Using APIs in libraries to increase efficiency in routine tasks

Robert Williams
Assistant Librarian (e-services)
Library Services
Birkbeck College
robert.williams@bbk.ac.uk

Introduction

In recent years there has been a discussion regarding whether librarians should learn computer programming (see for example Cope & Baker 2017, Schwartz 2017, Tillman *et al*. 2018, and Yelton 2012). Yelton identifies several benefits of learning to code for librarians. These include the ability to improve workflows, improved usability of services, better communication with vendors and IT staff, and increased creativity. If you do learn to code, how can you use your new skills in a library? There are plenty of opportunities to do so. This article focuses on just one: making use of Application Programming Interfaces (APIs) in order to work more efficiently and, in this instance, enhancing the digitsation service offered by Birkbeck College Library.

What is an API?

API stands for Application Programming Interface. What follows is an oversimplification of what an API is, but it is a useful starting point when trying to understand what they are and how they can help you.

An API allows one piece of software to communicate with another. Organisations can develop APIs to give others within their own organisation, or external users, access to some of their data or services. Programmers can integrate calls to these APIs in their own software. The call to the API is often just one part of a larger programme. The programme is likely to do some of the following tasks: collect some input (either from a user or from some other input device, such as a sensor), manipulate this into a form that can be read by the API, send a request to the API, receive and read the response, and take action based on the response. The end user is not interacting with the API directly; instead, they are interacting with a programme that interacts with the API. Once the programme has sent the request to the API, it has to wait for the API to process the request and send the response. It has no involvement in this processing, and the programmer who made the API call only needs to know what the API does and how the response will be formatted, not how it does whatever it does.

For example, the Copyright Licencing Agency (CLA) provide an API that returns whether an ISBN is covered by their copying licences. This can be used to create a programme that takes an ISBN / ISSN, creates a message, sends it to the API, receives and reads the response and then informs the end user whether that ISBN is covered by the licence. The code for gathering the inputs (ISBN, licence type etc.), formatting the inputs into a message that can be read by the API, sending the message, reading the response and displaying the result all have to be written by the programmer. The call to the API is only one step (but an important one) in this larger programme.

Essentially, an API takes in data and / or instructions, does something based on what it has received, and sends back a response. The developers of an API usually produce documentation that specifies:

- how the request must be formatted for it to be processed
- what functions the API can perform / what services are available
- how the API will format the response that it returns.

The end user probably has no idea, and does not need to know, that the software they are using may be communicating with other pieces of software.

Why use APIs in libraries?

1. Using APIs in libraries can increase efficiency in repetitive tasks. Thinking about the CLA example above, at first it may not seem worth the effort of creating a bespoke programme to check whether something is covered

The copyright in items published in *SCONUL Focus* remains the property of the author(s) or their employers as the case may be.

by the CLA licence when a member of staff could copy and paste the ISBN into a look-up tool on the CLA website. This provides exactly the same information as the call to the API, so why bother? Imagine if you had a spreadsheet containing tens or hundreds of ISBNs that need to be checked, and this is not a one-off task. Spending a small amount of time writing a programme that uses the CLA's API to automate this task can save library staff time.

2. As well as being much more time-efficient than manually checking all the ISBNs, using an API is more accurate. People can easily make mistakes, especially when doing repetitive, boring tasks, but computers don't.

3. If APIs are used to automate some time-consuming and repetitive tasks, people can devote time to more useful and value-adding activities.

4. Incorporating APIs into workflows is a good personal development opportunity for a member of staff. Most jobs now ask for excellent IT skills. If an individual can say that they have worked with APIs, they definitely fulfil this criterion. As we work in an increasingly digital environment and work with large amount of data, coding skills will become more useful in the future, so getting some experience now is a good way of future-proofing your skills.

Real-life examples of API use

Below are two examples of how Birkbeck College Library have used APIs to improve efficiency in their digitisation service. These are not intended to be instructions about how they were set up or how they work. They are just examples of the type of task that can be improved with the use of APIs.

