

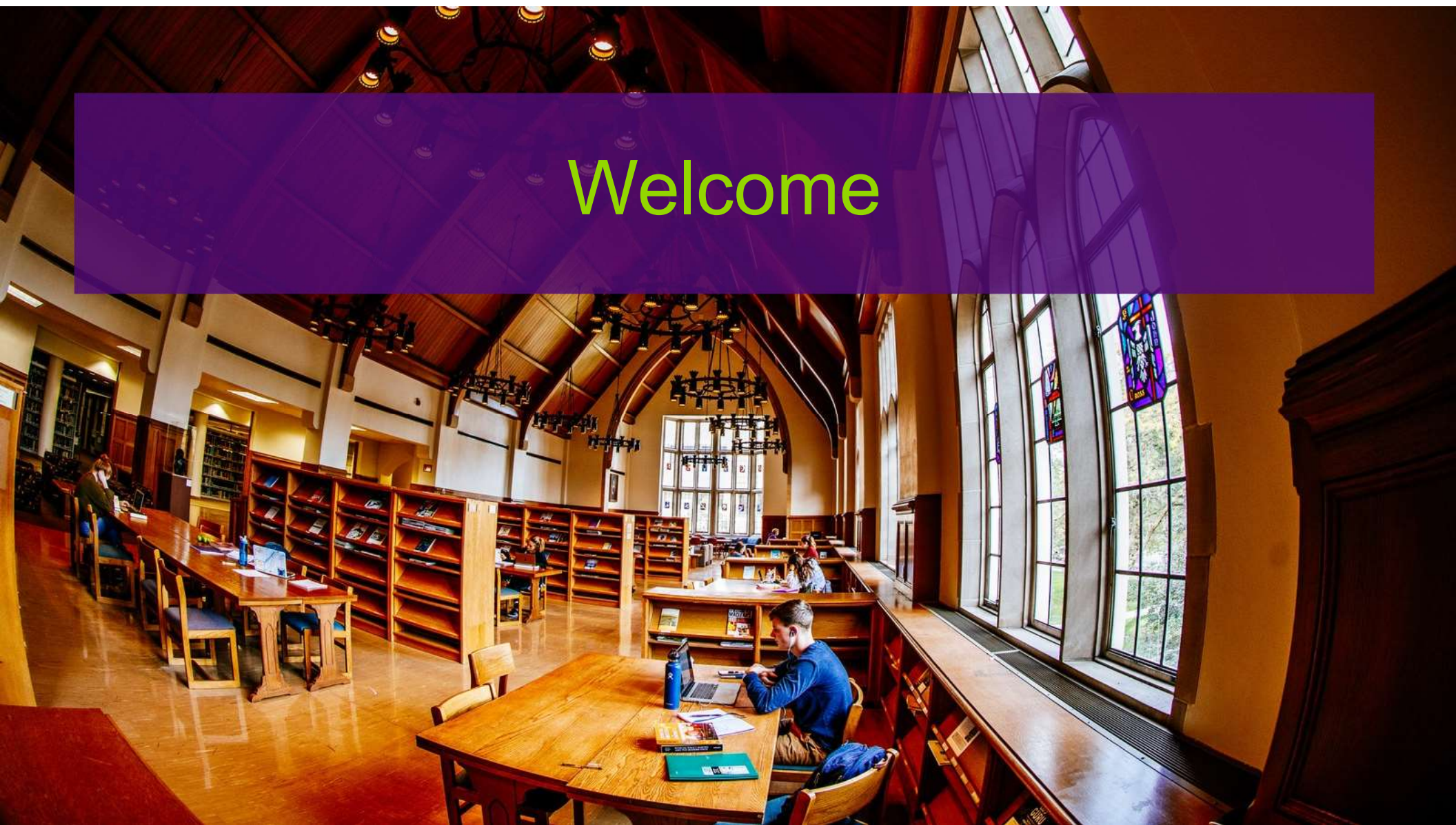
# Making APIs work for you: Creating integrations to enhance user services

Chad Kluck, Web Developer  
St. Thomas Libraries, User and Digital Services  
UMWUG: October 17, 2018

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

Presentation audio w/ slides: <https://youtu.be/eg2VVG2qCvo>

# 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.

**“Students Posing with Computer 1961”**

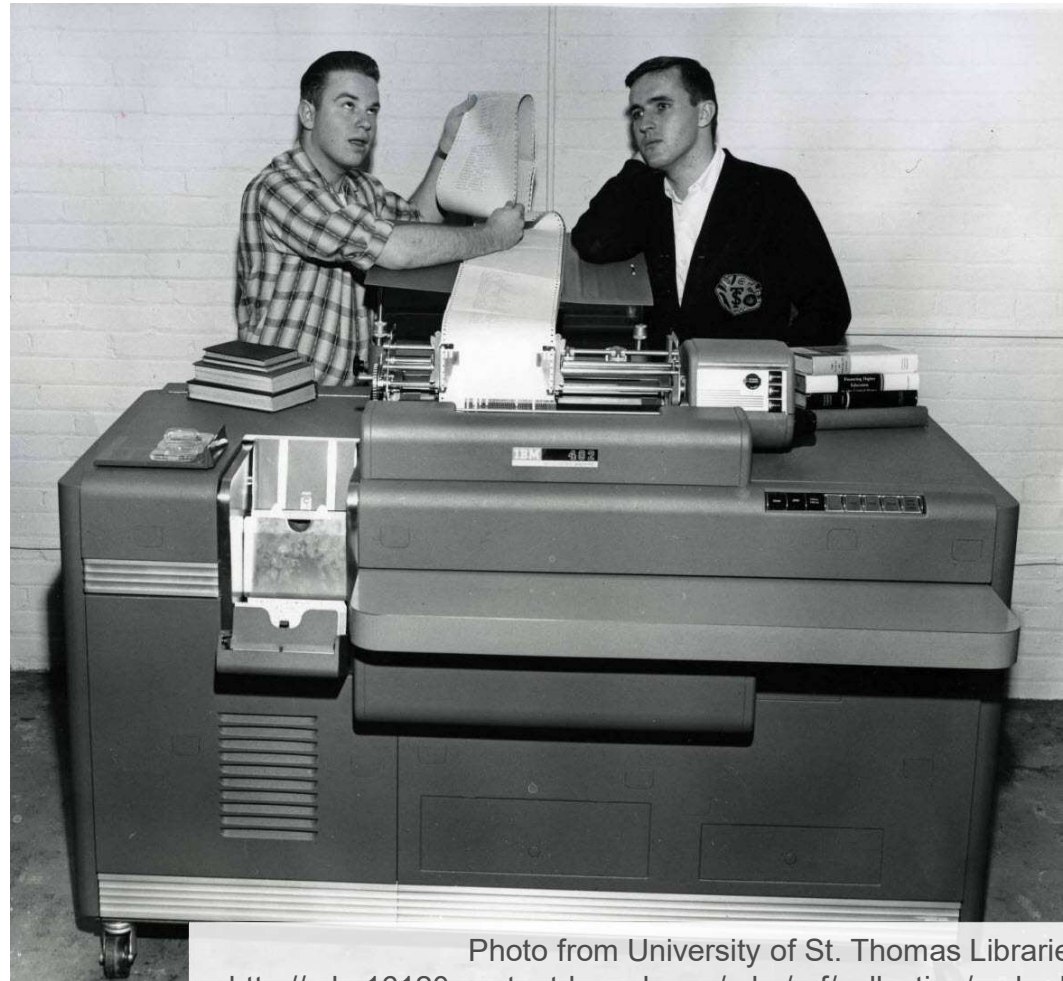


Photo from University of St. Thomas Libraries Archives  
<http://cdm16120.contentdm.oclc.org/cdm/ref/collection/arch-photo/id/903>





