



Intro to the API using ArchivesSnake

ArchivesSpace Member Forum
Washington DC

2018-08-14

Valerie Addonizio
vaddoniz@jhu.edu

Dave Mayo
dave_mayo@harvard.edu

Lora Woodford
lora.woodford@lyrasis.org

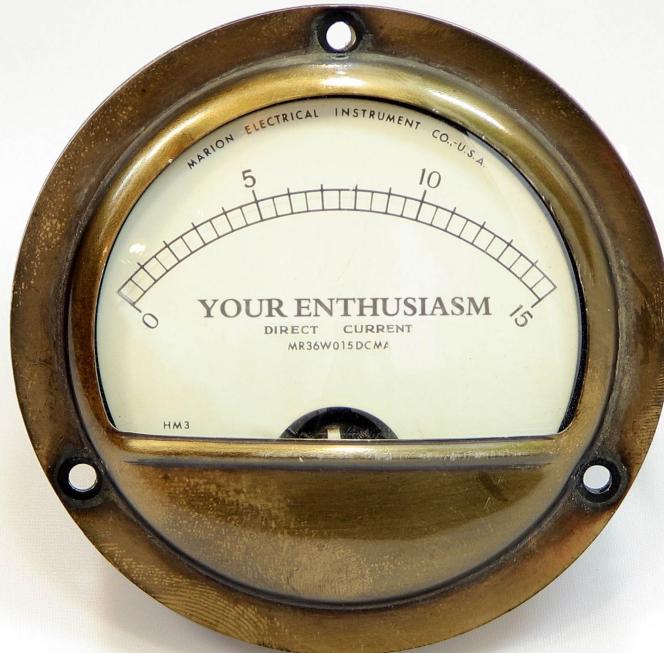
Introduction



A broad audience

You can be
here

Or here



Because we're here for everyone



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.

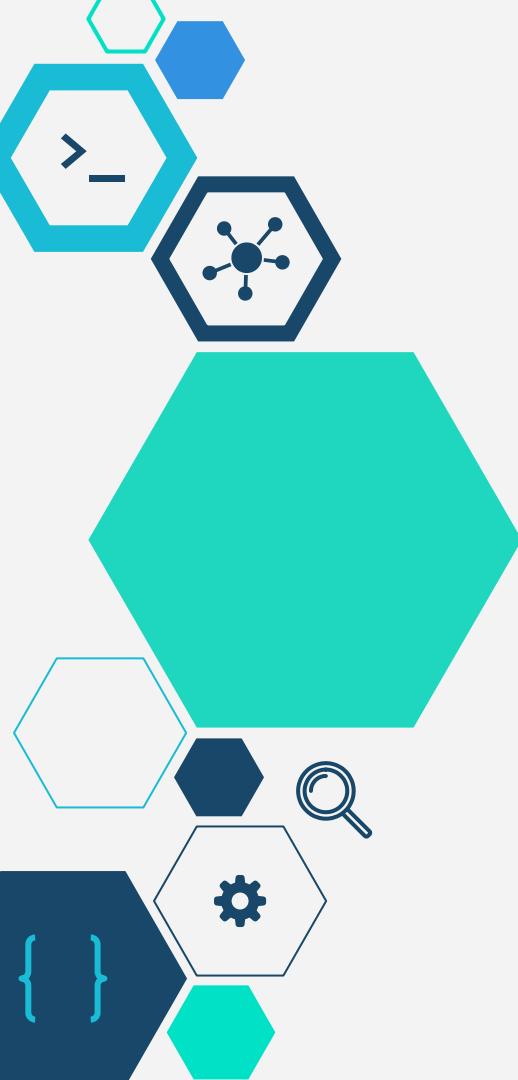
Are you a 2?

We hope you'll leave a
3.

Are you a 3?

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





Setting up your tech

We promise you're in the right workshop



Pitstop: Clone the repo

1. Open a browser and navigate to <https://github.com/pobocks/api-training>
2. You should see this:

Screenshot of the GitHub repository page for `pobocks / api-training`.

The repository was forked from `archivesspace/api-training`. It has 150 commits, 3 branches, 0 releases, and 3 contributors. The license is MIT.

This branch is 3 commits ahead, 9 commits behind `archivesspace:master`.

Recent commits:

Commit	Message	Date
.archivessnake.yml	Added .archivessnake.yml	Latest commit 94f6306 an hour ago
additional resources	Converted stuff in additional resources, deleted HTML masquerading as...	7 days ago
.archivessnake.yml	Added .archivessnake.yml	an hour ago
.gitignore	Update .gitignore	5 months ago
LICENSE	Initial commit	a year ago
Pre-Workshop_Mac.pdf	Adding pre-workshop intructions	5 months ago
Pre-Workshop_Windows.pdf	Adding pre-workshop intructions	5 months ago
README.md	Update README.md	5 months ago
archiveIt.py	API Code updated to use ASnake. Changes of note besides:	13 days ago



Pitstop: Clone the repo

1. Bookmark it!
2. Click the green button, click the little clipboard icon, and **Copy to clipboard**

A screenshot of a GitHub repository page for a project named "api-training". The page shows basic statistics: 1 watch, 0 stars, 0 forks, 0 releases, 1 contributor, and an MIT license. At the bottom, there are buttons for "Create new file", "Upload files", "Find file", and a prominent green "Clone or download" button. A large red arrow points from the top of the slide down to the "Clone or download" button.

A screenshot of a "Clone with HTTPS" modal window. It contains instructions: "Use Git or checkout with SVN using the web URL." Below is a text input field containing the URL "https://github.com/pobocks/api-training.git". To the right of the URL is a small icon of a clipboard with a checkmark, which is circled in red. Below the URL are two buttons: "Open in Desktop" and "Download ZIP".



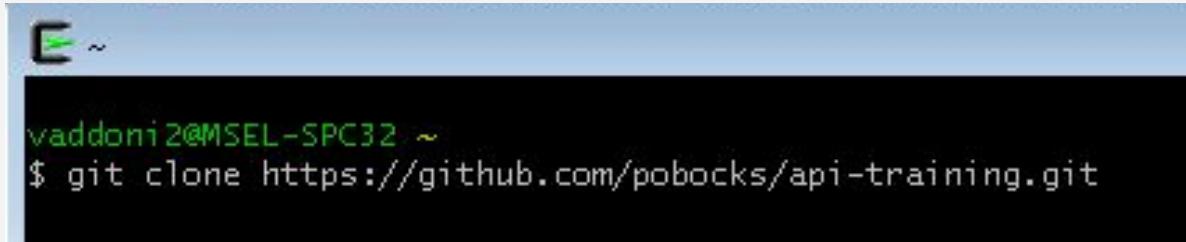
Pitstop: Clone the repo

Mac

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

PC

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



```
vaddoni2@MSEL-SPC32:~$ git clone https://github.com/pobocks/api-training.git
```

`git clone https://github.com/pobocks/api-training.git`



Pitstop: Clone the repo

You should see
text like this



```
vaddoni2@MSEL-SPC32 ~  
$ git clone https://github.com/pobocks/api-training.git  
Cloning into 'api-training'...  
remote: Counting objects: 518, done.  
remote: Compressing objects: 100% (205/205), done.  
remote: Total 518 (delta 321), reused 503 (delta 306), pack-reused 0  
Receiving objects: 100% (518/518), 25.59 MiB | 52.83 MiB/s, done.  
Resolving deltas: 100% (321/321), done.
```

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

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 “**api-training**,” is a direct clone of what you see in your browser on [github.com](https://github.com/pobocks/api-training).



Pitstop: Clone the repo

Find the folder named “api-training” and open the PDF of these slides.

- Mac users: check your desktop
- Windows users: look in C:\Cygwin\home\[username]

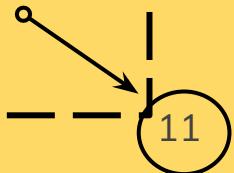
Everyone should have the slides open by the end of this slide. Raise your hand if you can't find it.

Name	Date modified	Type	Size
.git	8/13/2018 2:08 PM	File folder	
additional resources	8/13/2018 2:08 PM	File folder	
.archivessnake.yml	8/13/2018 2:08 PM	YML File	1 KB
.gitignore	8/13/2018 2:08 PM	Text Document	1 KB
archiveIt.py	8/13/2018 2:08 PM	PY File	5 KB
asLinkProfiles.py	8/13/2018 2:08 PM	PY File	3 KB
barcodes.csv	8/13/2018 2:08 PM	CSV File	1 KB
containerProfiles.json	8/13/2018 2:08 PM	JSON file	3 KB
dos.csv	8/13/2018 2:08 PM	CSV File	1 KB
LICENSE	8/13/2018 2:08 PM	File	2 KB
Member Forum API - Slides.pdf	8/13/2018 2:08 PM	Adobe Acrobat D...	200 KB
organizations.csv	8/13/2018 2:08 PM	CSV File	1 KB
postBarcodes.py	8/13/2018 2:08 PM	PY File	1 KB

You can follow along starting from this slide

This terrible shade of yellow should be easy to find.

Reference slide numbers at key points





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
- Cygwin ([Windows users](#))
 - A Unix emulator for Windows
 - Macs are already in a Unix environment (one reason for their preference among developers)





Pitstop: Pre-workshop steps

Pre-Workshop Instructions for Mac users



- Did you receive and complete the pre-workshop instructions?
- Did you install Atom?
- Postman?
- Did you check for gcc?
- Did you install Homebrew?

If you have not completed these steps, navigate to the api-training folder on your Desktop and open the Pre-Workshop instructions for Macs.

Pre-Workshop Instructions for Windows



- Did you receive and complete the pre-workshop instructions?
- Did you install Atom?
- Postman?
- Cygwin, with all the weird instructions?

If you have not completed these steps, navigate to C:/Cygwin/home/[username]/api-training and open the Pre-Workshop instructions for Windows.



Pitstop: Note this convention

If you see normal text, it's normal.

If you see a different font highlighted in a different color,
this indicates a command you need to type and execute.

Type or copy exactly what you see.

If you see weird stuff and lots of characters...
then type/copy weird stuff and lot of characters.