The CLA check permissions API

The CLA's check permissions API (https://cla.co.uk/our-services) provides information about whether a work is covered by their scanning licence. Every digitisation request must be checked for compliance with the licence before it is scanned. All requests are stored in an Access database that contains a form to view digitisation requests one at a time. As each request is processed, the data about it is displayed in the form and any missing information is added, and at this stage library staff check that the work is covered by the licence. To do this, staff used to copy and paste the ISBN into a search box on the CLA's website to get the information required, and this had to be done for every request. A script was written in a programming language called Visual Basic for Applications (VBA) that takes the ISBN from the form and uses it, plus some other parameters required by the CLA, to create a request that is sent to the check permissions API. The API receives this information and creates a response that contains the relevant permission information for that title, which Access reads and then displays as a pop-up message box. As well as containing the scanning permissions information, the CLA response also includes publisher information. Access checks to see if the publisher is missing from the database and adds it if needed. The script is triggered by clicking on a button on the database. To get a basic working version of this script took only a few hours, while much more time was spent testing to make sure that the information it provided was accurate.

The API is also used to do bulk checking of ISBNs. This is useful if a module is running the following year and the lecturer wants to reuse the same digitised readings. The resources are checked by exporting the requests into an Excel spreadsheet and running a programme that calls the API on each ISBN and colour codes the Excel cells green (for covered), red (for excluded), or orange (not found). With hundreds or thousands of requests reused each year, this can save a considerable amount of time.

The copyright in items published in *SCONUL Focus* remains the property of the author(s) or their employers as the case may be.

The final use of this API was to make an Excel add-in that includes a formula that calls the check permissions API. An ISBN can be checked by using the '=CLACHECK()' formula. The formula will display 'covered', 'excluded' or 'not found'. This works like any other built-in Excel formula. Every time the spreadsheet is opened the formula is re-evaluated, so if permissions change, the result of the formula is updated. This can be combined with conditional formatting to show all excluded requests in red, for example.

xISBN API

The xISBN API (https://www.worldcat.org/affiliate/webservices/xisbn/app.jsp) was developed by OCLC and returns information about ISBNs, including other editions of the work. This is useful as, unless specifically asked, library staff make the requested scans from the latest edition of the book that is in stock. Before using this API, at the permissions checking and data entry stage of processing a digitisation request, library staff would also check that the copy of the book listed on the database was the most recent edition in stock. To do this library staff would have to search the library catalogue. Using the API, we have been able to make another button that when clicked, takes the ISBN from the database and sends it to the xISBN API. This sends back to Access a list of ISBNs and the publication years of all other editions of the book. For each ISBN, the publication year is compared with the publication year on the digitisation database and for any that are more recent, the library catalogue is searched to see if it is available. When this process completes (usually a couple of seconds), a pop-up message provides information about more recent editions held by the library.

The API can also return other bibliographic information, including place of publication and author names. This data can be used to further reduce the amount of manual data entry by adding a small amount of code to check whether any of these fields are blank, and adding the data returned by the API if they are.

By using these two APIs we have moved from having to search two websites for information on each request to just clicking a few buttons. The data entry and permissions checking stage now takes a fraction of the time that it used to.

Fig. 1 shows the database form. Clicking on the 'Later Editions' button will call the xISBN API and list all the later editions available, the 'Later ed/ebook in stock' button checks to see if any of the later editions are in stock or whether an e-book is available, and the 'ISBN check' provides scanning permissions information. The pop-up box shows how the responses are displayed to the end user. The 'Get Data' button completes the 'Place of publication', 'Publication year', 'Author of source' and 'Volume' fields with information gathered from the xISBN API.