# Agenda

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

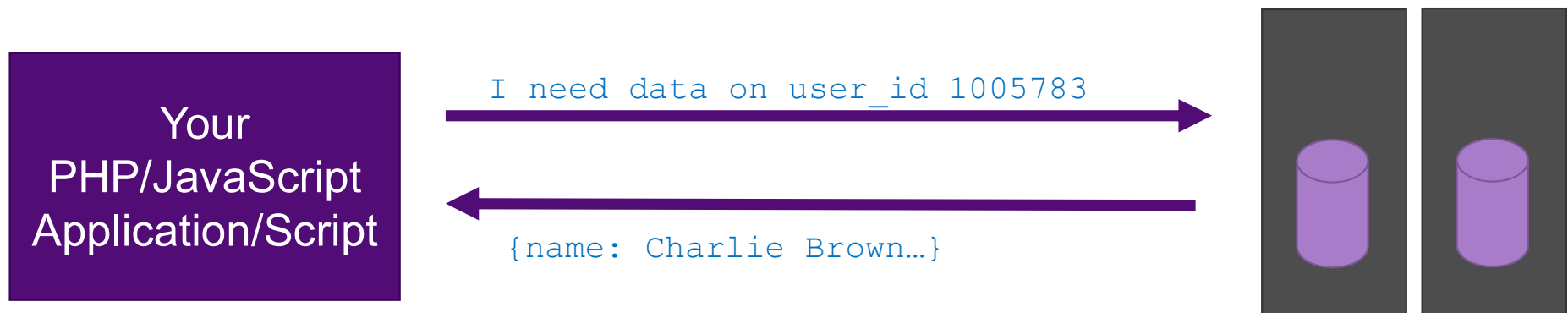
# Get Familiar: What is an API?

UNIVERSITY OF ST. THOMAS



# 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

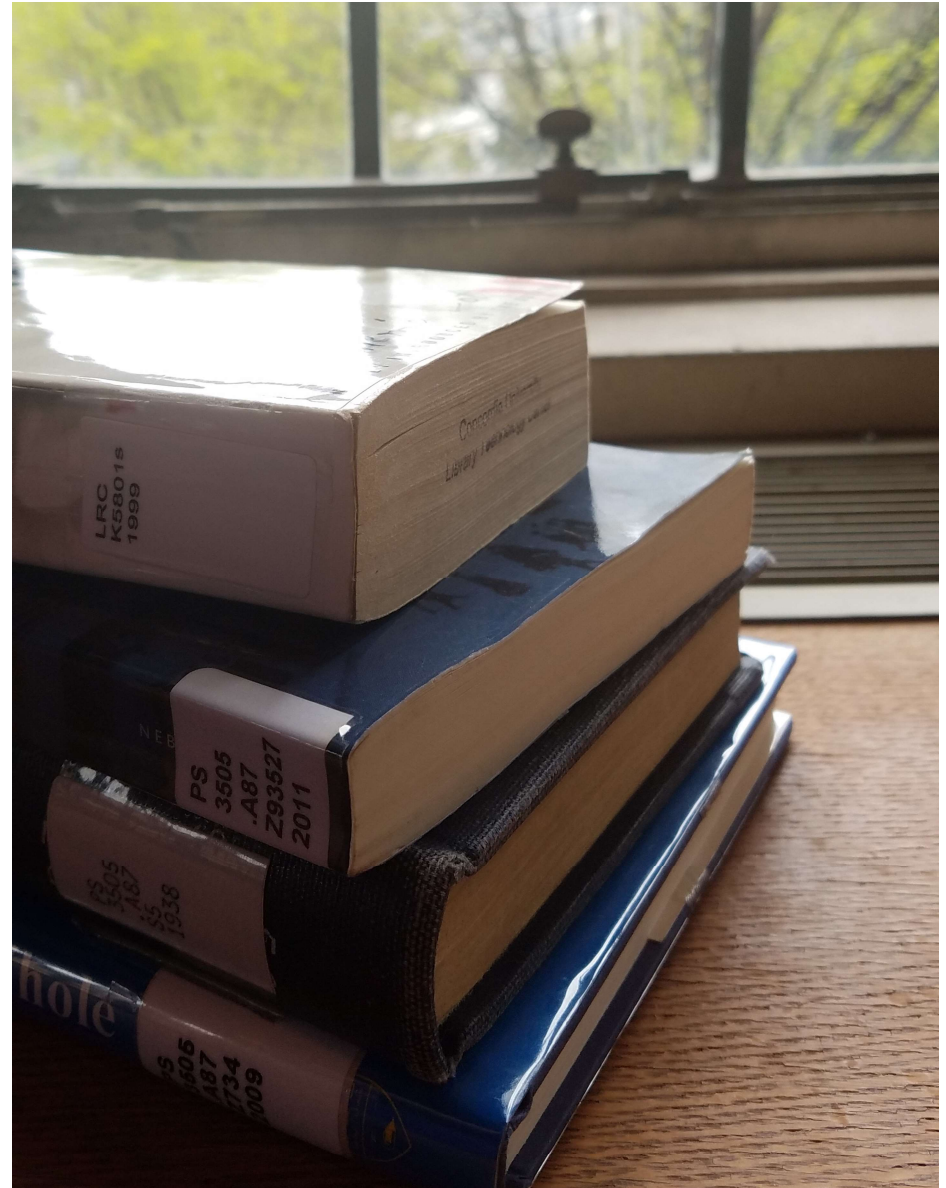




## Enhance User Services

- 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 where your patrons are in portals, on your website, and even new and bold places like voice or chat assistants
- Streamline the book request process
  - In house?
  - Consortium?
  - Interlibrary Loan?

Just let an app figure it out through a decision tree!





# I read, therefore I code!

Can you read this and visualize what it does?

```
<h1>Hello World!</h1>
<p>This is my page!</p>

<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/umwug>



## 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: JavaScript Object Notation (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/umwug?code={{3LetterCode}}
```

The 3 letter code is shown here



```
https://api.chadkluck.net/umwug?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/umwug?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/umwug?code=CPE> will get you:

