Using Metadata

The name of the song is called 'Haddocks' Eyes.'*"*

"Oh, that's the name of the song, is it?" Alice said, trying to feel interested.

"No, you don't understand," the Knight said, looking a little vexed. "That's what the name is called. *The name really is* 'The Aged Aged Man.'*"*

"Then I ought to have said 'That's what the song *is called'?" Alice corrected herself.*

"No, you oughtn't: that's quite another thing! The song *is called* 'Ways And Means'*: but that's only what it's* called*, you know!"*

"Well, what is *the song, then?" said Alice, who was by this time completely bewildered.*

"I was coming to that," the Knight said. "The song really is 'A-sitting On A Gate'*: and the tune's my own invention."*

—Lewis Carroll, *Through the Looking-Glass*

There's information in and around files which can help you figure out what they are. This information about data is naturally called "metadata." It can be anything from plain text describing a file to complicated documents with their own format specifications, but the term is normally reserved for something with a formal, computer-readable structure.

There are standards for metadata. Lots of them. There's metadata for preservation, for aggregation, for images, for sound, for video, for any purpose you can imagine. Jenn Riley's *Glossary of Metadata Standards* (Indiana University Libraries, http://www.dlib.indiana.edu/~jenlrile/metadatamap/seeingstandards_glossary_pamphlet.pdf) lists 105 metadata standards, from AACR2 through Z39.50.

Kinds of metadata

The Harvard Library describes six kinds of metadata:

Descriptive metadata identifies a resource and describes its intellectual content. Catalog records and finding aids are two examples of descriptive metadata. Descriptive metadata supports discovery, selection, acquisition and management of resources.

Administrative metadata includes information a repository needs to manage a resource, such as ownership and billing (if applicable). Administrative metadata is often locally defined rather than conforming to a shared standard.

Technical metadata focuses on how a digital object was created, its format, format-specific technical characteristics, storage and location, etc. Accurate technical metadata helps a repository deliver digital content appropriately to users and to manage digital objects over time and keep them usable.

Structural metadata documents the logical or physical organization of a digital object. Typically, it describes the relationships among individual files that together make up the object. For example, structural metadata tells how individual images of pages should be arranged to correspond to chapters of a book. It can also describe the logical structure within files, enabling users, for example, to move to different songs or scenes in audio or video.

Preservation metadata helps a repository keep digital objects usable over long periods of time. Almost any kind of metadata can play a role in preservation, but information about the technical characteristics and processing history of the object are crucial.

Rights metadata documents the rights holders, copyright status, permissions, agreements, terms and conditions, and licensing information associated with a resource.

Since this isn't a book for programmers, I won't go into all the details of metadata formats, but it's useful to know what kinds you're likely to run into and what you can create or are creating. That "are creating" is important; sometimes you create lots of metadata without even knowing it. When you take pictures with a good modern digital camera, for instance, it creates metadata about the time of the shot, the exposure conditions, and (if it's GPS-equipped) even where you took it. When you put your pictures up on Flickr, you might be telling the world more about your whereabouts than you really wanted to. On the other hand, this information makes it easier to reconstruct what a picture is showing and what its occasion was.

If your aim is to manage a relatively small collection of files over a long period of time, the most important metadata are descriptive and rights metadata. These can provide the answers to questions like:

1. Who created the content?

2. When was it created?

3. Where was it created?

4. What context did it originally appear in?

5. Who, if anyone, owns the rights to it?

What you usually most want to know is the information about the content, not the file. Distinctions as nitpicking as the ones the White Knight made become important; the date the file was created isn't necessarily the same as the date the content of the file was created.

Informal metadata

When putting together answers to these questions, it isn't always necessary to follow any particular data structure. The most important thing is to get the information down so that it can be recovered later. Simple descriptions take the least effort and can be the most human-readable. For large collections, it's better to follow a formal data scheme, but for describing a simple collection, it's better to get the information in place by simple means than not at all. There are three simple tools at your disposal: Text files, file names, and directory structures.

A simple text description of your files, answering the five questions above, goes a long way to providing the information people will later need about them. For instance, the description of a photograph might say: "Picture of Jane Doe in front of Faneuil Hall, Boston, photographed by John

Doe, July 4, 2011." If you have a folder full of pictures, one text file is enough, naming each file and describing it.

File names provide another layer of metadata. Typically the pictures that come from a digital camera have sequential file names that don't tell you anything except what order they were taken in. Naming them to something descriptive helps people to figure out what they are, even without looking at any other files. If that picture comes from the camera as IMG_489.JPG, you can rename it to Jane_FaneuilHall.JPG.

