Alma Elsewhere: Using APIs to Sprinkle Data Around Your Organization

Chad Kluck, Web Developer
St. Thomas Libraries, User and Digital Services
ELUNA: May 3, 2019

clkluck@stthomas.edu
github.com/USTLibraries
github.com/chadkluck

Presentation audio w/ slides: https://chadkluck.net/eluna-2019

UNIVERSITY OF St.Thomas


Welcome

What's Your Experience?

- How many have used an API?
- How many have created an Ex Libris API key for integration with another system?
- How many have at least looked at the API documentation?
- By the end of today you'll have done, or be able to do, all three.

St. Eligius/St. Elsewhere Hospital, Boston, MA
Photo by mmelody from https://mapio.net/pic/p-25305660/

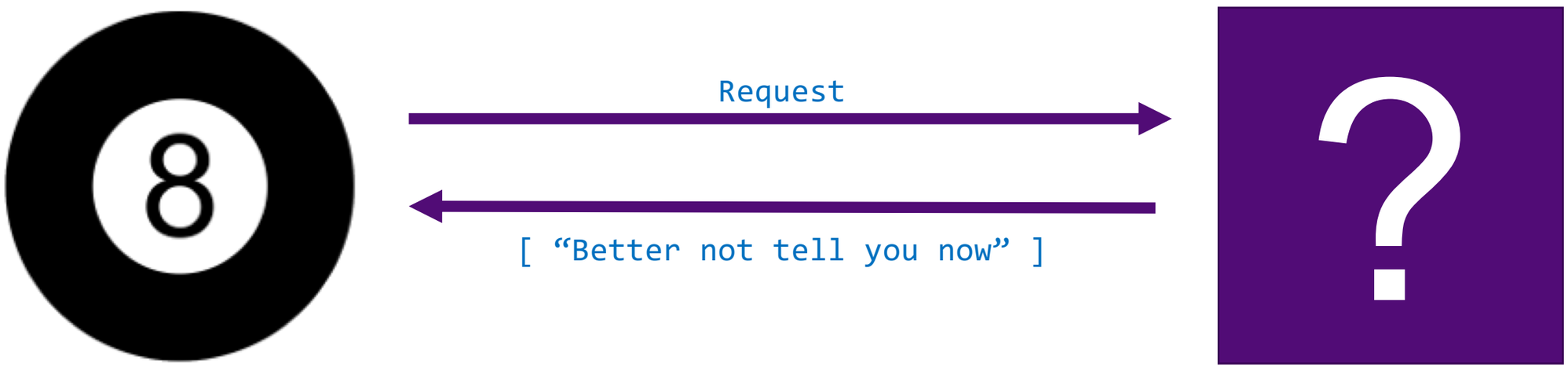
Agenda

- Get Familiar: What is an API?
- Ex Libris API Console: The Tour
- Postman: The Ease of Playing with APIs
- Security: (aka Officer Buzz Kill)
- A Framework & Other Goodies from GitHub

Get Familiar: What is an API?

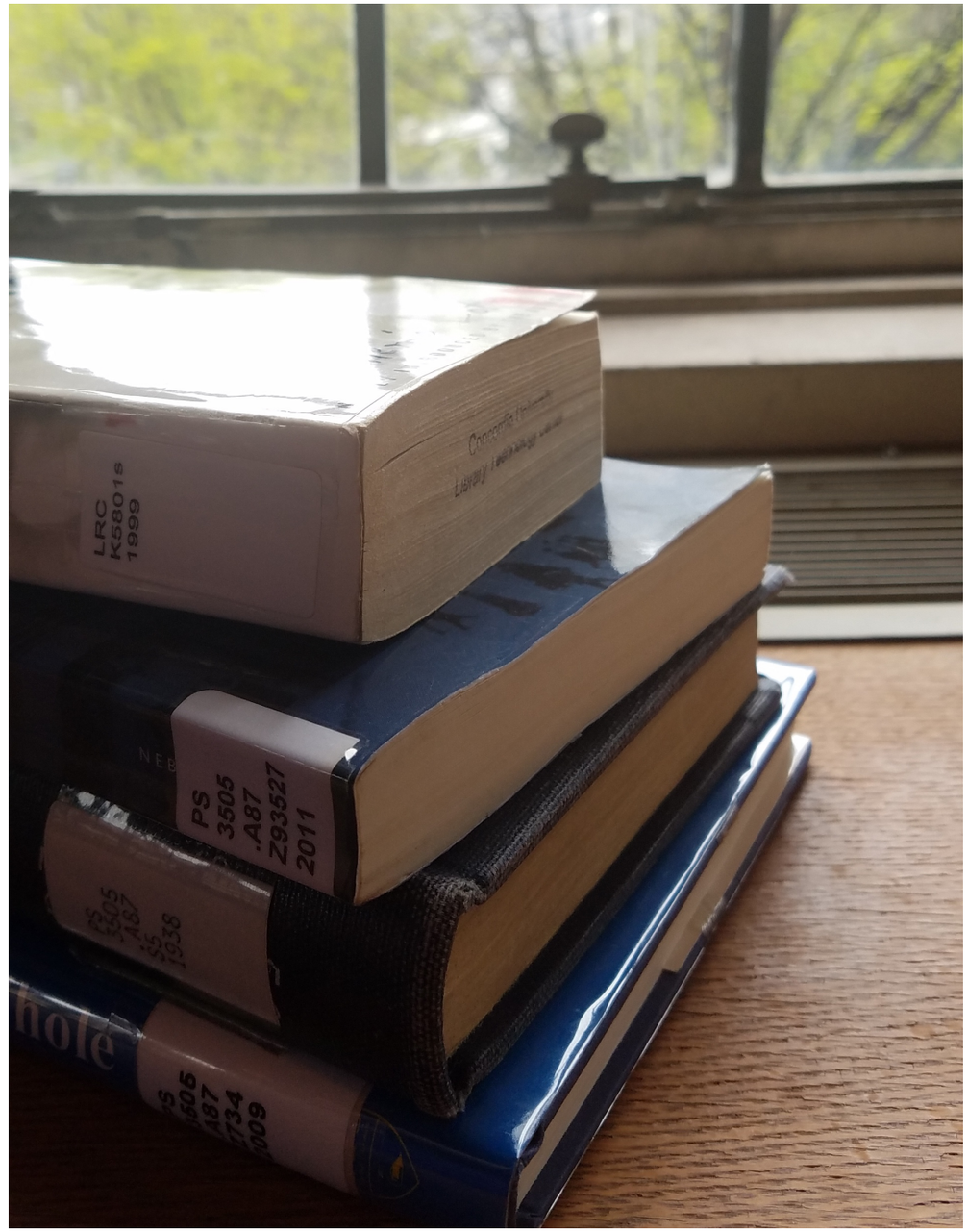
What is an API?

- Application Program Interface
- To put it simply: it's data that one application can request from another
- One application makes a REQUEST to a server
- Then the server gives back a RESPONSE



Enhance User Services

- You no longer need to rely on vendors to deliver a custom user interface or widget
- RSS feeds are still strong, but limited
- Bring the data to your patrons:
  - Student Portals
  - Course Management Systems
  - Chat and Virtual Assistants
- Streamline processes:
  - Invoices
  - Bulk Bib Edits



I read, therefore I code!

Can you read this and visualize what it does?

```
<h1>Hello World!</h1>
<p>This is my page!</p>
<img src=“images/underconstruction.gif” alt=”A highly animated and annoying GIF”>
<p><a href=“aboutme.html”>About me</a></p>
```

How many, the first time you saw HTML, said: “WTH? Maybe, if I stare at it long enough, just maybe I can make out what it is”?

Data Response Example (JSON)

```
["hello world"]
```

It is a single text string:

hello world

In fact, try it now in any browser by going to:
https://api.chadkluck.net/eluna

Data Response Example #2 (JSON)

What is this?

```
["banana", "apples", "oranges"]
```

It's a list:

Item 1: "banana"
Item 2: "apples"
Item 3: "oranges"

Or, if you think in arrays:

[0]: "banana"
[1]: "apples"
[2]: "oranges"

JSON: **J**ava**S**cript **O**bject **N**otation (but it can be used with PHP, Python, ASP…)

BAO

Data Response Example #2 (JSON)

What is this?

```
["banana", "apples", "oranges"]
```

A quick note: Examples and links from a slide:

`https://api.chadkluck.net/eluna?code={{3LetterCode}}`

The 3 letter code is shown here

`https://api.chadkluck.net/eluna?code=` BAO

Data Response Example #3 (JSON)

What about this?

```
[ {
   "name": "Charlie Brown",
   "id": "1005783",
   "fines": 38.93
},
{
   "name": "Linus van Pelt",
   "id": "1004378",
   "fines": 0,
   "loans": [{"title": "Bible (KJV)",
                "due": "20181226"}]
} ]
```

It's a multi-level array!

```
[0][name]: "Charlie Brown"
[0][id]: "1005783"
[0][fines]: 38.93

[1][name]: "Linus van Pelt"
[1][id]: "1004378"
[1][fines]: 0
[1][loans][0][title]: "Bible (KJV)"
[1][loans][0][due]: "Dec 26, 2018"
```

https://api.chadkluck.net/eluna?code=LVP

Wait, so you REQUEST a RESPONSE through a URL?

- Yep! No database connections
- It's the same as you would a web page, by using a URL
- Just as https://example.com/mypage might get you:

```
<html><head><title>Hello World!</title></head>
<body><h1>Hello World</h1></body></html>
```

- Requesting `https://api.chadkluck.net/eluna?code=8BL` will get you:

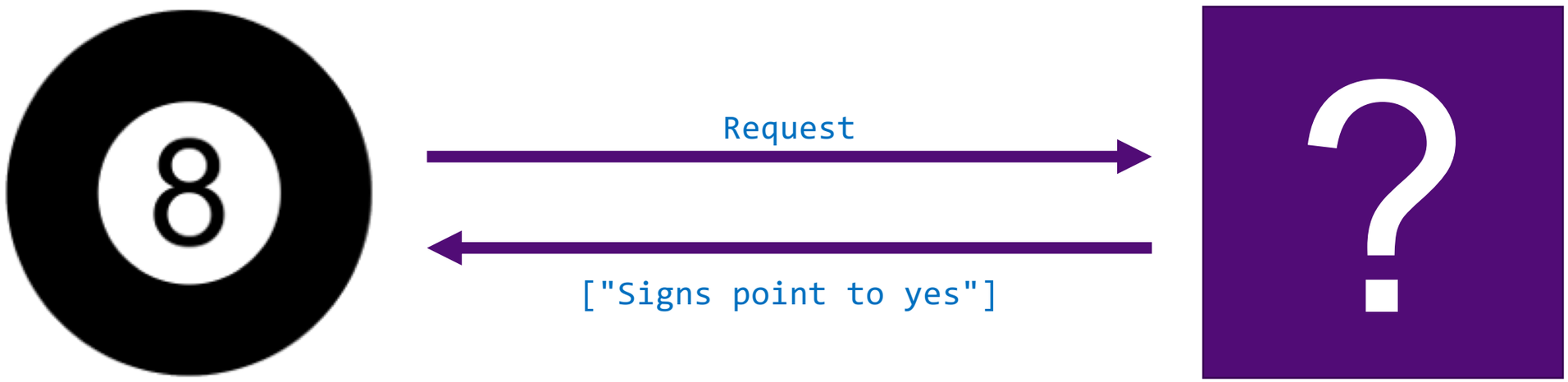
```
["Concentrate and ask again"]
```

(or some other random answer!)

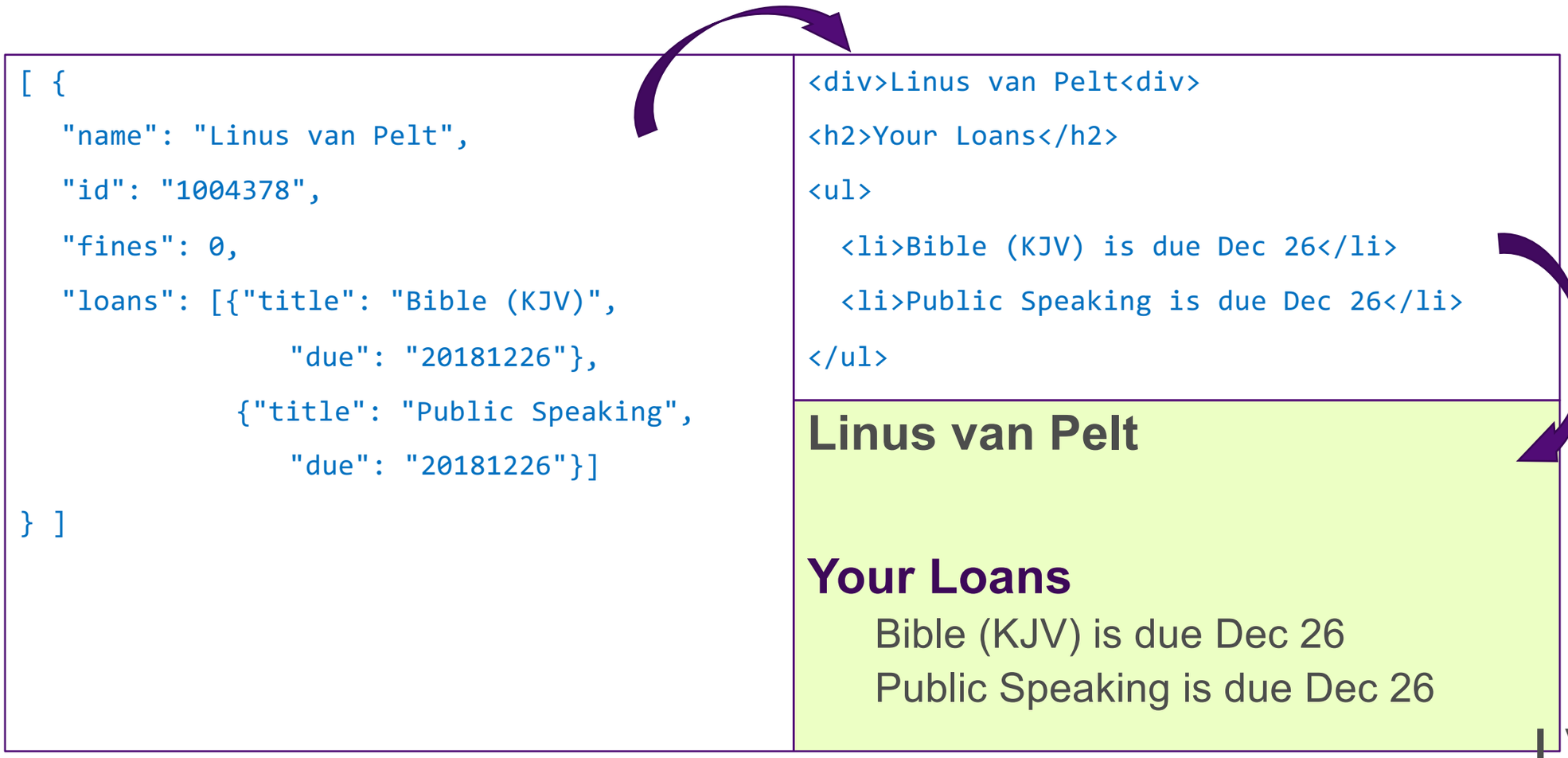
`https://api.chadkluck.net/eluna?code=` 8BL

It separates your UI from the Data

- Your application (8Ball) doesn't have to worry about random answers
- You just create an interface either Web (HTML), native app (iPhone/Android), or virtual assistant skill (ChatBot, Alexa, Siri, etc)



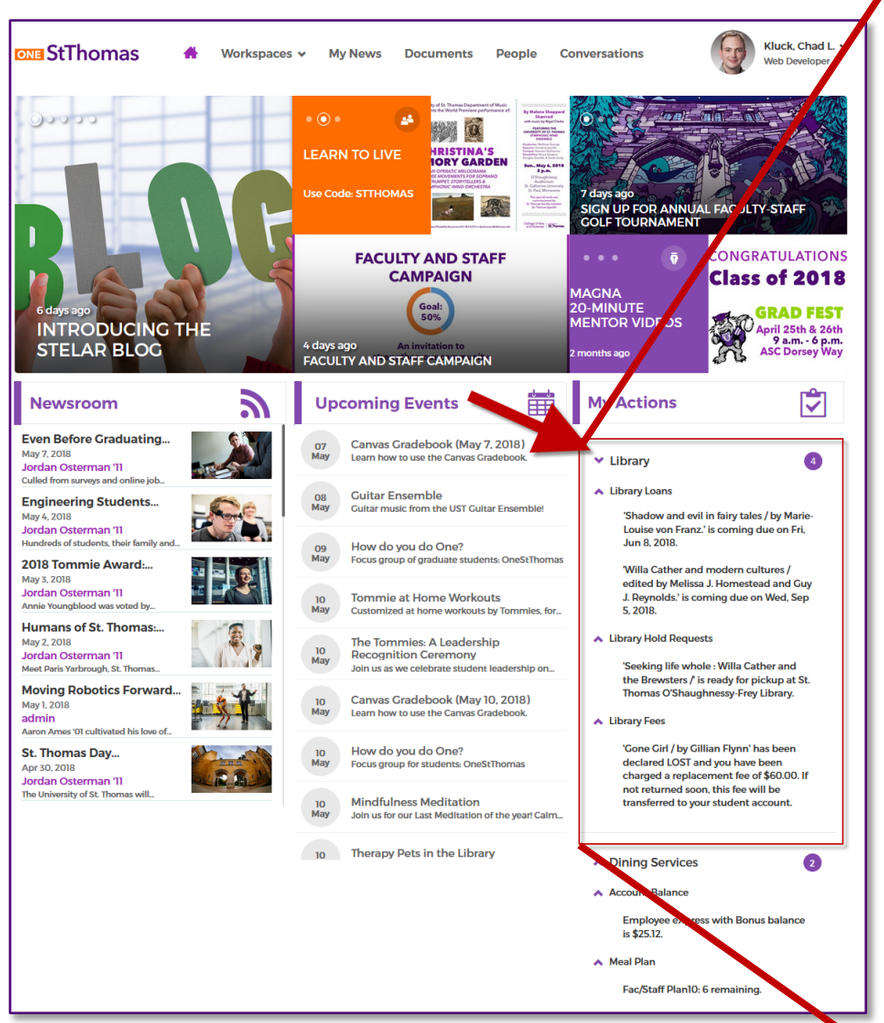
Okay, but what can I do with it?



LVP

Real Life Example: University Portal (One St. Thomas)

- When students and staff log in they see their library activity
  - Loaned items
  - Hold requests
  - Fees and fines



https://one.stthomas.edu

My Actions

Library 4

Library Loans

'Shadow and evil in fairy tales / by Marie-Louise von Franz.' is coming due on Fri, Jun 8, 2018.

'Willa Cather and modern cultures / edited by Melissa J. Homestead and Guy J. Reynolds.' is coming due on Wed, Sep 5, 2018.

Library Hold Requests

'Seeking life whole : Willa Cather and the Brewsters /' is ready for pickup at St. Thomas O'Shaughnessy-Frey Library.

Library Fees

'Gone Girl / by Gillian Flynn' has been declared LOST and you have been charged a replacement fee of $60.00. If not returned soon, this fee will be transferred to your student account.

Dining Services 2

Account Balance

Employee eXpress with Bonus balance is $25.12.

Meal Plan

Fac/Staff Plan10: 6 remaining.

Chad, it must have taken you weeks to code that!