```
{"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"]}
```

<https://api.chadkluck.net/umwug?code=CPE>



## Okay, but what can I do with it?

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

<div>Linus van Pelt</div>

<h2>Your Loans</h2>

<ul>

<li>Bible (KJV) is due Dec 26</li>

<li>Public Speaking is due Dec 26</li>

</ul>

**Linus van Pelt**

**Your Loans**

Bible (KJV) is due Dec 26

Public Speaking is due Dec 26

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

The screenshot shows the One St. Thomas university portal. The 'My Actions' section on the right is highlighted with a red box and a red arrow pointing to it from the top navigation bar. The 'My Actions' section includes links to Library, Library Loans, Library Hold Requests, and Library Fees, each with a notification badge indicating the number of items or requests.

<https://one.stthomas.edu>

### My Actions

#### Library

##### 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

##### 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 API code a little bird gave me and handed it over to IT



## 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 that little bird who told me?



Woodstock by Charles M. Schulz, *Peanuts*  
United Feature Syndicate Inc.



(Just Kidding)  
But seriously, I got it from the....

# Ex Libris API Center: The Tour



## 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 ExLibris Developer's Network
  - <https://developers.exlibrisgroup.com>
- Go to Applications

DEV

# Generate a key

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

ExLibris  
a ProQuest Company

Chad Kluck

Developer Network Docs Tech Blog Code & Apps Forum Dashboard

Search

Organization  
Applications  
Analytics  
My Profile

Ex Libris Developer Network / Dashboard / Applications

Display per page 10

Page 1 of 1

<input type="checkbox"/>	Application		API Key	Status	Platform
<input type="checkbox"/>	Canvas Integration		████████████████████47	Active	Web application
<input type="checkbox"/>	LibWebDev-Feeds		████████████████████8e	Active	Web application
<input type="checkbox"/>	MarcEdit		████████████████████f0	Active	Integration
<input type="checkbox"/>	Real Time Acq		████████████████████3a	Active	Integration
<input type="checkbox"/>	SpineOMatic		████████████████████ff	Active	Integration
<input type="checkbox"/>	StThomasAI		████████████████████4d	Active	Multi-purpose
<input type="checkbox"/>	USTWebDev-01		████████████████████6e		Web application

Actions Apply

Add Application

Use our [API console](#) for a birds-eye view.



# Adding an Application

The screenshot shows a dialog box titled "Adding an Application" with two tabs: "Application Information" (selected) and "API Management".

**Application Information Tab:**

- Application Name \***: A text input field containing "LibWebTest".
- Platform \***: A dropdown menu with "Web application" selected.
- Description**: A rich text editor with a toolbar (bold, italic, underline, bulleted list, numbered list, link, unlink, indent, outdent, undo, redo) and a text area containing "An API for use with testing and demos".

**Buttons:**

- Cancel**: A button on the bottom left.
- Next Step**: A button on the bottom right.
- Save**: A red button on the bottom right.



## Next Step: Add APIs

**Bibs**

The BIBs API allows access to Bibliographic records related information.

API Usage

Application Information

API Management

Current APIs \*

Add APIs

Choose an API

Acquisitions

Analytics

**Bibs**

campusM APIs

Configuration

Courses

Electronic

Next Step

Save

4d	Active	Multi-pu
----	--------	----------



## Note:

- You don't typically need a whole bunch of APIs
- For example our portal only needed "Users" as loans and fines are included
- Check documentation
- One weird thing, you've got to go back to the Application list and choose Edit. That will allow you to change "Plan" (Read-only, production, etc)

The screenshot shows a web interface with two tabs: 'Application Information' and 'API Management'. The 'API Management' tab is active. Below the tabs is a section titled 'Current APIs \*'. It contains a table with the following data:

API		Current Plan	Plan Change
Users	Unlimited Hits	Prod read-only	Change to -- Please select --



# Play

The screenshot shows the 'Applications' section of the Ex Libris Developer Network dashboard. On the left is a sidebar with navigation links: Organization, Applications (highlighted), Analytics, and My Profile. The main header includes the Ex Libris logo, 'Developer Network', and navigation tabs: Docs, Tech Blog, Code & Apps, Forum, and Dashboard. A search bar is located on the right.

Breadcrumbs indicate the current path: Ex Libris Developer Network / Dashboard / Applications. Below the breadcrumbs, there's a 'Display per page' dropdown set to 10 and a pagination indicator showing 'Page 1 of 1'.

<input type="checkbox"/>	Application		API Key	Status	Platform
<input type="checkbox"/>	Canvas Integration		[REDACTED]47	Active	Web application
<input type="checkbox"/>	LibWebDev-Feeds		[REDACTED]8e	Active	Web application
<input type="checkbox"/>	MarcEdit		[REDACTED]f0	Active	Integration
<input type="checkbox"/>	Real Time Acq		[REDACTED]3a	Active	Integration
<input type="checkbox"/>	SpineOMatic		[REDACTED]ff	Active	Integration
<input type="checkbox"/>	StThomasAI		[REDACTED]4d	Active	Multi-purpose
<input type="checkbox"/>	USTWebDev-01		[REDACTED]6e	Active	Web application

At the bottom of the table area are two buttons: 'Actions' (with a dropdown arrow) and 'Apply'. To the right of these is a red button labeled 'Add Application'.

A large purple arrow points from the bottom left towards the 'Actions' button. At the very bottom, a note states: 'Use our API console for a birds-eye view.'

# Formulate your request

- From the Explore screen:
  1. Choose the API (Users)
  2. Choose the Resource (Loans)
  3. Method (GET) – (typical)
  4. Enter a user\_id (your own)
  5. Add a parameter: format = json
  6. Choose an API Key
  7. Choose authentication (your app)
  8. Execute!

Ex Libris Developer Network / Alma / Explore

API  
1 Users

Resource  
2 /almaws/v1/users/{user\_id}/loans

Method  
3 GET

API Key  
6 LibWebTest

7 Authentication (API Key)

8! Execute Request

Request Query Response

GET https://api-eu.hosted.exlibrisgroup.com/almaws/v1/users/{user\_id}/loans

4 user\_id 1002548

user\_id\_type all\_unique

limit 10

offset 0

order\_by id

direction ASC

expand string

5 x format json

Add Parameter

# Eureka!

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

```
1. {
2.   "item_loan": {
3.     {
4.       "loan_id": "5200578480003692",
5.       "circ_desk":
6.         {
7.           "value": "DEFAULT_CIRC_DESK",
8.           "desc": "OSF Main"
9.         },
10.    "library":
11.      {
12.        "value": "IOSF",
13.        "desc": "St. Thomas O'Shaughnessy-Frey Library"
14.      },
15.    "user_id": "[REDACTED]",
16.    "item_barcode": "30516008019415",
17.    "due_date": "2019-02-08T05:59:00Z",
18.    "loan_status": "ACTIVE",
19.    "loan_date": "2018-10-10T16:33:22.063Z",
20.    "process_status": "NORMAL",
21.    "mms_id": "991006699759703691",
22.    "title": "American folklore & the historian [by
23.      ] Richard M. Dorson.",
24.    "author": "Dorson,
25.      Richard M.",
26.    "description": null,
27.    "publication_year": "[1971
28.
29.    {
30.      "value": "MAIN",
31.      "desc": "Main Collection"
```



## Query tab!

Request	Query	Response
<pre>GET https://api-eu.hosted.exlibrisgroup.com/almaws/v1/users/[REDACTED]/loans?user_id_type=all_unique&amp;limit=10&amp;offset=0&amp;order_by=id&amp;direction=ASC&amp;format=json</pre>		

Show Code Sample...

**It's a URL!**  
(Don't forget to append "&apikey=xxxx")

**But wait!**  
**There's more!**



# Python!

Request

Query

Response

GET https://api-eu.hosted.exlibrisgroup.com/almaws/v1/users/[REDACTED]/loans?user\_id\_type=all\_unique&limit=10&offset=0&order\_by=id&direction=ASC&format=json

Select Code

```
1. from urllib2 import Request, urlopen
2. from urllib import urlencode, quote_plus
3. url = 'https://api-eu.hosted.exlibrisgroup.com/almaws/v1/users/[REDACTED]/loans?user_id_type=all_unique&limit=10&offset=0&order_by=id&direction=ASC&format=json'
4. queryParams = '?' + urlencode({ quote_plus('user_id_type'): 'all_unique', 'limit': '10', 'offset': '0', 'order_by': 'id', 'direction': 'ASC', 'format': 'json' })
5. request = Request(url + queryParams)
6. request.get_method = lambda: 'GET'
7. response_body = urlopen(request).read()
8. print response_body
```

Python

Javascript

Curl

Node

Python

PHP

Ruby

Objective C

Java



# PHP!



Request	Query	Response
---------	-------	----------

GET https://api-eu.hosted.exlibrisgroup.com/almaws/v1/users/

██████████

/loans?user\_id\_type=all\_unique&limit=10&offset=0&order\_by=id&direction=ASC&format=json

PHP

Select Code

```
1. $ch = curl_init();
2. $url = 'https://api-eu.hosted.exlibrisgroup.com/almaws/v1/users/{user_id}/loans';
3. $templateParamNames = array('{user_id}');
4. $templateParamValues = array(urlencode('██████████'));
5. $url = str_replace($templateParamNames, $templateParamValues, $url);
6. $queryParams = '?' . urlencode('user_id_type') . '=' . urlencode('all_unique');
7. curl_setopt($ch, CURLOPT_URL, $url . $queryParams);
8. curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
9. curl_setopt($ch, CURLOPT_HEADER, FALSE);
10. curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'GET');
11. $response = curl_exec($ch);
12. curl_close($ch);
13.
14. var_dump($response);
```

<  >



# cURL!

Request

Query

Response

GET https://api-eu.hosted.exlibrisgroup.com/almaws/v1/users/[REDACTED]/loans?user\_id\_type=all\_unique&limit=10&offset=0&order\_by=id&direction=ASC&format=json

Select Code

1. curl --include --request GET \

2. 'https://api-eu.hosted.exlibrisgroup.com/almaws/v1/u

<

Curl

JavaScript

Curl

Node

Python

PHP

Ruby

Objective C

Java