Pitstop: Installing packages



Mac users only:

1. Open Terminal ➞
2. Type `brew update` and hit enter
3. Type `brew install python` and hit enter
4. Type `pip3 install ArchivesSnake` and hit enter



Pitstop: Installing packages



Windows users only:

1. Open Cygwin 
2. Type `python3 --version` and confirm you have **Python 3.6**
3. Type `easy_install-3.6 pip` 

```
Installed /usr/lib/python3.6/site-packages/pip-18.0-py3.6.egg  
Processing dependencies for pip  
Finished processing dependencies for pip
```

4. Wait for the command prompt to come back
5. Type `pip3 install ArchivesSnake` and hit enter 

```
running setup.py install for pyyaml...  
Successfully installed ArchivesSnake-0.6.0 a
```

The background of the image is a dramatic sky at either dawn or dusk. The upper portion of the sky is a deep, saturated blue. Below it, a layer of clouds is bathed in warm, fiery colors of orange, red, and yellow, transitioning into cooler blues and purples further down. The overall effect is one of tranquility and natural beauty.

Breathe



APIs in Action

Let's start elsewhere, away from ArchivesSpace



APIs are pretty basic; it's how you use them that gets complex

You have a few basic choices that you can combine:

COMMAND | Execution



Vocabulary pitstop: API Commands

- **GET, POST, and DELETE** are 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



In our first exercise, we will be using

GET with a GUI

(COMMAND | Execution)



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/powershell/terminal
- We will be using both: Postman is a GUI

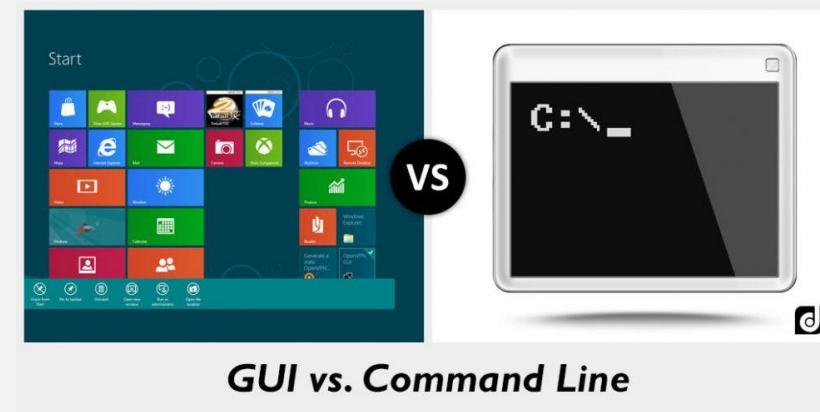


Image from <http://www.differencebtw.com/difference-between-gui-and-command-line/>



GET with GUI: ProPublica

Including:

- A lot of vocabulary
- Web searches versus APIs
- Repurposing data



GET with GUI

ProPublica.org emphasizes research-based journalism and provides its data to other reporters and the public

1. Navigate to ProPublica.org
2. Click News Apps
3. Scroll to the [Nonprofit Explorer](#)
4. Next slide...

ProPublica ProPublica Illinois Data Store

PROPUBLICA

TOPICS ▾ SERIES ▾ NEWS APPS GET INVOLVED IMPACT

FISCAL YEAR ENDING DEC. 2015

Total Revenue **\$12,491,149**
Total Functional Expenses **\$4,585,781**

Notable sources of revenue	Amount	Percent of total revenue
Contributions	\$6,882,164	0.4%
Program services	\$3,079	0.0%
Investment income	\$0	0%
Bond proceeds	\$0	0%
Royalties	\$0	0%
Net property income	\$0	0%
Net fundraising	\$0	0%
Sales of assets	\$0	0%
Net inventory sales	\$0	0%
Other revenue	\$23,891	0.2%

Notable expenses	Amount	Percent of total expenses
Professional compensation	\$1,272,433	10.2%
Professional fundraising fees	\$6,628,192	50.8%
Other salaries and wages	\$67,064	0.5%

Updated: Nonprofit Explorer

by Mike Tigas, Sisi Wei, and Alec Glassford,
Aug. 10, 2017, 11:37 a.m. EDT

We have added raw data from more than 1.9 million electronically filed Form 990 documents dating back to 2010.



GET with GUI

1. Do a search for “animal” in the Nonprofit Explorer and hit Search. This is a list of animal-based charities
2. Look at the address bar...

Search for Organizations Search for People

Search for a Nonprofit

Enter a nonprofit's name, a keyword, or city

Examples: ProPublica, Research or Minneapolis

State: Any State Major nonprofit categories: Any Category Org. Type: Any Type

SEARCH



Sidebar: Web URL vs API endpoint

Here's the address bar after that search. Note that your search term is in there:

https://projects.propublica.org/nonprofits/search?utf8=%E2%9C%93&q=animal&state%5Bid%5D=&ntee%5Bid%5D=&c_code%5Bid%5D=

- We tend to think of URLs as *locations*...
- We call your attention to this as we discuss Endpoints.



GET with GUI

Research Tax-Exempt Organizations

Search for a Nonprofit

Enter a nonprofit's name, a keyword, or city

animal

Examples: ProPublica, Research or Minneapolis

State

Major nonprofit categories ?

Org. Type ?

Any State

Any Category

Any Type

SEARCH

[Advanced Search](#) | [People Search](#)

Back to the results.

Click on the first result,

Animal Alliance

5072 organization results for animal. Results are ordered by relevance.

Note: The figure in the recent annual revenue column might be different than the revenue listed on the latest available Form 990. This is because the IRS provides more recent revenue data for some exempt organizations than is available in the publicly released detailed filings.

1 [2](#) [3](#) [4](#) ... [49](#) [50](#) [51](#) [Next >](#) [Last »](#)

Company Name	Location	NTEE Classification	Recent annual revenue
ANIMAL ALLIANCE	WOODLAND HLS, CA	Animal Protection and Welfare ↳ Animal-Related	\$18,019 2014
ANIMAL ANGELS	JACKSBORO, TX		
ANIMAL APPEAL	HERMITAGE, PA	Boys Clubs ↳ Youth Development	



Sidebar: Web URL vs API endpoint

Note that the address for Animal Alliance has a unique identifier in it.

<https://projects.propublica.org/nonprofits/organizations/731663130>



GET with GUI - ProPublica

1. Open Postman



The screenshot shows the Postman interface. The URL bar at the top has 'http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal' entered. Below the URL bar, there are tabs for Authorization, Headers, Body, Pre-request Script, and Tests. The Authorization tab is currently selected. In the bottom left, there's a dropdown for 'Type' set to 'No Auth'. On the right side, there are buttons for 'Params', 'Send', 'Save', 'Cookies', and 'Code'. A green box highlights the URL field, and a green circle highlights the 'Send' button. An arrow points from the text 'Type this next to the GET command:' in the list below to the URL field in the screenshot.

2. Type this next to the GET command:

`http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal`



GET with GUI - ProPublica

Do you see these results?

If not, check these settings.

Look at the first result.

There's that unique ID again.

It's an Employer
Identification Number (EIN)

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

- Method: GET
- URL: <http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal>
- Status: 200 OK
- Time: 759 ms
- Headers (21) tab is visible but not expanded.
- Body tab is selected, showing the JSON response.
- Pretty tab is available for viewing the JSON in a human-readable format.
- Raw tab is available for viewing the raw JSON text.
- Preview tab is available for viewing a preview of the JSON data.
- JSON dropdown menu is open, showing options: JSON, XML, YAML, and CSV.
- The JSON response body is as follows:

```
1 {  
2     "total_results": 5072,  
3     "organizations": [  
4         {  
5             "ein": 731663130,  
6             "strein": "73-1663130",  
7             "name": "ANIMAL ALLIANCE",  
8             "sub_name": "ANIMAL ALLIANCE",  
9             "city": "WOODLAND HLS",  
10            "state": "CA",  
11            "ntee_code": "D20",  
12            "raw_ntee_code": "D20",  
13            "subseccd": 3,  
14            "has_subseccd": true,  
15            "have_filings": null,  
16            "have_extracts": null,  
17            "have_pdfs": null,  
18            "score": 439.6073  
19        },  
20        {  
21            "ein": 752461428,  
22            "strein": "75-2461428",  
23            "name": "ANIMAL ANGELS",  
24            "sub_name": "ANIMAL ANGELS",  
25            "city": "JACKSBORO",  
26            "state": "TX",  
27        }  
28    ]  
29}  
30
```



GET with GUI - ProPublica

That was fun. Do it again:

<http://projects.propublica.org/nonprofits/api/v2/organizations/731663130.json>

```
1 <organization>: {  
2   "id": 731663130,  
3   "ein": 731663130,  
4   "name": "ANIMAL ALLIANCE",  
5   "careofname": "% LESTER J KNISPEL",  
6   "address": "21731 VENTURA BLVD STE 300",  
7   "city": "WOODLAND HLS",  
8   "state": "CA",  
9   "zipcode": "91364-1851",  
10  "exemption_number": 0,  
11  "subsection_code": 3,  
12  "affiliation_code": 3,  
13  "classification_codes": "1000",  
14  "ruling_date": "2004-04-01",  
15  "deductibility_code": 1,  
16  "foundation_code": 15,  
17 }  
18 }
```



Vocabulary pitstop: JSON

- JSON (jason or jay-sohn) 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”

EAD:

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

JSON:

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



Vocabulary pitstop: Endpoint

- An API Endpoint is simply a unique URL that
 - Calls a function, like a keyword search, or
 - Represents an object or a collection of objects
- In AS, an Endpoint can be a Resource record, an accession, a container, etc.
- All Endpoints are URLs, but not all URLs are Endpoints
- Constructing Endpoints is a fundamental requirement for using APIs
- You cannot paste what's in an address bar as an endpoint, but, addresses usually help you find your way



GET with GUI - ProPublica

Endpoints

`http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal`

Asking the API to perform a keyword search

`http://projects.propublica.org/nonprofits/api/v2/organizations/731663130.json`

Asking the API to display a specific record with a unique identifier



GET with GUI - ProPublica

Endpoints

`http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal`

Asking the API to perform a keyword search

`http://projects.propublica.org/nonprofits/api/v2/organizations/731663130.json`

Asking the API to display a specific record with a unique identifier

`http://aspace.library.jhu.edu/repositories/3/archival_objects/67503`

Hmm...

Sidebar: Web search vs API

5072 organization results for *animal*. Results are ordered by relevance.

