 'HOME', 'EXPLORE', 'LEARN MORE', and 'CONTACT US'. To the right of these links is a section with the text 'The leading web archiving service for collecting and accessing cultural heritage on the web' and 'Built at the Internet Archive'. Above this text are icons for Facebook, Twitter, and WordPress, followed by a 'Login' button.

Narrow Your Results

Type of Collecting Organization

Sort By: Count | (A-Z)

Colleges & Universities

Collecting Organization

Sort By: Count | (A-Z)

Johns Hopkins University (11)

Explore All Archives

Items in the archive are listed below. Narrow your results at left, or enter a search query below to find a collecting organization, collection, site, specific URL or to search the text of archived webpages.

Collection Name : Johns Hopkins University web collection

The following results were found for the term(s): library.jhu.edu

- 11 Sites were found.
- Additional results for library.jhu.edu may be found by searching [within the page text](#).

Sites

Search Page Text

Page 1 of 1 (11 Total Results)

Sort By: Best Match | Title (A-Z) | Title (Z-A) | URL (A-Z) | URL (Z-A)

URL: <http://library.jhu.edu>

Collection: Johns Hopkins University web collection

Organization: Johns Hopkins University

Captured 43 times between Aug 20, 2010 and Feb 28, 2017



App-to-app Communication

Does **Archive-It** have an **API** we can use to access this information we're seeing in our browsers?



YES!



Wayback Machine APIs

The Internet Archive Wayback Machine supports a number of different APIs to make it easier for developers to retrieve information about Wayback capture data.

The following is a listing of currently supported APIs. This page is subject to change frequently, please check back for the latest info.

Updated on September, 24, 2013

Wayback Availability JSON API

This simple API for Wayback is a test to see if a given url is archived and currently accessible in the Wayback Machine. This API is useful for providing a 404 or other error handler which checks Wayback to see if it has an archived copy ready to display. The API can be used as follows:

<http://archive.org/wayback/available?url=example.com>

which might return:

```
{  
  "archived_snapshots": {  
    "closest": {  
      "available": true,  
      "url": "http://web.archive.org/web/20130919044612/http://example.com/",  
      "timestamp": "20130919044612",  
      "status": "200"  
    }  
  }  
}
```

App-to-app Communication



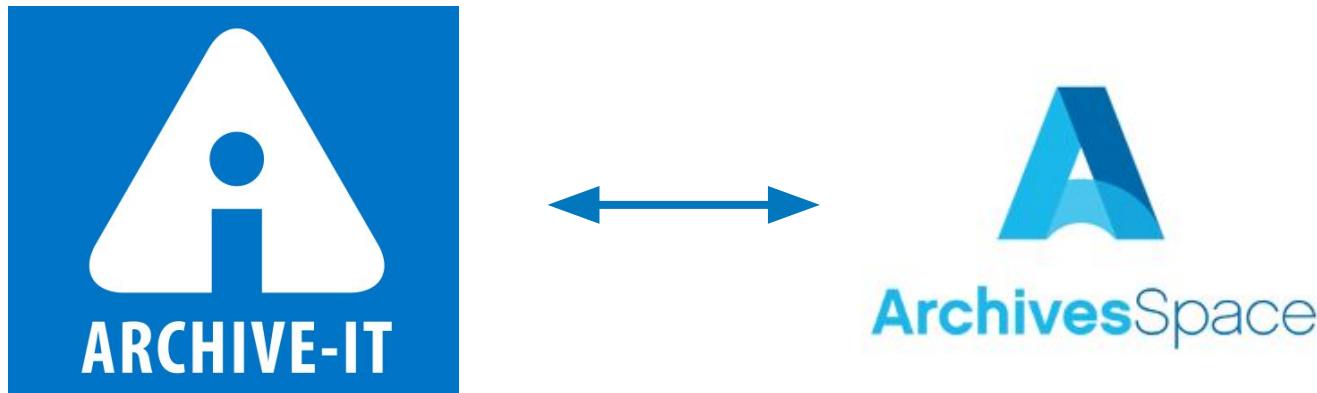
With the right amount of trial and error, we can also get information about our Archive-It holdings out of the Archive-It API with a **Python script** as well!

```
67     for crawl in crawlList:  
68         doid = 'https://wayback.archive-it.org' + '/' + archiveit_coll + '/' +  
69         crawl['timestamp'] + '/' + crawl['original']  
70         query = '/search?page=1&filter_term[]={"primary_type":"digital_object"}&q=' + doid  
71         existingdOID = requests.get(baseURL + query, headers=headers).json()  
72         if len(existingdOID['results']) > 0:
```

