MetadataDiscussion

Index

Introduction

This page has been created to host a discussion on how Tika returns metadata for different kinds of documents. The goal is to make sure that Tika users have a chance to get to all of the metadata created and/or extracted by Tika.

Original Problem

The original inspiration for this page was a Tika user who wanted to get access to the metadata for every document in an archive (e.g. zip, tar.gz, etc.). A way to get recursive metadata is described in the RecursiveMetadata article.

Goals for this Page

The goals for this page are bigger than the original problem. This page should hold a discussion about how to better meet different metadata needs for the different kinds of documents supported by Tika, and for the different kinds of users supported by Tika.

Since the title of this page is MetadataDiscussion, the page isn't limited to a particular set of goals, only to goals that relate to Tika's metadata handling. The goals for this page should at least include the following:

- Identify different kinds of documents and their metadata needs
- Identify different kinds of users and their metadata needs
- Identify potential metadata solutions, attempting to satisfy the needs of as many document types and user types as is practical

Document Types

This section attempts to identify document types from the point of view of metadata. This section is not trying to identify specific formats such as zip or tar, instead it is trying to identify categories of document formats that have different metadata needs such as simple, compound, and container.

The document types that have been identified so far are the following:

- Simple
- Structured
- Compound
- Simple Container
- Container with Text

Simple Document

A simple document is a single document contained in a single file. Some examples of simple documents include text files, xml files as parsed by DcXMLParser, etc. Any metadata associated with a simple document is for the entire document.

Structured Document

Like simple documents, structured documents are single documents in single files. What makes a structured document different is that the document has internal structure, and there is metadata associated specific parts of a document. For example, a PDF document has an internal structure for representing each page of the PDF, and there may be metadata associated with individual pages instead of with the entire document.

Compound Document

A compound document is logically a single document that is made up of many separate files. Usually the separate files are stored inside of a single container. Examples of compound documents include the following:

- .doc (several named streams inside of an OLE2 file)
- .xlsx (several named xml files inside of a zip file)

Simple Container

A simple container is a container file that contains other files. The container itself does not contain text, and instead it contains files that could be any document types identified so far. Examples include zip, tar, tar.gz, tar.bz2, etc.

Container with Text

Some documents have text of their own and contain other files. Examples include the following:

- an email with attachments
- A .doc with embedded spreadsheets

User Types

The following users have beed identified so far:

- ContentHandler implementers
- Tika class users
- XHTML users
- Bulk analysis users

For the most part the users are identified by the API they use Tika through, ranging from low level ContentHandler users to users who look at XHTML files generated by Tika.

The bulk analysis users aren't a specific API user, and instead are a reminder for an alternate use case. Some Tika users are using Tika to extract text and metadata from millions of documents, potentially all contained in a single container file. The metadata solution used by Tika should work even with container files that contain unreasonable numbers of files.

Potential Solutions That Don't Work

There are some good reasons why some seemingly good solutions don't actually work. This section captures solutions that have been rejected and why to help the discussion move on to solutions that will work.

A Naive Non-Solution

When I first started using Tika, I had the naive dream that I could point the AutoDetectParser at anything and it would automatically find the document boundaries that matter to me and make everything I consider a single document look like the following:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>...</title>
    <thismeta>...</thismeta>
    <thatmeta>...</thatmeta>
    <theothermeta>...</theothermeta>
  </head>
  <body>
    ...
  </body>
</html>
```

It turns out that, for lots of good reasons, the Tika developers decided not to make things work this way. Here is my understanding of why this is the case, and there is a good chance that my understanding is wrong:

1. The literal XHTML produced by Tika, and similarly the ContentHandler events generated by Tika, produce XHTML that a user could view with a browser and see something that the average person would consider the extracted text for a document.
    a. this means that all tags must be legal XHTML
    b. this also means that only metadata like "title" that a user expects to see when reading a document should appear in the XHTML in a way that a browser would display it

c. unfortunately since documents can be Structured, Compound, Simple Containers, or Containers with Text, short of reading a user's mind there is no way for Tika to normalize all of those documents into the above structure.

A Slightly Less Naive Non-Solution

This solution is like the first naive solution, except it uses legal XHTML

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>...</title>
    <meta name="description" content="Example XHTML" />
    <meta name="author" content="Paul Jakubik the Naive Programmer" />
  </head>
  <body>
    ...
  </body>
</html>
```