```
1. curl --include --request GET \
```

```
2. 'https://api-eu.hosted.exlibrisgroup.com/almaws/v1/u
```

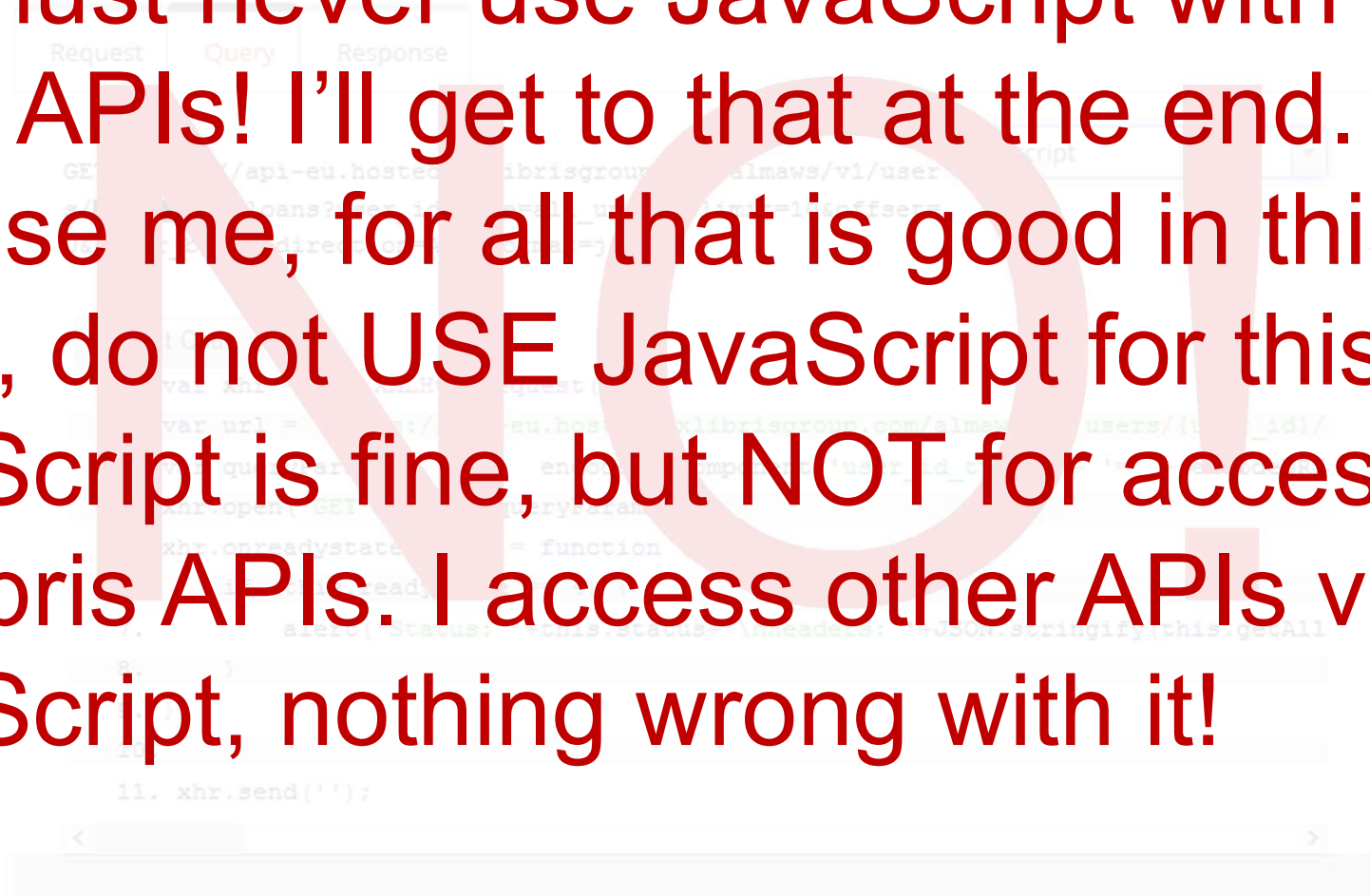


# JavaScript!

Request	Query	Response
<p>GET //api-eu.hosted-librisgroup.com/almaws/v1/users/{user_id}/loans?user_id={user_id}&amp;format=json&amp;limit=10&amp;offset=0&amp;direction=ASC</p> <p>Get Code</p> <pre>var xhr = new XMLHttpRequest(); var url = 'http://api-eu.hosted-librisgroup.com/almaws/v1/users/{user_id}/loans?user_id={user_id}&amp;format=json&amp;limit=10&amp;offset=0&amp;direction=ASC'; var queryParams = '&amp;format=json&amp;limit=10&amp;offset=0&amp;direction=ASC'; xhr.open('GET', url + queryParams); xhr.onreadystatechange = function () {     if (this.readyState == 4) {         alert('Status: ' + this.status + '\nHeaders: ' + JSON.stringify(this.getAllResponseHeaders()));     } }; 11. xhr.send('');</pre>		



**You must never use JavaScript with Ex Libris APIs! I'll get to that at the end. But promise me, for all that is good in this world, do not USE JavaScript for this! JavaScript is fine, but NOT for accessing Ex Libris APIs. I access other APIs via JavaScript, nothing wrong with it!**



But.

Not.

For.

This!

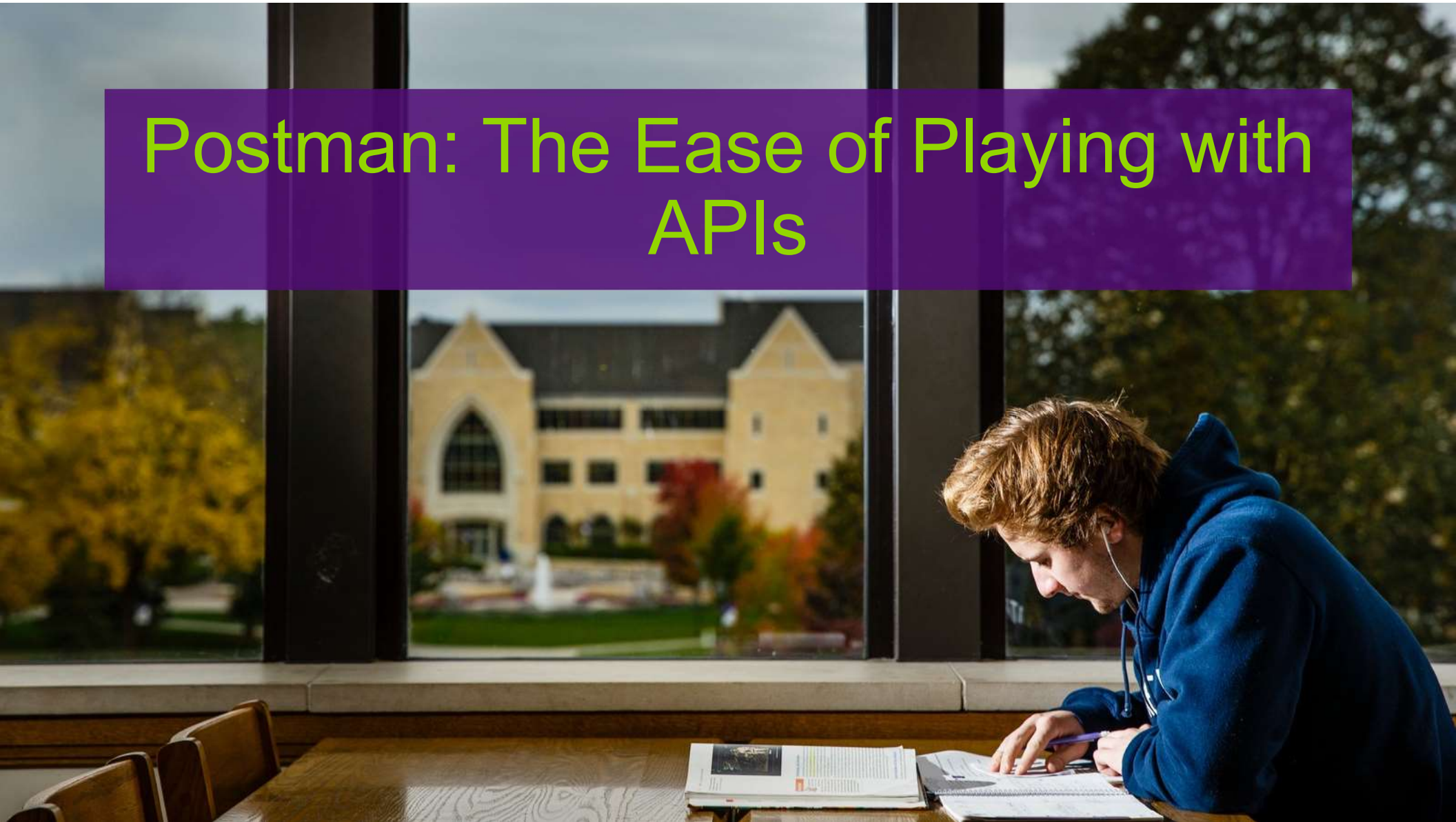




# Documentation

- You need to know what fields are available and how to make requests
- Ex Libris has extensive documentation on the Developers site
- For example, Users:
  - <https://developers.exlibrisgroup.com/alma/apis/users>

# Postman: The Ease of Playing with APIs





# Postman

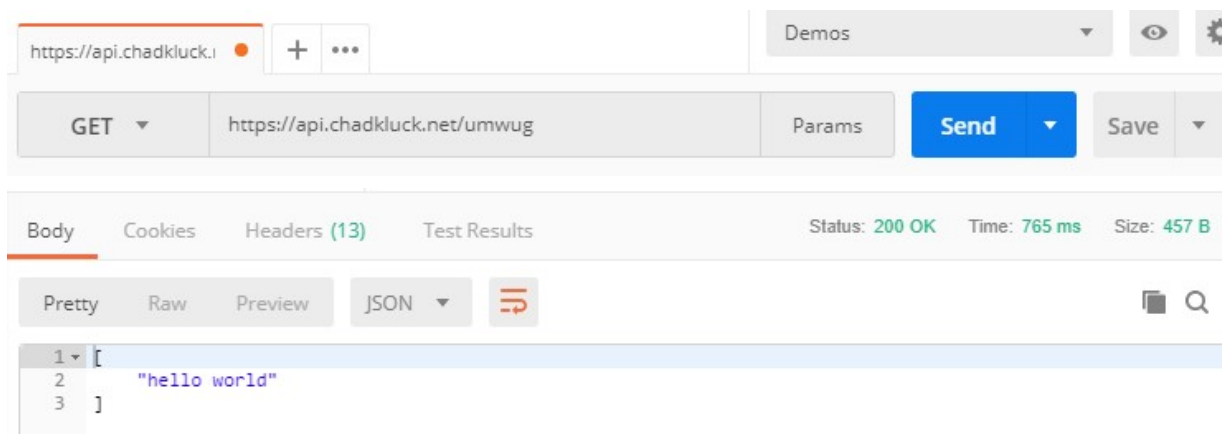
- A browser, but for APIs
- Remember that URL we used earlier?
  - <https://api.chadkluck.net/umwug>
- We'll create a new request and enter that URL
- But first we need to install the application





# Get Postman

- Download from site <https://www.getpostman.com> and install
- Create a new GET request for <https://api.chadkluck.net/umwug>
  - 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”

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	code	CPE			
	Key	Value	Description		

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?