Note: The figure in the recent annual revenue column might be different than the revenue listed on the IRS website because the IRS provides more recent revenue data for some exempt organizations than is available in the API.

1 2 3 4 ... 49 50 51 Next > Last »

Company Name	Location	NTEE Classification
ANIMAL ALLIANCE	WOODLAND HLS, CA	Animal Protection and Welfare ↳ Animal-Related
ANIMAL ANGELS	JACKSBORO, TX	
ANIMAL APPEAL	HERMITAGE, PA	Boys Clubs ↳ Youth Development

```
{  
    "total_results": 5072,  
    "organizations": [  
        {  
            "ein": 731663130,  
            "tax_id": "73-1663130",  
            "name": "ANIMAL ALLIANCE",  
            "sub_name": "ANIMAL ALLIANCE",  
            "city": "WOODLAND HLS",  
            "state": "CA",  
            "ntee_code": "D20",  
            "raw_ntee_code": "D20",  
            "subseccd": 3,  
            "has_subseccd": true,  
            "have_filings": null,  
            "have_extracts": null,  
            "have_pdfs": null,  
            "score": 439.6073  
        }  
    ]  
}
```

You don't get different results, you get the same results in a different format.



Sidebar: Repurposing results

If you don't want to work directly in JSON, that's cool. Just convert it to what you're comfortable with so that you can use the tools you know.

What's more important is getting it back into JSON.

Sidebar: Repurposing results

JSON

```
{  
    "ein": 731663130,  
    "strein": "73-1663130",  
    "name": "ANIMAL ALLIANCE",  
    "sub_name": "ANIMAL ALLIANCE",  
    "city": "WOODLAND HLS",  
    "state": "CA",  
    "ntee_code": "D20",  
    "raw_ntee_code": "D20",  
    "subseccd": 3,  
    "has_subseccd": true,  
    "have_filings": true,  
    "have_extracts": true,  
    "have_pdfs": true,  
    "score": 441.1612  
},  
{  
    "ein": 752461428,  
    "strein": "75-2461428",  
    "name": "ANIMAL ANGELS",  
    "sub_name": "ANIMAL ANGELS",  
    "city": "JACKSBORO",  
    "state": "TX",  
    "ntee_code": null,  
    "raw_ntee_code": null,  
    "subseccd": 3,  
    "has_subseccd": true,  
    "have_filings": true,  
    "have_extracts": true,  
    "have_pdfs": true,  
    "score": 441.1612  
},  
{  
    "ein": 232896507,  
    "strein": "23-2896507",  
    "name": "ANIMAL APPEAL",  
    "sub_name": "ANIMAL APPEAL",  
}
```

XML

```
<organizations>  
    <element>  
        <city>WOODLAND HLS</city>  
        <ein>731663130</ein>  
        <has_subseccd>true</has_subseccd>  
        <have_extracts>true</have_extracts>  
        <have_filings>true</have_filings>  
        <have_pdfs>true</have_pdfs>  
        <name>ANIMAL ALLIANCE</name>  
        <ntee_code>D20</ntee_code>  
        <raw_ntee_code>D20</raw_ntee_code>  
        <score>441.1612</score>  
        <state>CA</state>  
        <strein>73-1663130</strein>  
        <sub_name>ANIMAL ALLIANCE</sub_name>  
        <subseccd>3</subseccd>  
    </element>
```

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

Sidebar: Repurposing results

```
{  
    "ein": 731663130,  
    "strein": "73-1663130",  
    "name": "ANIMAL ALLIANCE",  
    "sub_name": "ANIMAL ALLIANCE",  
    "city": "WOODLAND HLS",  
    "state": "CA",  
    "ntee_code": "D20",  
    "raw_ntee_code": "D20",  
    "subsecd": 3,  
    "has_subsecd": true,  
    "have_filings": true,  
    "have_extracts": true,  
    "have_pdfs": true,  
    "score": 441.1612  
},  
  
{  
    "ein": 752461428,  
    "strein": "75-2461428",  
    "name": "ANIMAL ANGELS",  
    "sub_name": "ANIMAL ANGELS",  
    "city": "JACKSBORO",  
    "state": "TX",  
    "ntee_code": null,  
    "raw_ntee_code": null,  
    "subsecd": 3,  
    "has_subsecd": true,  
    "have_filings": true,  
    "have_extracts": true,  
    "have_pdfs": true,  
    "score": 441.1612  
},  
  
{  
    "ein": 232896507,  
    "strein": "23-2896507",  
    "name": "ANIMAL APPEAL",  
    "sub_name": "ANIMAL APPEAL",  
}
```

JSON

Spreadsheet

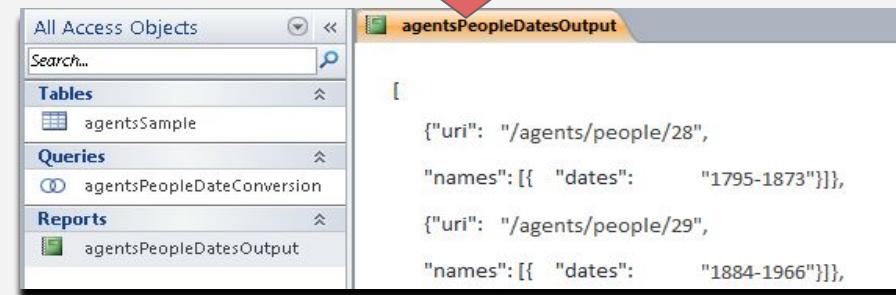
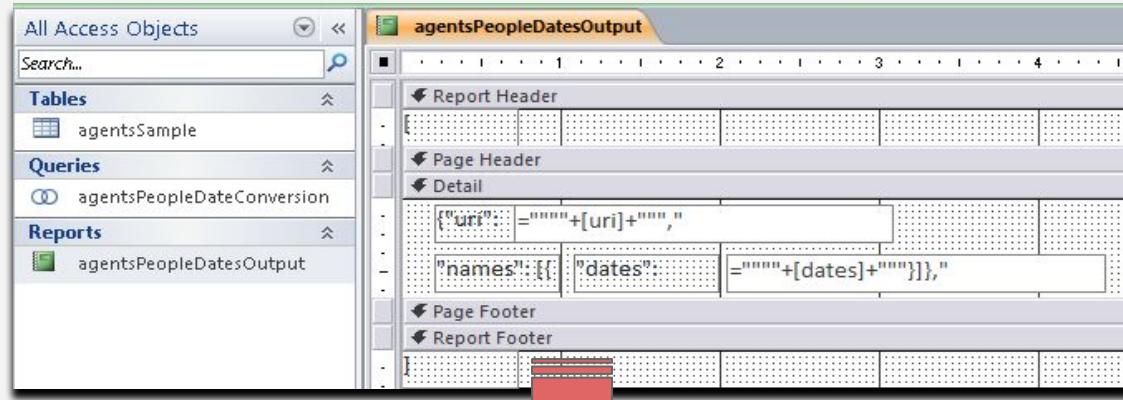
ein	strein	name	sub_name	city	state
731663130	73-1663130	ANIMAL ALLIANCE	ANIMAL ALLIANCE	WOODLAND HLS	CA
752461428	75-2461428	ANIMAL ANGELS	ANIMAL ANGELS	JACKSBORO	TX
232896507	23-2896507	ANIMAL APPEAL	ANIMAL APPEAL	HERMITAGE	PA
43654364	04-3654364	ANIMAL ASSISTANCE	ANIMAL ASSISTANCE	E BRUNSWICK	NJ
800099106	80-0099106	ANIMAL FAIR	ANIMAL FAIR	SEDALIA	MO

The above was again done with an online converter
(but remember that what you pass over the web is not secure).

Sidebar: Repurposing results

MS Access report

“JSON”



This is a breakdown of the endpoint we just used:

`http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal`

https://projects.propublica.org/nonprofits/api/v2	The address of the API. All requests must start with a similar string in order to point the request to where it needs to go. It essentially reads: go here.
/search.json	The search method set by the API. By appending this to the above URL, it essentially reads: go here, search, output that search in JSON.
?	Question marks denote that whatever follows is a parameter. Adding this to the above essentially reads: go here, and search, and use the following parameters. You can use more than one parameter.
q=	The parameter. The ProPublica API defines q as “A keyword search string. Searches using this parameter will search (in order) organization name, organization alternate name, city.”
animal	Any keyword supplied by the user. Go here, and use the keyword search parameter to search for <i>animal</i> , which will output as JSON.

This is a breakdown of the other endpoint we just used:

`http://projects.propublica.org/nonprofits/api/v2/organizations/731663130.json`

https://projects.propublica.org/nonprofits/api/v2	The address of the API. All requests must start with a similar string in order to point the request to where it needs to go. It essentially reads: go here.
/organizations	By appending this to the above URL, it essentially reads: go here; once you get there, go to this specific place
/:ein	The colon means that your input is required. This essentially reads: put EIN number here. Together, it reads: go here; once you get there, go to this specific place; and once you're there, find this exact record.
.json	Output the record as JSON



We just covered a whole lot.

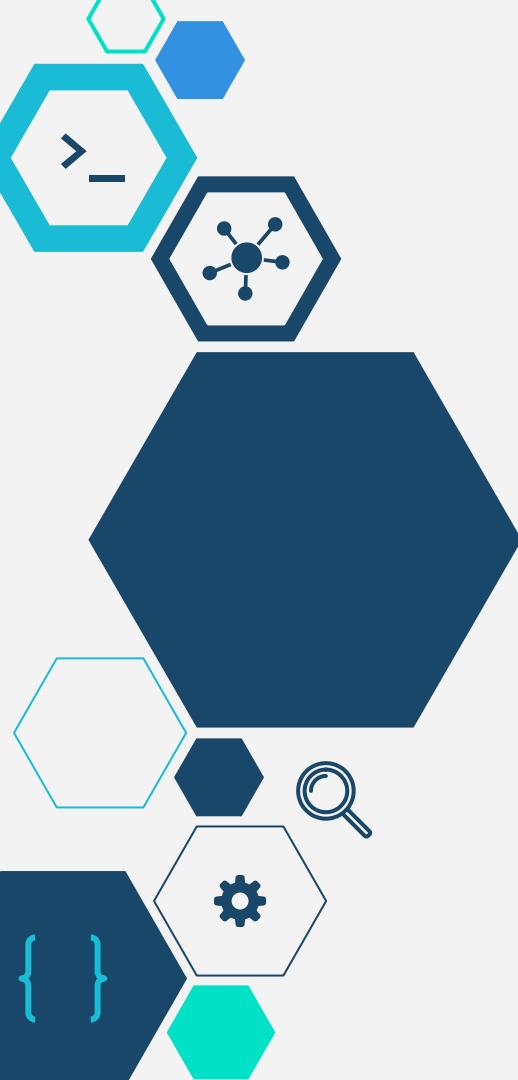
- We did a search on ProPublica.org, and then we looked at a specific result of that search.
- Then we used the GET command in a GUI interface to do the same search through an API, using a search endpoint.
- We got those results in a data standard called JSON, and they were exactly the same as the ones we got on the web.
- We then used the GET command to get an individual record by using its unique endpoint.
- Once we got the results, we explored ways of manipulating them to suit our skills or needs.



