Joe Cera (jcera@law.berkeley.edu)

The Benefits of Having Your Own Sandbox

We all have projects that we would like to build or tasks that we wish were more efficient. Having your own sandbox won't necessarily give you more time or magically solve your problems – it will probably do the opposite: It will take extra time and give you problems. Having this space is much more than finding a corner of the internet where you can tinker.

For those of us who are not programmers by education or occupation, the idea of wading into this area is intimidating. Think about what you might tell someone who wants to do a job like yours but doesn't have the work history or the training. Would you tell that person that it's impossible or too much work to get to where you are? I know I wouldn't. I would tell that person to ask questions of people who do a job like mine, find a way to do similar work, or try to get an internship. Having a programming sandbox basically achieves all of these tasks.

What do I mean by a sandbox? I mean that you should have a place where you can practice programming skills without the fear of something important breaking or being deleted. This could be a website that hosts the software and platforms you need without needing to learn how to configure them, it could be a server, or it could be a directory on your computer.

Skipping to the end…

Troubleshooting and success:
When you have a space to practice programming without the fear of losing important data or bricking a system, you will be more willing to try things – especially things that may not work. When these things don't work, you have an opportunity to troubleshoot (a massive part of programming). When they do work, you will have the confidence boost of success and an example to come back to when it doesn't work in a different context.

Solving real problems that you have:
There may be tools out there that almost solve a need you have. There may even be tools that do exactly what you want but you don't know about them. Either way, your sandbox is an opportunity to think about breaking this need into the smallest of steps. You may discover that you can do this task in another way. In the course of troubleshooting, you might discover someone else has already figured this out in a way you can use. You may also figure out that you should just stick to what you are doing.

Understanding the systems you use:
Creating a working prototype of something on a server is a success on its own. Maintaining one is equally a success. In these processes you can understand what goes into just getting an application to work. While many platforms have a person who specializes in each step of the process you have the ability to understand the relative difficulties of your requests relating to the platform and have a better ability to ask more specific and well-formed questions.

Gaining confidence:

You will have successes and failures in your sandbox. Each of these contributes differently to your confidence. Many of us are hesitant to take risks in our jobs, that is the nature of a librarian (except in The Librarian movie series). This is a place you can take risks and confidence will come quickly.

Self-promotion:
Many of us have felt that our job descriptions aren't representative of what we do. They can hold us back in both the tasks we tackle and how our colleagues and supervisors think about our jobs. This is mostly due to a lack of self-promotion. Working with repositories is one of the few positions in the library that is new enough to still be evolving – that is why our job descriptions don't match our actual jobs. Humans are bad at predicting the future, right? The longer you stay in this position, the less your position description will match your tasks. While your efforts won't change your job description, those efforts can change the way your job is viewed by others. Recognize that you have an opportunity to control your job and the way your job is viewed in a way that many librarians don't. Talk about what you are working on in your sandbox. That gulf you feel between your skills and what you think a programmer knows? It will shrink. Your ability to talk about what you are working on will let others know what you are doing, what you are learning, and how it contributes to the success of your institution. If you are lucky enough to have a supervisor that supports your efforts, you will find that the way you are viewed and how you view yourself will change quickly. If you have a supervisor that doesn't yet understand the importance of your growing skill set, keep talking about it. They will come around. If they don't, you will still have an important skill set that someone out there really wants.

My sandbox:
The project: law.technology (and then pads.law.technology)
The technology: A Python/Django application using PostgreSQL database and an etherpad instance
The purpose: Improving my Python/Django skills while trying do improve one aspect of a regular task
The justification: I am regularly adding faculty publications to the repository and many of these come from faculty CVs or forms submitted by faculty. Often, the citation abbreviations are correct. Sometimes, the abbreviations are close to correct. Rarely, the abbreviations are far from correct. Knowing these abbreviations by memory is not a strength of mine and looking them up took more time than I wanted to spend. I wanted to create something that was easy to use and existed exactly for this purpose.
Improvements: I have a couple colleagues who are checking citations from the repository that are not yet entered. I have then established a workflow to push these to the application. I also figured out what server blocks are and how to configure them in order to add a sub-domain with an etherpad instance on it.

Background of the project:

The idea of making abbreviations for citations easier to find and turning the task into an application actually started while I was working to refresh my Python/Django skills. I was walking through the book *Python Crash Course 2d* (https://nostarch.com/pythoncrashcourse2e) and while I was going through the various tutorials for different skills, the idea crept in and became a plan. The last few chapters of this book go through the process of creating a learning log application (try it here: https://safe-oasis-09143.herokuapp.com) which served as the model for the abbreviations application.

I would work on this for an hour or so most evenings. I would first follow a part of the tutorial on building the learning log and then I would follow the same steps to build the abbreviations application,

making adjustments as needed. Originally, I used this specifically for when I was adding publications into the repository and I wasn't confident about the abbreviation. It has grown slowly to be used for things like checking lists of faculty publications for law school publications to try to verify abbreviations in citations and also connecting journals with name changes (without always hopping into another database like Hein Online).