- Um, no. I didn't code that (though I could have!)
- I just went to our IT people and said: "I have this great idea for the new portal!"
- They said, "Yeah? What?"
- I said, "What if we let users know when their book request was available, see what they had checked out and if they had any fines!"
- They bowed down to me and exclaimed, "That's a brilliant idea!"
- I don't even know, or care, what language they used
- I simply copied a sample API GET request

But, I did need to interpret the data!

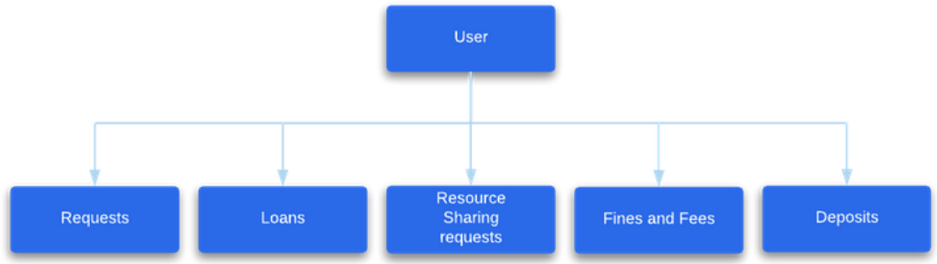
```
[ {
  "name": "Charlie Brown",
  "id": "1005783",
  "fines": 38.93
},
{
  "name": "Linus van Pelt",
  "id": "1004378",
  "fines": 0,
  "loans": [{"title": "Bible (KJV)",
             "due": "20181226"}]
} ]
```

- I told them in what fields to find the user data
- I told them what fields from the book request to display
- And I gave them the API request URL
- How did I know all this? I got it from the…

Ex Libris API Console: The Tour

Documentation

- You need to know what fields are available and how to make requests
- Ex Libris has extensive documentation on the Developers site



Alma provides a set of Web services for handling user information, enabling you to quickly and easily manipulate user details. These Web services can be used by external systems—such as student information systems (SIS)—to retrieve or update user data.

Users

API	Path
Retrieve users	`GET /almaws/v1/users`
Create user	`POST /almaws/v1/users`
Delete user	`DELETE /almaws/v1/users/{user_id}`

For example, Users: https://developers.exlibrisgroup.com/alma/apis/users

Get started

- Create your account
- Generate an API key
- Play

“Students in Computer Lab 1988”

Photo from University of St. Thomas Libraries Archives
http://cdm16120.contentdm.oclc.org/cdm/ref/collection/arch-photo/id/796

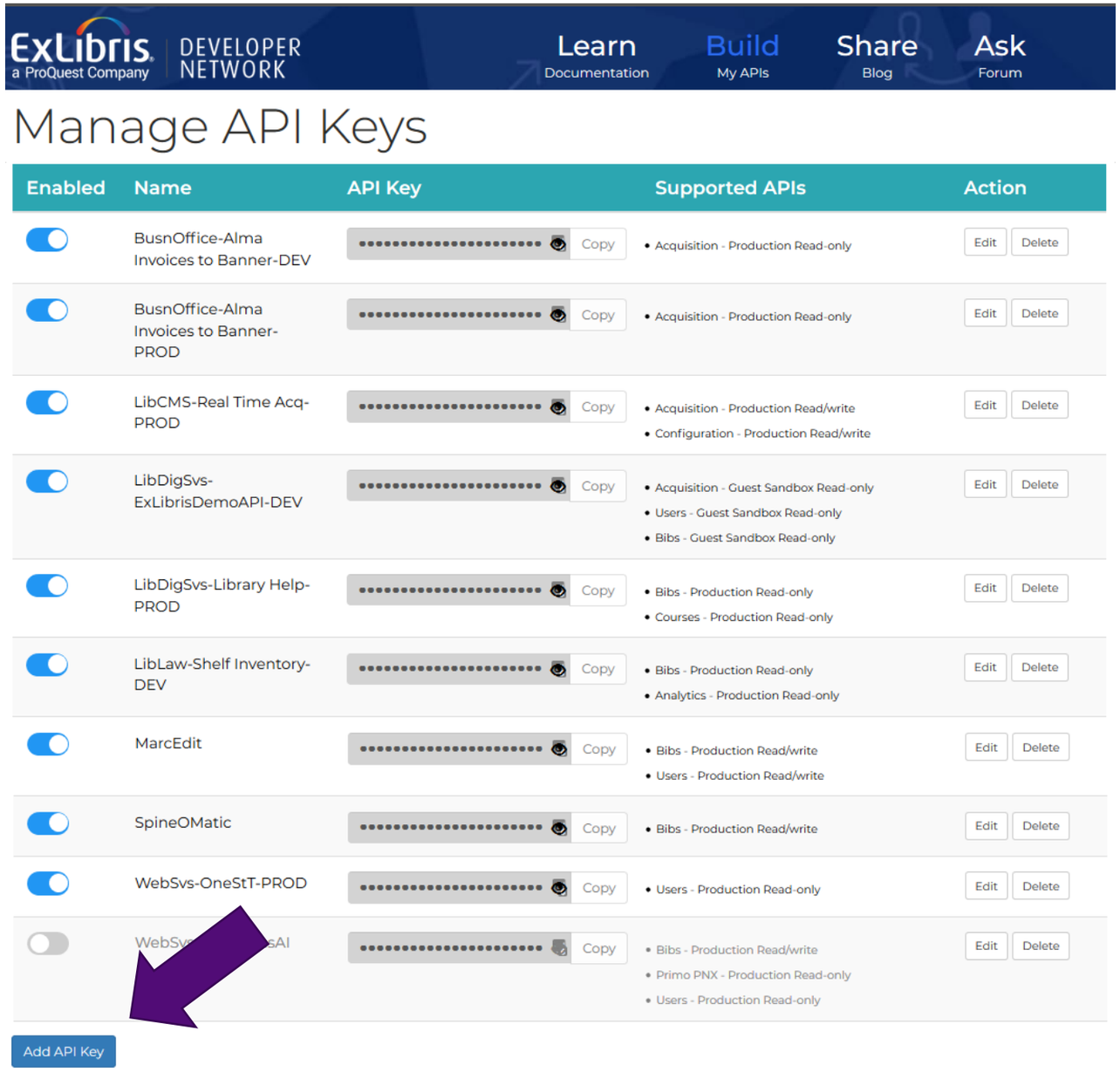
Set yourself up (if you have authorization)

- Who holds the keys? Take them out for coffee
- Get yourself an account
- Go to the Ex Libris Developer's Network
  - https://developers.exlibrisgroup.com
- Go to Build: My APIs

DEV

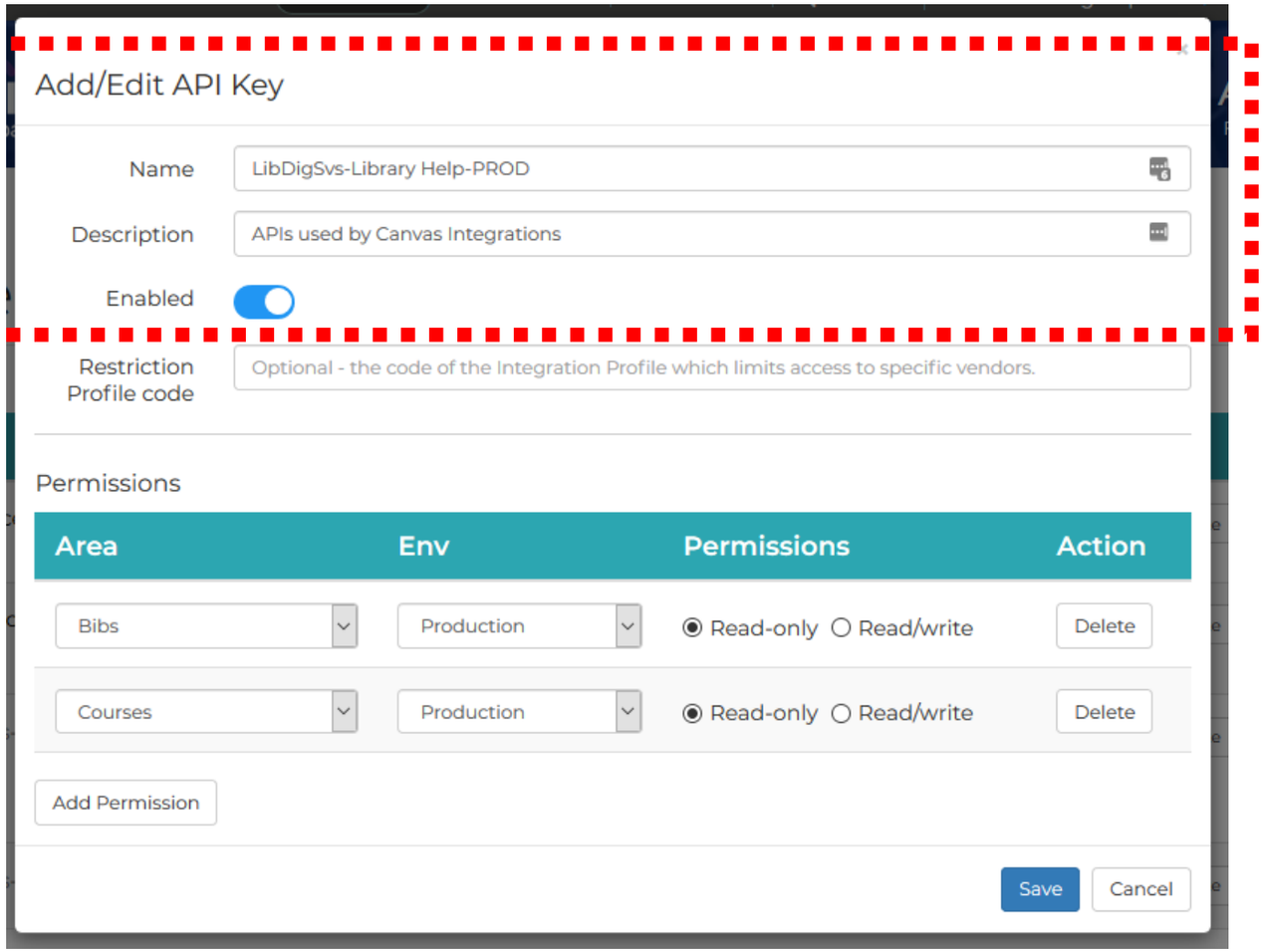
Generate a key

- Recommend one for each application
- Use meaningful names
- …that way you have an idea of what's "out there"
- …and you can revoke access if you need to
- For example, when we go from development to production
- To generate a key click on Add API Key



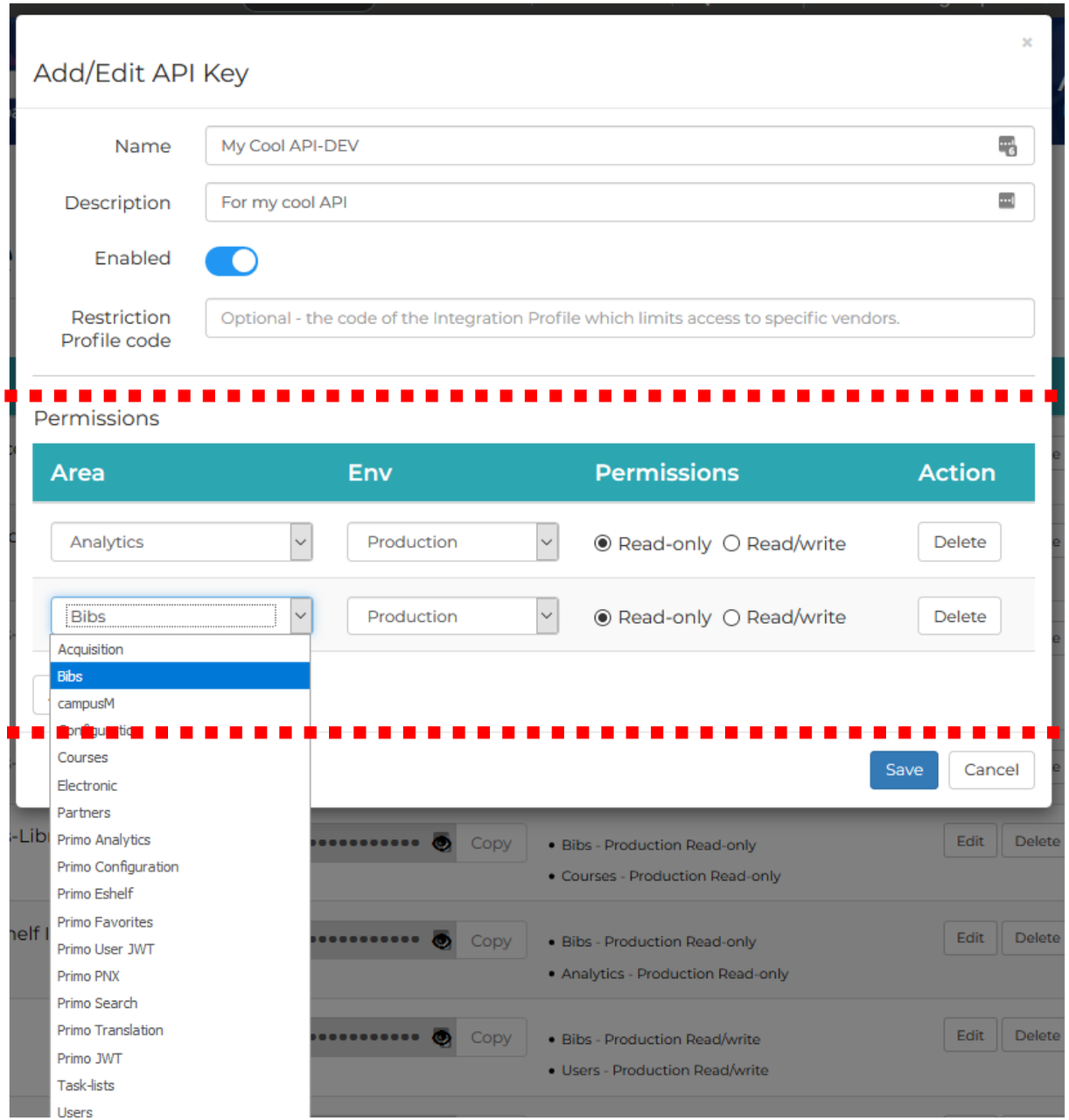
Adding an Application

- Give it a meaningful name. Add things like:
  - What library service area uses it
  - What application it is for
  - Whether it is in Production or Development
- You can disable keys you use for development when not in use!

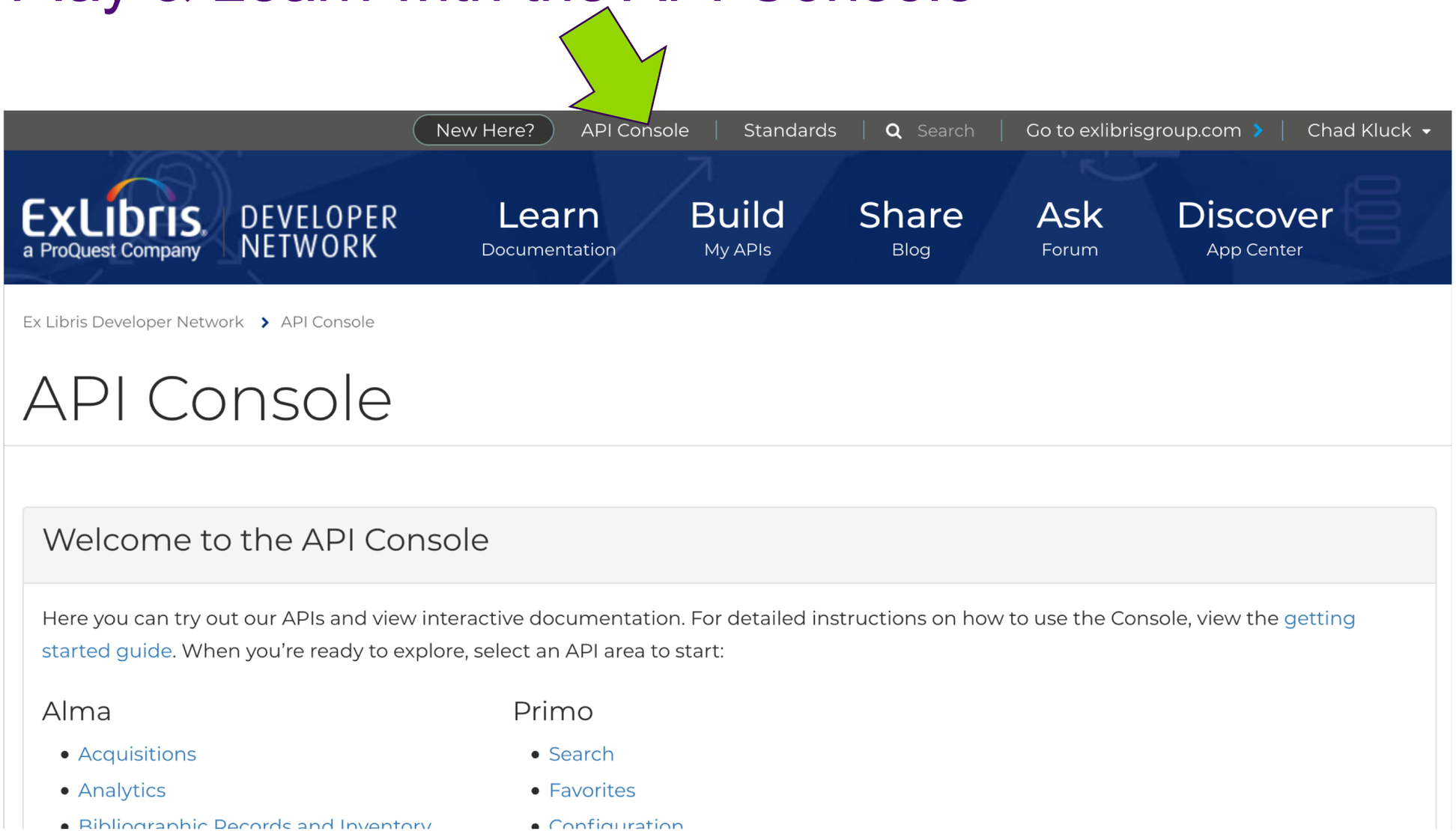


Next Step: Permissions

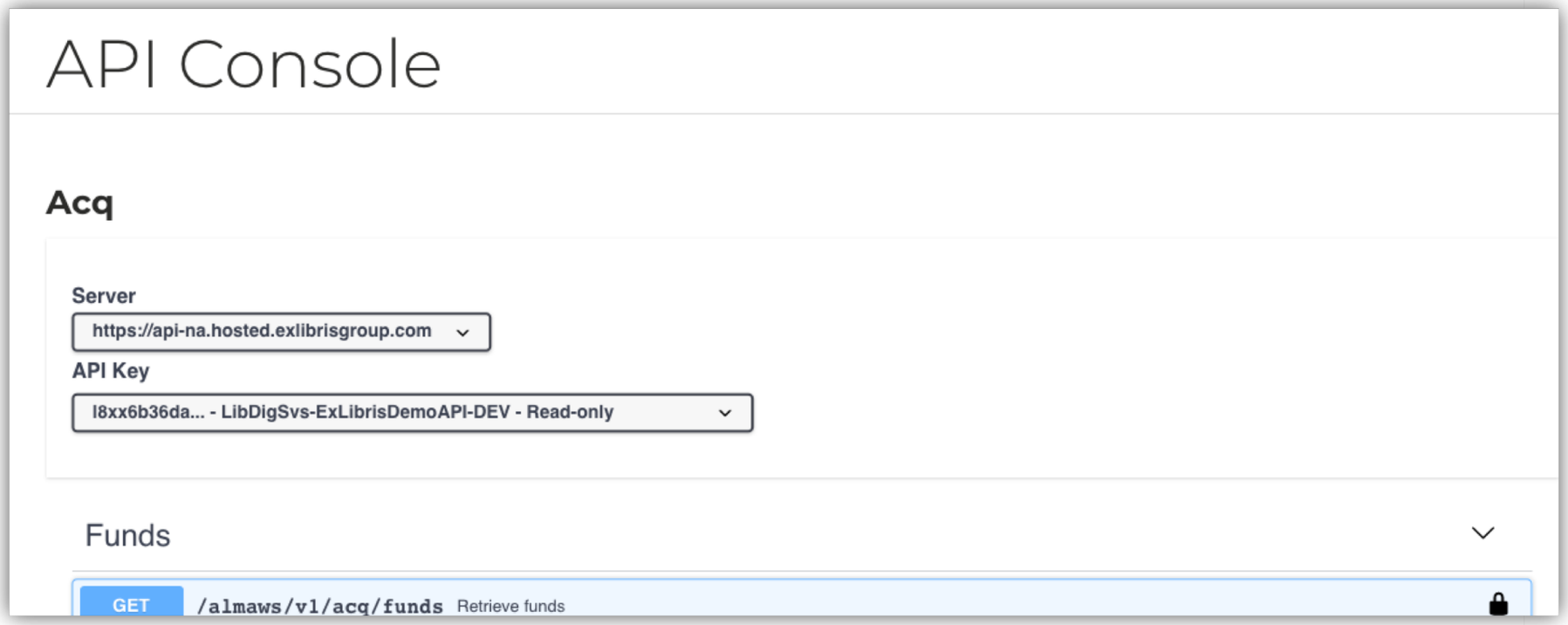
- Add an API “Area”
- You don’t typically need a whole bunch of “Areas”
- For example our portal only needed “Users” as loans and fines are included
- What environment should data come from?
- Does it have Read-Only or Read/write access?

