Small Archives

A guide to cataloguing small archives in an open and reliable way.

https://small.archives.org.au

December 21, 2025

This is a short guide to building websites for **small archives** such as those of a family, club, association, church, or school. It sets the goal of having the publicly-accessible physical and digital items well-preserved and catalogued, and available online in a cheap and reliable way for the long term. At its core it's what can be called a "Lib-Static[1]" methodology that uses Git hosting (e.g. GitHub[2]) and a static website to host the catalogue, transcriptions, and other text, along with sites such as Wikimedia Commons[3], the Internet Archive[4], and Flickr[5] to host the digitisation and other files. It sets out how to choose where to host what material (based primarily on copyright considerations), and focuses on the processes of digitisation, description, and the long-term preservation of the archive (including printing some parts of it, and producing redistributable data bundles).

The latest version of this guide is located at small.archives.org.au[6], and any issues can be reported on its Codeberg repository[7].

Introduction

Small archives are usually not someone's main job. They might do it as part of their employment, but not the central part, and very often it's a volunteer endeavour. This means that there will likely come a time when the archive is no longer being actively worked on or even thought about regularly. Responsibility for the archive might move from person to person over time, often with a lack of continuity, and so the archive system needs to be able to describe itself and how to work on it. It should not depend on people passing this information on.

It should also degrade gracefully: if the primary domain name is not renewed, a secondary one should be available; a search system or comment form could stop working, but this should not prevent access to the actual records; if some files become unavailable, others should keep working; the parts that need paying for should be able to be paid for by anyone; and in the worst case, the whole thing should be able to be resurrected by someone without special access rights. Failure recovery should be built-in and documented from the beginning, and be as simple as possible.

Ultimately, the work described here results in a physical set of storage folders, boxes, etc. whose contents are all digitised and catalogued. Some aspects of the catalogue are printed and

[1] `https://lib-static.github.io`

[2] `https://github.com`

[3] `https://commons.wikimedia.org`

[4] `https://archive.org`

[5] `https://flickr.com`

[6] `https://small.archives.org.au`

[7] `https://codeberg.org/freosam/smallarchives/issues`

kept with the stored material, with the aim that the whole lot can be handed over to a collecting institution that does not have the resources to do more than just store it as-is. The archive's website then serves as the main entry point for researchers.

In some ways, the procedures described here are a bit more technical than is usual for some catalogue systems. There is no friendly form to fill in with item metadata, for example. This is a trade-off, putting aside some ease of use for the people *preparing* the archive, in order to make it work better for people *using* it and (even more importantly) keeping it simpler for long-term hosting. The goal is for it to be possible to more-or-less "abandon" the archive for years at a time (or even forever) and still have a high liklihood of it being online and useful.

TL;DR

If this guide is all too long[8] and you don't want to read it, it mostly boils down to the following:

1. Set up a repository on Codeberg[9] or GitHub[10] and describe your archive in Yaml front-matter and Markdown.
2. Host a static website on statichost.eu[11], GitHub Pages[12], or Netlify[13], and set up an automatic or manual process to build and publish the repository's content as the website.
3. Upload files to Wikimedia Commons[14] (public domain or open licensed), the Internet Archive[15] (copyright but orphaned etc.), and Flickr[16] (t.b.d.).
4. Make it possible to print finding aids, cover sheets, folder and box labels, etc. so that the online part of the archive is easily found by anyone looking at the physical part.
5. Create a Fediverse account (on glammr.us[17], or ausglam.space[18] and post about your progress.

Prerequisites

- Accounts on a few services.
- Digitisation fundamentals such as how to scan photos.
- Some familiarity with working on the command-line can be useful.

Access and permissions

One of the key problems with making archives publicly accessible is that of copyright. This is a topic too broad to cover properly here, but a couple of general principles to keep in mind are that physical ownership does not imply ownership of copyright, and that

[8] `https://en.wikipedia.org/wiki/TL;DR`
[9] `https://codeberg.org`
[10] `https://github.com`
[11] `https://statichost.eu/`
[12] `https://pages.github.com`
[13] `https://netlify.com`
[14] `https://commons.wikimedia.org`
[15] `https://archive.org`
[16] `https://flickr.com`
[17] `https://glammr.us/`
[18] `https://ausglam.space`

Private or confidential material is not dealt with in this system. If you have material that must *not* be made public (for a generation or two, perhaps) then other systems need to be put in place and the public internet cannot really help. This material isn't covered here because having to enforce access controls means that you need to have users log in, and then you're stuck storing their account data and basically the hosting enters a whole other level of complexity, expense, and fragility. If we draw a line and say that everything must be public, everything becomes a lot more robust and simpler.