What was I supposed to learn/improve while doing this?
Python
Django

What did I learn/improve while doing this?
Python
Django
Git
Web design / Bootstrap
Postrgres
Linux user management
Linux permissions
Server management
Troubleshooting
More troubleshooting
Gunicorn and Nginx
Let's Encrypt SSL creation
Documentation (still learning)
Self-promotion
Project management

Let's get to the important part – what does this mean for you? The answer is that it depends on your experience.

If you have little to no programming experience:

Think about what you want to learn, not about what everyone else already knows (everyone knows less than you think they do). You will also surprise yourself by how quickly you can at least appear to know much more than you do. Pick something to learn (this could be a programming language, database management, server management, linux, etc.). It should be more specific than 'coding' or 'programming.'

Do you want to understand the command line? Install linux on a less-used computer
Do you want to learn about manipulating data? Python is a good and easy place to start, alternatively, learn SQL skills to learn how to interact with databases
Do you want to learn about evaluating statistics? Investigate R and/or Python
Do you want to build web applications? Ask others who have built applications or look for open source projects that have deployment instructions.
Do you want to have skills that will make you feel powerful (and popular)? Learn how to use Git really well (this is tough if you don't have other Git users causing conflicts) or spend time learning how to use regular expressions

Get a book with tutorials or follow online tutorials and ask around for suggestions on tools. Some of my favorite publishers are the No Starch Press books, Apress, O'Reilly, Packt publishing – there are even some good ones that may be available through your campus libraries as ebooks. You could also check your public library – they often have access to ebook collections that include these publishers and you might find that they provide access to platforms like Treehouse for some decent hand-holding. You can also look for some Humble Bundles that let you get collections of ebooks for a very reasonable price. Bonus: you will also find collections of cook books, DIY books, and more. For a great tutorial on getting started with Django, Django Girls is one of my favorite for clarity and ease of use.

Find some good places to practice your code or follow tutorials. You can use sites like repl.it, jupyter, even GitHub will let you deploy small projects as web pages. You could work locally on your Mac, Windows (extra skills to be learned here), or Linux machine. You could also ask for an investment of somewhere around $5 a month to have your own server through somewhere like Digital Ocean or Vultr (there are many others). You could even find a computer that is on its last legs that you don't really use and put a Linux distribution on there. You probably won't use it that much but it could be your 'coding computer.'

Ask for help. If you don't understand something, ask around. If you have a hard outer shell, try asking on stacksocial, reddit, or other coding forums (users can respond poorly to not well-formed questions). If you are like the rest of us, try asking someone who has some experience – almost everyone with experience is happy to share what they know.

Deploy things. Tear them down. Start over. Soon you will have programming experience.

I don't recommend trying to jump on GitHub to contribute to a project. Everyone suggests this but it is overwhelming and intimidating if you don't actually feel confident in your skills (even if you are doing the right thing).

If you have some programming experience:

Pick a project that includes some of what you know and some of what you don't. For the abbreviations application, it seems that building the app was the challenge but it wasn't. Everything outside of the application (deploying to a server, managing the server, hiding environment files, server blocks, etc.) involved things that I hadn't done before and the process of getting it to work taught me a great deal about things that probably get included in 'coding' but are outside of writing code.

Get a sandbox server and try deploying things that you are interested in. Ask others or put out a call for collaborators (like I am about to do).

If you are an experienced programmer:

Contact me so I can work on a project with you and you can show me how to do what I am doing better.

Joe Cera (jcera@law.berkeley.edu)

What's next?

My previous sandbox is no longer a real sandbox – it sees some regular use outside of my own. That means I am starting another one. If you would like to contribute to the abbreviations application, contact me and I would be happy for the help.

In my new sandbox (yet to be deployed), I would like to build an application that helps repository managers connect more easily. At a minimum, this would allow a user to easily find the contact person for any other repository. The growth for this is endless – it could include authors currently associated with an institution, it could serve as a way to communicate identifiers for authors, connect citations to publications, create a method to share download statistics of duplicated publications, or whatever else seems interesting. The limit, as always, is interest (and time).

What you could learn/improve/share?

Project management, git methods, web design skills, scripting skills, programming skills, database management skills, server management, Linux skills, and more.

How can you help?
The reality of this is that I would like to improve my own documentation for the abbreviations application. A section of this is about deploying the application and is intended to be more of a general guidance on deploying any Django application with a database while also serving as a resource for troubleshooting. The easiest way to do this is to build another application and actually document it as it is being built and deployed.

I have experience in this but I am not an expert. If you want to gain some experience in this, you would help me by contributing. I would rather go slow working with others and 'raise all boats' than just plow through this on my own. While many of the law schools feel the need to compete and that forces many of the libraries to compete (whatever that means), we can collaborate and leave the competition for others who have time for it.