Code snippet from archivelt.py

App-to-app Communication

1. In Terminal/Cygwin run `python archiveIt.py` and let's see what happens!
2. Go check out that “Records of the Johns Hopkins University Library” resource record once again.



Questions???

Icing and Advice

Icing: Interpreting (ASpace) API endpoints

Official: <http://archivesspace.github.io/archivesspace/api>

Non-Official: <https://gist.github.com/jgpawletko/18a1982ec91b290039a968fe4eb924e8>

GET /repositories/:repo_id/resources/:id

Description

Get a Resource

```
curl -H "X-ArchivesSpace-Session: $SESSION" "http://localhost:8089/repositories/:repo_id/resources/1"
```

Parameters

Integer id – The ID of the record

Integer repo_id – The Repository ID – The Repository must exist

[String] resolve – A list of references to resolve and embed in the response

Returns

200 – (:resource)

Icing: Interpreting (ASpace) API endpoints

The interface – just another lens on the same data – is helpful for constructing API requests.

Determining the repository number:

A screenshot of the Archivesspace interface. At the top, there's a navigation bar with 'Browse', 'Create', and a search bar. Below it, a breadcrumb trail shows 'Home / Repositories / Special Collections'. On the left, a sidebar lists 'Repository Fields' and 'Contact Details'. In the center, a large box contains the text 'Special Collections' with a blue 'Edit' button above it and a blue 'Repository' button to its right. A green callout box highlights this central area, and a green arrow points upwards from the bottom of the slide towards this box. Below the main text, a light blue box says 'Repository is Currently Selected'. The URL in the browser is <http://archivesspace.fakeu.edu/repositories/3>.

Determining an agent number:

A screenshot of the Archivesspace interface showing an agent record. At the top, a breadcrumb trail shows 'Home / Agents / Abbe, Cleveland, 1838-1916'. On the left, a sidebar lists 'Basic Information', 'Dates of Existence', 'Names', 'Contact Details', and 'Linked Records', each with a blue 'Edit' button. The main content area shows the agent's name 'Abbe, Cleveland, 1838-1916' with an orange 'Agent' button next to it. Below it is a section titled 'Basic Information' with fields for 'Agent Type' (Person) and 'Publish' (True). A note at the bottom says 'Created by admin 2016-05-13 1'. At the bottom, there's a section for 'Dates of Existence' with a single entry: 'Existence' from '1838 – 1916'. An orange arrow points upwards from the bottom of the slide towards the breadcrumb trail.

Sample endpoint from documentation: http://localhost:8089/repositories/:repo_id/resources/1

Example “fake” endpoint that mimics real life: <http://archivesspace.fakeu.edu:8089/repositories/3/resources/1>

http://localhost:8089	The address of your instance of ASpace. You will ONLY replace “local host,” the colon and port number remain. EX. http://archivesspace.fakeu.edu:8089
<code>/repositories</code>	The presence of “repositories” here means that this endpoint is repository-specific. Some non-repo specific requests in AS are for Agents and Access Points, which span all of AS. EX. http://archivesspace.fakeu.edu:8089/agents
<code>:repo_id</code>	The presence of this colon means this value will be unique to your institution. How can you determine the repository number? You can use the repo endpoint, or, from within AS navigate Systems > Manage Repositories > select repository > and look at the address bar. EX. http://archivesspace.fakeu.edu:8089/repositories/3
<code>/resources</code>	Other examples are /accessions or /top_containers. EX. http://archivesspace.fakeu.edu:8089/repositories/3/accessions
<code>/1</code>	The first resource. How can you determine resource numbers? Navigate to the resource in the interface and its number will be in the address bar. EX. http://archivesspace.fakeu.edu:8089/repositories/3/resources/1

Icing: What IS GitHub anyway?