```
Body Cookies Headers (13) Test Results
Pretty Raw Preview JSON ↻
1 {
2   "gamechoices": [
3     "Falken's Maze",
4     "Black Jack",
5     "Gin Rummy",
6     "Hearts",
7     "Bridge",
8     "Checkers",
9     "Chess",
10    "Poker",
11    "Fighter Combat",
12    "Guerrilla Engagement",
13    "Desert Warfare",
14    "Air-To-Ground Actions",
15    "Theaterwide Tactical Warfare",
16    "Theaterwide Biotoxic and Chemical Warfare",
17    "Global Thernonuclear War"
18  ],
19  "hiddengames": [
20    "Tic-Tac-Toe"
21  ]
22 }
```

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



## 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

```
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...



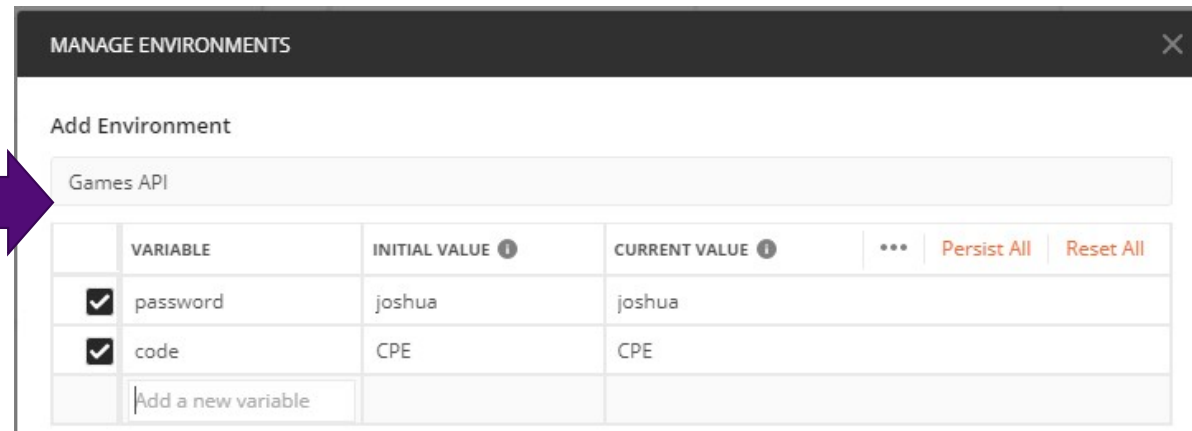
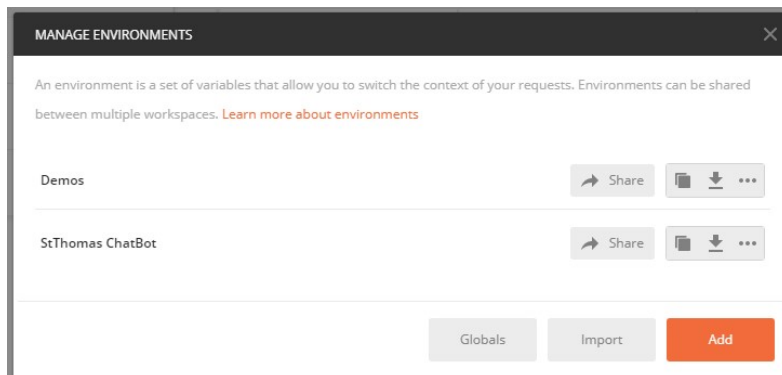
# Environment variables!



- 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/umwug?code={{code}}`



# Environment Variables: Manage/Add







## Use {{variable}} for the value

The screenshot shows a REST client interface. At the top, there's a tab labeled "Games API" with an eye icon and a settings gear icon. Below the tab, the HTTP method is set to "GET" and the URL is "https://api.chadkluck.net/umwug?code={{code}}". To the right of the URL are buttons for "Send" (in blue) and "Save". Below the URL bar is a "Params" section containing a table of query parameters.

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	code	{{code}}			
	Key	Value	Description		

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



# Environment Variables: Switch Between Values

https://api.chadkluck.i + ...