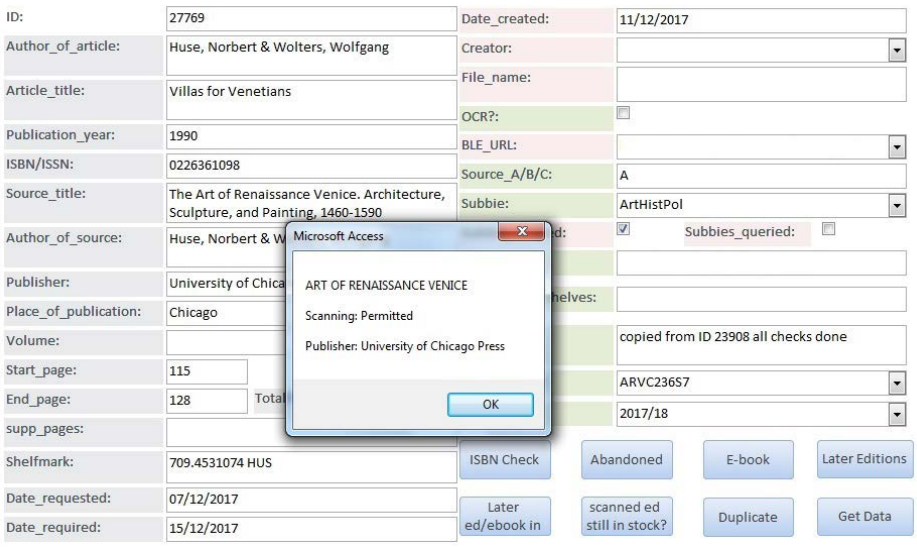

Fig. 1 xISBN database form

The copyright in items published in *SCONUL Focus* remains the property of the author(s) or their employers as the case may be.

Using both APIs together to enhance the digitsation workflow

The xISBN API has been incorporated into the CLA permissions checking spreadsheet mentioned above. For each entry on the spreadsheet, once the CLA API has performed its checks, the xISBN API is used to check whether any later editions are available in the library. The CLA tell us whether the book can be digitised and the xISBN API tell us whether we have a more recent edition that we should use (again, this is identified by colour coding). Red cells indicate definite problems. For example, a red ISBN cell means that the ISBN is not covered by the licence; a red later edition cell means a more recent edition of the title is available in the library. Orange cells indicate ISBNs that could not be identified by the CLA API and more investigation is required. Green cells confirm that no action is required (the CLA API confirmed that the ISBN is covered by the licence, and no later editions were identified). When the checks are finished, all the librarian needs to do is to look out for red or orange cells and take the appropriate action. Fig. 2 shows what the final spreadsheet looks like. Looking at this, it is easy to see which requests can be reused and which need some investigation. Fig. 3 shows a simplified flowchart of the process. The full process also involves checks to determine whether the book is still in stock (not withdrawn, gone missing, damaged, etc. since the scan was made) and whether an e-book is available. These processes do not use APIs and are therefore out of the scope of this article.

	A	B	C	D	E	F	G	H	AK	AL	AM
1	Reset ID	Author_of_article	Article_title	Publication_year	ISBN Check	Source_title	Publisher	Author_of_source	Later Edition in Stock?	Current Edition Available	E-Book Available
2	423	McLuhan, M.	The Medium is the Message (Chapter 1)	2001	0415253977	Understanding Media: the Extensions of Man	Routledge	McLuhan, M.	YES	YES	NO
3	424	Scannell, P.	Communication and technology (chap 5)	2007	9781412902694	Media and communication	Sage Publications	Scannell, P.	NO	YES	YES
4	425	Williams, R.	Effects of the Technology and its Uses (Chapter 5)	2003	9780415314565	Television: Technology and Cultural Form	routledge	Williams, R.	NO	YES	NO
5	426	Silverstone, R.	Technology (Chapter 3)	1999	0761964541	Why Study the Media	Sage Publications	Silverstone, R.	YES	YES	NO
6	428	Cook, S.D.N.	Technological revolutions and the Gutenberg Myth.	2006	0335217109	The new media theory reader.	Open University Press	Hassan, R. & Thomas, J. (eds.)	NO	YES	NO
7	450	Negroponte, N.	The DNA of information.	1996	9780679762904	Being digital	Vintage Books	Negroponte, N.	NO	YES	NO
8	451	Lax, S.	Mobile communications.	2009	1403998906	Media and communication technologies: a critical introduction	Palgrave Macmillan	Lax, S.	NO	NO	NO
9	452	Dwyer, T.	Interpreting media convergence.	2010	0335228739	Media convergence	Open University Press	Dwyer, T.	NO	YES	YES