Questions

?





GET with Script: ProPublica



In our second exercise, we will be using

GET with a **Script**

(COMMAND | Execution)



Scripts

Interacting with APIs through a GUI is a great way to “get it.”

But it isn’t the only way.

We’re about to step into some unfamiliar territory for some.



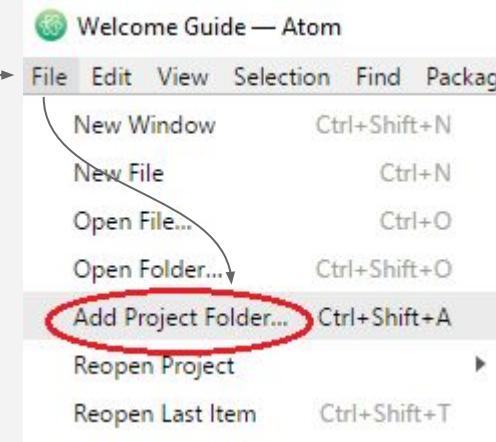
GET with Script- ProPublica

1. Open Atom
2. Add a project folder

3. Select the api-training folder

- Mac users: check your desktop
- Windows users: look in C:\Cygwin\home\[username]

3. You should now have a directory down the lefthand side, similar to this





GET with Script- ProPublica

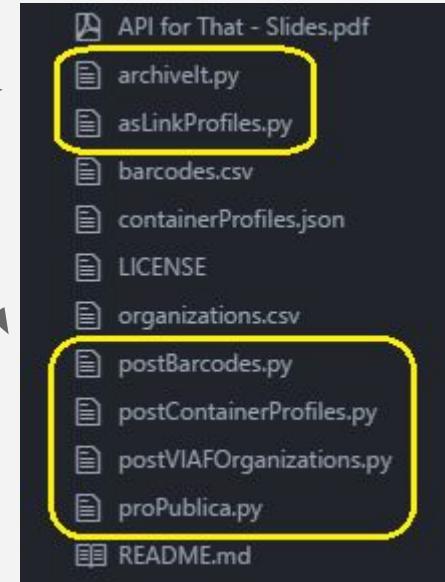
Take note of everything ending in .py

These are **python3** scripts, and we will be using them throughout the workshop.

We cannot teach you python3 today, but we can explore it together.

Open the first script of the day: **proPublica.py**

Locate the endpoint



```
7 raw_input('Press Enter to continue...')

8

9 endpoint = 'http://projects.propublica.org/nonprofits/api/v2/search.json?q=animal'
10

11 output = requests.get(endpoint).json()
```



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

Where are you and where do you want to go?

Mac

1. Open Terminal



PC

1. Open Cygwin





Command line bootcamp

Where are you?

Everyone type `pwd` and then hit enter.

Mac

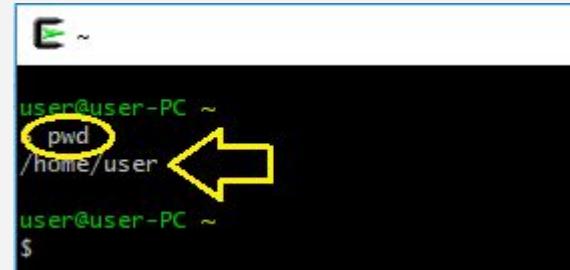
Mac users should see something like this

```
api-training — bash — 80x24
[Lyrasiss-MacBook-Pro-2:api-training woodford$ pwd
/Users/woodford/Documents/GitHub/api-training
Lyrasiss-MacBook-Pro-2:api-training woodford$
```

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.

PC

PC users should see something like this



```
user@user-PC ~
pwd
/home/user
user@user-PC ~
$
```



Command line bootcamp

Where are you?

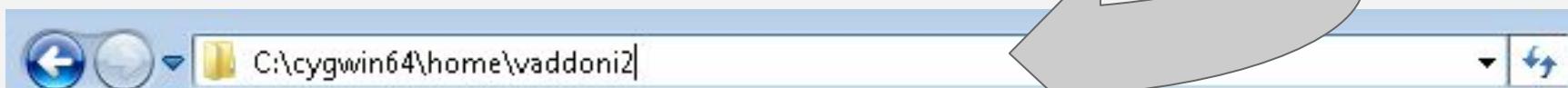
PC

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:

```
user@user-PC ~
: pwd
/home/user
user@user-PC ~
$
```

Is this location:





Unix commands for Mac and Cygwin

Where am I?

“print working directory”

`pwd`



Command line bootcamp

What is here?

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

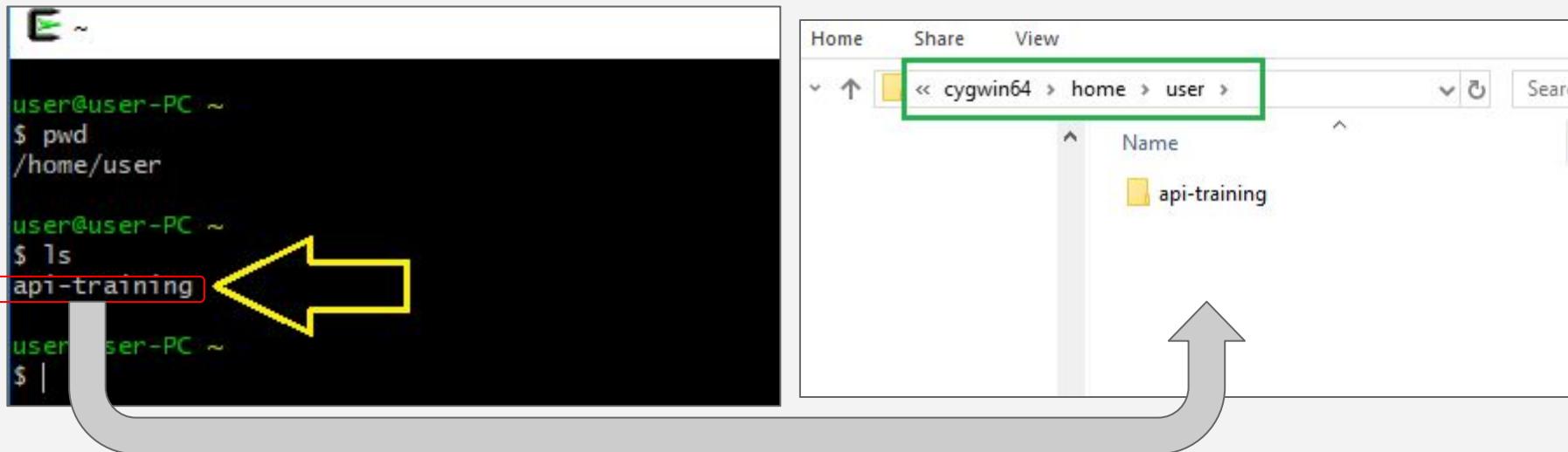
```
user@user-PC ~  
$ pwd  
/home/user  
  
user@user-PC ~  
$ ls  
api-training ←  
user@user-PC ~  
$ |
```



Command line bootcamp

What is here?

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 Windows 7)



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> and <code>ls -a</code>



Command line bootcamp

Move around

Now you're going to move from where you are into the api-training clone folder:

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

```
user@user-PC ~
$ pwd
/home/user

user@user-PC ~
$ ls
api-training

user@user-PC ~
$ cd api-training
$ cd api-training ← yellow arrow
user@user-PC ~/api-training ← yellow arrow
$
```



Command line bootcamp

Move around

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]\api-training
- And the same path in Cygwin looks like the new prompt, below:

A screenshot of a terminal window with a black background and white text. The text shows a yellow prompt: "user@user-PC ~/api-training \$ |". A cursor is visible at the end of the prompt line.



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> and <code>ls -a</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)



Command line bootcamp

What is really here?

Everyone type `ls` (that is L as in List)

Everyone type `ls -a` and then hit enter

The difference
between “list”
and “list all”

```
vaddoni2@MSEL-SPC32 ~/api-training
$ ls
'additional resources'    'Member Forum API - Slides.pdf'   Pre-Workshop_Windows.pdf
archiveIt.py               organizations.csv                 proPublica.py
asLinkProfiles.py          postBarcodes.py                README.md
barcodes.csv               postContainerProfiles.py     recordCenterProfile.json
containerProfiles.json     postDos.py                   runtime.py
dos.csv                   postVIAFOrganizations.py  updatePersonnames.py
LICENSE                   Pre-Workshop_Mac.pdf       viafReconciliationCorporate.py

vaddoni2@MSEL-SPC32 ~/api-training
$ ls -a
.
..
.archivessnake.yml
.git
.gitignore
'additional resources'
archiveIt.py               asLinkProfiles.py           postBarcodes.py       README.md
                           barcodes.csv            postContainerProfiles.py recordCenterProfile.json
                           containerProfiles.json  postDos.py           runtime.py
                           dos.csv              postVIAFOrganizations.py updatePersonnames.py
                           LICENSE              Pre-Workshop_Mac.pdf   viafReconciliationCorporate.py

vaddoni2@MSEL-SPC32 ~/api-training
$ |
```



Command line bootcamp

Go up one level

`cd ..` = “go up one”

You're going to move from the api-training clone folder back to the Cygwin home directory/Mac desktop

```
user@user-PC ~
$ pwd
/home/user

user@user-PC ~
$ ls
api-training

user@user-PC ~
$ cd api-training

user@user-PC ~/api-training  You were here
$ cd ..

user@user-PC ~
$ | Now you're back to where you started
```