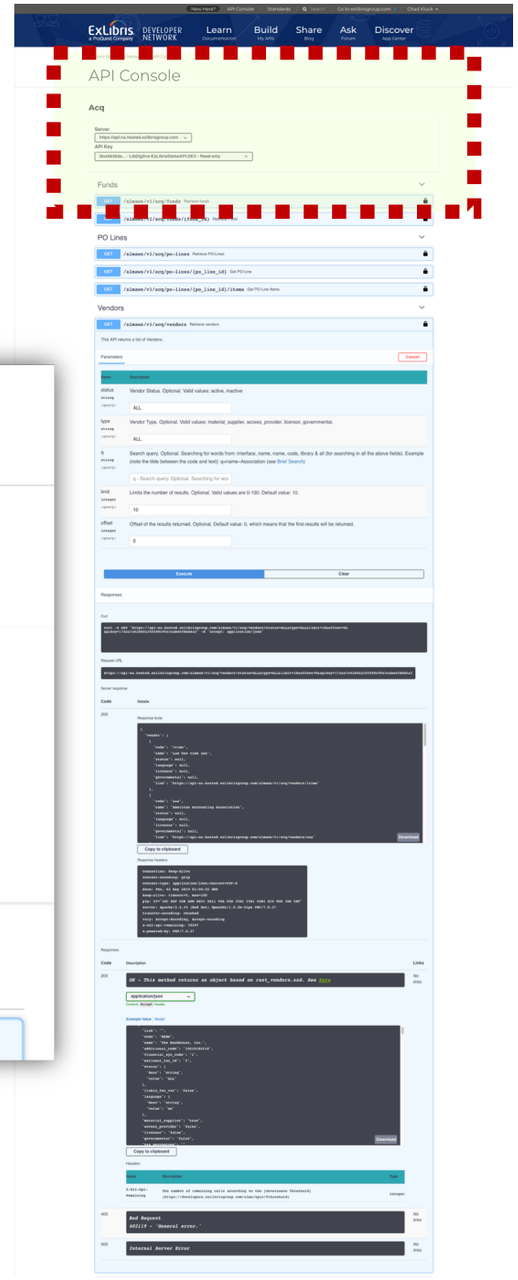


Play & Learn with the API Console



Choose your server and API key





Choose your request type and “Try it out”

PO Lines

GET **/almaws/v1/acq/po-lines** Retrieve PO-Lines

GET **/almaws/v1/acq/po-lines/{po_line_id}** Get PO-Line

GET **/almaws/v1/acq/po-lines/{po_line_id}/items** Get PO-Line Items

Vendors

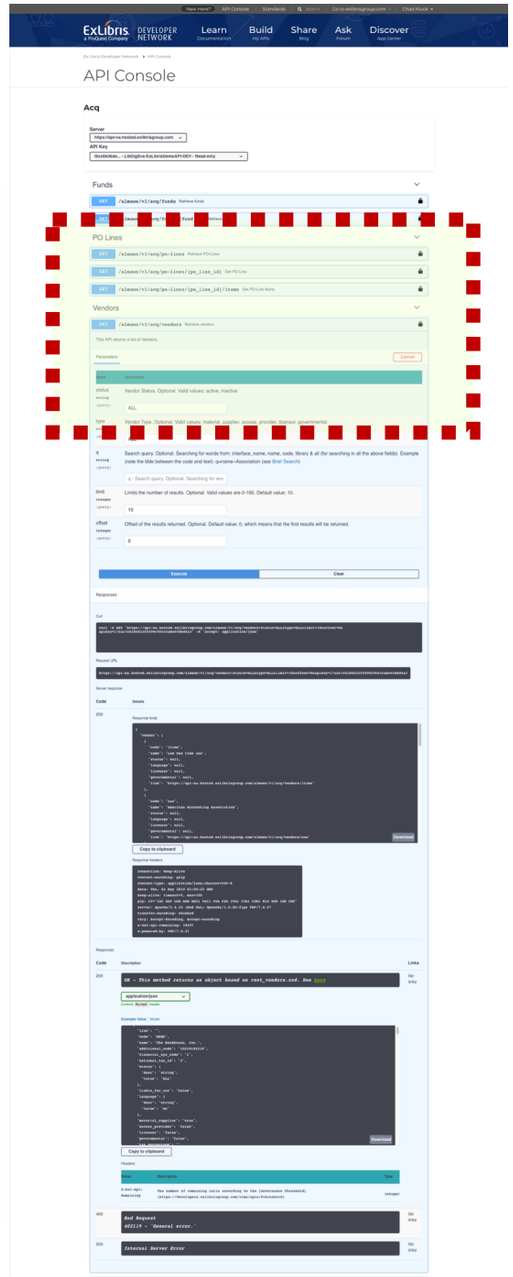
GET **/almaws/v1/acq/vendors** Retrieve vendors

This API returns a list of Vendors.

Parameters

Try it out

Name	Description
status **string** *(query)*	Vendor Status. Optional. Valid values: active, inactive *Default value* : ALL
type **string** *(query)*	Vendor Type. Optional. Valid values: material_supplier, access_provider, licensor, governmental. *Default value* : ALL



Enter Parameters and Execute

GET `/almaws/v1/acq/vendors` Retrieve vendors

This API returns a list of Vendors.

Parameters | Cancel

Name	Description
status **string** *(query)*	Vendor Status. Optional. Valid values: active, inactive ALL
type **string** *(query)*	Vendor Type. Optional. Valid values: material_supplier, access_provider, licensor, governmental. ALL
q **string** *(query)*	Search query. Optional. Searching for words from: interface_name, name, code, library & all (for searching in all the above fields). Example (note the tilde between the code and text): q=name~Association (see Brief Search) q - Search query. Optional. Searching for wor
limit **integer** *(query)*	Limits the number of results. Optional. Valid values are 0-100. Default value: 10. 10
offset **integer** *(query)*	Offset of the results returned. Optional. Default value: 0, which means that the first results will be returned. 0

Execute | Clear

View the response body

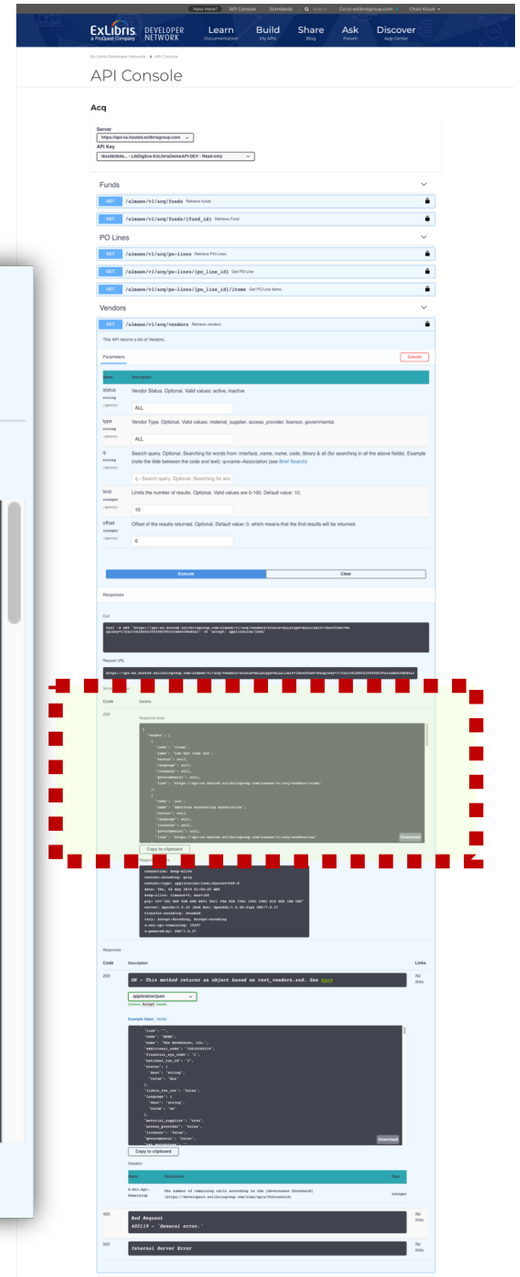
Server response

Code	Details
200	Response body


```
{
  "vendor": [
    {
      "code": "1time",
      "name": "Law One time use",
      "status": null,
      "language": null,
      "licensor": null,
      "governmental": null,
      "link": "https://api-na.hosted.exlibrisgroup.com/almaws/v1/acq/vendors/1time"
    },
    {
      "code": "aaa",
      "name": "American Accounting Association",
      "status": null,
      "language": null,
      "licensor": null,
      "governmental": null,
      "link": "https://api-na.hosted.exlibrisgroup.com/almaws/v1/acq/vendors/aaa"
```


Download

Copy to clipboard

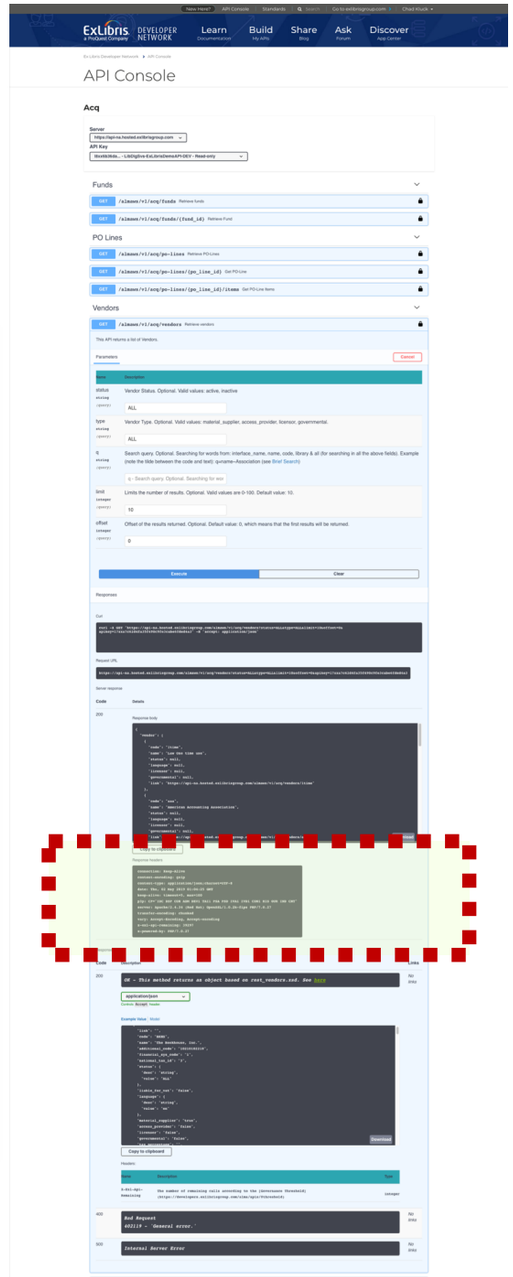


Look at the response headers

How many requests do you have left?

Response headers

```
connection: Keep-Alive
content-encoding: gzip
content-type: application/json;charset=UTF-8
date: Thu, 02 May 2019 01:04:25 GMT
keep-alive: timeout=5, max=100
p3p: CP="IDC DSP COR ADM DEVi TAIi PSA PSD IVAi IVDi CONi HIS OUR IND CNT"
server: Apache/2.4.34 (Red Hat) OpenSSL/1.0.2k-fips PHP/7.0.27
transfer-encoding: chunked
vary: Accept-Encoding, Accept-encoding
x-exl-api-remaining: 39297
x-powered-by: PHP/7.0.27
```



Eureka!

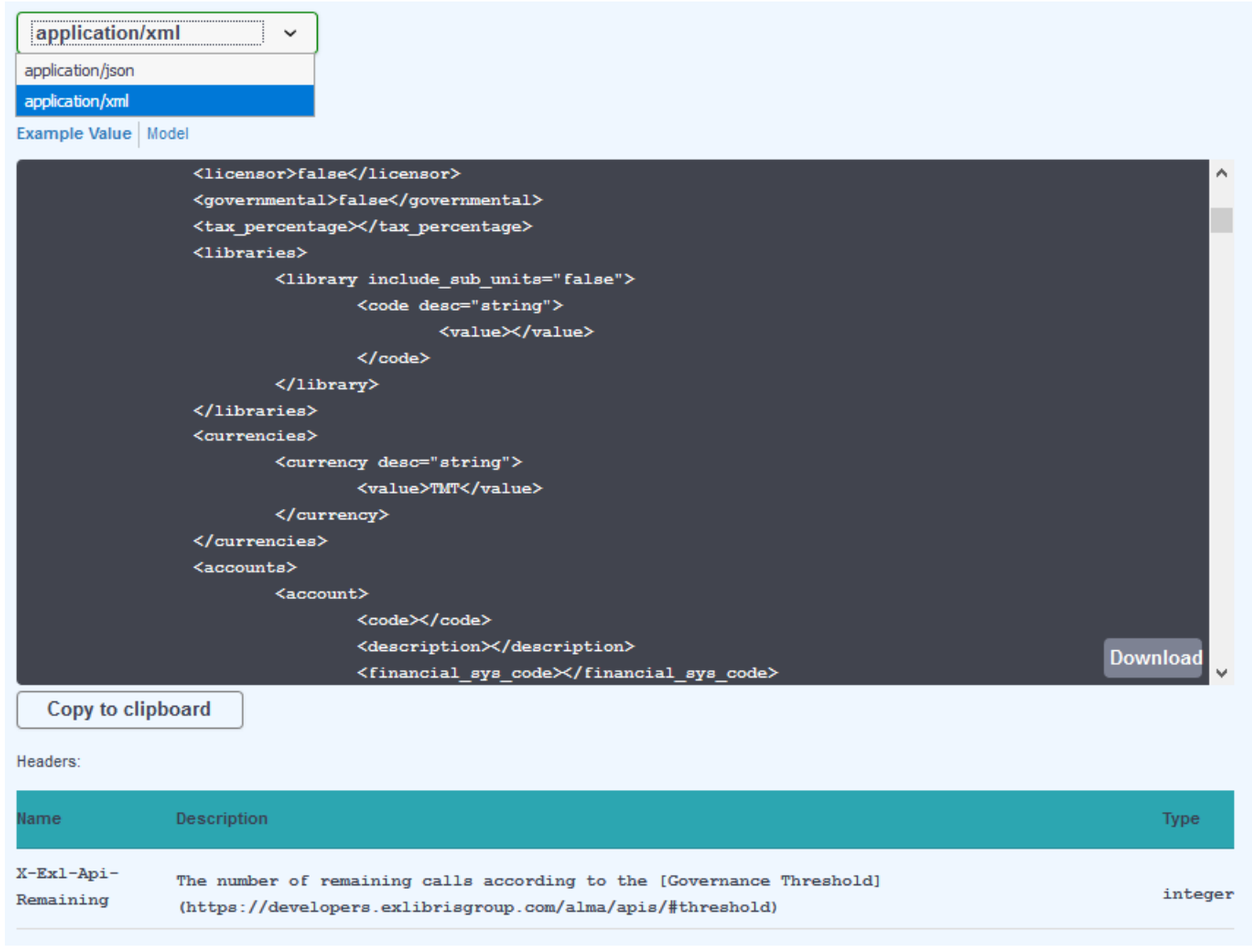
- You've got data!
- But now what?