GET https://api.chadkluck.net/umwug?code={{code}}

	KEY	VALUE
<input checked="" type="checkbox"/>	code	{{code}}

Key Value Description

Games API

No Environment

Demos

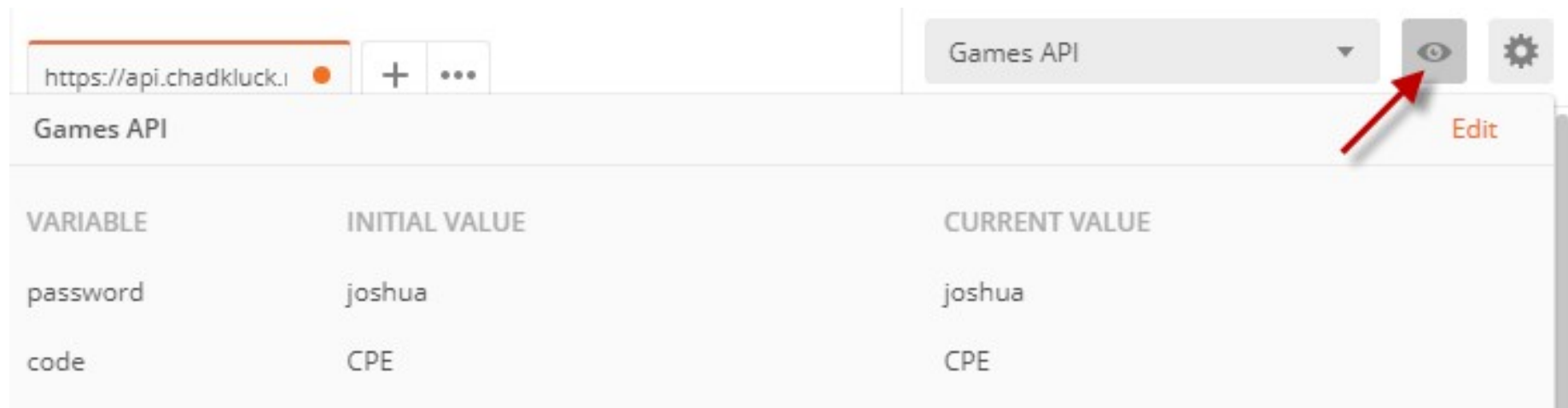
Games API

StThomas ChatBot

Save Bulk Edit



# Environment Variables: View Current Values



The screenshot shows a web application interface with a top bar containing a browser address bar with 'https://api.chadkluck.i', a '+' button, and a '...' button. To the right of the address bar is a dropdown menu labeled 'Games API', an eye icon, and a gear icon. A red arrow points to the eye icon. Below the top bar is a panel titled 'Games API' with an 'Edit' button in the top right corner. The panel contains a table with three columns: 'VARIABLE', 'INITIAL VALUE', and 'CURRENT VALUE'. The table has two rows of data: one for 'password' with initial value 'joshua' and current value 'joshua', and another for 'code' with initial value 'CPE' and current value 'CPE'.

VARIABLE	INITIAL VALUE	CURRENT VALUE
password	joshua	joshua
code	CPE	CPE



# 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
- Oh, and one more thing...



# Code snippets!

https://api.chadkluck.i

+

...

Games API

👁

⚙

GET

https://api.chadkluck.net/umwug?code={{code}}

Params

Send

Save

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	code	{{code}}			
	Key	Value	Description		

Authorization

Headers

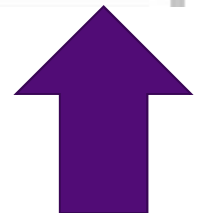
Body

Pre-request Script

Tests

Cookies

Code





# Python!

## GENERATE CODE SNIPPETS



Python Requests ▼

Copy to Clipboard

```
1 import requests
2
3 url = "https://api.chadkluck.net/umwug"
4
5 querystring = {"code":"CPE"}
6
7 headers = {
8     'Cache-Control': "no-cache",
9     'Postman-Token': "9a04dc4a-722e-4b6a-aa86-e2fcffa0b899"
10 }
11
12 response = requests.request("GET", url, headers=headers, params=querystring)
13
14 print(response.text)
```





# PHP!

GENERATE CODE SNIPPETS ✕

PHP HttpRequest ▼ Copy to Clipboard

```
1 <?php
2
3 $request = new HttpRequest();
4 $request->setUrl('https://api.chadkluck.net/umwug');
5 $request->setMethod(HTTP_METH_GET);
6
7 $request->setQueryData(array(
8     'code' => 'CPE'
9 ));
10
11 $request->setHeaders(array(
12     'Postman-Token' => 'cba57217-7a72-4dbf-87e6-0ce1dd802163',
13     'Cache-Control' => 'no-cache'
14 ));
15
16 try {
17     $response = $request->send();
18
19     echo $response->getBody();
20 } catch (HttpException $ex) {
21     echo $ex;
22 }
```



# cURL!

GENERATE CODE SNIPPETS



cURL ▼

Copy to Clipboard

```
1 curl -X GET \  
2 'https://api.chadkluck.net/umwug?code=CPE' \  
3 -H 'Cache-Control: no-cache' \  
4 -H 'Postman-Token: a66f5be3-47a3-411a-9e9a-028e5ea2aef6'
```





The background of the slide features a photograph of a multi-story building with light-colored stone or concrete walls and arched windows. In the foreground, there are several trees with dark trunks and branches, heavily laden with bright pink cherry blossoms. The blossoms are in full bloom, creating a dense canopy of pink over the building. To the right, there are some evergreen trees with dark green foliage.

Security: (aka Officer Buzz Kill)



## Security Checklist

- Are we exposing sensitive data in the API request?
  - Personal information
  - IDs that can be traversed in reverse
  - 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/umwug?code=CPE>
- Does the request:
  - Contain any personal info?
  - Reveal IDs that can be traversed in reverse?
  - Contain secrets like an API key?
  - Return any of the above?

No, No, No, and No!



```
1  {
2    "gamechoices": [
3      "Falken's Maze",
4      "Black Jack",
5      "Gin Rummy",
6      "Hearts",
7      "Bridge",
8      "Checkers",
9      "Chess",
10     "Poker",
11     "Fighter Combat",
12     "Guerrilla Engagement",
13     "Desert Warfare",
14     "Air-To-Ground Actions",
15     "Theaterwide Tactical Warfare",
16     "Theaterwide Biotoxic and Chemical Warfare",
17     "Global Thermonuclear War"
18   ],
19   "hiddengames": [
20     "Tic-Tac-Toe"
21   ]
22 }
```





## Applying the Security Checklist: Example #2

- Let's look at another example:
  - <https://api.chadkluck.net/umwug?code=CBL>
- Does the request:
  - Contain any personal info?
  - Reveal IDs that can be traversed in reverse?
  - Contain secrets like an API key?
  - Return any of the above?

No, No, No, **but YES!**

**DENIED!**

```
1 [
2   {
3     "name": "Charlie Brown",
4     "id": "1005783",
5     "fines": 38.93
6   }
7   {
8     "name": "Homer van Pelt",
9     "id": "1005784",
10    "fines": 38.93,
11    "loans": [
12      {
13        "title": "The Great Gatsby",
14        "due": "20181226"
15      },
16      {
17        "title": "Public Speaking",
18        "due": "20181226"
19      }
20    ]
21  }
22 ]
```



## Applying the Security Checklist: Example #3

- 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?
- Reveal IDs that can be traversed in reverse?
- Contain secrets like an API key?
- Return any of the above?

**DENIED!**

**YES, No, Yes, and YES!**

```
1 {  
2   "name": "Linus van Pelt",  
3   "id": "1004378",  
4   "age": 0,  
5   "books": [  
6     {  
7       "title": "Bible (KJV)",  
8       "id": "181226",  
9       "due": "20181226"},  
10    {  
11      "title": "The Hobbit",  
12      "id": "181227",  
13      "due": "20181227"},  
14    ]  
15 }
```

**REDACTED!**



# Applying the Security Checklist: Example #4

- Bibs

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

- Does the request:

- Contain any personal info?
- Reveal IDs that can be traversed in reverse?
- Contain secrets like an API key?
- Return any of the above?