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> and <code>ls -a</code>
How do I move from here to there?	“change directory”	<code>cd</code> [type the name of the directory] <small>(don't type this, we mean an action)</small>
Move up one level		<code>cd ..</code>



Command line bootcamp

Repeat commands

Lastly, let's go back into the api-training 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 api-training 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

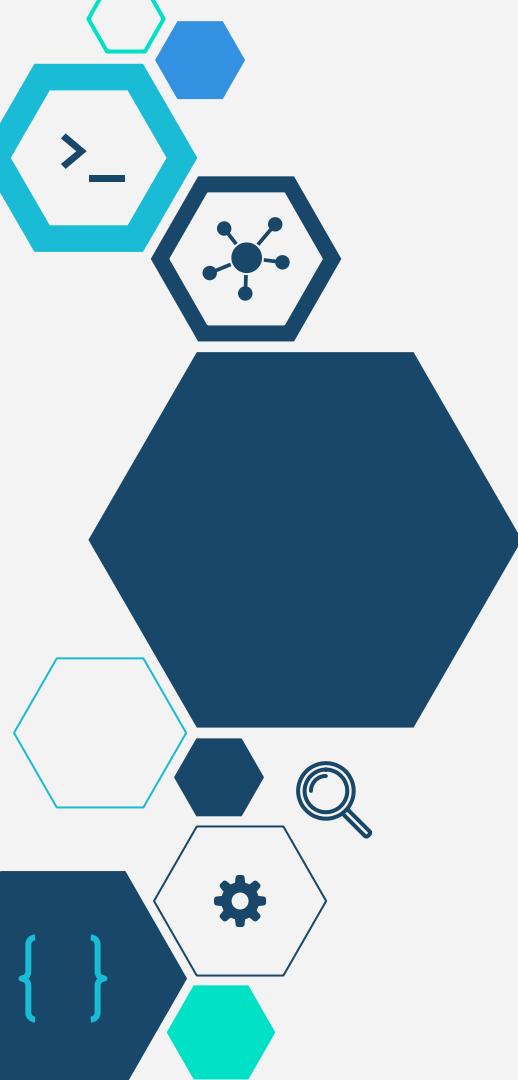
Where am I?	“print working directory”	<code>pwd</code>
What is here?	“list” (remember L as in List)	<code>ls</code> and <code>ls -a</code>
How do I move from here to there?	“change directory”	<code>cd</code> [type the name of the directory] ↑ (don't type this, we mean an action)
Move up one level		<code>cd ..</code>
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/Powershell; those are MS-DOS commands. This is why Mac users are smug.

The background of the image is a dramatic sky at either sunrise or sunset. The upper portion of the sky is a deep, saturated blue. Below it, a layer of clouds is bathed in warm, golden-orange light, transitioning into a lighter, more ethereal color towards the horizon. The overall effect is one of tranquility and natural beauty.

Breathe



And now back to...

GET with Script: ProPublica



GET with Script- ProPublica

Mac

1. In the Finder navigate to your api-training directory
2. Ctrl+click the api-training directory, and select “New Terminal at Folder”
3. Type `ls` and examine the contents of that folder
4. Type `python3 proPublica.py` and hit enter
5. Read the prompt
6. Next slide...

PC

1. Open Cygwin
2. Type `cd api-training` to enter the api-training directory
3. Type `ls` and examine the contents of that folder
4. Type `python3 proPublica.py` and hit enter
5. Read the prompt
6. Next slide...



GET with Script- ProPublica

Mac

6. A new file called “proPublicaRecord.json” should now be in your api-training directory
7. Go back to Atom and open “proPublicaRecord.json”

Directory reminder: Desktop/api-training

8. *Click Packages > Atom Beautify > Beautify*
9. Take a look!

PC

6. A new file called “proPublicaRecord.json” should now be in your api-training directory
7. Go back to Atom and open “proPublicaRecord.json”

Directory reminder:
C:\cygwin\home\[username]\api-training

8. *Click Packages > Atom Beautify > Beautify*
9. Take a look!



Questions

?



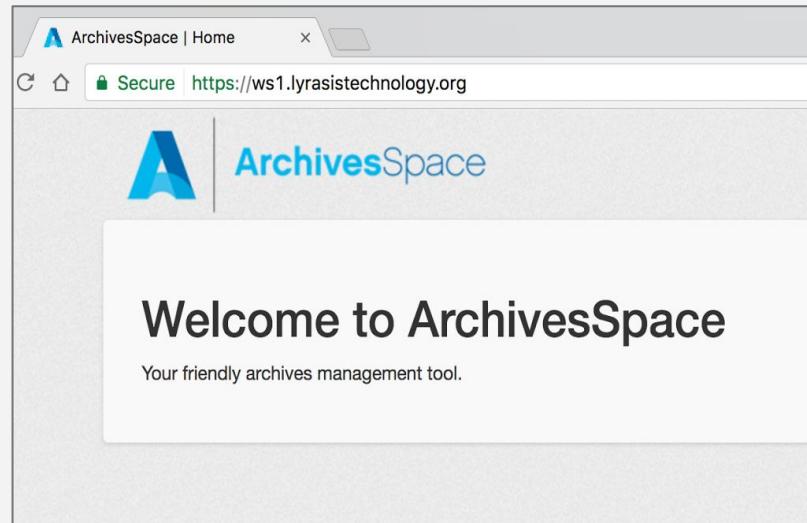
The ASpace API

Finally!



ArchivesSpace!

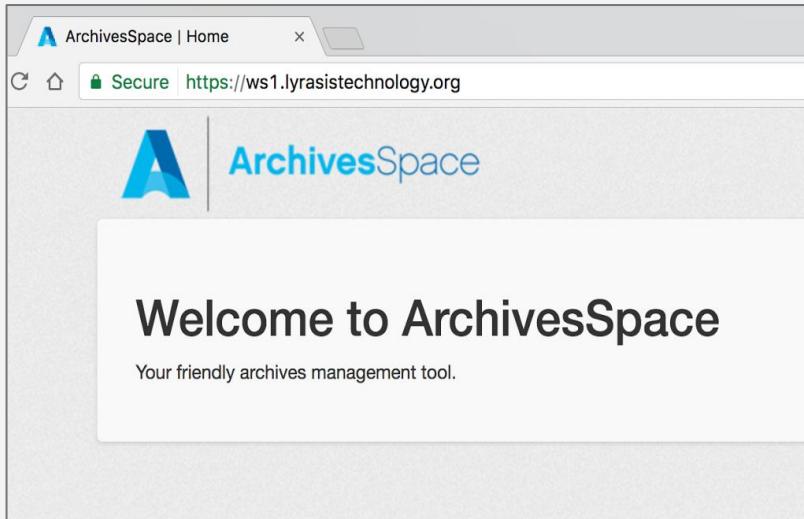
- You've each been provided a test instance of ArchivesSpace by our gracious hosts at Lyrasis
- The address of the **staff interface** of your instance is:
[https://ws\[your#\].lyrtech.org/staff/](https://ws[your#].lyrtech.org/staff/)
(ex. <https://ws0.lyrtech.org/staff/>)
- The address of the **API** of your instance is:
[https://ws\[your#\].lyrtech.org/staff/api](https://ws[your#].lyrtech.org/staff/api)
- Go check out the **staff interface** now!
username: admin
password: admin
- You already know your number!

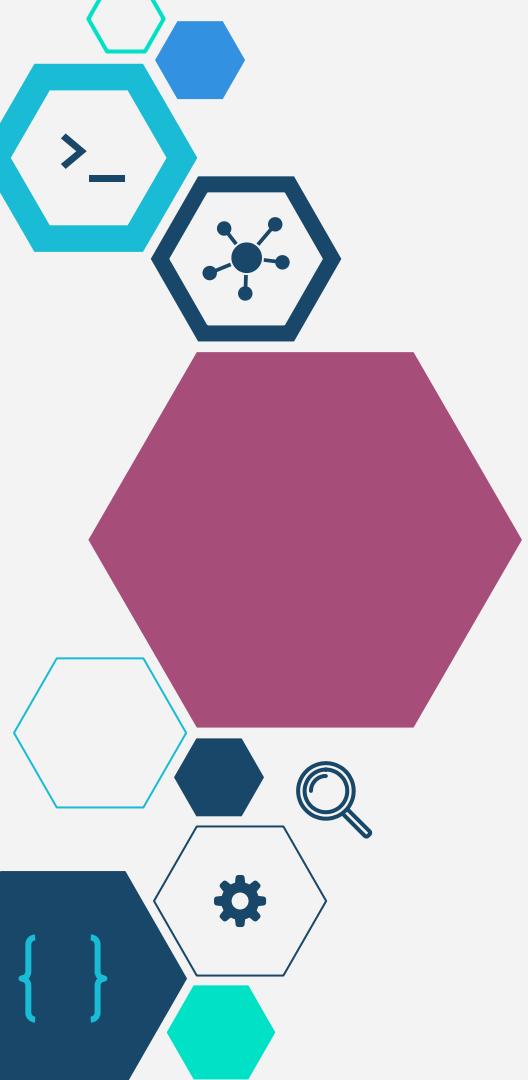




ArchivesSpace!

- We will be in ASpace for the rest of the day
- We will be starting with three simple actions, then stepping up in complexity.
- These scenarios are specific and may not help you; think about your own work as a series of steps, not circumstances
- Ask us questions!





Authenticate and GET

ASpace with GUI



Authenticate to AS with GUI

Before we do anything, we need to authenticate (just like you do when logging in) so let's do that in 6 steps:

The screenshot shows the Postman interface with the following configuration:

- Method: POST (Step 1)
- Endpoint: Enter endpoint here (Step 2)
- Body type: form-data (Step 3)
- Key: password (Step 4)
- Value: admin (Step 5)
- Send button (Step 6)

Type **password** under Key ; type **admin** under Value

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



Authenticate to AS with GUI

We just POSTed our username and password. In return, AS acknowledges that we have permission to proceed and provides a “session key.” We need to copy that.