Grab the Request URL

- As stated earlier, an API (GET) request is basically just a URL
- This formulates your URL with all the parameters and even the API key
- Note that if you want JSON data you either send that fact in the request header, or you add `&format=json` to your URL

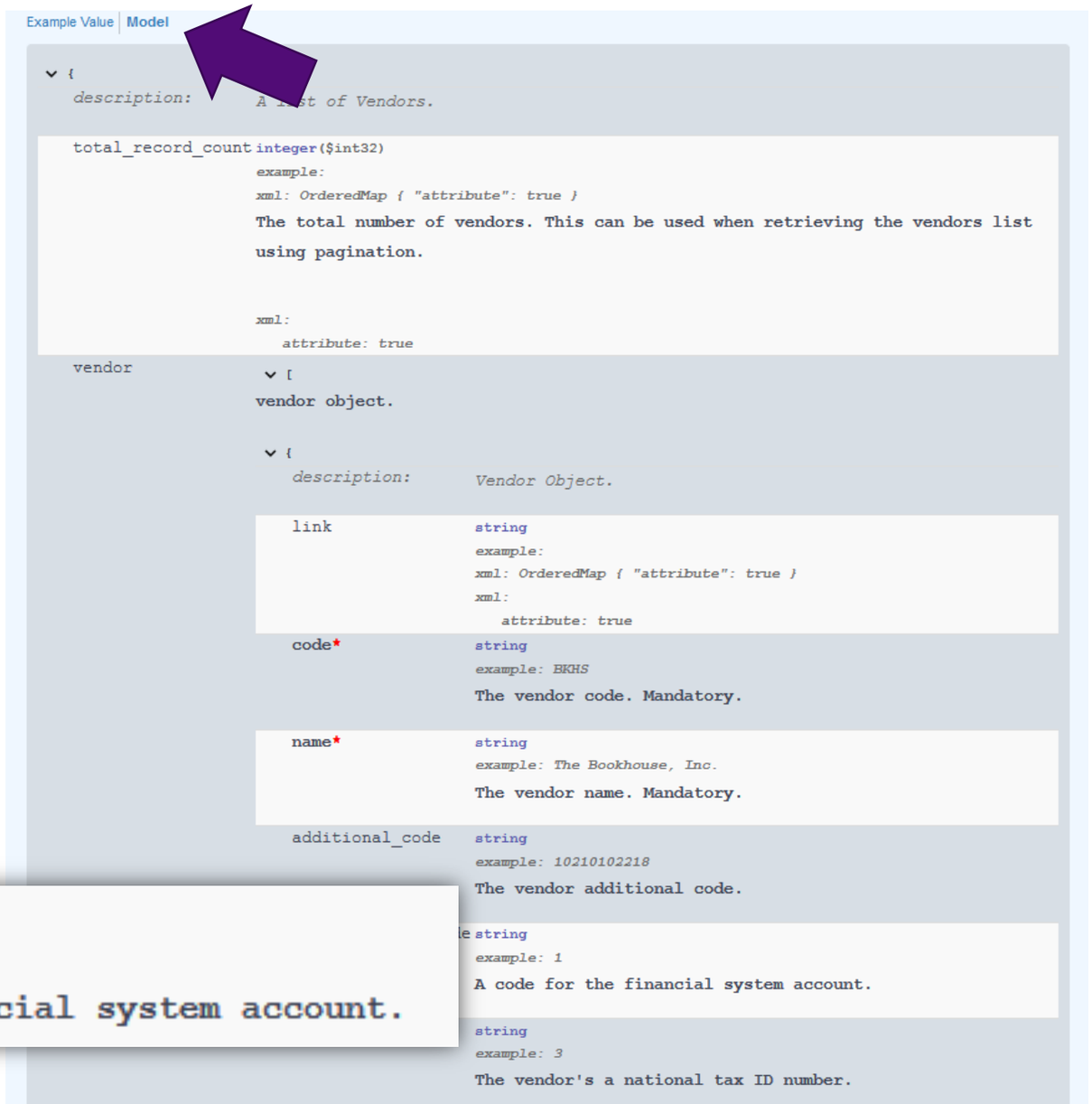
XML or JSON

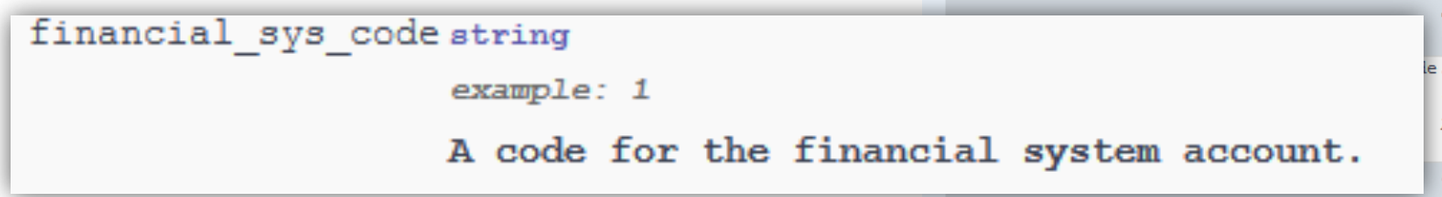
- Default format returned is XML, but you can switch it to JSON



Model view

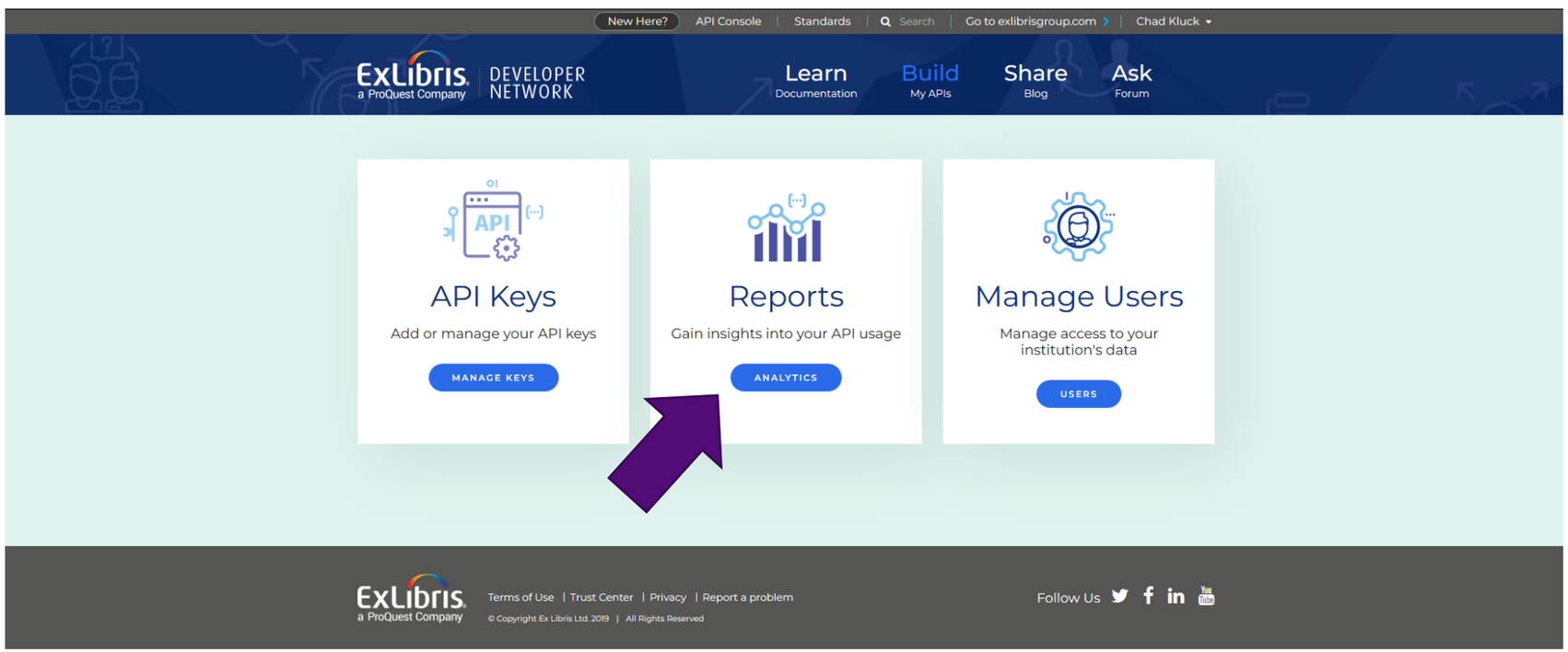
- Empty fields may not display in the response, so how do you know what fields are available?
- Model View!



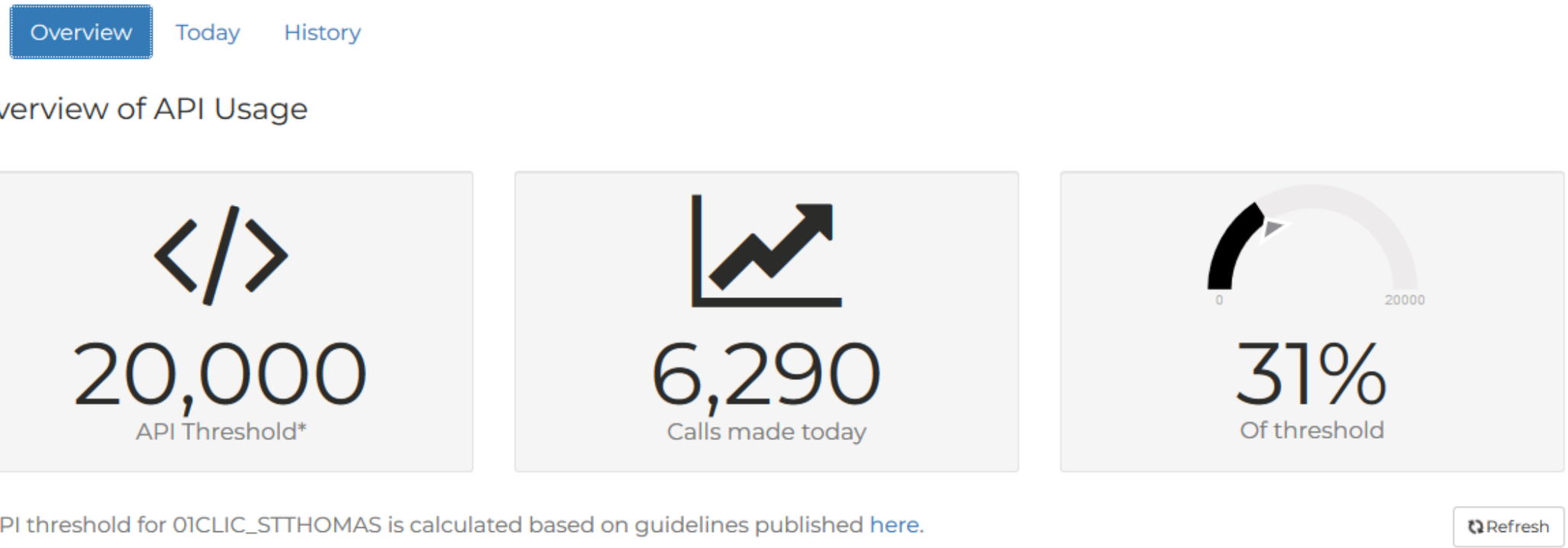


Reports

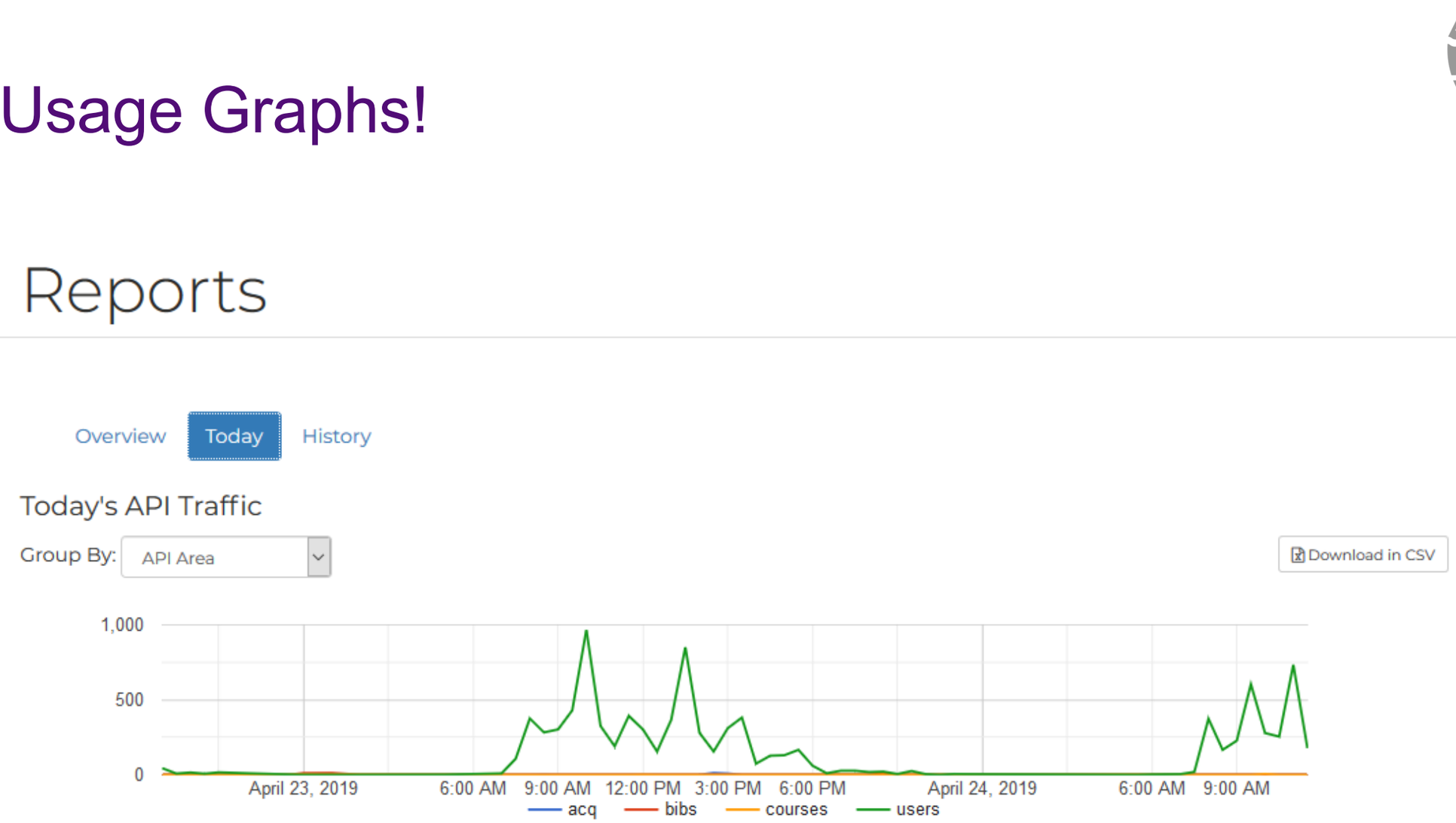
- Ex Libris likes to report on how many API requests they process
- But how many are yours? And for what?
- Well, find out!



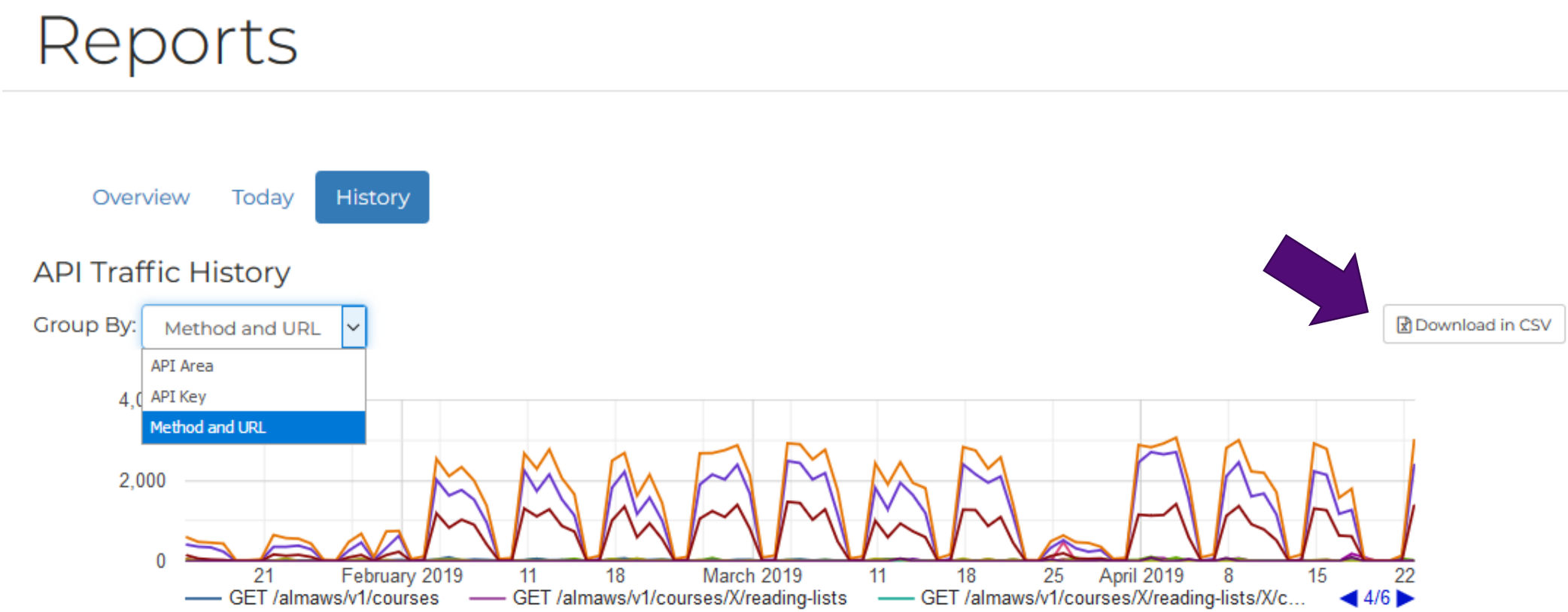
Be a good steward of resources



Usage Graphs!



Download any report to CSV



Daily usage by Area

	A	B	C	D	E	F	G
1		acq	bibs	courses	items	users	
2	Jan 14, 2019, 6:00:00 PM	0	62	2	51	1,167	
3	Jan 15, 2019, 6:00:00 PM	0	0	3	0	900	
4	Jan 16, 2019, 6:00:00 PM	0	0	4	0	841	
5	Jan 17, 2019, 6:00:00 PM	0	0	10	0	701	
6	Jan 18, 2019, 6:00:00 PM	0	0	9	0	29	
7	Jan 19, 2019, 6:00:00 PM	0	0	5	0	35	
8	Jan 20, 2019, 6:00:00 PM	0	0	12	0	60	
9	Jan 21, 2019, 6:00:00 PM	0	9	16	11	1,171	
10	Jan 22, 2019, 6:00:00 PM	0	9	25	70	1,060	
11	Jan 23, 2019, 6:00:00 PM	0	0	16	0	1,097	
12	Jan 24, 2019, 6:00:00 PM	0	0	16	0	831	
13	Jan 25, 2019, 6:00:00 PM	0	0	3	0	37	
14	Jan 26, 2019, 6:00:00 PM	0	0	0	0	30	
15	Jan 27, 2019, 6:00:00 PM	0	10	5	19	828	
16	Jan 28, 2019, 6:00:00 PM	0	41	10	52	1,293	
17	Jan 29, 2019, 6:00:00 PM	0	0	9	0	140	
18	Jan 30, 2019, 6:00:00 PM	0	14	19	33	1,218	
19	Jan 31, 2019, 6:00:00 PM	0	0	33	0	1,613	
20	Feb 1, 2019, 6:00:00 PM	0	0	17	0	57	
21	Feb 2, 2019, 6:00:00 PM	0	0	18	0	172	
22	Feb 3, 2019, 6:00:00 PM	0	20	49	37	5,737	
23	Feb 4, 2019, 6:00:00 PM	0	85	112	41	4,559	
24	Feb 5, 2019, 6:00:00 PM	0	16	22	35	5,125	
25	Feb 6, 2019, 6:00:00 PM	0	6	45	0	4,422	
26	Feb 7, 2019, 6:00:00 PM	0	7	39	0	2,749	
27	Feb 8, 2019, 6:00:00 PM	0	0	17	0	76	

Usage by half-hour!