No,        No, and No

```
1  {
2    "mms_id": "991006699759703691",
3    "record_format": "marc21",
4    "linked_record_id": {
5      "value": "9910041082403692",
6      "type": "NZ"
7    },
8    "title": "American folklore & the historian",
9    "author": "Dorson, Richard M.",
10   "issn": null,
11   "isbn": "0226158683",
12   "network_number": [
13     ".b13584923",
14     "478949",
15     "(OCoLC)ocm00211518",
16     "(MnSST)b13584923-01cllic_stthomas",
17     "(EXLNZ-01CLIC_NETWORK)9910041082403692"
18   ],
19   "place_of_publication": "Chicago,",
20   "date_of_publication": "[1971]",
21   "publisher_const": "University of Chicago Press",
22   "holdings": {
23     "value": null,
24     "link": "https://api-na.hosted.exlibrisgroup.com/alma
25   },
26   "created_by": "import",
27   "created_date": "2017-05-23Z",
28   "last_modified_by": "system",
29   "last_modified_date": "2018-05-18Z",
30   "suppress_from_publishing": "false",
31   "originating_system": "ILS",
32   "originating_system_id": "b13584923-01cllic_stthomas",
33   "cataloging_level": {
34     "value": "00",
35     "desc": "Default Level"
36   },
37 }
```



## Applying the Security Checklist: Example #4

- Bibs

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

- Does the request:

- Contain any personal info?
- Reveal IDs that can be traversed in reverse?
- Contain secrets like an API key?
- Return any of the above?

No, YES, No, but YES!



```
▼ item_loan:
  ▼ 0:
    loan_id: "542897"
    desk: "DEFAULT_CIRC_DESK"
    desc: "OSF Main"
    ▼ library:
      value: "Shaughnessy-Frey Library"
      desc: "Shaughnessy-Frey Library"
    user_id: "100437"
    item_barcode: "30515"
    due_date: "2018-12-26T05:55"
    loan_status: "ACTIVE"
    loan_date: "2018-09-10T16:33:22.063Z"
    call_number: "GR105.3 .A47 1987"
    renewable: null
```

**REDACTED!**



# 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."

We've all had that one door  
in the library we're curious  
about.

For weeks you glanced at it  
as you passed.

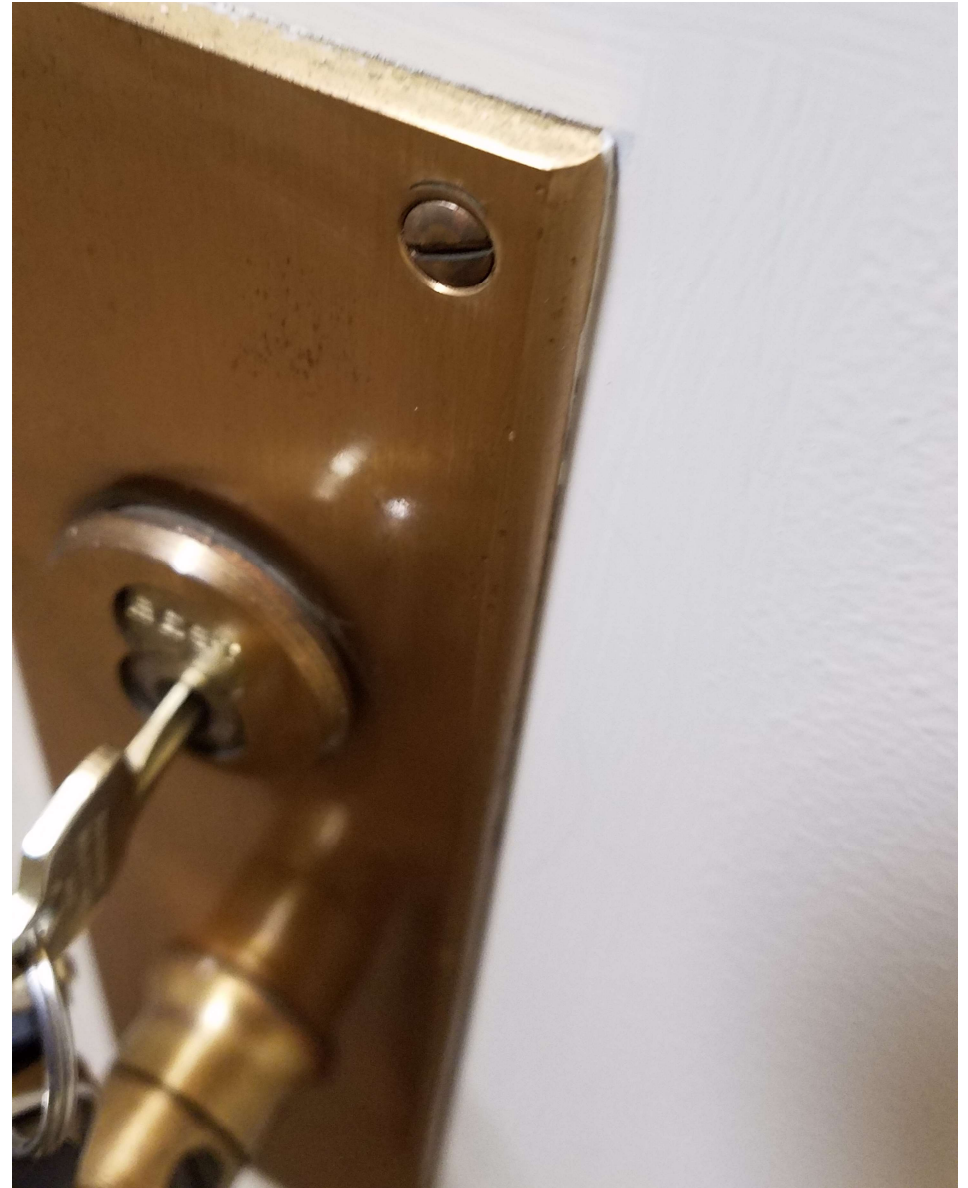
Then, one day you wondered  
if the key they entrusted to  
you opens it.





You slipped the key in.  
Perhaps you tried more than  
one, disappointed on each  
failed attempt, but still eager  
to try again.

The knob might have turned  
hard, you jiggled the key  
because you were  
determined to make it work.





Finally it worked! It opened!

You looked inside with satisfaction.

You thought, “Am I really supposed to have a key for this?”

Then you quietly closed the door and promised yourself never to tell anyone.



But, sometimes, maybe even last week, as you walked past, you checked to make sure no one was looking before you tested to see if the key still worked.



It did!

“Yes! No one has caught me yet!”

Someday, eventually, you’ll have the urge to brag in front of your peers. Then at some conference you’ll mention the thrill you felt when you had once been bored and curious.

And therefore you know what it’s like to be a hacker.