Body Cookies Headers (9) Tests

Status: 200 OK Time: 396 ms Size: 2.9 KB

Pretty Raw Preview JSON ↻

```
1 {  
2   "session": "3fbf4887b333cdfc68bcab2b0306a8e0fcfc152fdc3342ee15a4ca4616aaa94e6",  
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  }  
19}
```

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



Authenticate to AS with GUI

Now we're going to take our authentication info and put it where it should go, in **3 steps**:

The screenshot shows the Postman interface for making a POST request. The method is set to 'POST', and the URL field contains 'Enter request URL'. The 'Params' button is visible. Below the method, there's an 'Authorization' section with a 'Headers (1)' tab selected. A green box highlights this tab, and a green arrow labeled '1.' points from it to the 'Headers' tab. The table below lists one header entry:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> X-ArchivesSpace-Session	f7772be9e02eb7eb83a9a8bf9ae1ab8bb4b1cad67ee11c13d...			
Key	Value	Description		

Type **X-ArchivesSpace-Session** under Key ; paste from the clipboard under Value



GET from AS with GUI

With our authentication info in place, let's GET a resource record in **3 steps**.

- 1.
- 2.
- 3.

The screenshot shows the Postman interface for making a GET request. Step 1 highlights the 'GET' method dropdown. Step 2 highlights the 'Endpoint' input field where the URL is being typed. Step 3 highlights the 'Send' button. Below the main form, the 'Headers (1)' tab is selected, showing an authorization header 'X-ArchivesSpace-Session' with the value '600db5a3f8ec1ed8d60508e78bcf515afa5e29f25750c8cc88ef...'. There is also a 'Body' tab and a 'Tests' section.

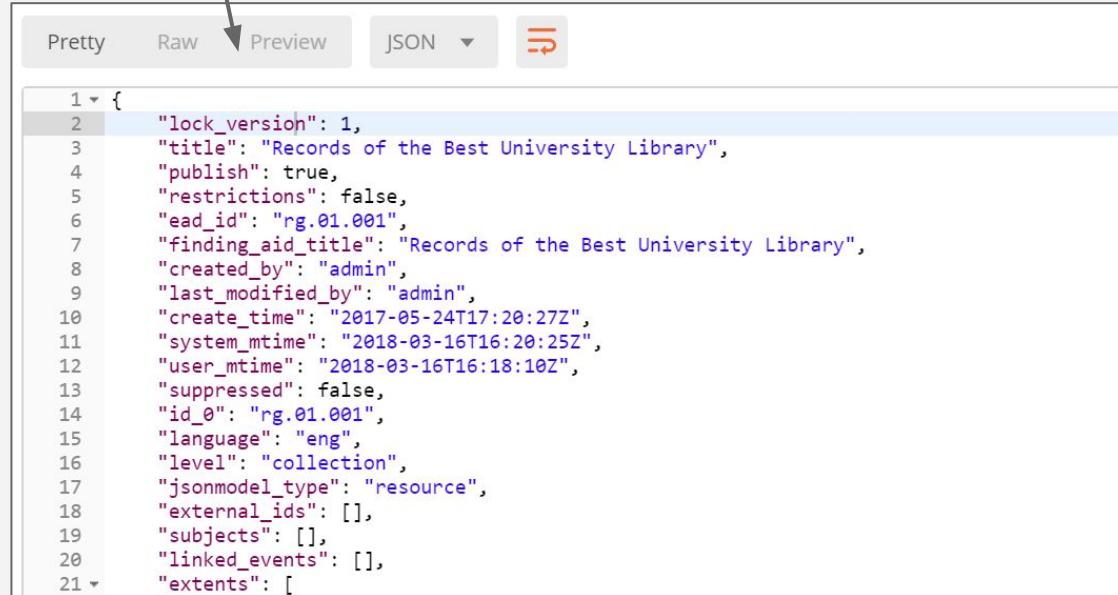
KEY	VALUE	DESCRIPTION	...	Bulk Edit
X-ArchivesSpace-Session	600db5a3f8ec1ed8d60508e78bcf515afa5e29f25750c8cc88ef...			
Key	Value	Description		

Endpoint: **[https://ws\[your#\].lyrtech.org/staff/api/repositories/2/resources/2](https://ws[your#].lyrtech.org/staff/api/repositories/2/resources/2)**



GET from AS with GUI

Did you get a result? 



The screenshot shows a JSON viewer interface with the following interface elements:

- Top navigation: Pretty, Raw, Preview, JSON (selected), and a refresh icon.
- Document content: A JSON object with 21 numbered lines of data.

Line numbers 1 through 21 are listed on the left, and the corresponding JSON key-value pairs are listed to the right. The data includes fields such as lock_version, title, publish, restrictions, ead_id, finding_aid_title, created_by, last_modified_by, create_time, system_mtime, user_mtime, suppressed, id_0, language, level, jsonmodel_type, external_ids, subjects, linked_events, and extents.

```
1 ▾ {  
2   "lock_version": 1,  
3   "title": "Records of the Best University Library",  
4   "publish": true,  
5   "restrictions": false,  
6   "ead_id": "rg.01.001",  
7   "finding_aid_title": "Records of the Best University Library",  
8   "created_by": "admin",  
9   "last_modified_by": "admin",  
10  "create_time": "2017-05-24T17:20:27Z",  
11  "system_mtime": "2018-03-16T16:20:25Z",  
12  "user_mtime": "2018-03-16T16:18:10Z",  
13  "suppressed": false,  
14  "id_0": "rg.01.001",  
15  "language": "eng",  
16  "level": "collection",  
17  "jsonmodel_type": "resource",  
18  "external_ids": [],  
19  "subjects": [],  
20  "linked_events": [],  
21  "extents": [  
  ]  
}
```



GET from AS with GUI

The screenshot shows a web browser window for ArchivesSpace. The URL in the address bar is highlighted with a green oval: `https://ws40.lyrtech.org/staff/resources/2#tree::resource_2`. The main content area displays a list of records under the heading "Records of the Best University Library". On the left, there is a sidebar with navigation links: Basic Information, Dates, Extents, Finding Aid Data, and Assessments. The main content area has tabs at the top: Edit, Add Event, Publish All, View Published, Export, Merge, More, Suppress, and Delete. The "Resource" tab is selected. Below it, the title "Records of the Best University Library" is displayed. A red box highlights the "Resource" tab. The detailed view for the first record shows the following fields:

Title	Records of the Best University Library
Identifier	rg.01.001
Level of Description	Collection
Language	English

A red text overlay "It's the same record!!!" is overlaid on the detailed view area.

Instructor cue: Auth and GET if you haven't already

POST to ASpace



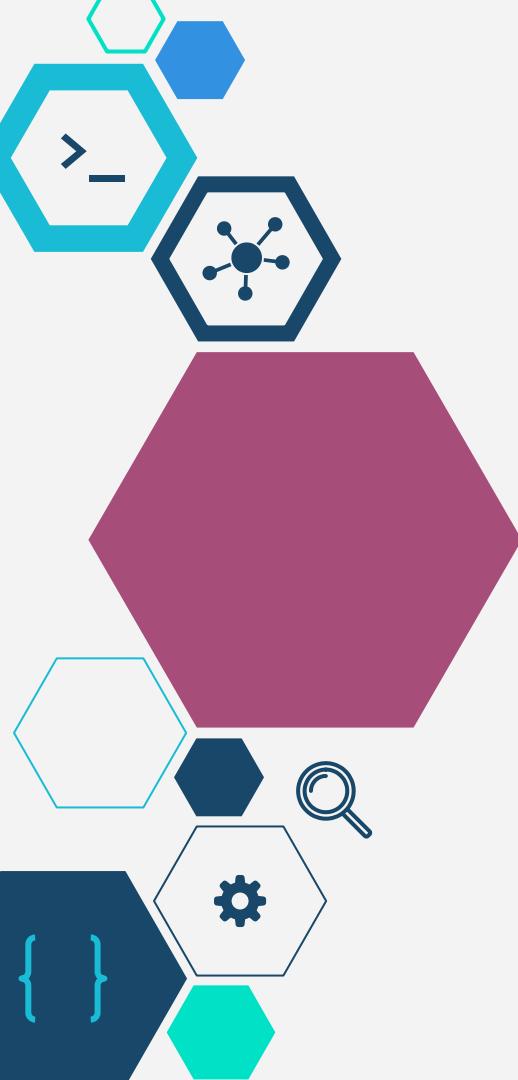
POST to AS

Scenario: Back when JHU migrated to 1.5.4, we found ourselves with the following problems. They are simple but good introductions to project work with the API.

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

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



POST to ASpace with GUI

Container profiles

Creating new records



Container profiles

What's a container profile?

A simple record consisting of dimensions.
Height x Width x Depth.

They are an optional feature of the container management module.

They are very SIMPLE, which is why we're using them.





POST to AS with GUI

Container profiles → Creating new records

The JSON for a profile is simple and easy to read:

1. Open Atom
2. Open “recordCenterProfile.json” from the directory with our cloned GitHub repo
3. Packages > Atom Beautify > Beautify
4. Here is the container profile for a record center carton in JSON, ready to go
5. Copy the whole thing (including brackets), and go back to Postman

```
recordCenterProfile.json
{
  "name": "Record center box",
  "extent_dimension": "width",
  "height": "10.5",
  "width": "13.1",
  "depth": "15.75",
  "dimension_units": "inches",
  "jsonmodel_type": "container_profile"
}
```



POST to AS with GUI

Container profiles → Creating new records

1. POST

2. Enter endpoint here

3. Body

4. JSON (application/json)

5.

6. Paste JSON, hit send

Endpoint:
[https://ws\[your#\].lyrtech.org/staff/api/container_profiles](https://ws[your#].lyrtech.org/staff/api/container_profiles)

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 → Creating new records

```
Pretty Raw Preview JSON ↻ Save Response  
1  {  
2  "status": "Created",  
3  "id": 1,  
4  "lock_version": 0,  
5  "stale": null,  
6  "uri": "/container_profiles/1",  
7  "warnings": []  
8 }
```

Success!!



POST to AS with GUI

Container profiles → Creating new records

Let's do more!

1. Go back to Atom and open “containerProfiles.json”
2. Packages > Atom Beautify > Beautify
3. Here are *all* the profiles, ready to go
4. Copy everything, 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 → Creating new records

1. POST

2. Enter endpoint here

3. Body

4. JSON (application/json)

5.