The least most helpful thing you'll hear is, "It's in our GitHub!"

If you're serious about learning to script, you should watch the 10 million GitHub intro videos on YouTube

Even casual users will benefit from using other people's scripts (that's how devs work!)

Let's go look at our repo together, we made it for you!

https://github.com/jhu-archives-and-manuscripts/ASpace_API_Workshop

Icing: There will ALWAYS be “gotchas”

We purposely made you “fail” a few times today. Get used to it!

- 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.

Icing: A frequent ASpace “gotcha”

it·er·a·tion

īdē' rāSH(ə)n/

noun

1. the repetition of a process or utterance.
 - repetition of a mathematical or computational procedure applied to the result of a previous application, typically as a means of obtaining successively closer approximations to the solution of a problem.

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,
```

Icing: A frequent ASpace frustration

Session time and page limits

- Ever been timed out of your bank account? Frustrating but vital
- The amount of time you have after authenticating is called “session time”
- ASpace default is very short
- Ask your ASpace tech person to up the session time in the AS config; we’ve provided instructions in the take home document (which really will happen, we promise)

Icing: What about XYZ application?

Many other applications provide robust APIs for your use:

- ILSes like [Voyager](#);
- Digital exhibition applications like Omeka;
- Digital collections/repository applications like ContentDM or Islandora; and,
- Cloud-based sharing applications like Dropbox/Drive/Box.

While we didn't work through exercises with these applications today, hopefully you now know the **steps to take** to do future API work of your own, namely:

- Research the API, including authentication requirements and endpoint documentation;
- Play around in Postman (or another GUI API application);
- Determine whether your desired tasks can be accomplished through the GUI, or if you need a scripting language;
 - If the latter, determine if someone else has already tackled your task (for example, on GitHub)
- Iteratively test (in a *non-Production environment!*)
- Profit!

User stories

Can the API create reports?

- Yes... but an API is the wrong tool for reporting
- Like using a jack-hammer indoors: yes it will work, but it will be more effort with the wrong tool
- For AS users in particular:
 - Wait for reporting to improve, because it will
 - But in the meantime (or for more customized reporting) we suggest connecting AS to MS Access. If you don't know Access, it's easier to learn that than learning to script just for reports.
 - You can find explanatory slides in the workshop GitHub > additional resources
 - Credit to Nancy Enneking, Head of Institutional Records at the Getty and Celia Caust-Ellenbogen, Friends Historical Library of Swarthmore College for the method

Can APIs change the staff/user interface?

- No: the API is only a way of manipulating data
- Look at the API endpoints and see if one relates to the change you want to make
- We did write something that [changed enumerations](#)

Can APIs improve my agent records?

- Yes!
- Hopkins made changes to almost all of our Agent and Subject records through the API
- We added VIAF ids to Agents and converted LCSH to FAST
- You already have improved Corporate Names from earlier
- Let's put those to work!

Load, GET, and compare - VIAF

API possibilities

Get data out → Do something to it → Put it back in



We are
here

Load, GET, and compare - VIAF

Scenario: You have an existing spreadsheet containing a number of organizational names that are either subjects or creators of some of your collections. Now, you want to take this manually-made spreadsheet and actually do some authority control work!

Load, GET, and compare - VIAF

1. Take a look at “organizations.csv” in GitHub:

https://github.com/jhu-archives-and-manuscripts/ASpace_API_Workshop/blob/master/organizations.csv

2. From Terminal/Cygwin type