# 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

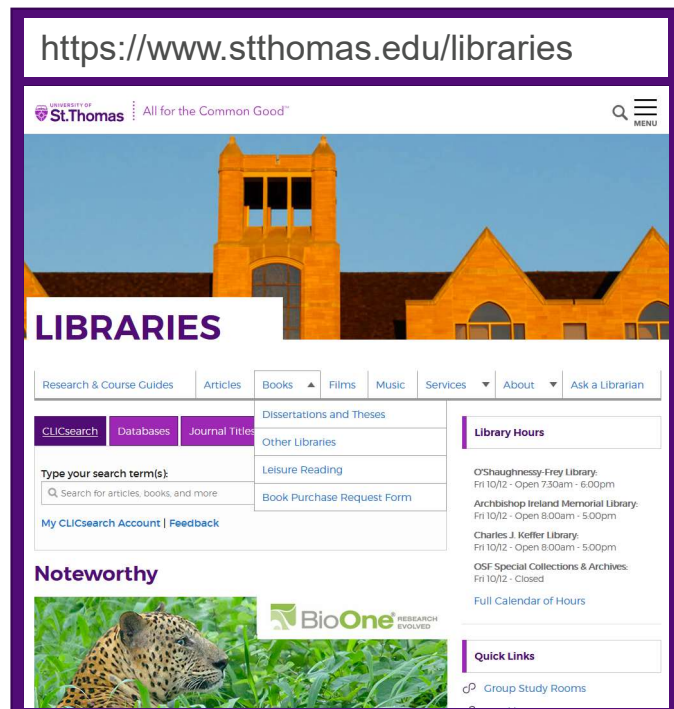
- Public feeds (blogs, event calendars, new book lists)
- At St. Thomas Libraries we use JavaScript API calls for a lot of public content
  - However we typically TRANSFORM the data before passing it on to the client-side
  - We host a variety of intermediary scripts (written in PHP)
  - We also restrict 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

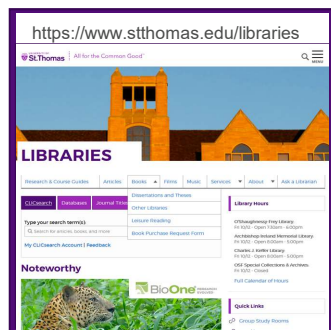
- <https://lib-api.stthomas.edu/courses?list=all>
- [/courses/index.php](#) is a server-side script that holds the API key and only returns pre-defined (non-sensitive) data fields suitable for public consumption.
- The server-side script also checks to make sure the request was made from a St. Thomas 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

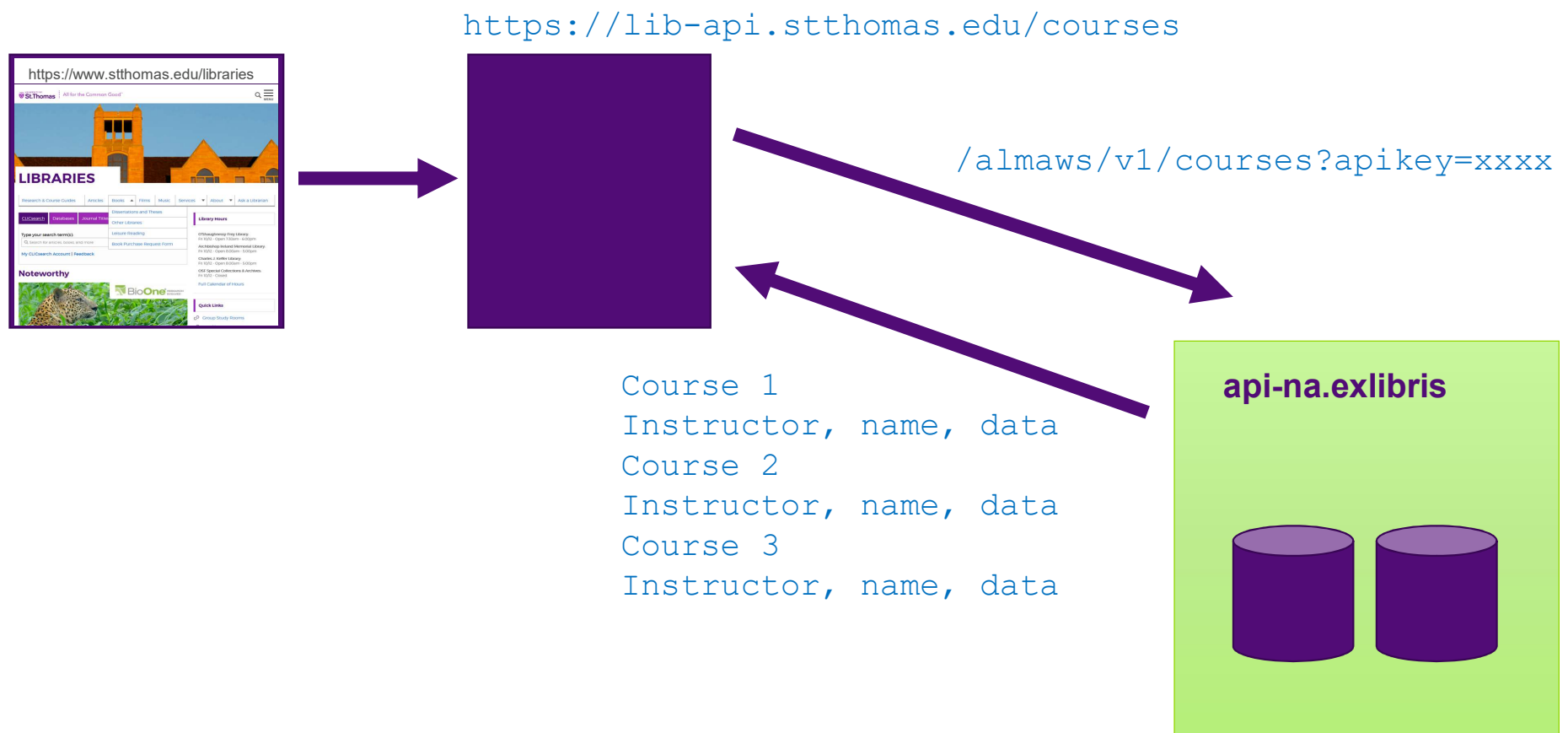
<https://lib-api.stthomas.edu/courses>

## // API Script Pseudo Code

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

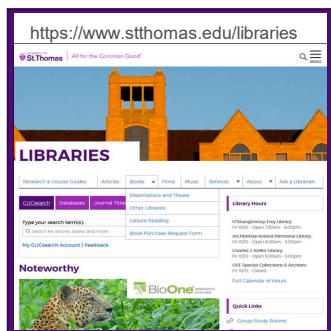


# The “backend” call





# The intermediary passes back “clean” data



<https://lib-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();
```



# Live Demo

- <https://libraries.aws.stthomas.edu/api/exlibris-api-example/>



# 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 photograph of a stone building with arched windows and a person walking on a path next to it, with a large red tree in the background. The image is used as a background for the title text.

# 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)
  - Amazon Web Services (AWS)
  - Microsoft
  - Google sheets
- 
- Be careful, you don't want a host that is prone to being compromised
  - And you don't want to store sensitive data where you shouldn't
  - Be sure to vet your hosting provider





## Server Side

- Anywhere you can run server side scripts to create a data set and then utilize a public API from that
- WordPress, Drupal
- 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](https://github.com/USTLibraries)
  - [github.com/chadkluck](https://github.com/chadkluck)

GIT



# Framework

- Developed to quickly deploy APIs and microsites
- 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
- <https://github.com/chadkluck/php-project-framework>
- Here's something to get you started:
  - <https://github.com/ustlibraries/exlibris-api-example>



# 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 (heavy scripting)
  - <https://diversion.chadkluck.net/8ball>
  - <https://github.com/chadkluck/8ball-api> (the api service)
  - <https://github.com/chadkluck/8ball-js> (the JavaScript client)



## A final piece of code

The code for `api.chadkluck.net/umwug`:

- <https://github.com/ustlibraries/umwug-2018>



## What next?

- Even if you can't set up a server environment, get familiar and play
- Play with Postman, Ex Libris APIs, 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
- This time next year: Proposition for a place to host your APIs
  - I gave you everything you need to get started





# Thank You! Questions?

[ckluck@stthomas.edu](mailto:ckluck@stthomas.edu)  
[github.com/USTlibraries](https://github.com/USTlibraries)  
[github.com/chadkluck](https://github.com/chadkluck)



## 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&sc  
ope=stthomas&apikey={{apikey}}
```



## 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}}
```



## 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}}
```



# Video of Presentation Slides with Audio

- I recorded the presentation slides along with audio:  
<https://youtu.be/eg2VVG2qCvo>