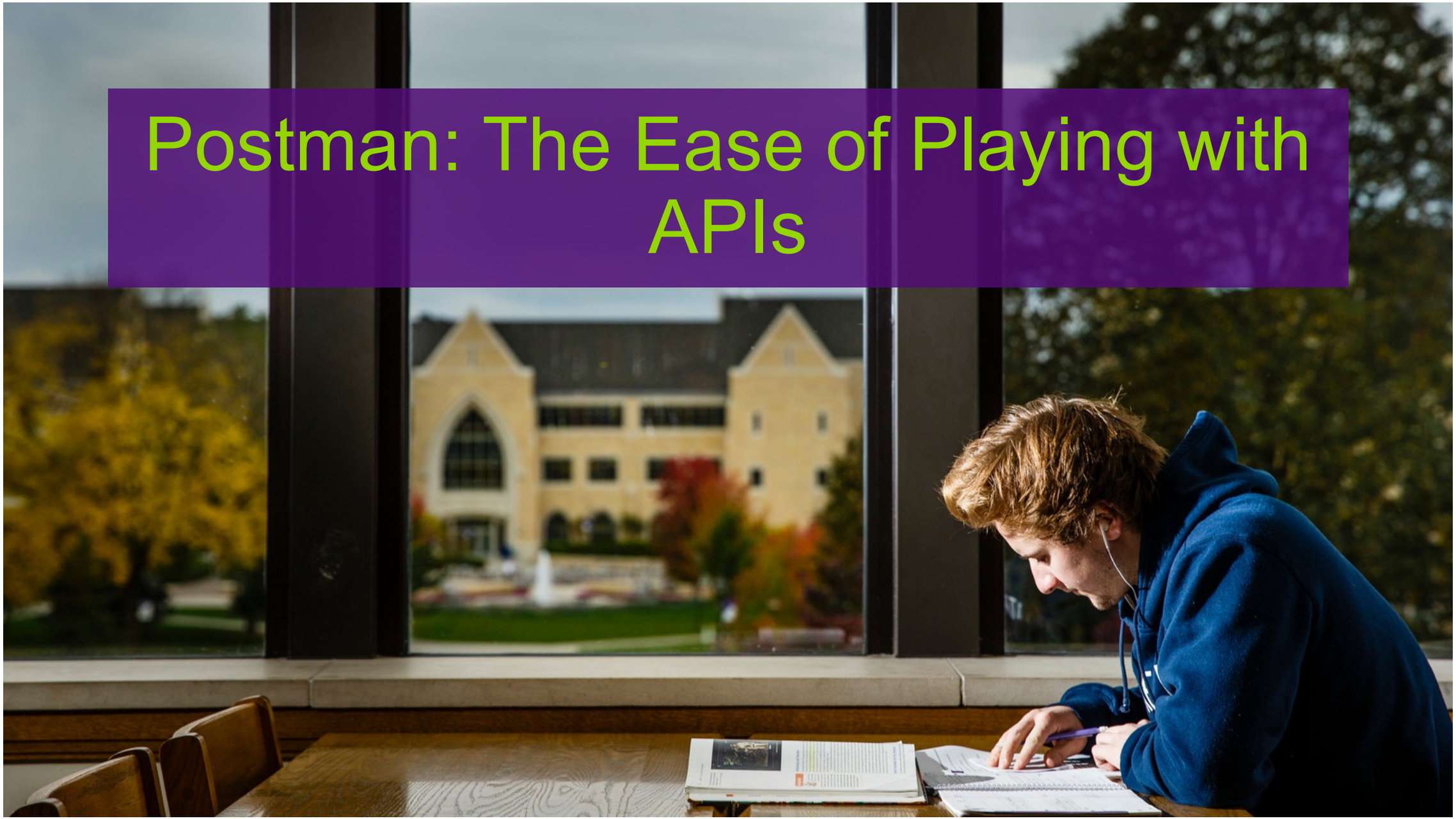
	A	B	C	D	E
1		acq	bibs	courses	users
2	Apr 22, 2019, 7:00:00 PM	0	0	0	44
3	Apr 22, 2019, 7:30:00 PM	0	0	0	8
4	Apr 22, 2019, 8:00:00 PM	0	0	0	15
5	Apr 22, 2019, 8:30:00 PM	0	0	0	7
6	Apr 22, 2019, 9:00:00 PM	0	0	0	16
7	Apr 22, 2019, 10:00:00 PM	0	0	0	10
8	Apr 22, 2019, 11:30:00 PM	0	0	1	2
9	Apr 23, 2019, 12:00:00 AM	0	11	2	0
10	Apr 23, 2019, 1:00:00 AM	0	12	1	0
11	Apr 23, 2019, 2:00:00 AM	0	0	1	0
12	Apr 23, 2019, 5:00:00 AM	0	0	0	1
13	Apr 23, 2019, 7:00:00 AM	0	0	0	9
14	Apr 23, 2019, 7:30:00 AM	0	0	0	107
15	Apr 23, 2019, 8:00:00 AM	0	0	0	375
16	Apr 23, 2019, 8:30:00 AM	1	0	0	282
17	Apr 23, 2019, 9:00:00 AM	0	0	0	302
18	Apr 23, 2019, 9:30:00 AM	0	0	0	429
19	Apr 23, 2019, 10:00:00 AM	0	0	0	965
20	Apr 23, 2019, 10:30:00 AM	0	0	0	325
21	Apr 23, 2019, 11:00:00 AM	0	0	0	190
22	Apr 23, 2019, 11:30:00 AM	0	0	0	392
23	Apr 23, 2019, 12:00:00 PM	0	0	1	302
24	Apr 23, 2019, 12:30:00 PM	0	0	0	155
25	Apr 23, 2019, 1:00:00 PM	0	0	0	366

Even daily usage by key

- This is where giving your keys useful names help!
- And giving each application it's own key!
- Find the API hogs and rogue scripts!

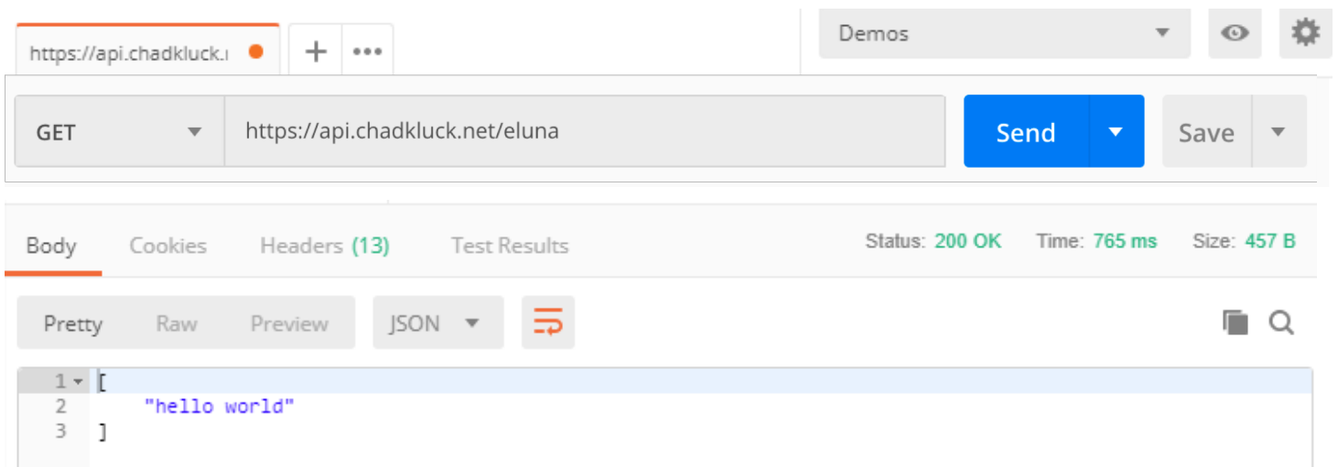
	A	B	C	D	E	F
1		BusnOffice-Invoices-DEV	LibDigSvs-DemoAPI-DE	LibDigSvs-Library H	SpineOMatic	WebSvs-OneStT-PROD
2	Mar 30, 2019, 7:00:00 PM	0	0	35	0	130
3	Mar 31, 2019, 7:00:00 PM	20	0	47	31	6,478
4	Apr 1, 2019, 7:00:00 PM	166	0	124	0	6,654
5	Apr 2, 2019, 7:00:00 PM	75	0	33	25	6,707
6	Apr 3, 2019, 7:00:00 PM	0	0	112	0	7,180
7	Apr 4, 2019, 7:00:00 PM	83	0	15	0	4,107
8	Apr 5, 2019, 7:00:00 PM	0	0	2	0	118
9	Apr 6, 2019, 7:00:00 PM	0	0	4	0	228
10	Apr 7, 2019, 7:00:00 PM	116	0	28	2	6,027
11	Apr 8, 2019, 7:00:00 PM	70	0	68	32	6,811
12	Apr 9, 2019, 7:00:00 PM	0	0	15	6	4,740
13	Apr 10, 2019, 7:00:00 PM	0	0	30	0	4,651
14	Apr 11, 2019, 7:00:00 PM	0	0	16	0	3,380
15	Apr 12, 2019, 7:00:00 PM	1	0	2	0	83
16	Apr 13, 2019, 7:00:00 PM	0	0	53	0	227
17	Apr 14, 2019, 7:00:00 PM	0	0	36	43	6,446
18	Apr 15, 2019, 7:00:00 PM	8	0	39	56	6,185
19	Apr 16, 2019, 7:00:00 PM	0	0	10	16	3,377
20	Apr 17, 2019, 7:00:00 PM	283	0	4	0	3,677
21	Apr 18, 2019, 7:00:00 PM	95	0	6	0	65
22	Apr 19, 2019, 7:00:00 PM	0	0	11	0	29
23	Apr 20, 2019, 7:00:00 PM	0	0	16	0	19
24	Apr 21, 2019, 7:00:00 PM	1	0	69	0	189
25	Apr 22, 2019, 7:00:00 PM	20	0	30	0	6,852
26						

Postman: The Ease of Playing with APIs



Get Postman

- A browser, but for APIs
- Download from site https://www.getpostman.com and install
- Create a new GET request for https://api.chadkluck.net/eluna
  - Hit “Send”



Beautiful, isn’t it?

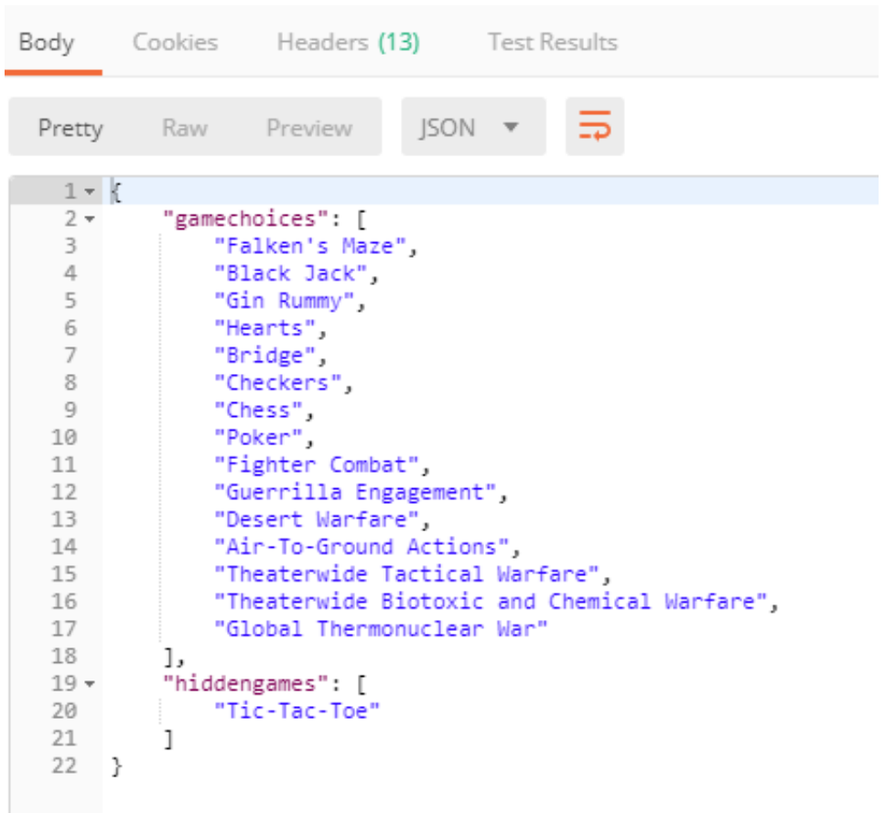
Add a query parameter (key + value pair)

1. Add “code” as a key and enter “CPE” for the value
2. Hit “Send”

You’ll notice that the parameters are added to the query string in real-time. You can paste in a full URL with query string and it will parse out the parameters for you.

Would you like to play a game?

`https://api.chadkluck.net/eluna?code=CPE`




```
{
    "gamechoices": [
        "Falken's Maze",
        "Black Jack",
        "Gin Rummy",
        "Hearts",
        "Bridge",
        "Checkers",
        "Chess",
        "Poker",
        "Fighter Combat",
        "Guerrilla Engagement",
        "Desert Warfare",
        "Air-To-Ground Actions",
        "Theaterwide Tactical Warfare",
        "Theaterwide Biotoxic and Chemical Warfare",
        "Global Thermonuclear War"
    ],
    "hiddengames": [
        "Tic-Tac-Toe"
    ]
}
```


- Go ahead and try changing the value for “code” and submit:
  - BAO
  - LVP
  - CBL
  - 8BL

Careful when playing around!

- Doing a GET: You're safe
- DELETE, POST, PUT? Know what you're doing!
- This is where provisioning the keys is important

Let's try another request

- Remember our university portal example?
  - Loans

    ```
    https://api-na.hosted.exlibrisgroup.com/almaws/v1/users/{{user_id}}/loans?
    user_id_type=all_unique&limit=10&offset=0&order_by=due_date&format=json&direction=ASC
    &apikey={{apikey}}
    ```

  - Holds/Requests

    ```
    https://api-na.hosted.exlibrisgroup.com/almaws/v1/users/{{user_id}}/requests?
    request_type=HOLD&user_id_type=all_unique&limit=10&offset=0&status=active&format=json
    &apikey={{apikey}}
    ```

  - Fines

    ```
    https://api-na.hosted.exlibrisgroup.com/almaws/v1/users/{{user_id}}/fees?
    user_id_type=all_unique&status=ACTIVE&format=json&apikey={{apikey}}
    ```

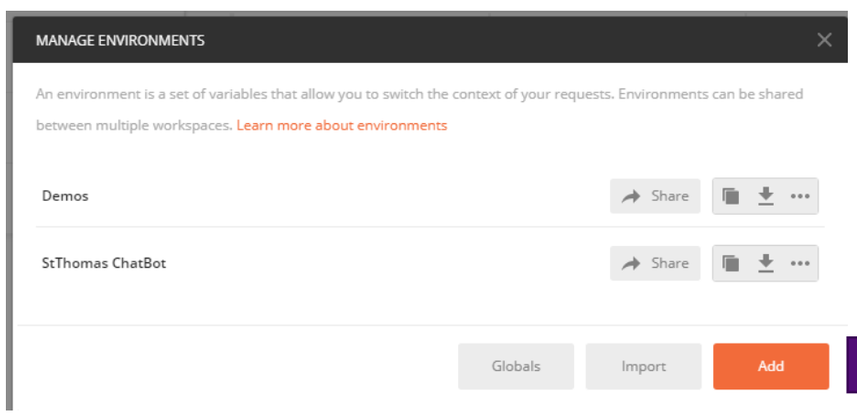
- Replace `{{user_id}}` and `{{apikey}}` with your own
- Or don't, because…

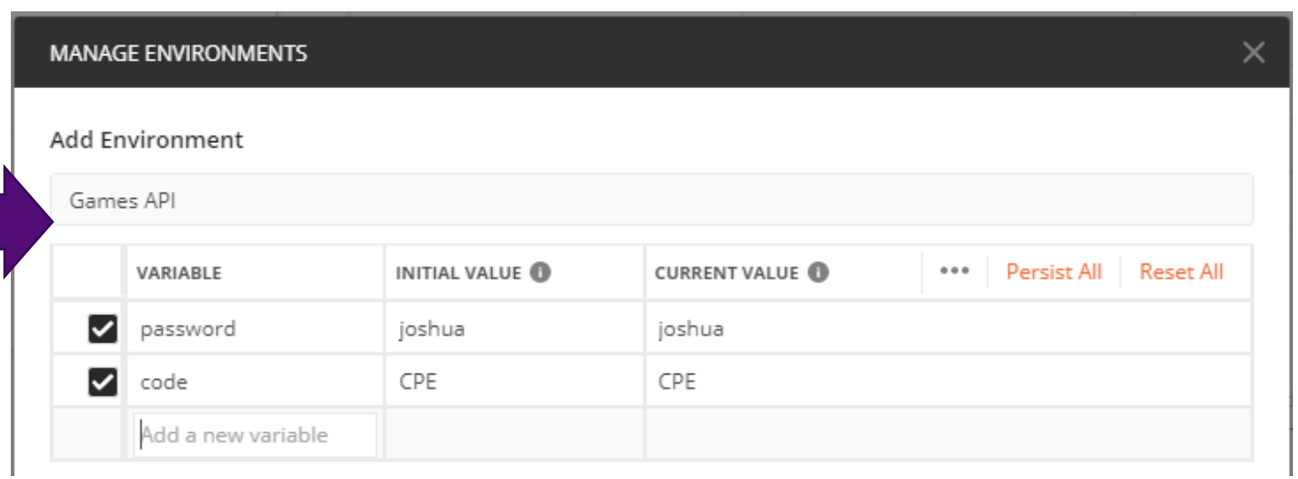
LHF

Environment variables!

- You can add frequently used values for testing, such as mm_id, user_id, etc.
- Access them by placing the variable name in between {{ }}
- https://api.chadkluck.net/eluna?code={{code}}

Environment Variables: Manage/Add



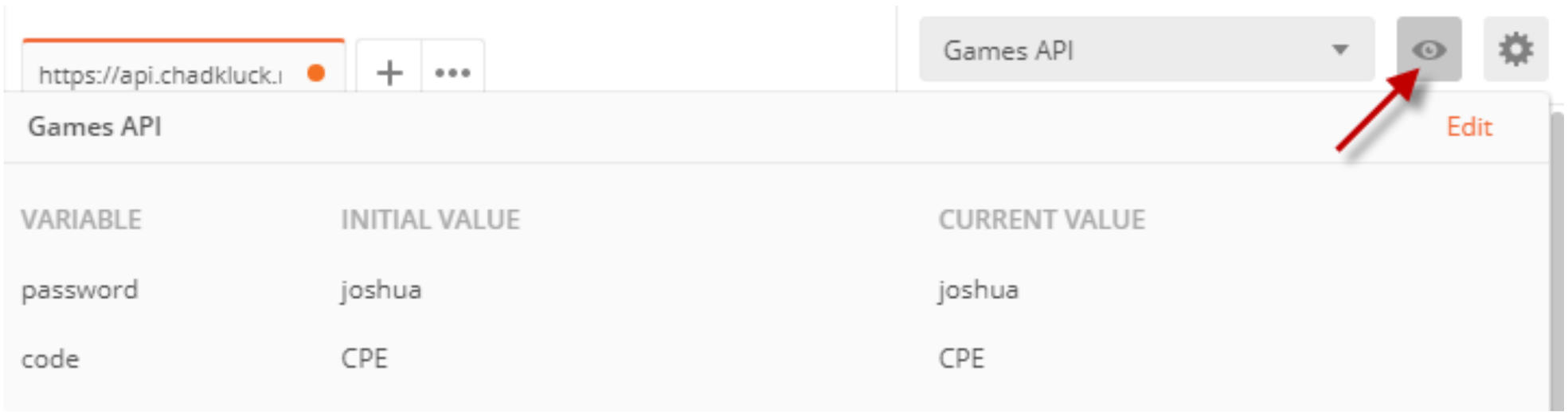


Use {{variable}} for the value

- Switch between environments to change the value of the variable
- Multiple test users, loan ids, book ids, etc