6. Paste JSON, hit send

Endpoint:
[https://ws\[your#\].lyrtech.org/staff/api/container_profiles](https://ws[your#].lyrtech.org/staff/api/container_profiles)

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 → Creating new records

POST localhost:8089/container_profiles

Authorization Headers (2) Body Pre-request Script Tests

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

```
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
```

Body Cookies Headers (0) Tests

Pretty Raw Preview JSON

```
1 [ { "name": "Postcard box", "extents_dimension": "width", "height": "5.5", "width": "3.5", "depth": "", "dimension_units": "inches", "jsonmodel_type": "container_profile" }, { "name": "Postcard cover box", "extents_dimension": "width", "height": "10.5", "width": "13.1", "depth": "15.75", "dimension_units": "inches", "jsonmodel_type": "container_profile" } ]
```

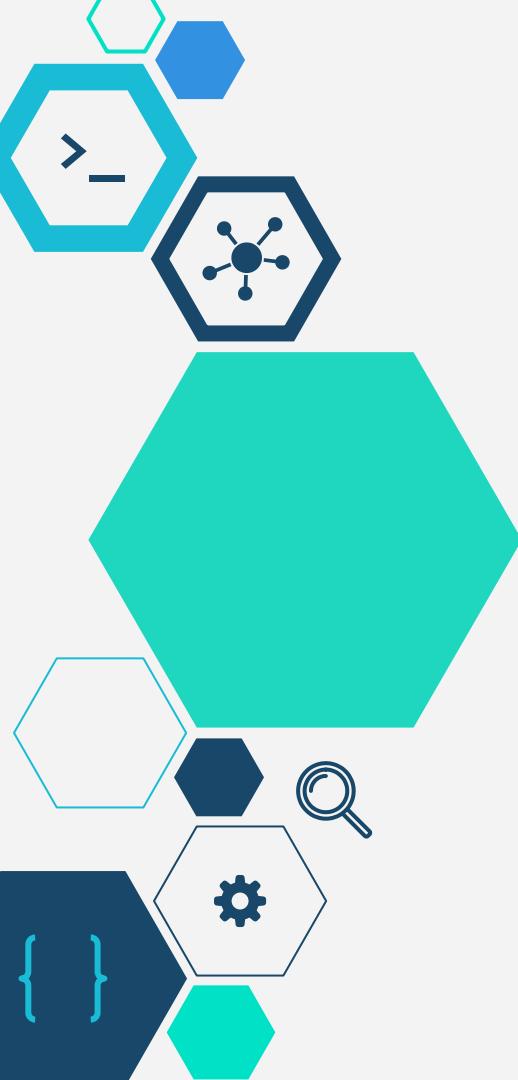
```
1 [ { "error": "can't convert String into Integer" } ]
```

Don't hate us: you cannot post multiple records through the GUI.
This frustrating exercise might save you a month.

(use your month wisely: take a vacation from computers)

The background of the slide is a photograph of a sky at dusk or dawn. The clouds are scattered across the frame, with colors ranging from deep blue and purple to bright orange and yellow, suggesting the light of the sun just above or below the horizon.

Breathe
(hydrate)

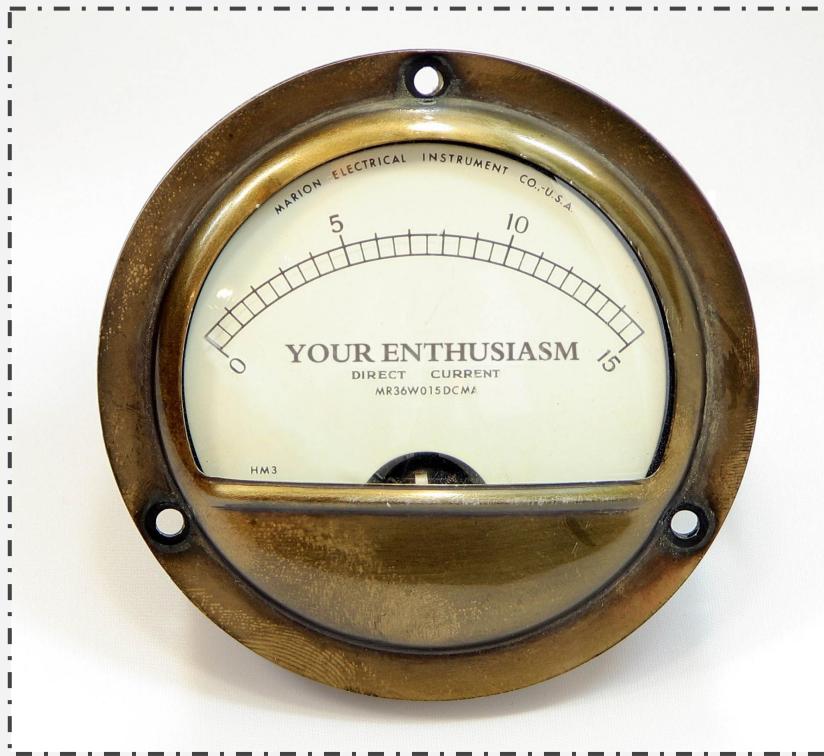


Scripting

>_



Scripting

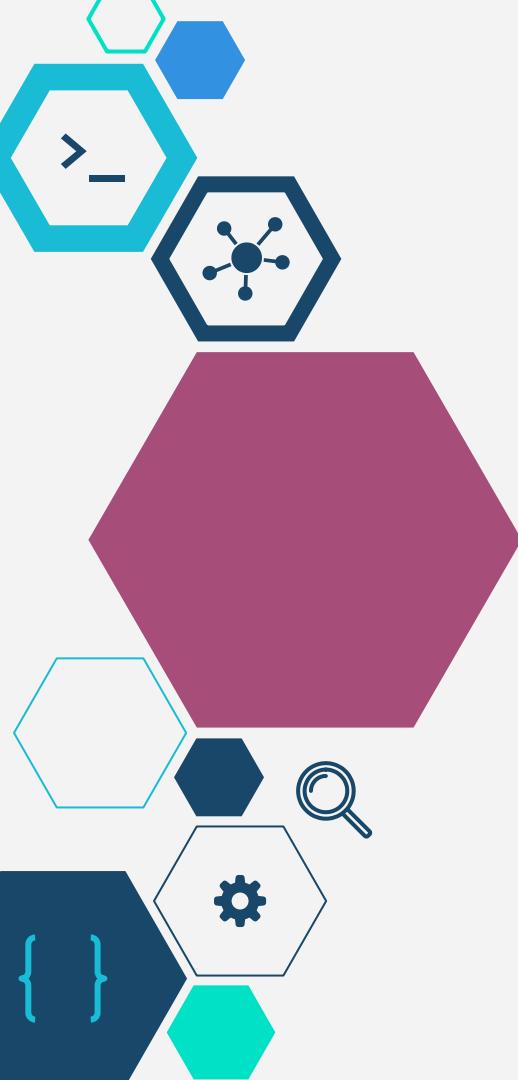




Scripting - Why?

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:

- 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!**



POST to ASpace with Script

Container profiles

Creating new records



POST to AS with Script

Container profiles → Creating new records

- We're all connecting to different instances of ArchivesSpace (so we don't clash with each others' work!), so we need to tell **ArchivesSnake** where each of our individual instances live.
- Open **.archivessnake.yml** in Atom
- Edit the line starting with **baseurl** to include your instance number:
baseurl: 'https://ws[your#].lyrtech.org/staff/api'

```
.archivessnake.yml
5 # Replace [your#] with the number for your instance
6 baseurl: "https://ws[your#].lyrtech.org/staff/api"
7 username: "admin"
8 password: "admin"
```

- Go to **File > Save As** and save **.archivessnake.yml** in your **home directory**

OSX

/Users/[username]

Windows

C:\cygwin\home\[username]



POST to AS with Script

Container profiles → Creating new records

Mac

1. In the Finder navigate to your api-training directory
2. Ctrl+click the api-training directory, and select “New Terminal at Folder”
3. Type `ls` and examine the contents of that folder
4. Type `python3 postContainerProfiles.py` (case sensitive!) and hit enter
5. Navigate back to AS in your browser ([https://ws\[your#\].lyrtech.org/staff/container_profiles](https://ws[your#].lyrtech.org/staff/container_profiles))

PC

1. Open Cygwin
2. Type `cd api-training` to enter the api-training directory
3. Type `ls` and examine the contents of that folder
4. Type `python3 postContainerProfiles.py` (case sensitive!) and hit enter
5. Navigate back to AS in your browser ([https://ws\[your#\].lyrtech.org/staff/container_profiles](https://ws[your#].lyrtech.org/staff/container_profiles))



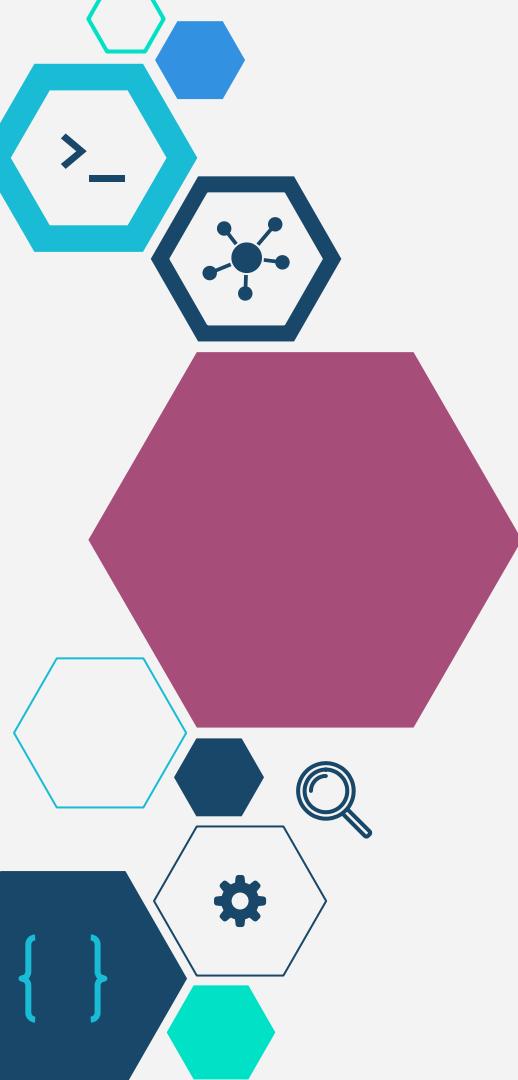
POST to AS with Script

Container profiles → Creating new records

Scenario: Back when JHU migrated to 1.5.4, we found ourselves with the following problems. They are simple but good introductions to project work with the API.