```
python viafReconciliationCorporate.py
```

In theory, we could make changes to this .csv file (add/change lines, etc.), but we’re not going to have you do this today. Feel free to try it at home, though, with the caveat that some spreadsheet editors (particularly Excel on Macs) output messy .csv’s that cause errors when the script is run.

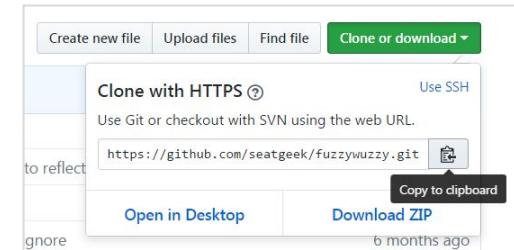
Load, GET, and compare - VIAF

Uh oh...

Technical Pitstop - Installing extra packages

Downloading the fuzzy wuzzy python package:

1. Google “fuzzy wuzzy github” and it should be the first result
2. Click the green “Clone or Download” button, click the little clipboard icon, and copy the path to the clipboard
3. Confirm that terminal/cygwin is still in the ASpace_API_Workshop folder
4. Type `git clone` then paste the path, which should look like:
`git clone https://github.com/seatgeek/fuzzywuzzy.git`
5. Hit enter



Technical Pitstop - Installing extra packages

Installing the fuzzy wuzzy python package:

1. Type `cd fuzzywuzzy` to enter the newly created fuzzywuzzy directory
2. Type `ls` to see what's in the directory and note the script “`setup.py`”
3. To execute that setup script, type `python setup.py install`
4. The package is **installed!**
5. Type `cd ..` to return you to the `ASpace_API_Workshop` directory

Load, GET, and compare - VIAF

Let's try this again!

1. Type or up-arrow `python viafReconciliationCorporate.py`
2. Success!
3. Go back to your spreadsheet program and open the newly created “viafCorporateResults.csv” file from the ASpace_API_Workshop directory

Mac users: Desktop/ASpace_API_Workshop

Windows users: C:\cygwin64\home\[username]\ASpace_API_Workshop

4. The script created this new file for you. Let's inspect it!

POST with script - VIAF

Post from a CSV - VIAF

Before posting these VIAF corporate entities into ArchivesSpace, we must first add VIAF as a valid “name source” in ArchivesSpace

1. Go to the ArchivesSpace staff interface at:

[https://ws\[your#\].lyrasistechology.org](https://ws[your#].lyrasistechology.org)

2. In the top right select “System > Manage Controlled Value Lists”
3. In the drop-down you’re provided, select “Name Source (name_source)”
4. In the middle right click “Create Value”
5. Name this value “vifaf”
 - a. Note: The punctuation here is important since it must match what is in our Python script. All lowercase, no spaces.
6. Click “Create Value”

Post from a CSV - VIAF

With the VIAF name source added to ArchivesSpace, next:

1. Confirm you still have a file called “viafCorporateResults.csv” in your ASpace_API_Workshop directory
2. From within the ASpace_API_Workshop directory in Terminal/Cygwin execute `python postVIAFOrganizations.py`
3. Go back to the ArchivesSpace staff interface at:
[https://ws\[your#\].lyrasistechology.org](https://ws[your#].lyrasistechology.org)
4. In the top left click “Browse > Agents”
5. Voila!

Wrap up!

Thanks!

This workshop is a heavily abbreviated/modified version of the series of API workshops co-taught for MARAC by Lora Woodford and **Valerie Addonizio**. For more on these workshops, see: https://github.com/jhu-archives-and-manuscripts/MARAC_API_Workshop

Thanks to **Eric Hanson**, Metadata Librarian at JHU, for his troubleshooting and all-around good spirits (YAY ERIC!)

Several of these concepts and/or exercises were tested on JHU Sheridan Libraries' staff, including NDSR Resident **Elizabeth England**.

Anyone who has ever shared help/advice/support on blogs/listservs/bar stools who are too numerous to fully name (but we'll try in person if you ask us to!).