Environment Variables: Switch Between Values

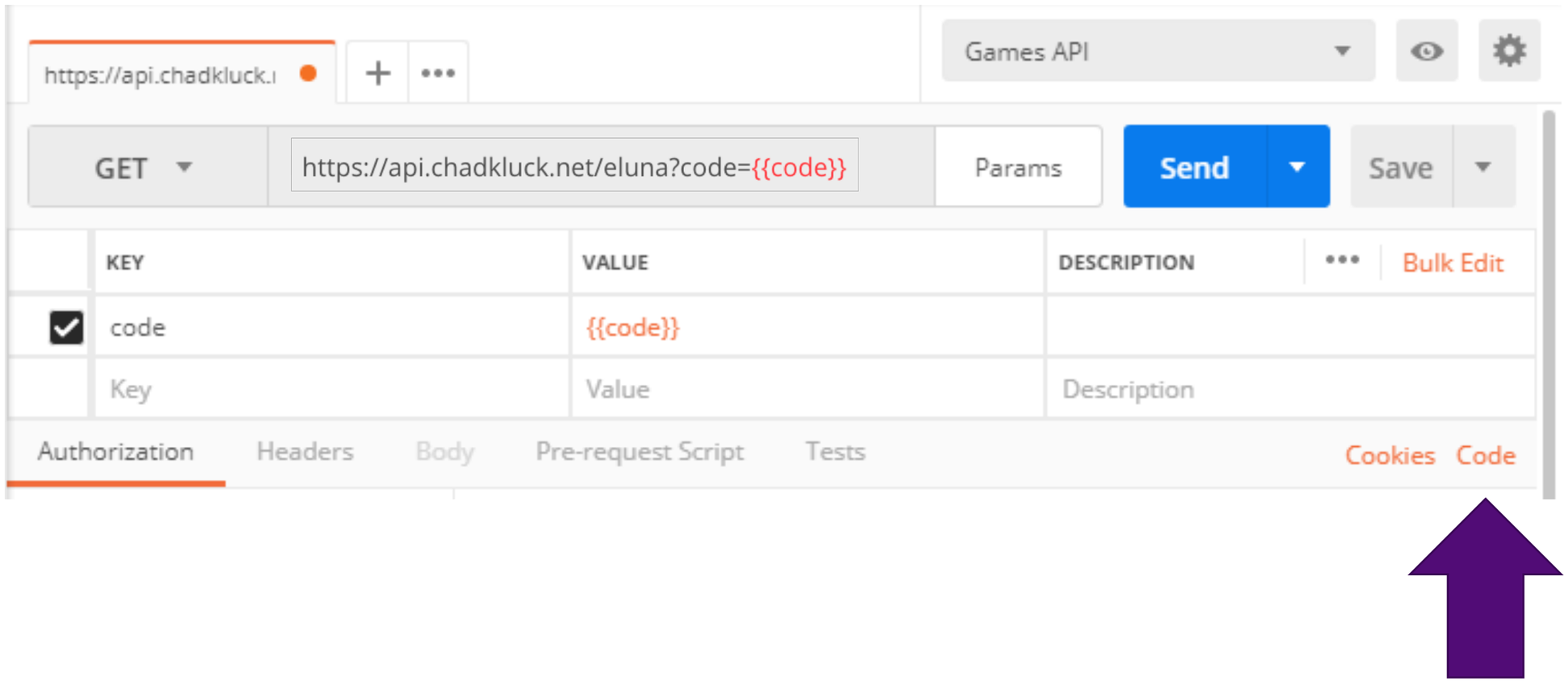
Environment Variables: View Current Values



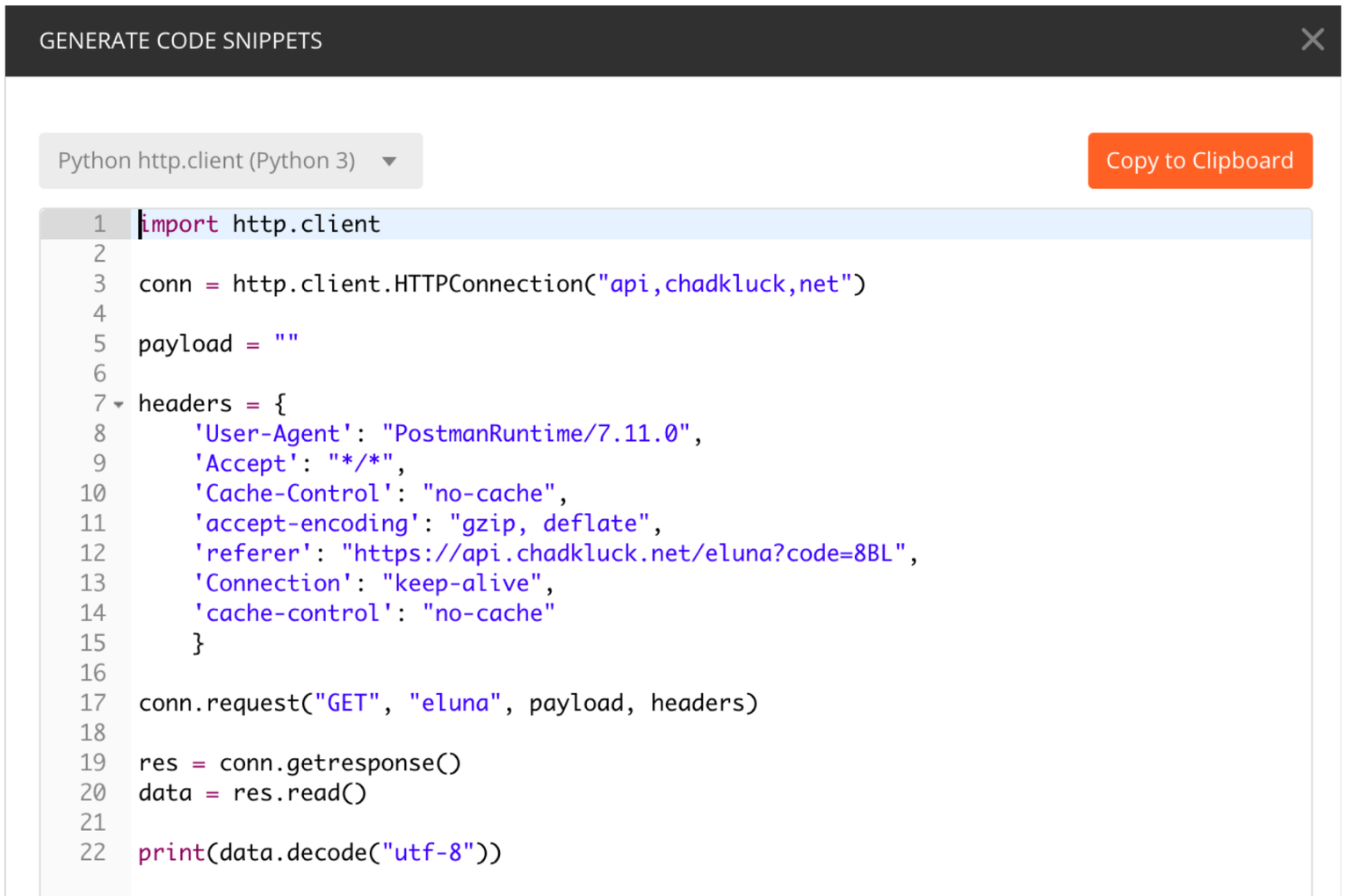
Postman is fully featured and free!

- Save multiple environment variable sets
- Organize your requests in folders
- Create a free account and access your work from the cloud across multiple machines
- Are my keys safe? Check out https://www.getpostman.com/security
  - "AES-256-GCM encrypted at the application layer before storage"
- Oh, and one more thing…

Code snipits!



Python!


GENERATE CODE SNIPPETS
Python http.client (Python 3)
Copy to Clipboard
import http.client
conn = http.client.HTTPConnection("api,chadkluck,net")
payload = ""
headers = {
'User-Agent': "PostmanRuntime/7.11.0",
'Accept': "*/*",
'Cache-Control': "no-cache",
'accept-encoding': "gzip, deflate",
'referer': "https://api.chadkluck.net/eluna?code=8BL",
'Connection': "keep-alive",
'cache-control': "no-cache"
}
conn.request("GET", "eluna", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))

PHP!

GENERATE CODE SNIPPETS

PHP HttpRequest

Copy to Clipboard

```php
<?php

$request = new HttpRequest();
$request->setUrl('https://api.chadkluck.net/eluna');
$request->setMethod(HTTP_METH_GET);

$request->setQueryData(array(
  'code' => '8BL'
));

$request->setHeaders(array(
  'cache-control' => 'no-cache',
  'Connection' => 'keep-alive',
  'referer' => 'https://api.chadkluck.net/eluna?code=8BL',
  'accept-encoding' => 'gzip, deflate',
  'Cache-Control' => 'no-cache',
  'Accept' => '*/*',
  'User-Agent' => 'PostmanRuntime/7.11.0'
));

try {
  $response = $request->send();

  echo $response->getBody();
} catch (HttpException $ex) {
  echo $ex;
}
```

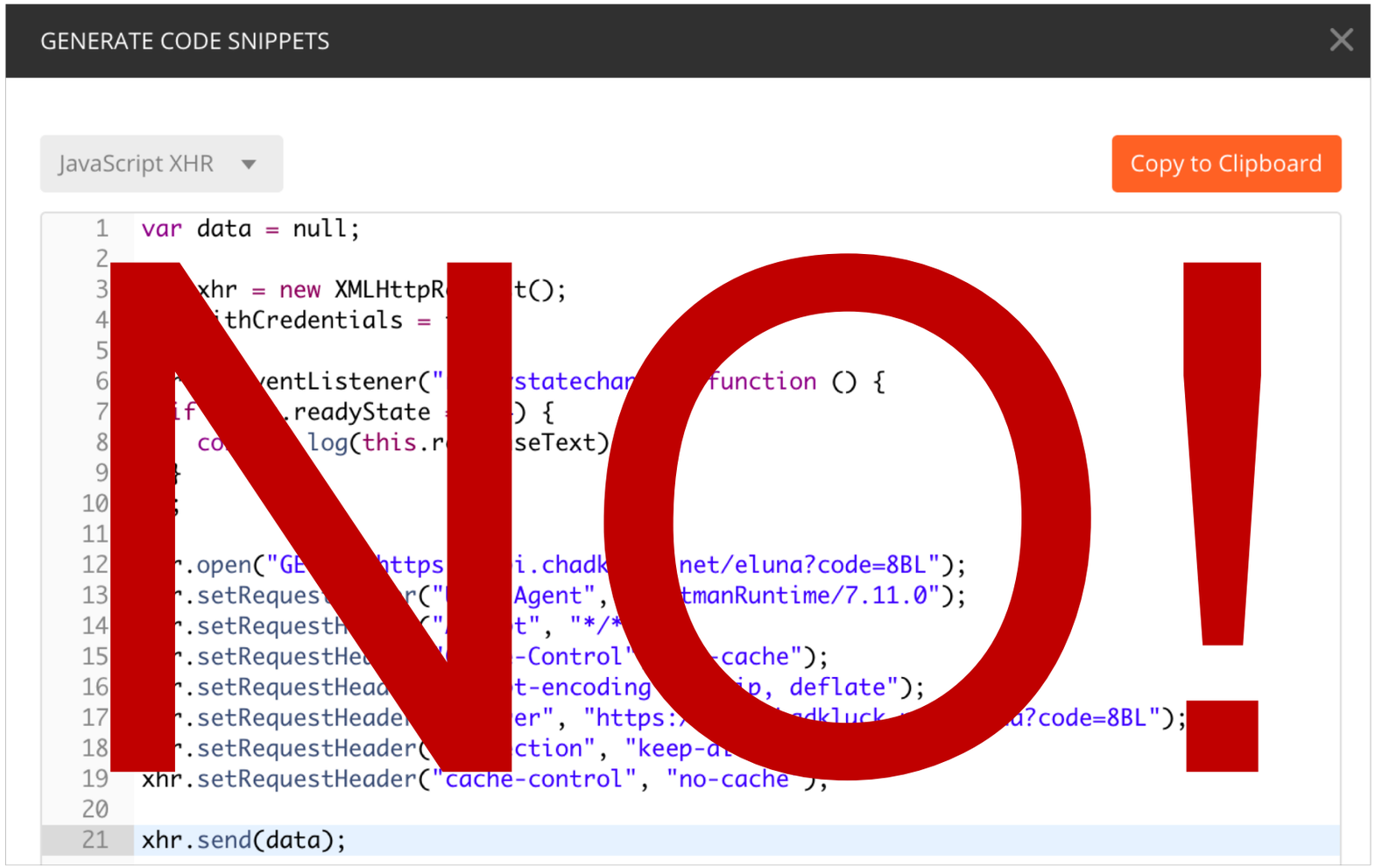
cURL!

cURL

Copy to Clipboard

```
curl -X GET \
  'https://api.chadkluck.net/eluna?code=8BL' \
  -H 'Accept: */*' \
  -H 'Cache-Control: no-cache' \
  -H 'Connection: keep-alive' \
  -H 'User-Agent: PostmanRuntime/7.11.0' \
  -H 'accept-encoding: gzip, deflate' \
  -H 'cache-control: no-cache' \
  -H 'referer: https://api.chadkluck.net/eluna?code=8BL'
```

JavaScript!


GENERATE CODE SNIPPETS
JavaScript XHR
Copy to Clipboard
var data = null;
xhr = new XMLHttpR t();
ithCredentials =
ventListener(" statechar function () {
.readyState) {
log(this.r seText)
.open("GE https i.chadk net/eluna?code=8BL");
.setReques r(" Agent", tmanRuntime/7.11.0");
.setRequestH (" t", "*/*
.setRequestHe -Control" -cache");
.setRequestHead t-encoding ip, deflate");
.setRequestHeade er", "https:/ adkluck a?code=8BL");
.setRequestHeader(ction", "keep-a
xhr.setRequestHeader("cache-control", "no-cache");
xhr.send(data);
NO!

You must never, ever use JavaScript with Ex Libris APIs! Promise me, for all that is good in this world, do not USE JavaScript for this! JavaScript is fine, but NOT for accessing APIs Client-Side with keys.

Only use JavaScript on public APIs or with Oauth or Server-Side!

But.

Not.

For.

This!

Security: (aka Officer Buzz Kill)

JavaScript?

There are exceptions, right?

Security Checklist

- Are we exposing sensitive data in the API request?
  - Personal information
  - Secrets like an API key or access tokens
  - Does it have the potential to return any of the above?
- If you answer “yes” to any of these then you need to implement the request server-side
- All Ex Libris APIs will have an answer of “Yes”
- Why? API Key

“Computer Dance 1965”

St. Thomas freshmen were matched to dates based on answers to a questionnaire

Photo from University of St. Thomas Libraries Archives
http://cdm16120.contentdm.oclc.org/cdm/ref/collection/arch-photo/id/903

Applying the Security Checklist: Example #1

- Let's look at that last example again:
  - https://api.chadkluck.net/eluna?code=CPE

- Does the request:
  - Contain any personal info?
  - Contain secrets like an API key?
  - Return any of the above?

No, No, and No!


```
{
    "gamechoices": [
        "Falken's Maze",
        "Black Jack",
        "Gin Rummy",
        "Hearts",
        "Bridge",
        "Checkers",
        "Chess",
        "Poker",
        "Fighter Combat",
        "Guerrilla Engagement",
        "Desert Warfare",
        "Air-To-Ground Actions",
        "Theaterwide Tactical Warfare",
        "Theaterwide Biotoxic and Chemical Warfare",
        "Global Thermonuclear War"
    ],
    "hiddengames": [
        "Tic-Tac-Toe"
    ]
}
```

Applying the Security Checklist: Example #2

- Loans

  https://api-blahblah.com/almaws/v1/users/1004378/loans?user_id_type=all_unique&limit=10&offset=0&order_by=due_date&apikey=trfKkutTj3ljsXfZyEHB4D

- Does the request:
  - Contain any personal info?
  - Contain secrets like an API key?
  - Return any of the above?

YES, Yes, and YES!

Applying the Security Checklist: Example #3

- Bibs

  https://api-blahblah.com/almaws/v1/bibs/235487

- Does the request:
  - Contain any personal info?
  - Contain secrets like an API key?
  - Return any of the above?

No, No, and ___


```
{
    "mms_id": "991006699759703691",
    "record_format": "marc21",
    "linked_record_id": {
        "value": "9910041082403692",
        "type": "NZ"
    },
    "title": "American folklore & the historian",
    "author": "Dorson, Richard M.",
    "issn": null,
    "isbn": "0226158683",
    "network_number": [
        ".b13584923",
        "478949",
        "(OCoLC)ocm00211518",
        "(MnSST)b13584923-01clic_stthomas",
        "(EXLNZ-01CLIC_NETWORK)9910041082403692"
    ],
    "place_of_publication": "Chicago,",
    "date_of_publication": "[1971]",
    "publisher_const": "University of Chicago Press",
    "holdings": {
        "value": null,
        "link": "https://api-na.hosted.exlibrisgroup.com/alma
    },
    "created_by": "import",
    "created_date": "2017-05-23Z",
    "last_modified_by": "system",
    "last_modified_date": "2018-05-18Z",
    "suppress_from_publishing": "false",
    "originating_system": "ILS",
    "originating_system_id": "b13584923-01clic_stthomas",
    "cataloging_level": {
        "value": "00",
        "desc": "Default Level"
    },
```

Applying the Security Checklist: Example #3

- Bibs

  https://api-blahblah.com/almaws/v1/bibs/235487**/loans**

- Does the request:
  - Contain any personal info?
  - Contain secrets like an API key?
  - Return any of the above?

No, No, and **YES!**

API Best Practices

- Never reveal secrets
- Never place secrets in the client's hands (JavaScript in browser, URLs, POST requests, compiled code in a mobile app)
  - Definitely never hard code them!
  - Two words: View Source
- Anything in the client's hands is able to be reversed engineered
  - A hard lesson to learn, many app devs still do it and it comes back to bite them!
- Revealing a key to someone makes them look at it and think, "Huh, I wonder what else this opens."

The problem with the Ex Libris API

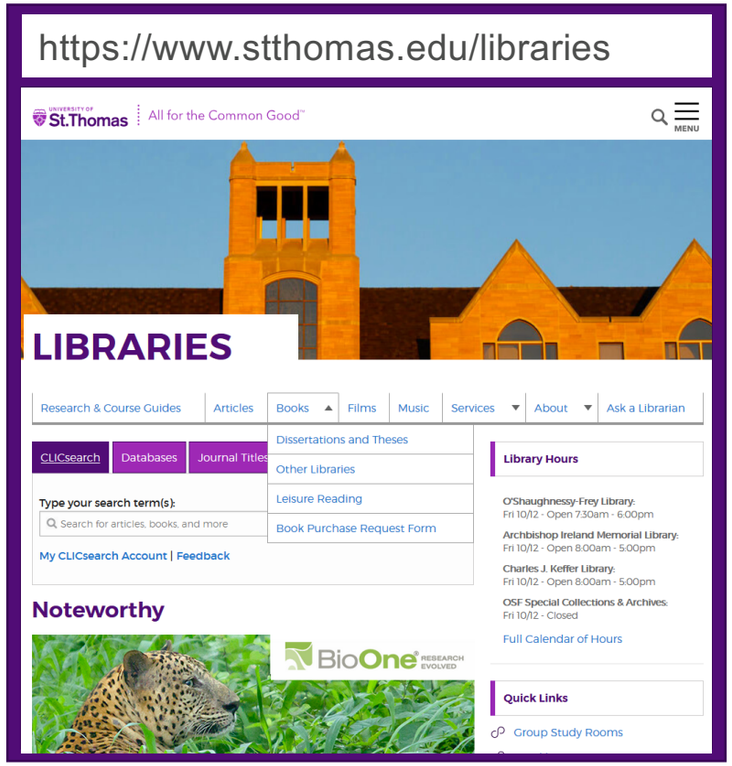
- It's not really a problem
- The limitations it imposes actually increases its flexibility and usefulness
- It is API key based and therefore meant to be used server to server
- It is a key that opens many doors, even ones you don't consider
- An API key, even one provisioned for just BIBs, allows someone to poke around
- It is up to you to develop a (secure) public interface

When to use client-side API calls

- Public feeds (blogs, event calendars, new book lists)
- At St. Thomas Libraries we use JavaScript API calls for a lot of public content
  - However we typically TRANSFORM the data before passing it on to the client-side
  - We host a variety of intermediary scripts (written in PHP)
  - We also restrict which websites can request the data through the embedded scripts
  - If the requestor for the data is not from a *.stthomas.edu webpage or a pre-approved domain, we don’t honor the API request.