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

Pro-tip: Remember that this container profile scenario represented “create” in AS; if you have records you need to create, this is how to start thinking about them.



POST to ASpace with Script

Edit barcodes

Modify existing records



Barcodes

Every container in your collections 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”

AS assigns its own unique number to every top_container, but you might also want to use YOUR unique number.

At JHU, we wanted to use our actual barcodes.

Container 1: [31151030080422]		Top Container
Container Profile	Record center box	<input checked="" type="checkbox"/>
Indicator	1	
Barcode	31151030080422	
Exported to ILS	Not exported	
Legacy Restricted?	False	



POST to AS with Script

Edit barcodes → Modify existing records

1. Navigate to the Gérard Defaux papers
([https://ws\[your#\].lyrtech.org/staff/resources/1](https://ws[your#].lyrtech.org/staff/resources/1))
2. Expand Research Materials > click on any file > scroll down to Instances > see fake barcodes

[aspace.60240.box.1]
[aspace.60240.box.1]
[aspace.60240.box.1]
[aspace.60240.box.1]
[aspace.60240.box.1]
[aspace.60240.box.1]

These are the barcodes generated by the barcoder plugin. Hopkins has thousands of them.

3. Navigate to the [GitHub](#) and look at barcodes.csv



POST to AS with Script

Edit barcodes → Modify existing records

Mac

1. Type `ls` and examine the contents of that folder
2. Type `python3 postBarcodes.py` (case sensitive!) and hit enter
3. Navigate back to AS in your browser
([https://ws\[your#\].lyrtech.org/staff/resources/1](https://ws[your#].lyrtech.org/staff/resources/1))

PC

1. Type `ls` and examine the contents of that folder
2. Type `python3 postBarcodes.py` (case sensitive!) and hit enter
3. Navigate back to AS in your browser
([https://ws\[your#\].lyrtech.org/staff/resources/1](https://ws[your#].lyrtech.org/staff/resources/1))



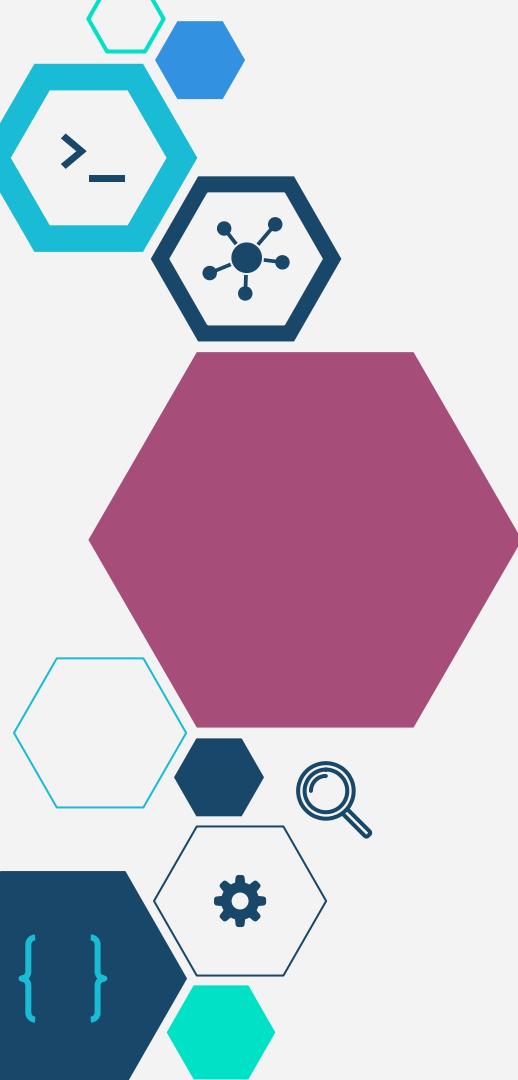
POST to AS with Script

Edit barcodes → Modify existing records

Scenario: Back when JHU migrated to 1.5.4, we found ourselves with the following problems. They are simple but good introductions to project work with the API.

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

Pro-tip: Remember that this scenario represented “edit record” in AS; if you have records you need to edit, this is how to start thinking about it.



POST to ASpace with Script

Link profiles

Link existing records



Linking profiles to containers

Link profiles —→ Link existing records



1. Type `ls`
2. Type `python3 asLinkProfiles.py` (case sensitive!)
3. The script should prompt you for something...



POST to AS with Script

Link profiles —> Link existing records

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 Resource view. The URL in the browser address bar is `localhost:8080/resources/1#tree::resource_1`, with the entire path highlighted by a large red circle. The page displays the title "ArchivesSpace" and a navigation menu with "Browse" and "Create" options. Below the menu, the breadcrumb trail shows "Home / Resources / Gérard Defaux papers". A tree view on the left shows "Gérard Defaux papers" expanded, with "Research Materials" listed under it.

View a container profile for its profile number:

A screenshot of the ArchivesSpace Container Profile view. The URL in the browser address bar is `localhost:8080/container_profiles/12`, with the entire path highlighted by a large red circle. The page displays the title "ArchivesSpace" and a navigation menu with "Browse" and "Create" options. Below the menu, the breadcrumb trail shows "Home / Container Profiles / Record center box". A red circle highlights the "Record center box" section on the right side of the screen, which contains tabs for "Basic Information" and "Edit".



POST to AS with Script

Link profiles —→ Link existing records

Scenario: Back when JHU migrated to 1.5.4, we found ourselves with the following problems. They are simple but good introductions to project work with the API.

1. ~~We didn't have container profiles in ArchivesSpace and wanted to, so we needed to create some~~
2. ~~In following the migration instructions for 1.5, we had to add faux codes to our containers; we'd like to use our actual barcodes~~
3. ~~Once we created container profiles, we needed to link them to actual top containers~~

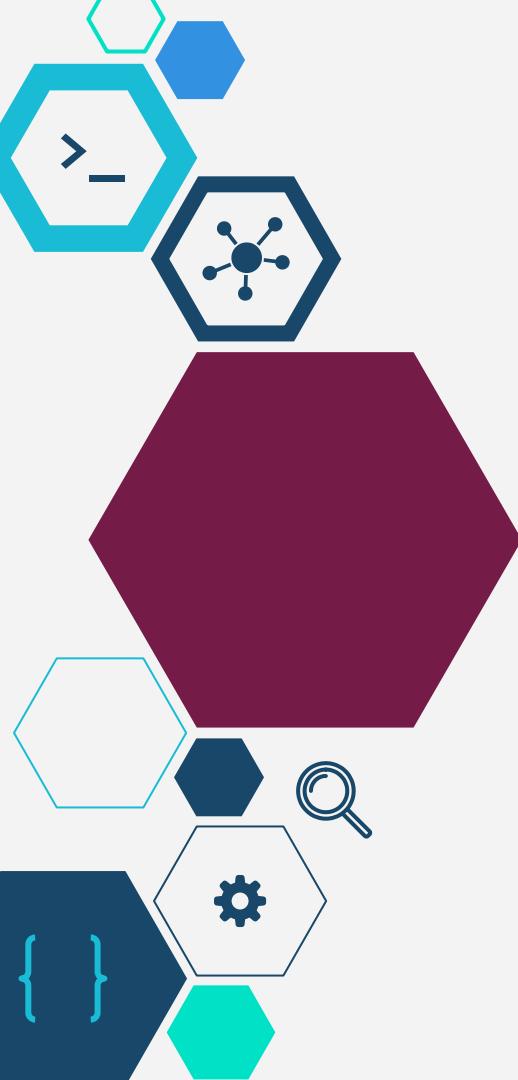
Pro-tip: Remember that this link scenario represented any “link” in AS; if you have records you need to link, this is how to start thinking about it.



Remember this

- You did it!
- Yes, we wrote the scripts.
- But now they're yours.

- If you wish to move forward, remember these. Examine them, and those we provide at the end of the workshop.



App-to-app communications

Automating (some) archival description



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 Best University Library” resource
2. Expand Subgroup 7: 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 **python3 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



ArchivesSpace



App-to-app communication

The screenshot shows the Archive-It search interface. At the top, there's a navigation bar with links for HOME, EXPLORE, LEARN MORE, and CONTACT US. To the right of the navigation is a logo for "The leading web archiving service for collecting and accessing cultural heritage on the web. Built at the Internet Archive". Below the navigation, there are two sections: "Narrow Your Results" and "Explore All Archives". The "Narrow Your Results" section includes filters for "Type of Collecting Organization" (set to "Colleges & Universities") and "Collecting Organization" (set to "Johns Hopkins University"). The "Explore All Archives" section has a search bar containing "library.jhu.edu", a "Search" button, and a "Clear" button. Below the search bar, it says "Collection Name : Johns Hopkins University web collection". It then lists the search results: "The following results were found for the term(s): library.jhu.edu" followed by a bulleted list: "• 11 Sites were found." and "• Additional results for library.jhu.edu may be found by searching within the page text." At the bottom, there are tabs for "Sites" and "Search Page Text", with "Sites" being the active tab. A summary box at the bottom right shows "Page 1 of 1 (11 Total Results)" and provides sorting options: "Sort By: Best Match | Title (A-Z) | Title (Z-A) | URL (A-Z) | URL (Z-A)". Below this, a green box highlights the URL "http://library.jhu.edu". Further down, it lists the collection and organization details: "Collection: Johns Hopkins University web collection", "Organization: Johns Hopkins University", and "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!

INTERNET ARCHIVE
WayBack Machine

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!

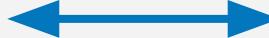
```
67     for crawl in crawlList:  
68         doid = 'https://wayback.archive-it.org' + '/' + archiveit_coll + '/' +  
69         crawl['timestamp'] + '/' + crawl['original']  
70         query = '/search?page=1&filter_term[]={"primary_type":"digital_object"}&q=' + doid  
71         existingdOID = 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 `python3 archiveIt.py` and let's see what happens!
2. Go check out that “Records of the Best University Library” resource record once again.



Questions

?

ArchivesSnake



Thanks!

Join us at api-alums.slack.com

Valerie Addonizio
vaddoniz@jhu.edu

Dave Mayo
dave_mayo@harvard.edu

Lora Woodford
Lora.Woodford@lyrasis.org