Familiy archives have a particularly useful attribute, with regards to copyright. If you are a descendent of a copyright holder, then you may be an heir to their intellectual property. Often, this will be along with other descendents, but there is at least a possiblity of all heirs agreeing to release the material under an open license such as the Creative Commons Attribution Share-Alike, and if they do this then you can work with the material in much the same way as if it were in the public domain.

The physical storage of archive items should also follow the same access control principles as the digital items. This means that each folder, box, etc. only contains items that are public.

Digitisation

There are two main categories of digitisation that we deal with here: items such as photographs and artwork which we aim to represent as faithfully as possible; and text-based works such as letters and other documents, where the information they contain is (for the most part) separate from the format. These require different approaches to imaging, in order to find a good balance of cost, complexity, speed, and utility.

Artwork (photographs, paintings, engravings, etc.) should be scanned as single TIFF files, at least 600 DPI (or the long dimension being at least 3,000 pixels) and 24 bit colour depth, in the sRGB colourspace. Do not use multi-page TIFFs. The reverse side should also be digitised if it has anything on it. If it is blank, that should be noted in the description.

Text documents can be of lesser quality usually, as the important information is text and the digital copy only need to be sufficient to read the text. This *can* at times necessitate high resolution, for things like hard-to-read handwriting, but mostly 300 DPI is okay and keeping the individual pages smaller can speed up the processing. (If speed is not a concern, then it's probably worth doing things at a higher quality.) Photographing text documents can be much faster than scanning. You should use a flat-copy set-up, which need not be expensive: a good smartphone on a stand with a ringlight might be sufficient. Just make sure to use the camera's 2-second timer, in order to avoid camera shake when pressing the shutter button.

The output of both scanning and photographing will be a set of TIFF, JPEG, or PNG files. These should be given meaningful filenames that ensure the files are sorted in the correct order (e.g. a common prefix, and a zero-padded integer suffix).

In some cases (e.g. for Wikimedia Commons) it can be useful to prepare a PDF from these; one simple way is using img2pdf[19]. This program is better than ImageMagick and many other PDF-creation systems such as those bundled with many consumer scanners because it doesn't change the images when adding them to the PDF.

See the 'Uploading' section below for more information about what to do next with the digitised files. and the 'Cataloge' section for ideas about identifying and storing the physical items.

[19] `https://pypi.org/project/img2pdf/`

Repository

Set up a Git repository on one of the public hosts, such as Codeberg[20] or GitHub[21]. This is where all of the textual and data content of your archive will live, and none of the digitized files.

These code forges (as they're called) have systems of issue tracking, and these can serve as very useful tools for making 'to do' notes about things that need to be done with your archive. Note however that the contents of the issues are not actually part of the archive, so should not be the only location for any crucial data. For example, some archives use the issue trackers as a way for people to give feedback and more information about individual items. This works very well, and is recommended, but any information thus provided should also be copied into the catalogue record of an item.

Catalogue

The catalogue is created, item by item, as plain text pages in the repository. Each of these pages comprises two parts: a YAML-formatted frontmatter section, and a Markdown body. The structure is bespoke, and should be designed to meet the needs of the repository. The filenames and their directories should fit the collection, and the fields used within the frontmatter are entirely customisable. A good simple starting point is as follows:

- Each archive item gets a unique ID number by creating its file in a common directory, e.g. `items/123.md`.
- The YAML frontmatter should at a minimum contain the basic Dublin Core terms[22], especially title, date, description, type, and format. (Note that identifier is left out, as it's already in the filename and need not be repeated.)

Uploading

After setting up a repository and the basic structure of your catalogue, it's time to upload digital images to a host platform. Your choice of platform should be informed by considerations such as what material they accept, the long-term sustainability (economic and technical) of the platform, and the ways in which you can incorporate the uploaded material in your archival website.

The following platforms are listed in order of priority; files should be uploaded to the first of these that will accept the files.

Wikimedia Comnmons

Wikimedia Commons[23] accepts public domain or open-licenced material that is of "educational value". The scope of that is very broad, and it is usually licencing rather than scope that is the limiting factor in what you can upload. To stay on the safe side, it is best to only upload material that you know to either a) be public domain in the country of origin and in the US (where the project is hosted); or b) that you have created yourself and so are able to release under a Creative Commons license.