Creating an intermediary

Displaying a list of courses we have in Course Reserves is pretty safe, right?
But we can't just put `GET /almaws/v1/courses?apikey=xxxx` out there in JavaScript



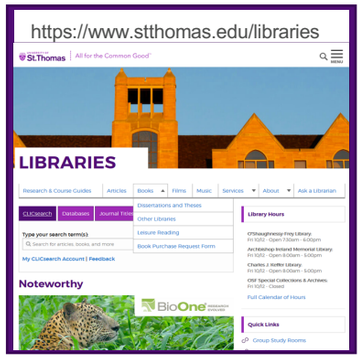

```
<script>
    var display = function (apiRequest) {
        /* some code that makes the request
        and then formats it in HTML for
        display on page */
    }
    // call the display func with API url
    display("https://lib-
api.stthomas.edu/courses?list=all");
</script>
```

We'll host an intermediary script on our application server

- `https://libraries-api.stthomas.edu/courses?list=all`
- `/courses/index.php` is a server-side script that holds the API key and only returns pre-defined (non-sensitive) data fields suitable for public consumption.
- The server-side script also checks to make sure the request was made from a St. Thomas website (don't confuse this with IP Address checking!)
  - Why check for a *.stthomas.edu address? So that our server doesn't respond to blanket requests that didn't originate from a web page hosted by us.
  - Responding to illegitimate requests (even for public information) uses bandwidth and server processing time.

The Intermediary

It has tunnel vision, it doesn't know about the other API requests that are possible. Doesn't even think about poking around the other APIs it has access to. It is a very trustworthy staff member.



`?list=all`

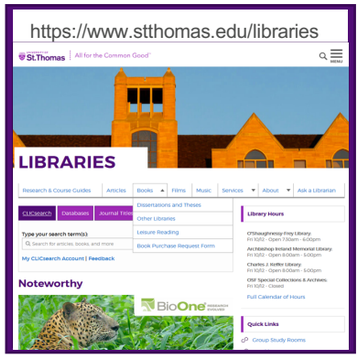
`https://libraries-api.stthomas.edu/courses`

```
// API Script Pseudo Code

$params = getParams();
$url = "/ almaws/ v1/ courses?$params&apikey=$apikey";
callExLibrisAPI(url);
generateResponse();
```

The “backend” call

https://libraries-api.stthomas.edu/courses



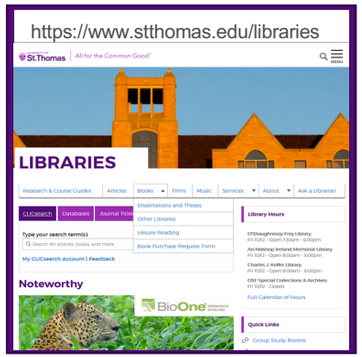
St. Thomas Intermediary API

/almaws/v1/courses?apikey=xxxx

courses{…}

Alma API

The intermediary passes back “clean” data



https://libraries-api.stthomas.edu/courses

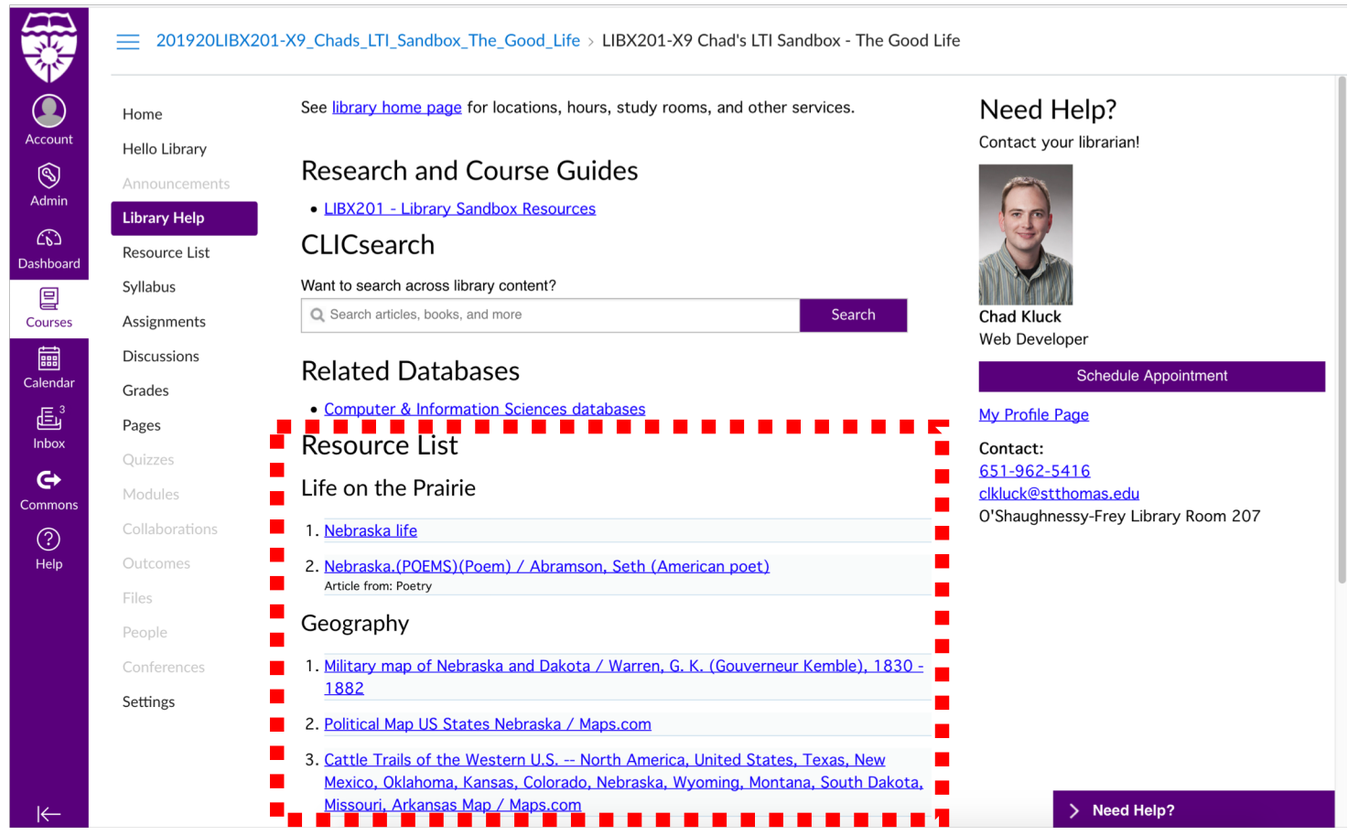
```
[  {"title": "Geology 101",
   "code": "GEOG101"},
   {"title": "Biology 203",
   "code": "BIOL203"}
]
```

```
// API Script Pseudo Code

$params = getParams();
$url = “/ almaws/ v1/ courses?$params&apikey=$apikey”;
callExLibrisAPI(url);
generateResponse();
```

Reading Lists in our campus LMS (Canvas)

- We use the Alma API to display Reading Lists for a given course in Canvas
- This page uses an API service we developed for Library Help
- The API is accessible by any of our campus applications
- Imagine not only having the Reading List in Canvas, but also on a personalized library page for each student in our campus portal



See library home page for locations, hours, study rooms, and other services.

Research and Course Guides

- LIBX201 - Library Sandbox Resources

CLICsearch

Want to search across library content?

Search articles, books, and more | Search

Related Databases

- Computer & Information Sciences databases

Resource List

Life on the Prairie

1. Nebraska life
2. Nebraska.(POEMS)(Poem) / Abramson, Seth (American poet)
   Article from: Poetry

Geography

1. Military map of Nebraska and Dakota / Warren, G. K. (Gouverneur Kemble), 1830 - 1882
2. Political Map US States Nebraska / Maps.com
3. Cattle Trails of the Western U.S. -- North America, United States, Texas, New Mexico, Oklahoma, Kansas, Colorado, Nebraska, Wyoming, Montana, South Dakota, Missouri, Arkansas Map / Maps.com

Need Help?

Contact your librarian!

Chad Kluck
Web Developer

Schedule Appointment

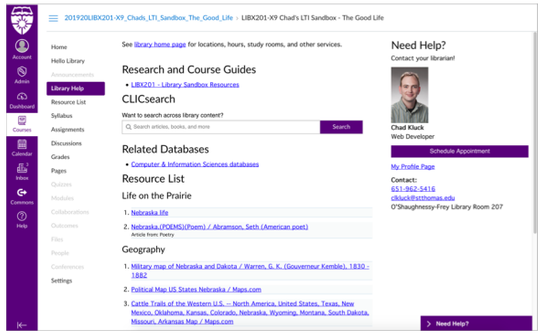
My Profile Page

Contact:
651-962-5416
clkluck@stthomas.edu
O'Shaughnessy-Frey Library Room 207

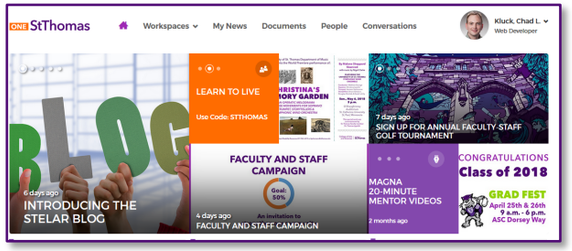
Remember how I said the UI is separate from Data?

- The logic involved in searching for and displaying Alma reading lists is very complex, a ruleset of about a dozen IF statements
- Plus there are “don’t show before” and “Don’t show after” dates for many sections and items
- Instead of coding this search and display logic for each application (University portal, Canvas, campus mobile app) we code the logic once and serve it up as an API
- Therefore our App/UI only needs to know the course and the Intermediary API takes care of the logic behind the search, data, and formatting!

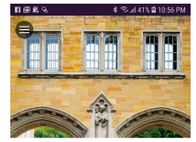
User Interfaces

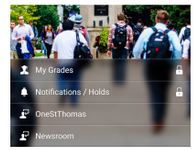


canvas.stthomas.edu



one.stthomas.edu





Mobile app

PHP (or Python or anything) hosted on a Web Server (LAMP stack) or serverless environment

libraries-api.stthomas.edu

ExLibris
a ProQuest Company

api-na.hosted.exlibrisgroup.com

Wow. That's a lot of coding. Sigh.

- "Chad, you made me sit through this presentation thinking I would walk away with something I could actually use."
- Yes, but I have some good news…


A Framework
& Other Goodies from GitHub

But you'll need a place to host your code

- If you can't have your own IT group provision a server, there are other options (as long as you are allowed to host your code elsewhere)
  - Web Hosting Service w/ a LAMP stack (Linux, Apache, MySQL, PHP/Py/Perl)
  - Amazon Web Services (AWS) Lambda
  - Microsoft Azure Functions
  - Google Functions
- Read more on this blog post by Asaf Yigal: AWS Lambda vs. Azure Functions vs. Google Functions
  - https://logz.io/blog/serverless-guide

Load Times

- Be careful! API requests take time and can delay loading of a web page
  - That's why we use JavaScript where possible
  - Page loads and when API calls are done the JavaScript updates the page

GitHub

- There's a lot out there
- But these will get you coding for now:
  - github.com/USTLibraries
  - github.com/chadkluck

Framework

- Here's something to get you started:
  - https://github.com/ustlibraries/exlibris-api-example
- Based on a framework developed to quickly deploy APIs and microsites:
  - https://github.com/chadkluck/php-project-framework
- All you need to do is:
  - Set up a few lines in the custom config file (CORS request)
  - And plop your code into the generateResponse() function consisting of:
    - An API request to Ex Libris
    - Running through/looping through the data
    - Generating an array of data suitable to return

JavaScript Template

- Already has code to make API calls
- Place your code in the execute() function
- You're ready to roll
- https://github.com/chadkluck/js-template

- Example: 8 Ball
  - https://diversion.chadkluck.net/8ball
  - https://github.com/chadkluck/8ball-api (the api service)
  - https://github.com/chadkluck/8ball-js (the JavaScript client/UI)

A final piece of code

The code for api.chadkluck.net/eluna:

- https://github.com/ustlibraries/eluna-2019

What next?

- Even if you can't set up a server environment, get familiar and play
- Play with Postman, Ex Libris API Console, your own APIs
- Understand what is available, and what you can do
  - Start small, remember I just handed 3 API requests over to our university developers
- By this time next year: Proposition for a place to host your APIs
  - I gave you everything you need to get started

Thank You!
Questions?

clkluck@stthomas.edu
github.com/USTlibraries
github.com/chadkluck

Video of Presentation Slides with Audio

- I recorded the presentation slides along with: https://chadkluck.net/eluna-2019
- This ELUNA presentation was based on/extends the presentation I gave at UMWUG (Upper Midwest User Group) Fall: https://chadkluck.net/umwug-2018
- More on the Library Help LTI: “Drag, Drop, Done, Implementing Library Help LTI” presentation given at LibTech 2019: https://chadkluck.net/libtech-2019


BONUS MATERIAL

PHP – Return an array as JSON

```
// ------------------------------------------------
// PHP - Return an array as JSON

$data = array();
$data[] = "Hello World"; // put data in the array
$cache = 60; //in seconds
$origin = "https://www.yoursitedomain.com"; // CORS allowed
origin

$ts = gmdate("D, d M Y H:i:s", time() + $cache) . " GMT";
header("Expires: ".$ts);
header("Pragma: cache");
header("Cache-Control: max-age=".$cache);
header("Access-Control-Allow-Origin: ".$origin); // CORS
header("Content-type: application/json");
echo json_encode($array);
```

PHP – Request JSON and turn into array

```
// ------------------------------------------------------
// PHP - Request json data and turn into array

function getDataFromJSON($url) {
	$results = array();
	try {
		$contents = @file_get_contents($url); // @ means suppress warnings
		$results = json_decode($contents, true);
	} catch (Exception $e) {
		echo "Caught exception: ". $e->getMessage();
	}
	return $results;
}

$apikey = ""; // enter an API key here - needs access to Courses
$url = "https://api-
na.hosted.exlibrisgroup.com/almaws/v1/courses?limit=10&offset=0&order_by=code%2Cse
ction&direction=ASC&format=json&apikey=".$apikey; // change this to whatever you
want, be sure the key has access
$data = getDataFromJSON($url);
```


Loans, Fines, Fees

User Loans, Requests, and Fines for university portal – Part 1

- First, determine if the user has any loans, fines and requests.
- Remember we want to be good stewards of API requests. Why make 3 calls if we can just make 1 and determine the user doesn't have anything to display?
- Of course, at most we'll make 4 calls, but chances are we won't.
- Call #1: Get User Info:

```
GET https://api-na.hosted.exlibrisgroup.com/almaws/v1/users/
{{user_id}}?user_id_type=all_unique&view=full
&expand=loans,fees,requests&apikey={{apikey}}&format=json
```

User Loans, Requests, and Fines for university portal – Part 2

- In the response you'll find something like this:


```
"requests": {
        "value": 0,
        "link": "https://api-na.hosted.exlibrisgroup.com/almaws/v1/users/1065237/requests"
    },
    "loans": {
        "value": 3,
        "link": "https://api-na.hosted.exlibrisgroup.com/almaws/v1/users/1065237/loans"
    },
    "fees": {
        "value": 0,
        "link": "https://api-na.hosted.exlibrisgroup.com/almaws/v1/users/1065237/fees"
    },
```


- This user only has loans, so we end up making 2 requests instead of 3

User Loans, Requests, and Fines for university portal – Part 3

- Loans

  ```
  https://api-na.hosted.exlibrisgroup.com/almaws/v1/users/{{user_id}}/loans?
  user_id_type=all_unique&limit=10&offset=0&order_by=due_date&format=json&direction=ASC
  &apikey={{apikey}}
  ```

- Holds/Requests

  ```
  https://api-na.hosted.exlibrisgroup.com/almaws/v1/users/{{user_id}}/requests?
  request_type=HOLD&user_id_type=all_unique&limit=10&offset=0&status=active&format=json
  &apikey={{apikey}}
  ```

- Fines

  ```
  https://api-na.hosted.exlibrisgroup.com/almaws/v1/users/{{user_id}}/fees?
  user_id_type=all_unique&status=ACTIVE&format=json&apikey={{apikey}}
  ```


Search for, and request an item

Bonus API Calls: Step 1

- Try these out in Postman, but here's a sequence to get you on your way to a book request service (well, once the user is authenticated—there is that)
- Each of the following leads to the next
- First a user performs a search query:

```
https://api-na.hosted.exlibrisgroup.com/primo/v1/pnxs?q={{query}}&lang=eng&offset=1&limit=10&view=full&vid=STTHOMAS&scope=stthomas&apikey={{apikey}}
```

BNS

Bonus API Calls: Step 2

- When a user selects an item we get the BIB:

  https://api-na.hosted.exlibrisgroup.com/almaws/v1/bibs/?view=full&expand=p_avail,e_avail,d_avail&nz_mms_id={{nz_mms_id}}&format={{format}}&apikey={{apikey}}

- Holding Link:

  https://api-na.hosted.exlibrisgroup.com/almaws/v1/bibs/{{mms_id}}/holdings?format={{format}}&apikey={{apikey}}

- Get PID:

  https://api-na.hosted.exlibrisgroup.com/almaws/v1/bibs/{{mms_id}}/holdings/{{holding_id}}/items/?format={{format}}&apikey={{apikey}}

BNS

Bonus API Calls: Step 3

- We make the request for the user:

```
https://api-na.hosted.exlibrisgroup.com/almaws/v1/bibs/
{{mms_id}}/holdings/{{holding_id}}/items/{{item_pid}}/requests?
user_id={{user_id}}&format={{format}}&apikey={{apikey}}
```

BNS

Import invoices into finance system

Import invoices into finance system

- This spring we partnered with our university budget and finance group to automate the import of invoices from Alma into our university finance system.
- Easy, right?

The 3 APIs for finance

Get Fund Codes:

```
/almaws/v1/acq/funds?apikey={{apikey}}&view=full&limit=100&offset=0
```

Get Active Invoices:

```
/almaws/v1/acq/invoices/?apikey={{apikey}}&base_status=ACTIVE
&limit=100&view=full&offset=0
```

Get Each Vendor (based on invoices):

```
/almaws/v1/acq/vendors/{{vendorCode}}?apikey={{apikey}}
```

Piece of cake! Right?