Directories are useful for grouping files, and you can have as many levels as you want. They cost next to nothing in storage. If that picture is part of a set that you shot on a trip to Boston, you can name the directory "Boston_4-Jul-2011." Or if you were in Boston for a week, you might name it "Boston_Jul-2011" and have a subfolder, such as "4-Jul-2011," for each day of the trip. It's easy for the date on a file to change, so this gives you a record of when the picture was taken. The folder for the trip, in turn, might be in a folder called "Vacations." Other folders might be named "Family," "Cats," "School," and so on. These give your collection some structure, so you can figure out where to look for something.

That much is easy, but for serious work it's important to know about formal metadata as well.

Understanding RDF

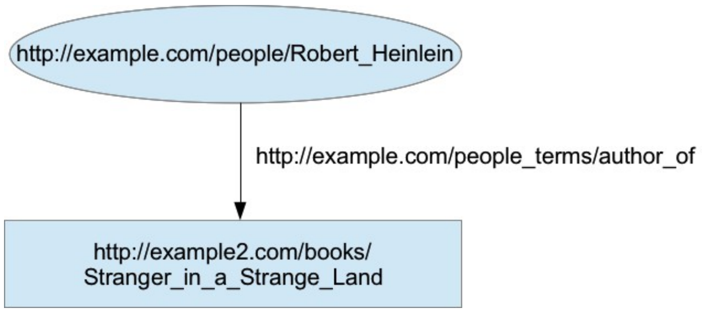
RDF (Resource Description Framework) is a way of describing relationships among items of information. Dry but authoritative material can be found from W3C at (http://www.w3.org/RDF/). It's used in a number of metadata models, so it's important to understand it at the outset. The basic idea of RDF is being able to say "X has an attribute Y with the value Z." For instance, "Grass has the color green," "Wolfgang Amadeus Mozart has the father Leopold Mozart." Statements like these are called *triples*, and their parts are called the *subject*, *predicate*, and *object*. Don't confuse these with the subject, predicate, and object of grammar; the idea is similar but the parts don't really match up. A subject, predicate, and object expressing a relationship constitute a *triple*.

RDF makes heavy use of URIs. The subject and predicate of a triple must be a URI, and the object can be either a URI or a literal value. The power of this approach is that it can connect objects from different places. An object which is a URI takes you to a new resource that can treat it as the subject of more triples. A predicate URI identifies it as part of a family of related predicates.

For example, let's take this triple:

Subject: http://example.com/people/Robert_Heinlein
Predicate: http://example.com/people_terms/author_of
Object: http://example2.com/books/Stranger_in_a_Strange_Land

The standard graphic representation for this is:


http://example.com/people/Robert_Heinlein
http://example.com/people_terms/author_of
http://example2.com/books/ Stranger_in_a_Strange_Land

The subject tells you that this could be one of a collection of subjects grouped under the domain example.com and the subcategory “people.” It could be a URL pointing to a page of information about Heinlein. The predicate tells you that example.com has one or more predicates grouped under “people terms.” The object tells you that the relationship is satisfied by an entity which a second domain, example2.com, has put under the category “books.” There can be triples which give more information about *Stranger in a Strange Land*, and their objects may lead to still more information.

A group of URI references that cover a common subject matter is a *vocabulary*. It’s common, but not required, for URIs in a vocabulary to have a common base path.

The triples themselves don’t have to come from either example.com or example2.com. They might come from your local library. RDF allows for very freely organized information structures built on simple patterns, which anyone can provide. In the cases we’ll be looking at, the metadata of a file provides the triples.

RDF is a data model, not a format. This is a distinction that will be important all through the discussion of metadata. So far I haven’t said anything about how triples are represented in a file. There are several well-known ways to represent RDF, including XML, JSON, PHP, and plain text. RDF/XML is the commonest, and often people will say just “RDF” when they mean RDF/XML. RDF/XML provides a lot of shortcuts, since using sets of three URIs everywhere can get seriously verbose. It takes the concept “vocabulary” a step further by letting XML element names in a namespace stand in for URIs. This technique is so common that people often forget that there are URIs lurking behind everything and think of the element names as the components of a vocabulary. This shortcut actually makes a lot of sense, so I’ll use it in discussing RDF metadata vocabularies.

Metadata models

Sometimes formal data structures are available to you, because the software that created the file provides them, or because someone went to the trouble to generate them. Some kinds are proprietary, but many are standard ones used across many different formats, allowing them to be kept when you migrate from one format to another. Often multiple kinds of metadata will be embedded in a single file. Covering all the metadata formats known to archivists would take a book by itself; here I’ll cover just ones which are used in files that you’re likely to run into, and which have significance outside a single file format.