This solution is better since it is legal XHTML, and since meta tags aren't displayed by a browser. Unfortunately this solution doesn't handle any of the following:

- Structured documents with metadata associated with parts of the document
- Compound documents with metadata associated with parts of the compound document
- Simple containment (no representation of the contained documents and their relationship to the container)
- Containers with text (same issue)

Div Sections: No Place for Metadata

The first two non-solutions ignored that decisions have already been made about how Tika will represent structured documents and simple containers in XHTML. Tika represents a simple container document something like the following:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>...</title>
  </head>
  <body>
    <div class="package-entry">
      <h1>testfiles/5.txt</h1>
      <p>...</p>
    </div>
    <div class="package-entry">
      <h1>testfiles/getty11.zip</h1>
      <div class="package-entry">
        <h1>testfiles/getty11.txt</h1>
        <p>...</p>
      </div>
    </div>
  </body>
</html>
```

Tika uses the `<div>` tag to show subparts of a document. The subparts can have their own subparts by nesting `<div>` sections inside of other `<div>` sections. The result is a structured XHTML document that can show arbitrary levels of nesting and can represent the text extracted from nearly any kind of document.

The problem is that there is no place to put the metadata that is legal XHTML. The `<meta>` tags can only appear in the `<head>` section. Even if we wanted to put all metadata in the `<head>` section, doing so would mean that Tika could not stream the XHTML events, and instead of have to parse entire containers in two passes: once to gather the metadata, and a second time to output all of the text.

If XHTML had a way to specify arbitrary name-value pairs somewhere in the `<div>` section, that could be used as a place to associate metadata with a `<div>` section. As far as I can tell from the specification [http://www.w3schools.com/tags/tag_div.asp] there isn't a place for arbitrary name-value pairs.

Potential Solutions That Could Work

Hopefully we can find some solutions that actually work, and work for many kinds of users. It doesn't look like there is a way to represent metadata for nested sections or nested documents in XHTML, but there may be other ways to make metadata nested metadata available to some users.

Metadata for ContentHandler Implementors: Metadata Stack in ParseContext

If you are going to the effort of implementing a ContentHandler, the RecursiveMetadata page describes how you can ret access to recursive metadata.

A Solution for Users Who Don't Implement ContentHandler

If XHTML doesn't offer a legal way to associate arbitrary name-value pairs with a `<div>` section, then there don't seem to be options for providing full metadata in a single XHTML document. There are at least a couple of possibilities for providing a better-than-nothing solution for users who want all of the metadata without having to write their own ContentHandler.

The possibilities identified so far are the following:

- Add a ConvertToMultiDocContentHandler that produces multiple XHTML files, one file per line of the output file.
- Add a ConvertToXmlDocContentHandler that produces an XML file that is similar to XHTML, but allows `<meta>` tags in each `<div>` section
- Add a ConvertToXmlDocContentHandler that produces an XML file that is completely unlike XHTML to avoid confusion, and is capable of representing nested sections, each having their own metadata.

Once there is a way to implement a ContentHandler and get all of the nested metadata associated with each `<div>` section, special content handlers could be written that translate the events into something other than a single XHTML document.

If a ContentHandler was going to create an output file with multiple XHTML documents in it, it would have to decide where the document boundaries are (while not perfect for everyone, probably good enough for 80% of users), and it would have to have a way of encoding newlines so the each XHTML document could be put on a single line (for easy parsing later). This could be as simple as surrounding each line with `<p></p>` and dropping the `\r` and `\n` characters.

Yet another alternate content handler could convert the output to XML that has a place for associating metadata with each subsection. This XML could resemble XHTML, or to avoid confusion, could be completely different from XHTML.

Next Steps

- Identify other use cases
- Identify other solutions
- Discuss pros and cons of solutions
- Reach consensus on an adequate path forward for nested metadata