Fig. 2 Final spreadsheet

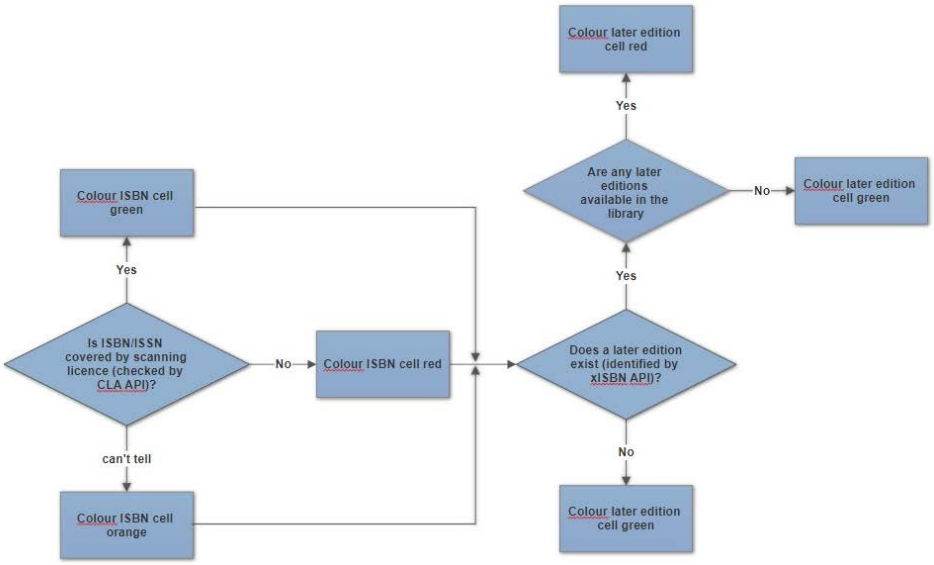

Fig. 3 Simplified flowchart of the process

The copyright in items published in *SCONUL Focus* remains the property of the author(s) or their employers as the case may be.

How the digitisation service has benefited from using APIs

The biggest benefit of automating copyright checks and data entry is the amount of time saved. Many requests only require one button click to confirm that they can be legally scanned, and another to add all the missing bibliographic data. Prior to this, the ISBN for every request would have to be copied and pasted into the CLA's website to check that the requests could be scanned, and bibliographic data would have to be found in the physical book and manually typed into the database. Data entered into the database is more accurate because there are no human errors, such as misspellings and typos.

The time saved by using APIs has two important benefits:

1. Requests can be delivered to the lecturers more quickly. This has allowed us to cope better with last-minute requests and we are more likely to be able to respond on time, so students are able to prepare for their classes more easily.

2. Staff are involved in a greater variety of tasks. Less time is spent on data entry, so staff involved in this process have been able to get involved in other areas, such as ordering material from the British Library, helping to collect books from the shelves and working on other one-off projects.

How to get started or learning to code

To use an API a member of library staff will need some coding experience. They do not need to be an expert, or even a good programmer to make a simple API call, and there are plenty of free online resources to get them going. Platforms such as Coursera, EdX and FutureLearn offer free versions of programming courses made by good universities. Websites like Codecademy have free versions of their courses. There are also many blogs and personal websites and free online books that offer good introductions to programming languages.

If you have never programmed before, the learning curve can seem steep at first, but once you have mastered a few basic skills and learned how to approach a problem as a programmer, you will be well on your way to being able to incorporate APIs into your workflows.

Links

CLA Check Permissions API: https://cla.co.uk/our-services

xISBN API: https://www.worldcat.org/affiliate/webservices/xisbn/app.jsp

Coursera: https://www.coursera.org/

Edx: https://www.edx.org/

FutureLearn: https://www.futurelearn.com/

Codecademy: https://www.codecademy.com/

References

Cope, J. & Baker, J. (2017) Library carpentry: software skills training for library professionals. *International journal of data curation*, *12* (2). pp. 266–73. Available at http://sro.sussex.ac.uk/69975/ [Accessed 07/11/2018]

Schwartz, M. (2016) Top skills for tomorrow's librarians | Careers 2016. Available at https://www.libraryjournal.com/?detailStory=top-skills-for-tomorrows-librarians-careers-2016 [Accessed 07/11/2018]

Tillman, R. *et al.* (2018) Roundtable: Should library workers learn to code? *Partnership: The Canadian journal of library and information practice and research*, 12(2). Available at https://doi.org/10.21083/partnership.v12i2.4121 [Accessed 07/11/2018]

Yarnold, A. (2012) Why should librarians learn python? (a better answer). Available at https://andromedayelton.com/2012/08/28/why-should-librarians-learn-python-a-better-answer/ [accessed 07/11/2018]

The copyright in items published in *SCONUL Focus* remains the property of the author(s) or their employers as the case may be.