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<funds total_record_count="13">
    <fund link="https://api-na.hosted.exlibrisgroup.com/almaws/v1/acq/funds/1332706870000521">
        <id>1332706870000521</id>
        <code>EHum</code>
        <name>E: Humanities</name>
        <external_id>12345-743</external_id>
        <type desc="General">General</type>
        <status desc="Active">ACTIVE</status>
        <fiscal_period desc="2018/06/30 - 2019/06/29">41</fiscal_period>
        <currency desc="US Dollar">USD</currency>
        <available_balance>92123.00</available_balance>
    </fund>
    <fund link="https://api-na.hosted.exlibrisgroup.com/
        <id>1332706790000521</id>
        <code>ESCI</code>
        <name>E: Sciences</name>
        <external_id>12345-744 </external_id>
        <status desc="Active">ACTIVE</status>
        <fiscal_period desc="2018/06/30 - 2019/06/29">41</fiscal_period>
        <currency desc="US Dollar">USD</currency>
        <available_balance>82115.27</available_balance>
    </fund>
    <fund link="https://api-na.hosted.exlibrisgroup.com/almaws/v1/acq/funds/133270671000052
        <id>1332706710000521</id>
        <code>ESS</code>
        <name>E: Social Sciences</name>
```

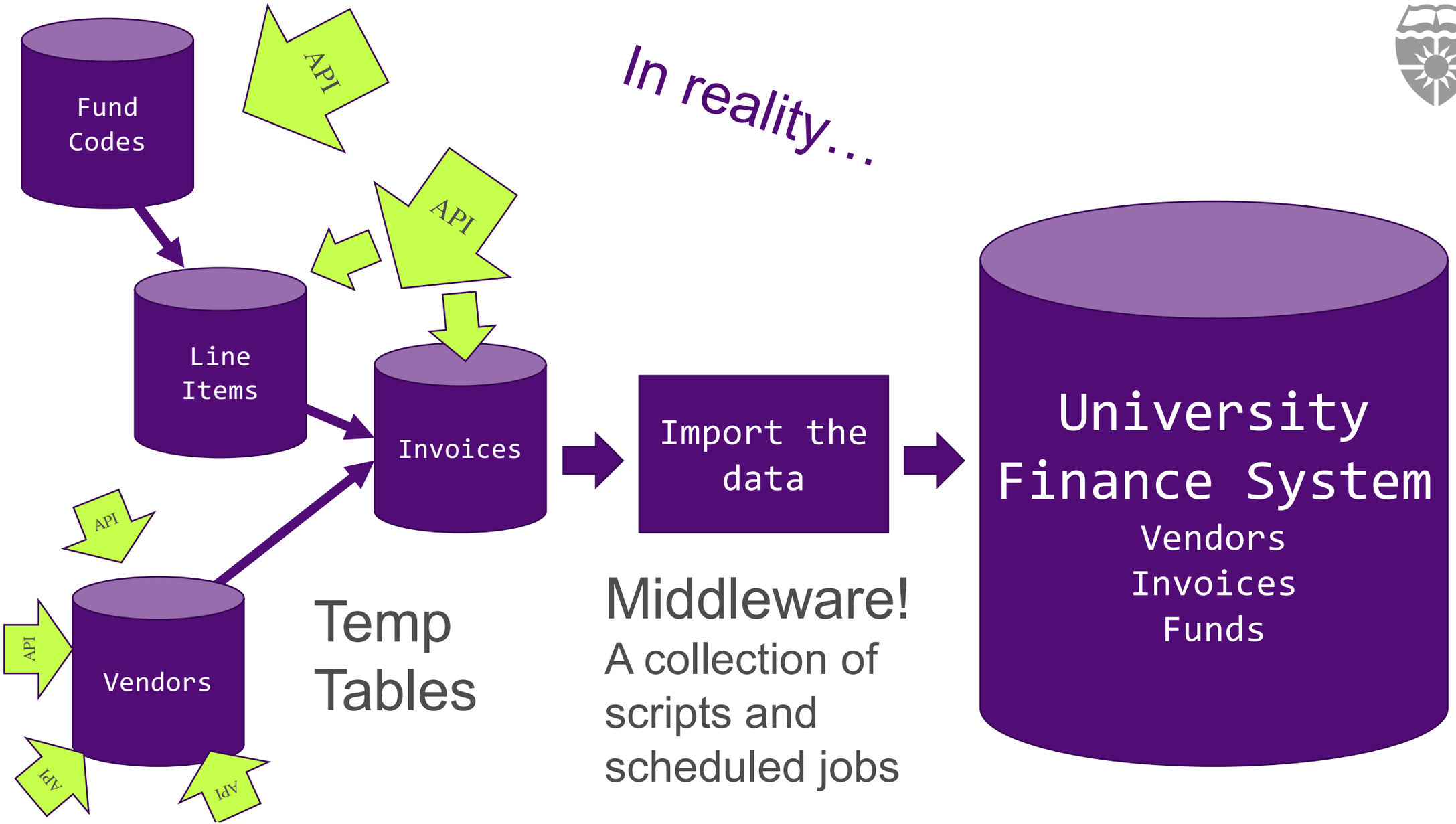
API

University Finance System

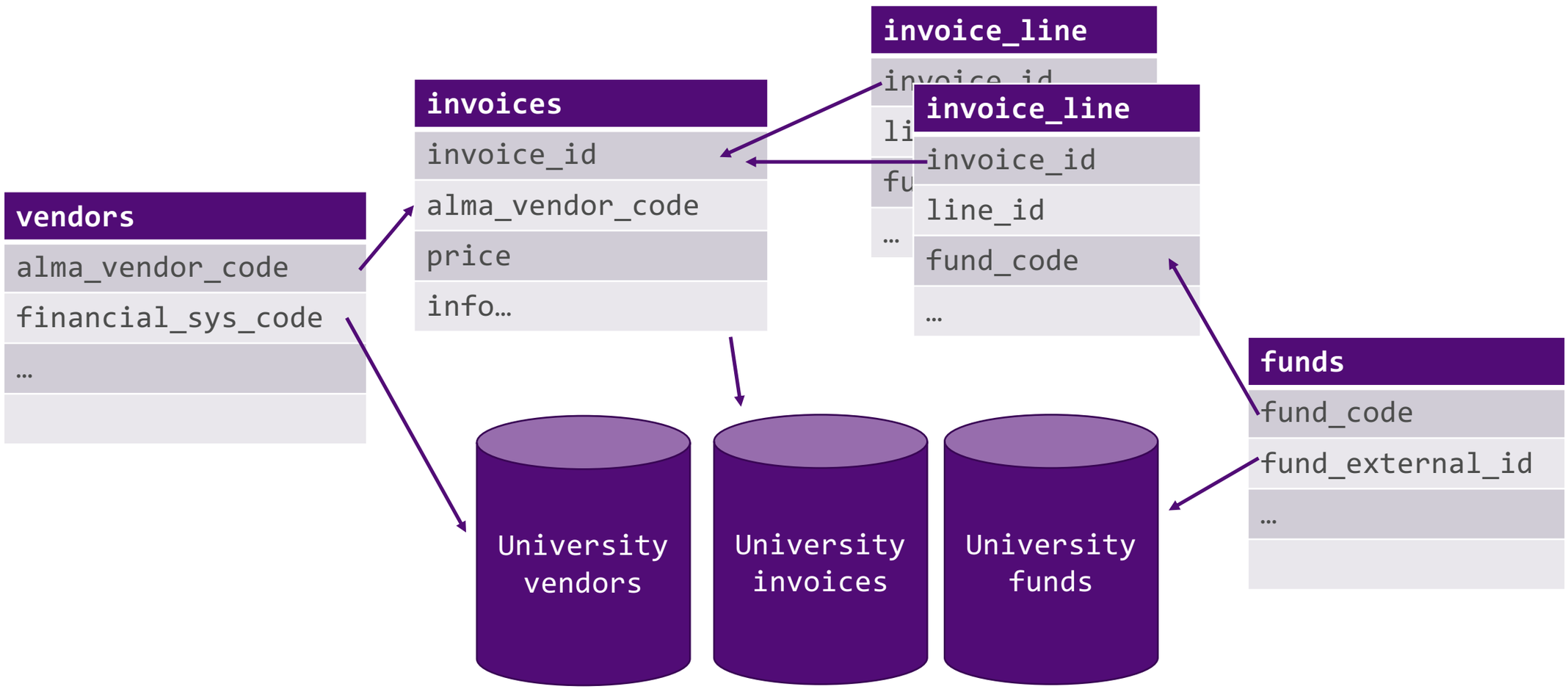
Vendors

Invoices

Funds


In reality…
Fund Codes
API
API
Line Items
Invoices
API
API
API
API
Vendors
Temp Tables
Import the data
Middleware!
A collection of scripts and scheduled jobs
University Finance System
Vendors
Invoices
Funds

Because…


invoice_line
invoice_id
li
fu
…
invoices
invoice_id
alma_vendor_code
price
info…
invoice_line
invoice_id
line_id
fund_code
…
vendors
alma_vendor_code
financial_sys_code
…
funds
fund_code
fund_external_id
…
University vendors
University invoices
University funds

Import invoices into finance system

- You need to find the common IDs/Codes that make your Alma data link to your University IDs. Such as Fund Codes and Vendor Codes.
- …And you may need to use tables or mapped arrays to make things relational (temporarily at least) as you churn the data.
- Depending on how your university likes to ingest data, your mileage will vary. Everyone has different batch automation systems… even within the same university.
- Our finance group likes XML and Tables.
- Though it uses Acquisitions, it can be used as a guide to create any relational view.

Import invoices into finance system

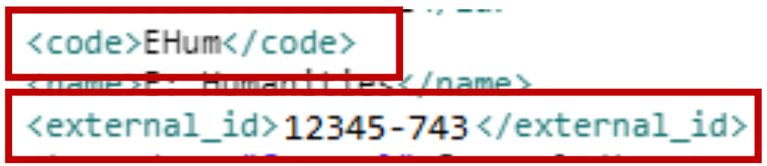
- I will only provide logic, API URLs, and pseudo-code here.
- First we needed to clean up our Alma data
- Each vendor needed a Financial System Code that would correspond to what is in the university system.
- Plus we also needed to provide each Fund Code with a code that would match the university system.
- So Vendors and Funds have two identifiers: the Alma Identifier, and the University Identifier.

Step 1: Get the fund codes

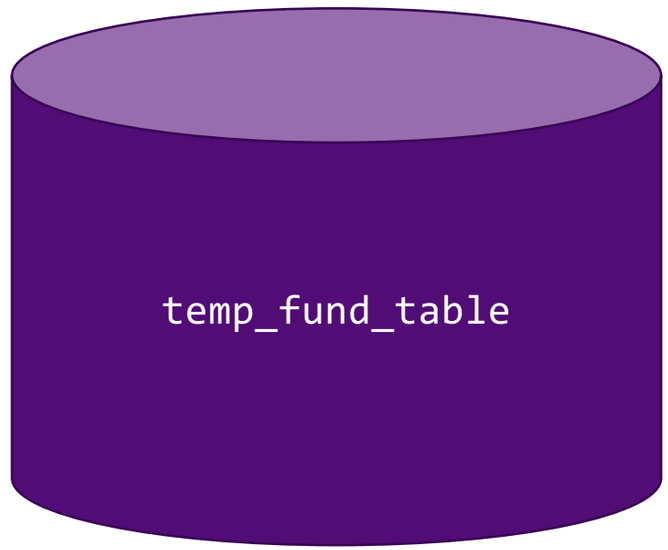
- Our finance development group likes working with tables (and XML) so the first step was to populate a temporary (or refreshable) fund code mapping table using the Funds API.
- This will map the Alma fund code to our Finance System Fund Code.
- Note that we can only retrieve up to 100 codes at a time.
- A loop was created to grab 1-100, then 101-200, etc.

```
offset = 0;
do {
    myData = GET “https://api-na.hosted.exlibrisgroup.com/almaws/v1/acq/funds
        ?apikey={{apikey}}&view=full&limit=100&offset={{offset}}”;
    addToTempFundable(myData);
    offset += 100;
} while (numOfFundsReturned == 100);
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<funds total_record_count="13">
    <fund link="https://api-na.hosted.exlibrisgroup.com/almaws/v1/acq/funds/1332706870000521">
        <id>1332706870000521</id>
        <code>EHum</code>
        <name>E: Humanities</name>
        <external_id>12345-743</external_id>
        <type desc="General">General</type>
        <status desc="Active">ACTIVE</status>
        <fiscal_period desc="2018/06/30 - 2019/06/29">41</fiscal_period>
        <currency desc="US Dollar">USD</currency>
        <available_balance>92123.00</available_balance>
    </fund>
    <fund link="https://api-na.hosted.exlibrisgroup.com/almaws/v1/acq/funds/1332706790000521">
        <id>1332706790000521</id>
        <code>ESCI</code>
        <name>E: Sciences</name>
        <external_id>12345-744 </external_id>
        <status desc="Active">ACTIVE</status>
        <fiscal_period desc="2018/06/30 - 2019/06/29">41</fiscal_period>
        <currency desc="US Dollar">USD</currency>
        <available_balance>82115.27</available_balance>
    </fund>
    <fund link="https://api-na.hosted.exlibrisgroup.com/almaws/v1/acq/funds/1332706710000521">
        <id>1332706710000521</id>
        <code>ESS</code>
        <name>E: Social Sciences</name>
```



alma_code	external_id
ESci	12345-744
ERel	12345-732
EHum	12245-743



Step 2: Get the ACTIVE invoices

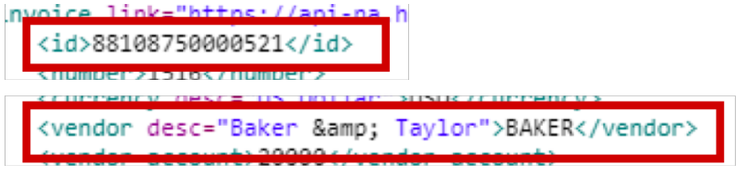
- Now we populate a temporary Invoice table using the Invoice API.
- Again note that each call will only return up to 100 invoices.

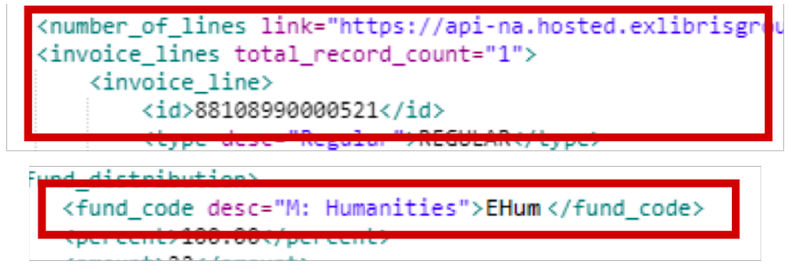
```
offset = 0;
do {
    myData = GET “https://api-na.hosted.exlibrisgroup.com/almaws/v1/acq/invoices/
        ?apikey={{apikey}}&base_status=ACTIVE&limit=100&view=brief&
        offset={{offset}}”;
    addToTempInvoiceTable(myData);
    offset += 100;
} while (numOfInvoicesReturned == 100);
```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<invoices total_record_count="152">
<invoice link="https://api-na.hosted.exlibrisgroup.com/almaws/v1/acq/invoices/88108750000521">
<id>88108750000521</id>
<number>1516</number>
<invoice_date>2014-03-04Z</invoice_date>
<total_amount>14</total_amount>
<total_invoice_lines_amount>14.00</total_invoice_lines_amount>
<currency desc="US Dollar">USD</currency>
<vendor desc="Baker & Taylor">BAKER</vendor>
<vendor_account>20000</vendor_account>
<vendor_contact_person/>
<payment_method desc="Accounting Department">ACCOUNTINGDEPARTMENT</payment_method>
<creation_form desc="Manually">MANUALLY</creation_form>
<owner desc="API Guest Institution">API_GUEST_INST</owner>
<invoice_status desc="Active">ACTIVE</invoice_status>
<invoice_workflow_status desc="In Approval">InApproval</invoice_workflow_status>
<invoice_approval_status desc="Approved">APPROVED</invoice_approval_status>
<number_of_lines link="https://api-na.hosted.exlibrisgroup.com/almaws/v1/acq/invoices/88108920000521/lines">1</number_of_lines>
<invoice_lines total_record_count="1">
<invoice_line>
<id>88108990000521</id>
<type desc="Regular">REGULAR</type>
<number>1</number>
<status desc="Ready">READY</status>
<po_line>POL-5251</po_line>
<price>22</price>
<total_price>22.00</total_price>
<quantity>1</quantity>
<vat_note>Approximately 0.00 included in line Total Price.</vat_note>
<check_subscription_date_overlap>true</check_subscription_date_overlap>
<fully_invoiced>true</fully_invoiced>
<additional_info></additional_info>
<release_remaining_encumbrance>false</release_remaining_encumbrance>
<reporting_code>Null</reporting_code>
<secondary_reporting_code>Null</secondary_reporting_code>
<tertiary_reporting_code>Null</tertiary_reporting_code>
<invoice_line_vat>
<vat_code></vat_code>
<percentage>0.0</percentage>
<vat_amount>0</vat_amount>
</invoice_line_vat>
<fund_distributions>
<fund_distribution>
<fund_code desc="M: Humanities">EHum</fund_code>
<percent>100.00</percent>
<amount>22</amount>
</fund_distribution>
</fund_distributions>
</invoice_line>
</invoice_lines>
</invoice>

There could be more than one line item per invoice

There could be more than one fund distribution per line item





invoice_id	alma_vendor_code	line_id	fund_code
88108560000521	WTCOX	88108730000521	EHum
88108560000521	WTCOX	88108730000521	ERel
88108750000521	BAKER	88108990000521	EHum

temp_line_table

invoice_id	alma_vendor_code	amt
88108750000521	WTCOX	…
88108750000521	BAKER	…

temp_invoice_table

Vendors

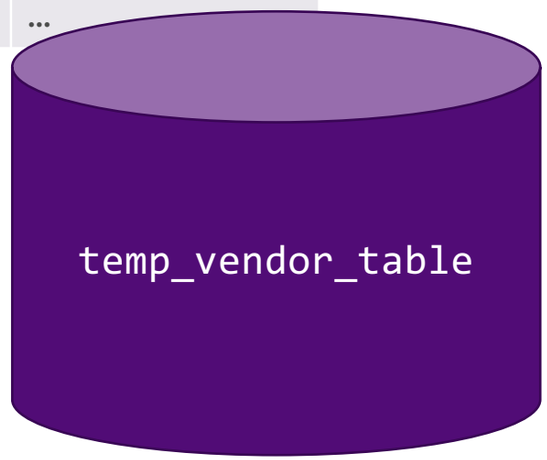
- Using the Invoice table, we can reference the list of vendors we need to get data on.
- We need to make 1 call for each vendor, but not necessarily each invoice.
- We could have 3 invoices for one vendor so we want to group first.

```
vendorList = removeDuplicates(getAllInvoiceVendors()); // only need to access each vendor once
foreach (vendorList as vendor) {
    vendorData = GET “https://api-na.hosted.exlibrisgroup.com/almaws/v1/
        acq/vendors/{{vendor.alma_vendor_code}}?apikey={{apikey}}”;
    addToTempVendorTable(vendorData);
}
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<vendor>
    <code>BAKER</code>
    <name>Baker &amp;amp; Taylor</name>
    <financial_sys_code>101098689</financial_sys_code>
    <status desc="Active">ACTIVE</status>
    <liable_for_vat>false</liable_for_vat>
    <language desc="English">en</language>
    <material_supplier>true</material_supplier>
    <access_provider>false</access_provider>
    <licensor>false</licensor>
    <governmental>false</governmental>
    <vendor_libraries>
        <library include_sub_units="true">
            <code desc="API Guest Institution">API_GUEST_INST</code>
        </library>
    </vendor_libraries>
    <vendor_currencies>
        <currency desc="US Dollar">USD</currency>
        <currency desc="Euro">EUR</currency>
        <currency desc="Pound Sterling">GBP</currency>
        <currency desc="Australian Dollar">AUD</currency>
        <currency desc="New Zealand Dollar">NZD</currency>
        <currency desc="Canadian Dollar">CAD</currency>
        <currency desc="Norwegian Krone">NOK</currency>
    </vendor_currencies>
```

```
<code>BAKER</code>
<name>Baker &amp;amp; Taylor</name>
<financial_sys_code>101098689</financial_sys_code>
```

alma_vendor_code	financial_sys_code	payment_addr	…
WTCOX	101098392	123 Library Lane	…
BAKER	101098689	221 Baker St.	…



Combine all the ingredients

- Take your fund codes, vendor codes, invoices, line items, and mix it all together.
- Load what you need into your finance system.
- Stick a fork in it—it's done!