MetadataRoadmap

Metadata roadmap

Introduction

With Tika’s focus on mimetype detection and content extraction, the support for content metadata is somewhat suboptimal as of today. There is not much usage of standard metadata semantics and related namespaces and when trying to map the information to a data model like XMP, the current implementation is fairly limited.

This page shall provide a roadmap about how to improve the support for metadata in Tika going forward and shall serve as a basis to discuss this. The idea of providing a unified access to a common set of format independent metadata is one main aspect of this discussion. The output of metadata as XMP data model is another.

Current situation

The Metadata implementation in Tika as of April 2012:

1. Each parser fills a Metadata map which is a simple key-value list where values can also be multi-values
2. Mostly the keys for the Metadata map are taken from fixed lists which are defined as interfaces in the Metadata class
3. Those keys are usually Property objects, where the Property class also serves as a static list which registers every property that is created in the Metadata interfaces. This property class resembles the XMP data model to some extend but does not store e.g. any hierarchical information. And it leaves every client the choice to store property names with prefixes or not.
4. Any metadata outputter just iterates over the Metadata map and could query the Property list for additional information.
5. In case of the XMP outputter (XMPContentHandler) only those properties are outputted which are stored with a prefix in the Property list.

The general idea of what to accomplish

Tika should support a unifying access to common metadata properties like title, description, keywords, creator, rating, etc. So there should be a clear semantic for those common properties regardless of the underlying implementation in various metadata containers. And the access to these properties should be easy and fast. On the other hand, Tika should allow to access and manage file format specific metadata and its underlying semantic in a consolidated and flexible way, i.e. using one data model to provide information to clients.

While the current Metadata map can be used to offer easy access to the common set of properties, an XMP output could be used to offer a more extensive, flexible and semantically clearer access to a file's metadata.

The recommendation is to use Dublin Core and the semantic of the ISO part of XMP - which builds on top of DC - for common and file format neutral Tika properties as both are already being used by several standards like IPTC and MWG for this purpose.

Roadmap

The following steps shall provide a roadmap to reach the above goal

1. **Reorganize metadata keys internally**

The Metadata keys and their interfaces should be reorganized and renamed in two groups: First in namespaces and second in standards which just contain lists of aliases with the properties they use from the namespace interfaces. The reason is that only those two concepts have unambiguous and clearly defined semantics where each client knows what to do with it.

Properties which are currently not connected to a namespace (like the properties from MSOffice interface) would also be moved to an appropriate namespace interface.
To not have to prefix each namespace property, the namespace interfaces should be removed from Metadata class and aliases be added to the class to keep backwards compatibility.

No parser or client has to be changed.

1. **Move XMP output to an extra XMP module of Tika**

Add an extra Tika module that takes the metadata map from Tika-core and transforms it into XMP. Add XMPCore library from Maven.org to this module and use create XMP output.

Add a static Tika-to-XMP mapping table for the common set of properties and file formats to have a first working version of XMP output.

1. **Correct parsers where necessary**

Adjust the parsers to map metadata not only to the current mappings but also to the correct set of common properties and namespaces (i.e. DublinCore and XMP ones) and maybe add file format specific properties.

Declare current mappings deprecated if needed.
Still no client changes needed.

1. **Add support for structured data to metadata class**

There is need to have a data model which is able to faithfully store all metadata information, also structured one. There are several potential ways to solve this. Either the current HashMap is extended to also support structured properties or other structured data models like XMP could be used as internal representation for that. The Metadata API will be kept as is, just the internal representation of the data will be changed. Additional APIs should be introduced to manipulate structured properties.

Any client provided data that cannot be mapped to existing namespaces, will be stored in a special Tika namespace.
Still no client has to change.

1. **Introduce versioning scheme for metadata mappings**

This is very useful if mappings of metadata properties need to be changed in the future. Such changes are versioned and Clients can then pass the mapping version they are interested in through the parsing context. This will ensure backwards compatibility while allowing for changes and improvements.

The default implementation provides the latest version.

1. **Introduce the ability for clients to define own mappings**

This is an optional step that would allow clients to pass in own metadata mappings they are interested in, in case they want to have access to data that the default mapping is not providing.