Dublin Core

Bibliographic data can come in a lot of different models. One of the best known and simplest is Dublin

Core (http://dublincore.org/). It seems appropriate for it to be named for the home of famous authors like Joyce and George Bernard Shaw — except that it comes from Dublin, Ohio. It offers fifteen "core" metadata elements:

- contributor
- coverage
- creator
- date
- description
- format
- identifier
- language
- publisher
- relation
- rights
- source
- subject
- title
- type

For more detailed description, it has a set of "terms" which include the fifteen elements already mentioned plus these. Some of the terms definitely aren't self-explanatory, so I've added a few explanations.

- abstract
- accessRights
- accrualMethod (how items are added to a collection)
- accrualPeriodicity
- accrualPolicy
- alternative (name)
- audience
- bibliographicCitation
- conformsTo
- created
- dateAccepted
- dateCopyrighted
- dateSubmitted
- educationLevel (of the target audience)
- extent (page or word count, etc.)
- hasFormat (a reference to the same resource in another format)
- hasPart (a related resource included in this resource)
- hasVersion (a related resource which is a different version of this)
- instructionalMethod (which this resource supports)
- isFormatOf (about the same as hasFormat)
- isPartOf (a related resource which this is part of)
- isReferencedBy
- isReplacedBy
- isRequiredBy
- issued (date)

- isVersionOf (about the same as hasVersion)
- license
- mediator (an “entity that mediates access to the resource”)
- medium (physical form or carrier)
- modified (date)
- provenance (history of ownership or custody)
- references
- replaces (a resource this one replaces)
- requires (a resource needed for this to be usable or comprehensible)
- rightsHolder
- spatial (dimensions, etc.)
- tableOfContents
- temporal (“temporal characteristics”)
- valid (date or date range)

Dublin Core is built on RDF. The terms just listed all stand for URIs; for example, “subject” stands for `http://purl.org/dc/elements/1.1/subject` . These terms are the predicates of RDF triples. Dublin Core can be represented in all the ways RDF can. XML is just the most common. Here’s a simple example:

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rss="http://purl.org/rss/1.0/"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
    <dc:title>The Chronicles of Narnia</dc:title>
    <dc:description>
      All of Lewis's Narnia novels in one volume
    </dc:description>
    <dc:identifier>urn:ISBN:0-06-623850-1</dc:identifier>
    <dc:publisher>Harper Collins</dc:publisher>
    <dc:rights>Copyright 1950, 1951, 1952, 1953, 1954, 1955, 1956
      by C. S. Lewis Ptc. Ltd.</dc:rights>
</rdf:RDF>
```

If you’re trying to understand that in terms of RDF triples, you may wonder where the subject is. It’s implicit; the subject is the object which contains that piece of XML. This happens a lot in RDF representations. A more formal representation will wrap the DC tags in an element with a URI, e.g.,