There are various ways of uploading to Wikimedia Commons, depending on how many files you have to upload and what support you need for different ways of handling metadata. The

[20] `https://codeberg.org`

[21] `https://github.com`

[22] `https://www.dublincore.org/specifications/dublin-core/dcmi-terms/\#section-3`

[23] `https://commons.wikimedia.org`

simplest way to get started is with the UploadWizard[24], which is always accessible via the 'Upload file' link in the sidebar of every page on Wikimedia Commons.

Internet Archive

Uploading[25] to the Internet Archive should be done as a single `_images.zip` file containing all the images you want in TIFF, JPEG, PNG, or other supported raster format.

It is also possible to upload whatever files you wish to an IA item, and to use whatever filenames you like, but if you upload them directly like this then you also have to upload derivative "display" formats (e.g. smaller JPEG versions of large TIFFs). The only derivatives that will be created automatically are JPEG versions of TIFF files, but these will have the same dimensions as the TIFF and so in many cases still be too large to serve as a convenient display copy. The default display for the item will load all JPEG and PNG images in an image "carousel", and if they're large files this results in slow loading and navigation. No automatic derivatives are created for JPEG or PNG images.

So instead of individual files, we recommend the `_images.zip` approach (what IA calls a "generic raw book zip"). Make sure the images have appropriate filenames and that they sort correctly (e.g. give each a `-001.jpg`, `-002.jpg`, etc. suffix). Zip them together so that the files are at the top level of the Zip file, and not inside any directory. The IA will then derive smaller JPEGs for each image, and display them using the "Bookreader" interface, which makes it easy to browse and search. The derivation process will also attempt to OCR any text in the images, and that text can be searched via the bookreader interface, or downloaded for use elsewhere.

Two crucial pieces of metadata when uploading to IA are the item *type* and a *backlink*. The item's "mediatype[26]" determines things like how it is displayed and what derivative formats are created. It defaults to `texts`, and can also be `audio`, `movies`, `image`, or `data` (note that it can only be changed after an item has been created by emailing the IA helpdesk). The "description[27]" field is where you can add some information about the item, but as that is already included in the item's page in your own catalogue it is simpler to just include a link to that page.

Once the files are uploaded and the derivation process has completed, display version of individual images are available at a URL of the form `https://archive.org/download/<ID>/page/leaf<number>_w1000` (where the leaf number starts from 0, and the `w1000` indicates the width in pixels; see the docs[28] for more details).

Items can also be given 'subjects', which are the same as 'tags' or 'keywords' on other platforms. It can be useful to add a unique subject to the items you upload (for example, the domain name of your archive), so that all your Internet Archive items can be grouped together.

Elsewhere

You may have some items that are not able to be hosted on Wikimedia Commons or the Internet Archive. For these, one simple approach is to rent "object storage" from a vendor such as Digital Ocean or Hetzner. This is a cheap way to host files, with the downside being that you have to manage the creation of derivative files yourself. For many digitisation projects this isn't necessarily too hard, as you may only need a single viewing-sized image for each main TIFF or PDF you are archiving.

[24] `https://commons.wikimedia.org/wiki/Special:UploadWizard`

[25] `https://archive.org/create/`

[26] `https://archive.org/developers/metadata-schema/\#mediatype`

[27] `https://archive.org/developers/metadata-schema/\#description`

[28] `https://openlibrary.org/dev/docs/bookurls\#pageimages`

Future-proofing

There are various ways in which a small archive, structured as described above, might fail. It is crucial to think of these in advance, and if possible either guard against them or define recovery processes.

- Domain name not renewed.
- Hosting not renewed (or they change the rules about their free tier).

Another principle of future-proofing is to make it possible for researchers to make a copy of your archive, or at least of parts of it. This may be difficult, especially as the digital files may be spread out over multiple platforms, but the architecture described in this guide does at least make it reasonably simple for the core textual data to be cloned as a Git repository. It is worth going beyond this though, and also making other 'dumps' available, such as the raw HTML and assets of the site (which generally will be much larger than the Git repostiroy but also much smaller than a full dump including all full-resolution images etc.). A full one-click download of the entire dump is generally not feasible, because it would have to be pre-generated and stored somewhere and in doing so more than double the storage requirements of the archive. But it can be possible to give people a somewhat easy means to download