```
<rdf:Description
  rdf:about="http://example2.com/books/The_Chronicles_of_Narnia">
    ... Dublin core elements ...
</rdf:Description>
```

Dublin Core data can be embedded in HTML files using `meta` and `link` tags. There are tools such as the Firefox Dublin Core Viewer for reading it.

RDF/XML can be created and edited with any XML or text editor.

Published works of any kind can be described with Dublin Core. Quite a few applications, including the Open Office and Microsoft Office Open XML suites, use it, but not in a way that advertises itself as such. If you run into terms like the ones listed here, understanding how they fit into the Dublin Core model may help to make sense of how they're used.

Exif

In photographs, the metadata model you're likely to run into most often is Exif, which stands for "Exchangeable Image File Format for Digital Still Camera." If it sounds a little like a translation from Japanese, that's because it is. Japan is home to some of the biggest camera manufacturers, and the standard is issued by the Japan Electronic Industry Development Association (JEIDA). Exif defines not just an abstract model but a format for metadata. Curiously, its structure is based on TIFF, even when it's used in a non-TIFF file. Originally it was intended as a common file format for cameras, much like DNG (which is also based on TIFF); today it lives on mainly as a metadata format, a TIFF file minus the image but plus a lot of tags not in the TIFF spec.

There's no official English-language website for Exif, but http://www.exif.org/ is very useful.

Broadly speaking, Exif provides metadata in the following categories:

- Basic image description (pixel dimensions, resolution, bit depth, timestamp, etc.)
- Camera or scanner information (make, model, serial number)
- Technical photographic information (shutter speed, aperture, flash, focus, color profile, etc.),
- GPS data (where the picture was taken)
- Descriptive information provided by user (description, creator, copyright, etc.)

Although its native format uses the TIFF structure, Exif information is often represented as XML. There doesn't seem to be any one dominant schema for it, so its organization might look different from different sources.

Images from most digital cameras come with Exif data, popular image formats support it, and good image manipulation software, including Photoshop and Gimp, knows how to read and write it. If you don't mind dealing with the command line, ExifTool (http://www.sno.phy.queensu.ca/~phil/exiftool/) is a versatile and free application for reading, writing, and editing all kinds of metadata, including (as you might guess) Exif. Another tool for extracting metadata, including Exif, is JHOVE. I wrote most of its code, and there's a guide to JHOVE in Appendix 4.

IPTC

IPTC is the abbreviated name of the International Press Telecommunications Council and the name of its metadata standard, more properly called the IPTC Information Interchange Model or IPTC Recommendation 7901, which goes all the way back to the 1970s. It was created as a message transmission protocol rather than a format to embed in files. Its information model was incorporated into XMP, but you may encounter IPTC data on older Photoshop files. Information on it is available at (http://www.iptc.org/cms/site/index.html?channel=CH0109). In recent versions of Photoshop you'll see IPTC data described in separate panels of the File Information dialog, but it's actually part of the XMP data.

XMP

Getting as many kinds of metadata under one overall design can be useful, and this is what XMP (Extensible Metadata Platform) aims at. Adobe published the standard and makes heavy us of it in its products. Information on it is available at (http://www.adobe.com/products/xmp/). XMP's data model is based on RDF, but it's also closely tied to XML, which is the format almost always used for it. It includes other RDF vocabularies, including Dublin Core and Exif. Photoshop lets you export XMP metadata, which can be useful if you need to store or process it separately. An example of XMP exported by Photoshop, which is too long to put here, is provided in Appendix 1.

XMP uses several different XML metadata schemas:

The *Dublin Core Schema* allows the use of Dublin Core terms in XMP.

The *XMP Basic Schema* defines terms that provide basic descriptive information which isn't in Dublin Core, such as creator tool, nickname, and thumbnails.

The *XMP Rights Management Schema* provides details about ownership and conditions of use.

The *XMP Media Management Schema* is for use by asset management systems.

The *XMP Job Ticket Schema* provides simple information for workflows.

The *XMP Paged Text Schema* provides a couple of elements for describing documents with pages.

The *XMP Dynamic Media Schema* supports audio and video metadata.

The *Camera Raw Schema* supports metadata from "raw" camera images.

The *Adobe PDF Schema* has terms for describing a PDF source.

The *Photoshop Schema* covers some miscellaneous information from Photoshop, including geographic location.

The *Exif Schemas* allow Exif data to be represented as XML. There's one schema for TIFF properties (i.e., the properties which Exif takes straight out of the TIFF spec) and another for Exif-specific properties.

If you want to read or edit XMP, ExifTool does that too. JHOVE can extract XMP as raw XML from files.

ID3

The most widely used standard for MP3 metadata has an odd history, never having had any official status in MPEG's array of specifications. Really, the name ID3 covers two completely different metadata formats. The reference site for ID3 is (http://id3.org).

The original format, ID3V1, was devised by Eric Kemp in 1996. It consists of a 128-byte block appended to an MP3 file. You can't get much information into that little space, but it gives you this:

- Identifier "TAG": 3 bytes
- Song title: 30 bytes
- Artist: 30 bytes
- Album: 30 bytes
- Year: 4 bytes
- Comment: 30 bytes
- Genre: 1 byte

The genre is an index into a fixed list of terms and obviously reflects what was hot in the mid-nineties in the US and Europe. ID3 V1.1 adds a trick to put the track number into the last byte of the comment field.

A group of programmers developed ID3V2 to overcome the limitations of the old ID3. V2 is defined in frames, allowing more kinds of data and larger amounts of it. The size of a frame is devised a a 28-bit integer (actually a 32-bit integer in which only 7 bits of each byte are used), so a frame can be up to 256 megabytes in size. ID3V2 has gone through several revisions and is currently at 2.4. The list of frames used in version 2.4 is in Appendix 2.

Today ID3 is used not only in MP3, but in MP4 (both audio and video), AIFF, WAVE, and probably other formats.

There are a number of free tools for editing ID3 metadata. You can do some editing in player applications such as iTunes, but full control requires a dedicated ID3 editor. Lots of free ones can be found with a Web search. Oh, and ExifTool even does ID3.

Conclusions

Knowing about these metadata formats can be helpful in figuring out what's in a file, or in putting information into it so others will be able to tell what's in it in the future. Their full details are of interest mostly to programmers, but it's valuable to know the kinds of information they can hold. If you want to get practice just sniffing around metadata to see what's in your files, get ExifTool. You'll be impressed by how much descriptive information is hidden in your files.

Whether you pay attention to formal metadata or not, the informal metadata of file and directory names and accompanying text descriptions can go a long way toward letting people understand what files are about. For a level 1 strategy, they may be your main source of documentation. At any level, don't neglect them.